

and providing a reasonably uniform sampling of the environment. Automatically making this selection and smoothly transitioning between representations based on the extent of the panorama is an active area of current research (Kopf, Uyttendaele, Deussen *et al.* 2007).

An interesting recent development in panoramic photography has been the use of stereographic projections looking down at the ground (in an outdoor scene) to create “little planet” renderings.¹³

View selection. Once we have chosen the output parameterization, we still need to determine which part of the scene will be *centered* in the final view. As mentioned above, for a flat composite, we can choose one of the images as a reference. Often, a reasonable choice is the one that is geometrically most central. For example, for rotational panoramas represented as a collection of 3D rotation matrices, we can choose the image whose z -axis is closest to the average z -axis (assuming a reasonable field of view). Alternatively, we can use the average z -axis (or quaternion, but this is trickier) to define the reference rotation matrix.

For larger, e.g., cylindrical or spherical, panoramas, we can use the same heuristic if a subset of the viewing sphere has been imaged. In the case of full 360° panoramas, a better choice might be to choose the middle image from the sequence of inputs, or sometimes the first image, assuming this contains the object of greatest interest. In all of these cases, having the user control the final view is often highly desirable. If the “up vector” computation described in Section 9.2.1 is working correctly, this can be as simple as panning over the image or setting a vertical “center line” for the final panorama.

Coordinate transformations. After selecting the parameterization and reference view, we still need to compute the mappings between the input and output pixels coordinates.

If the final compositing surface is flat (e.g., a single plane or the face of a cube map) and the input images have no radial distortion, the coordinate transformation is the simple homography described by (9.5). This kind of warping can be performed in graphics hardware by appropriately setting texture mapping coordinates and rendering a single quadrilateral.

If the final composite surface has some other analytic form (e.g., cylindrical or spherical), we need to convert every pixel in the final panorama into a viewing ray (3D point) and then map it back into each image according to the projection (and optionally radial distortion) equations. This process can be made more efficient by precomputing some lookup tables, e.g., the partial trigonometric functions needed to map cylindrical or spherical coordinates to 3D coordinates or the radial distortion field at each pixel. It is also possible to accelerate this process by computing exact pixel mappings on a coarser grid and then interpolating these values.

When the final compositing surface is a texture-mapped polyhedron, a slightly more sophisticated algorithm must be used. Not only do the 3D and texture map coordinates have to be properly handled, but a small amount of *overdraw* outside the triangle footprints in the texture map is necessary, to ensure that the texture pixels being interpolated during 3D rendering have valid values (Szeliski and Shum 1997).

¹³ These are inspired by *The Little Prince* by Antoine De Saint-Exupery. Go to <http://www.flickr.com> and search for “little planet projection”.

Sampling issues. While the above computations can yield the correct (fractional) pixel addresses in each input image, we still need to pay attention to sampling issues. For example, if the final panorama has a lower resolution than the input images, pre-filtering the input images is necessary to avoid aliasing. These issues have been extensively studied in both the image processing and computer graphics communities. The basic problem is to compute the appropriate pre-filter, which depends on the distance (and arrangement) between neighboring samples in a source image. As discussed in Sections 3.5.2 and 3.6.1, various approximate solutions, such as MIP mapping (Williams 1983) or elliptically weighted Gaussian averaging (Greene and Heckbert 1986) have been developed in the graphics community. For highest visual quality, a higher order (e.g., cubic) interpolator combined with a spatially adaptive pre-filter may be necessary (Wang, Kang, Szeliski *et al.* 2001). Under certain conditions, it may also be possible to produce images with a higher resolution than the input images using the process of *super-resolution* (Section 10.3).

9.3.2 Pixel selection and weighting (de-ghosting)

Once the source pixels have been mapped onto the final composite surface, we must still decide how to blend them in order to create an attractive-looking panorama. If all of the images are in perfect registration and identically exposed, this is an easy problem, i.e., any pixel or combination will do. However, for real images, visible seams (due to exposure differences), blurring (due to mis-registration), or ghosting (due to moving objects) can occur.

Creating clean, pleasing-looking panoramas involves both deciding which pixels to use and how to weight or blend them. The distinction between these two stages is a little fluid, since per-pixel weighting can be thought of as a combination of selection and blending. In this section, we discuss spatially varying weighting, pixel selection (seam placement), and then more sophisticated blending.

Feathering and center-weighting. The simplest way to create a final composite is to simply take an *average* value at each pixel,

$$C(\mathbf{x}) = \sum_k w_k(\mathbf{x}) \tilde{I}_k(\mathbf{x}) \Bigg/ \sum_k w_k(\mathbf{x}), \quad (9.37)$$

where $\tilde{I}_k(\mathbf{x})$ are the *warped* (re-sampled) images and $w_k(\mathbf{x})$ is 1 at valid pixels and 0 elsewhere. On computer graphics hardware, this kind of summation can be performed in an *accumulation buffer* (using the *A* channel as the weight).

Simple averaging usually does not work very well, since exposure differences, mis-registrations, and scene movement are all very visible (Figure 9.14a). If rapidly moving objects are the only problem, taking a *median* filter (which is a kind of pixel selection operator) can often be used to remove them (Figure 9.14b) (Irani and Anandan 1998). Conversely, center-weighting (discussed below) and *minimum likelihood* selection (Agarwala, Dontcheva, Agrawala *et al.* 2004) can sometimes be used to retain multiple copies of a moving object (Figure 9.17).

A better approach to averaging is to weight pixels near the center of the image more heavily and to down-weight pixels near the edges. When an image has some cutout regions,

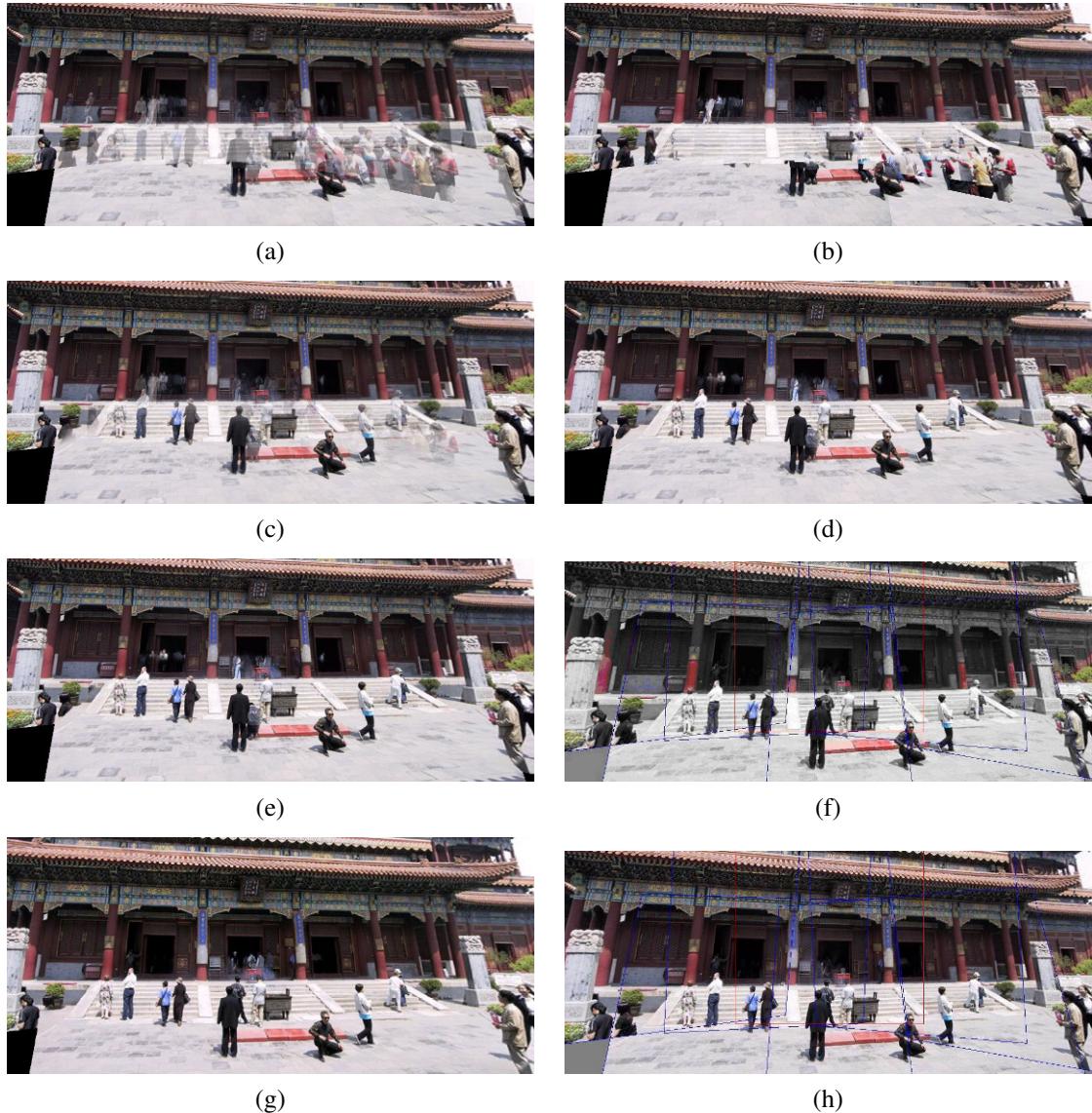


Figure 9.14 Final composites computed by a variety of algorithms (Szeliski 2006a): (a) average, (b) median, (c) feathered average, (d) p -norm $p = 10$, (e) Voronoi, (f) weighted ROD vertex cover with feathering, (g) graph cut seams with Poisson blending and (h) with pyramid blending.

down-weighting pixels near the edges of both cutouts and the image is preferable. This can be done by computing a *distance map* or *grassfire transform*,

$$w_k(\mathbf{x}) = \arg \min_{\mathbf{y}} \{ \|\mathbf{y}\| \mid \tilde{I}_k(\mathbf{x} + \mathbf{y}) \text{ is invalid } \}, \quad (9.38)$$

where each valid pixel is tagged with its Euclidean distance to the nearest invalid pixel (Section 3.3.3). The Euclidean distance map can be efficiently computed using a two-pass raster algorithm (Danielsson 1980; Borgefors 1986).

Weighted averaging with a distance map is often called *feathering* (Szeliski and Shum 1997; Chen and Klette 1999; Uyttendaele, Eden, and Szeliski 2001) and does a reasonable job of blending over exposure differences. However, blurring and ghosting can still be problems (Figure 9.14c). Note that weighted averaging is *not* the same as compositing the individual images with the classic *over* operation (Porter and Duff 1984; Blinn 1994a), even when using the weight values (normalized to sum up to one) as *alpha* (translucency) channels. This is because the over operation attenuates the values from more distant surfaces and, hence, is not equivalent to a direct sum.

One way to improve feathering is to raise the distance map values to some large power, i.e., to use $w_k^p(\mathbf{x})$ in Equation (9.37). The weighted averages then become dominated by the larger values, i.e., they act somewhat like a *p-norm*. The resulting composite can often provide a reasonable tradeoff between visible exposure differences and blur (Figure 9.14d).

In the limit as $p \rightarrow \infty$, only the pixel with the maximum weight is selected,

$$C(\mathbf{x}) = \tilde{I}_{l(\mathbf{x})}(\mathbf{x}), \quad (9.39)$$

where

$$l = \arg \max_k w_k(\mathbf{x}) \quad (9.40)$$

is the *label assignment* or *pixel selection* function that selects which image to use at each pixel. This hard pixel selection process produces a visibility mask-sensitive variant of the familiar *Voronoi diagram*, which assigns each pixel to the nearest image center in the set (Wood, Finkelstein, Hughes *et al.* 1997; Peleg, Rousso, Rav-Acha *et al.* 2000). The resulting composite, while useful for artistic guidance and in high-overlap panoramas (*manifold mosaics*) tends to have very hard edges with noticeable seams when the exposures vary (Figure 9.14e).

Xiong and Turkowski (1998) use this Voronoi idea (local maximum of the grassfire transform) to select seams for Laplacian pyramid blending (which is discussed below). However, since the seam selection is performed sequentially as new images are added in, some artifacts can occur.

Optimal seam selection. Computing the Voronoi diagram is one way to select the *seams* between regions where different images contribute to the final composite. However, Voronoi images totally ignore the local image structure underlying the seam.

A better approach is to place the seams in regions where the images agree, so that transitions from one source to another are not visible. In this way, the algorithm avoids “cutting through” moving objects where a seam would look unnatural (Davis 1998). For a pair of images, this process can be formulated as a simple dynamic program starting from one edge

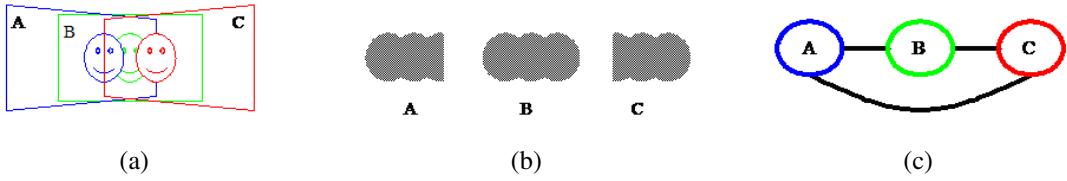


Figure 9.15 Computation of regions of difference (RODs) (Uyttendaele, Eden, and Szeliski 2001) © 2001 IEEE: (a) three overlapping images with a moving face; (b) corresponding RODs; (c) graph of coincident RODs.

of the overlap region and ending at the other (Milgram 1975, 1977; Davis 1998; Efros and Freeman 2001).

When multiple images are being composited, the dynamic program idea does not readily generalize. (For square texture tiles being composited sequentially, Efros and Freeman (2001) run a dynamic program along each of the four tile sides.)

To overcome this problem, Uyttendaele, Eden, and Szeliski (2001) observed that, for well-registered images, moving objects produce the most visible artifacts, namely translucent looking *ghosts*. Their system therefore decides which objects to keep and which ones to erase. First, the algorithm compares all overlapping input image pairs to determine *regions of difference* (RODs) where the images disagree. Next, a graph is constructed with the RODs as vertices and edges representing ROD pairs that overlap in the final composite (Figure 9.15). Since the presence of an edge indicates an area of disagreement, vertices (regions) must be removed from the final composite until no edge spans a pair of remaining vertices. The smallest such set can be computed using a *vertex cover* algorithm. Since several such covers may exist, a *weighted vertex cover* is used instead, where the vertex weights are computed by summing the feather weights in the ROD (Uyttendaele, Eden, and Szeliski 2001). The algorithm therefore prefers removing regions that are near the edge of the image, which reduces the likelihood that partially visible objects will appear in the final composite. (It is also possible to infer which object in a region of difference is the foreground object by the “edginess” (pixel differences) across the ROD boundary, which should be higher when an object is present (Herley 2005).) Once the desired excess regions of difference have been removed, the final composite can be created by feathering (Figure 9.14f).

A different approach to pixel selection and seam placement is described by Agarwala, Dontcheva, Agrawala *et al.* (2004). Their system computes the label assignment that optimizes the sum of two objective functions. The first is a per-pixel *image objective* that determines which pixels are likely to produce good composites,

$$\mathcal{C}_D = \sum_{\mathbf{x}} D(\mathbf{x}, l(\mathbf{x})), \quad (9.41)$$

where $D(\mathbf{x}, l)$ is the *data penalty* associated with choosing image l at pixel \mathbf{x} . In their system, users can select which pixels to use by “painting” over an image with the desired object or appearance, which sets $D(\mathbf{x}, l)$ to a large value for all labels l other than the one selected by the user (Figure 9.16). Alternatively, automated selection criteria can be used, such as *maximum likelihood*, which prefers pixels that occur repeatedly in the background (for object removal), or *minimum likelihood* for objects that occur infrequently, i.e., for moving object retention. Using a more traditional center-weighted data term tends to favor objects that are



Figure 9.16 Photomontage (Agarwala, Dontcheva, Agrawala *et al.* 2004) © 2004 ACM. From a set of five source images (of which four are shown on the left), Photomontage quickly creates a composite family portrait in which everyone is smiling and looking at the camera (right). Users simply flip through the stack and coarsely draw strokes using the designated source image objective over the people they wish to add to the composite. The user-applied strokes and computed regions (middle) are color-coded by the borders of the source images on the left.

centered in the input images (Figure 9.17).

The second term is a *seam objective* that penalizes differences in labelings between adjacent images,

$$\mathcal{C}_S = \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{N}} S(\mathbf{x}, \mathbf{y}, l(\mathbf{x}), l(\mathbf{y})), \quad (9.42)$$

where $S(\mathbf{x}, \mathbf{y}, l_x, l_y)$ is the image-dependent *interaction penalty* or *seam cost* of placing a seam between pixels \mathbf{x} and \mathbf{y} , and \mathcal{N} is the set of \mathcal{N}_4 neighboring pixels. For example, the simple color-based seam penalty used in (Kwatra, Schödl, Essa *et al.* 2003; Agarwala, Dontcheva, Agrawala *et al.* 2004) can be written as

$$S(\mathbf{x}, \mathbf{y}, l_x, l_y) = \|\tilde{I}_{l_x}(\mathbf{x}) - \tilde{I}_{l_y}(\mathbf{x})\| + \|\tilde{I}_{l_x}(\mathbf{y}) - \tilde{I}_{l_y}(\mathbf{y})\|. \quad (9.43)$$

More sophisticated seam penalties can also look at image gradients or the presence of image edges (Agarwala, Dontcheva, Agrawala *et al.* 2004). Seam penalties are widely used in other computer vision applications such as stereo matching (Boykov, Veksler, and Zabih 2001) to give the labeling function its *coherence* or *smoothness*. An alternative approach, which places seams along strong consistent edges in overlapping images using a watershed computation is described by Soille (2006).

The sum of these two objective functions gives rise to a *Markov random field* (MRF), for which good optimization algorithms are described in Sections 3.7.2 and 5.5 and Appendix B.5. For label computations of this kind, the α -expansion algorithm developed by Boykov, Veksler, and Zabih (2001) works particularly well (Szeliski, Zabih, Scharstein *et al.* 2008).

For the result shown in Figure 9.14g, Agarwala, Dontcheva, Agrawala *et al.* (2004) use a large data penalty for invalid pixels and 0 for valid pixels. Notice how the seam placement algorithm avoids regions of difference, including those that border the image and that might result in objects being cut off. Graph cuts (Agarwala, Dontcheva, Agrawala *et al.* 2004) and vertex cover (Uyttendaele, Eden, and Szeliski 2001) often produce similar looking results, although the former is significantly slower since it optimizes over all pixels, while the latter is more sensitive to the thresholds used to determine regions of difference.



Figure 9.17 Set of five photos tracking a snowboarder’s jump stitched together into a seamless composite. Because the algorithm prefers pixels near the center of the image, multiple copies of the boarder are retained.

9.3.3 Application: Photomontage

While image stitching is normally used to composite partially overlapping photographs, it can also be used to composite repeated shots of a scene taken with the aim of obtaining the best possible composition and appearance of each element.

Figure 9.16 shows the *Photomontage* system developed by Agarwala, Dontcheva, Agrawala *et al.* (2004), where users draw strokes over a set of pre-aligned images to indicate which regions they wish to keep from each image. Once the system solves the resulting multi-label graph cut (9.41–9.42), the various pieces taken from each source photo are blended together using a variant of Poisson image blending (9.44–9.46). Their system can also be used to automatically composite an all-focus image from a series of bracketed focus images (Hasinoff, Kutulakos, Durand *et al.* 2009) or to remove wires and other unwanted elements from sets of photographs. Exercise 9.10 has you implement this system and try out some of its variants.

9.3.4 Blending

Once the seams between images have been determined and unwanted objects removed, we still need to blend the images to compensate for exposure differences and other mis-alignments. The spatially varying weighting (feathering) previously discussed can often be used to accomplish this. However, it is difficult in practice to achieve a pleasing balance between smoothing out low-frequency exposure variations and retaining sharp enough transitions to prevent blurring (although using a high exponent in feathering can help).

Laplacian pyramid blending. An attractive solution to this problem is the Laplacian pyramid blending technique developed by Burt and Adelson (1983b), which we discussed in Section 3.5.5. Instead of using a single transition width, a frequency-adaptive width is used by creating a band-pass (Laplacian) pyramid and making the transition widths within each level

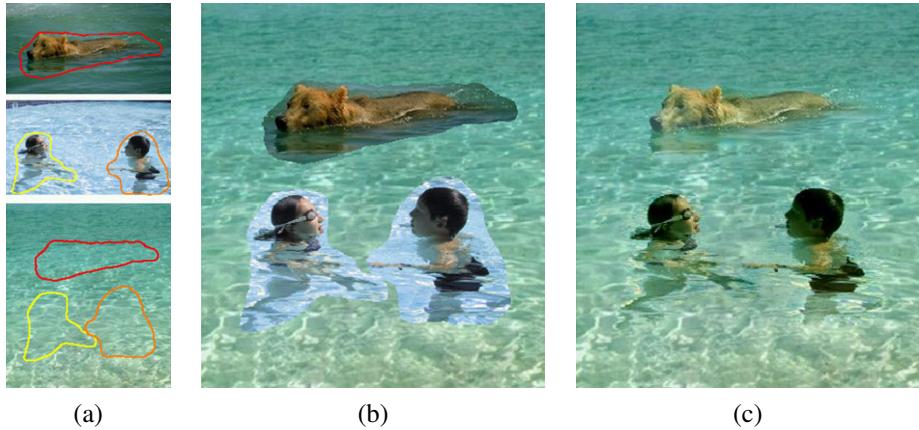


Figure 9.18 Poisson image editing (Pérez, Gangnet, and Blake 2003) © 2003 ACM: (a) The dog and the two children are chosen as source images to be pasted into the destination swimming pool. (b) Simple pasting fails to match the colors at the boundaries, whereas (c) Poisson image blending masks these differences.

a function of the level, i.e., the same width in pixels. In practice, a small number of levels, i.e., as few as two (Brown and Lowe 2007), may be adequate to compensate for differences in exposure. The result of applying this pyramid blending is shown in Figure 9.14h.

Gradient domain blending. An alternative approach to multi-band image blending is to perform the operations in the *gradient domain*. Reconstructing images from their gradient fields has a long history in computer vision (Horn 1986), starting originally with work in brightness constancy (Horn 1974), shape from shading (Horn and Brooks 1989), and photometric stereo (Woodham 1981). More recently, related ideas have been used for reconstructing images from their edges (Elder and Goldberg 2001), removing shadows from images (Weiss 2001), separating reflections from a single image (Levin, Zomet, and Weiss 2004; Levin and Weiss 2007), and *tone mapping* high dynamic range images by reducing the magnitude of image edges (gradients) (Fattal, Lischinski, and Werman 2002).

Pérez, Gangnet, and Blake (2003) show how gradient domain reconstruction can be used to do seamless object insertion in image editing applications (Figure 9.18). Rather than copying pixels, the *gradients* of the new image fragment are copied instead. The actual pixel values for the copied area are then computed by solving a *Poisson equation* that locally matches the gradients while obeying the fixed *Dirichlet* (exact matching) conditions at the seam boundary. Pérez, Gangnet, and Blake (2003) show that this is equivalent to computing an additive *membrane* interpolant of the mismatch between the source and destination images along the boundary.¹⁴ In earlier work, Peleg (1981) also proposed adding a smooth function to enforce consistency along the seam curve.

Agarwala, Dontcheva, Agrawala *et al.* (2004) extended this idea to a multi-source formulation, where it no longer makes sense to talk of a destination image whose exact pixel values must be matched at the seam. Instead, *each* source image contributes its own gradient field and the Poisson equation is solved using *Neumann* boundary conditions, i.e., dropping any

¹⁴ The membrane interpolant is known to have nicer interpolation properties for arbitrary-shaped constraints than frequency-domain interpolants (Nielson 1993).

equations that involve pixels outside the boundary of the image.

Rather than solving the Poisson partial differential equations, Agarwala, Dontcheva, Agrawala *et al.* (2004) directly minimize a *variational problem*,

$$\min_{C(\mathbf{x})} \|\nabla C(\mathbf{x}) - \nabla \tilde{I}_l(\mathbf{x})(\mathbf{x})\|^2. \quad (9.44)$$

The discretized form of this equation is a set of gradient constraint equations

$$C(\mathbf{x} + \hat{\mathbf{i}}) - C(\mathbf{x}) = \tilde{I}_l(\mathbf{x})(\mathbf{x} + \hat{\mathbf{i}}) - \tilde{I}_l(\mathbf{x})(\mathbf{x}) \text{ and} \quad (9.45)$$

$$C(\mathbf{x} + \hat{\mathbf{j}}) - C(\mathbf{x}) = \tilde{I}_l(\mathbf{x})(\mathbf{x} + \hat{\mathbf{j}}) - \tilde{I}_l(\mathbf{x})(\mathbf{x}), \quad (9.46)$$

where $\hat{\mathbf{i}} = (1, 0)$ and $\hat{\mathbf{j}} = (0, 1)$ are unit vectors in the x and y directions.¹⁵ They then solve the associated sparse least squares problem. Since this system of equations is only defined up to an additive constraint, Agarwala, Dontcheva, Agrawala *et al.* (2004) ask the user to select the value of one pixel. In practice, a better choice might be to weakly bias the solution towards reproducing the original color values.

In order to accelerate the solution of this sparse linear system, Fattal, Lischinski, and Werman (2002) use multigrid, whereas Agarwala, Dontcheva, Agrawala *et al.* (2004) use hierarchical basis preconditioned conjugate gradient descent (Szeliski 1990b, 2006b) (Appendix A.5). In subsequent work, Agarwala (2007) shows how using a quadtree representation for the solution can further accelerate the computation with minimal loss in accuracy, while Szeliski, Uyttendaele, and Steedly (2008) show how representing the per-image offset fields using even coarser splines is even faster. This latter work also argues that blending in the log domain, i.e., using multiplicative rather than additive offsets, is preferable, as it more closely matches texture contrasts across seam boundaries. The resulting seam blending works very well in practice (Figure 9.14h), although care must be taken when copying large gradient values near seams so that a “double edge” is not introduced.

Copying gradients directly from the source images after seam placement is just one approach to gradient domain blending. The paper by Levin, Zomet, Peleg *et al.* (2004) examines several different variants of this approach, which they call *Gradient-domain Image STitching* (GIST). The techniques they examine include feathering (blending) the gradients from the source images, as well as using an L1 norm in performing the reconstruction of the image from the gradient field, rather than using an L2 norm as in Equation (9.44). Their preferred technique is the L1 optimization of a feathered (blended) cost function on the original image gradients (which they call GIST1- l_1). Since L1 optimization using linear programming can be slow, they develop a faster iterative median-based algorithm in a multigrid framework. Visual comparisons between their preferred approach and what they call *optimal seam on the gradients* (which is equivalent to the approach of Agarwala, Dontcheva, Agrawala *et al.* (2004)) show similar results, while significantly improving on pyramid blending and feathering algorithms.

Exposure compensation. Pyramid and gradient domain blending can do a good job of compensating for moderate amounts of exposure differences between images. However, when the exposure differences become large, alternative approaches may be necessary.

¹⁵ At seam locations, the right hand side is replaced by the average of the gradients in the two source images.

Uyttendaele, Eden, and Szeliski (2001) iteratively estimate a local correction between each source image and a blended composite. First, a block-based quadratic transfer function is fit between each source image and an initial feathered composite. Next, transfer functions are averaged with their neighbors to get a smoother mapping and per-pixel transfer functions are computed by *splining* (interpolating) between neighboring block values. Once each source image has been smoothly adjusted, a new feathered composite is computed and the process is repeated (typically three times). The results shown by Uyttendaele, Eden, and Szeliski (2001) demonstrate that this does a better job of exposure compensation than simple feathering and can handle local variations in exposure due to effects such as lens vignetting.

Ultimately, however, the most principled way to deal with exposure differences is to stitch images in the radiance domain, i.e., to convert each image into a radiance image using its exposure value and then create a stitched, high dynamic range image, as discussed in Section 10.2 (Eden, Uyttendaele, and Szeliski 2006).

9.4 Additional reading

The literature on image stitching dates back to work in the photogrammetry community in the 1970s (Milgram 1975, 1977; Slama 1980). In computer vision, papers started appearing in the early 1980s (Peleg 1981), while the development of fully automated techniques came about a decade later (Mann and Picard 1994; Chen 1995; Szeliski 1996; Szeliski and Shum 1997; Sawhney and Kumar 1999; Shum and Szeliski 2000). Those techniques used direct pixel-based alignment but feature-based approaches are now the norm (Zoghami, Faugeras, and Deriche 1997; Capel and Zisserman 1998; Cham and Cipolla 1998; Badra, Qumsieh, and Dudek 1998; McLauchlan and Jaenicke 2002; Brown and Lowe 2007). A collection of some of these papers can be found in the book by Benosman and Kang (2001). Szeliski (2006a) provides a comprehensive survey of image stitching, on which the material in this chapter is based.

High-quality techniques for optimal seam finding and blending are another important component of image stitching systems. Important developments in this field include work by Milgram (1977), Burt and Adelson (1983b), Davis (1998), Uyttendaele, Eden, and Szeliski (2001), Pérez, Gangnet, and Blake (2003), Levin, Zomet, Peleg *et al.* (2004), Agarwala, Dontcheva, Agrawala *et al.* (2004), Eden, Uyttendaele, and Szeliski (2006), and Kopf, Uyttendaele, Deussen *et al.* (2007).

In addition to the merging of multiple overlapping photographs taken for aerial or terrestrial panoramic image creation, stitching techniques can be used for automated whiteboard scanning (He and Zhang 2005; Zhang and He 2007), scanning with a mouse (Nakao, Kashitani, and Kaneyoshi 1998), and retinal image mosaics (Can, Stewart, Roysam *et al.* 2002). They can also be applied to video sequences (Teodosio and Bender 1993; Irani, Hsu, and Anandan 1995; Kumar, Anandan, Irani *et al.* 1995; Sawhney and Ayer 1996; Massey and Bender 1996; Irani and Anandan 1998; Sawhney, Arpa, Kumar *et al.* 2002; Agarwala, Zheng, Pal *et al.* 2005; Rav-Acha, Pritch, Lischinski *et al.* 2005; Steedly, Pal, and Szeliski 2005; Baudisch, Tan, Steedly *et al.* 2006) and can even be used for video compression (Lee, ge Chen, lung Bruce Lin *et al.* 1997).

9.5 Exercises

Ex 9.1: Direct pixel-based alignment Take a pair of images, compute a coarse-to-fine affine alignment (Exercise 8.2) and then blend them using either averaging (Exercise 6.2) or a Laplacian pyramid (Exercise 3.20). Extend your motion model from affine to perspective (homography) to better deal with rotational mosaics and planar surfaces seen under arbitrary motion.

Ex 9.2: Featured-based stitching Extend your feature-based alignment technique from Exercise 6.2 to use a full perspective model and then blend the resulting mosaic using either averaging or more sophisticated distance-based feathering (Exercise 9.9).

Ex 9.3: Cylindrical strip panoramas To generate cylindrical or spherical panoramas from a horizontally panning (rotating) camera, it is best to use a tripod. Set your camera up to take a series of 50% overlapped photos and then use the following steps to create your panorama:

1. Estimate the amount of radial distortion by taking some pictures with lots of long straight lines near the edges of the image and then using the plumb-line method from Exercise 6.10.
2. Compute the focal length either by using a ruler and paper, as in Figure 6.7 (Debevec, Wenger, Tchou *et al.* 2002) or by rotating your camera on the tripod, overlapping the images by exactly 0% and counting the number of images it takes to make a 360° panorama.
3. Convert each of your images to cylindrical coordinates using (9.12–9.16).
4. Line up the images with a translational motion model using either a direct pixel-based technique, such as coarse-to-fine incremental or an FFT, or a feature-based technique.
5. (Optional) If doing a complete 360° panorama, align the first and last images. Compute the amount of accumulated vertical mis-registration and re-distribute this among the images.
6. Blend the resulting images using feathering or some other technique.

Ex 9.4: Coarse alignment Use FFT or phase correlation (Section 8.1.2) to estimate the initial alignment between successive images. How well does this work? Over what range of overlaps? If it does not work, does aligning sub-sections (e.g., quarters) do better?

Ex 9.5: Automated mosaicing Use feature-based alignment with four-point RANSAC for homographies (Section 6.1.3, Equations (6.19–6.23)) or three-point RANSAC for rotational motions (Brown, Hartley, and Nistér 2007) to match up all pairs of overlapping images.

Merge these pairwise estimates together by finding a spanning tree of pairwise relations. Visualize the resulting global alignment, e.g., by displaying a blend of each image with all other images that overlap it.

For greater robustness, try multiple spanning trees (perhaps randomly sampled based on the confidence in pairwise alignments) to see if you can recover from bad pairwise matches (Zach, Klopschitz, and Pollefeys 2010). As a measure of fitness, count how many pairwise estimates are consistent with the global alignment.

Ex 9.6: Global optimization Use the initialization from the previous algorithm to perform a full bundle adjustment over all of the camera rotations and focal lengths, as described in Section 7.4 and by Shum and Szeliski (2000). Optionally, estimate radial distortion parameters as well or support fisheye lenses (Section 2.1.6).

As in the previous exercise, visualize the quality of your registration by creating composites of each input image with its neighbors, optionally blinking between the original image and the composite to better see mis-alignment artifacts.

Ex 9.7: De-ghosting Use the results of the previous bundle adjustment to predict the location of each feature in a consensus geometry. Use the difference between the predicted and actual feature locations to correct for small mis-registrations, as described in Section 9.2.2 (Shum and Szeliski 2000).

Ex 9.8: Compositing surface Choose a compositing surface (Section 9.3.1), e.g., a single reference image extended to a larger plane, a sphere represented using cylindrical or spherical coordinates, a stereographic “little planet” projection, or a cube map.

Project all of your images onto this surface and blend them with equal weighting, for now (just to see where the original image seams are).

Ex 9.9: Feathering and blending Compute a feather (distance) map for each warped source image and use these maps to blend the warped images.

Alternatively, use Laplacian pyramid blending (Exercise 3.20) or gradient domain blending.

Ex 9.10: Photomontage and object removal Implement a “PhotoMontage” system in which users can indicate desired or unwanted regions in pre-registered images using strokes or other primitives (such as bounding boxes).

(Optional) Devise an automatic moving objects remover (or “keeper”) by analyzing which inconsistent regions are more or less typical given some consensus (e.g., median filtering) of the aligned images. Figure 9.17 shows an example where the moving object was kept. Try to make this work for sequences with large amounts of overlaps and consider averaging the images to make the moving object look more ghosted.

Chapter 10

Computational photography

| | |
|---|-----|
| 10.1 Photometric calibration | 412 |
| 10.1.1 Radiometric response function | 412 |
| 10.1.2 Noise level estimation | 415 |
| 10.1.3 Vignetting | 416 |
| 10.1.4 Optical blur (spatial response) estimation | 416 |
| 10.2 High dynamic range imaging | 419 |
| 10.2.1 Tone mapping | 427 |
| 10.2.2 <i>Application:</i> Flash photography | 434 |
| 10.3 Super-resolution and blur removal | 436 |
| 10.3.1 Color image demosaicing | 440 |
| 10.3.2 <i>Application:</i> Colorization | 442 |
| 10.4 Image matting and compositing | 443 |
| 10.4.1 Blue screen matting | 445 |
| 10.4.2 Natural image matting | 446 |
| 10.4.3 Optimization-based matting | 450 |
| 10.4.4 Smoke, shadow, and flash matting | 452 |
| 10.4.5 Video matting | 454 |
| 10.5 Texture analysis and synthesis | 455 |
| 10.5.1 <i>Application:</i> Hole filling and inpainting | 457 |
| 10.5.2 <i>Application:</i> Non-photorealistic rendering | 458 |
| 10.6 Additional reading | 460 |
| 10.7 Exercises | 461 |

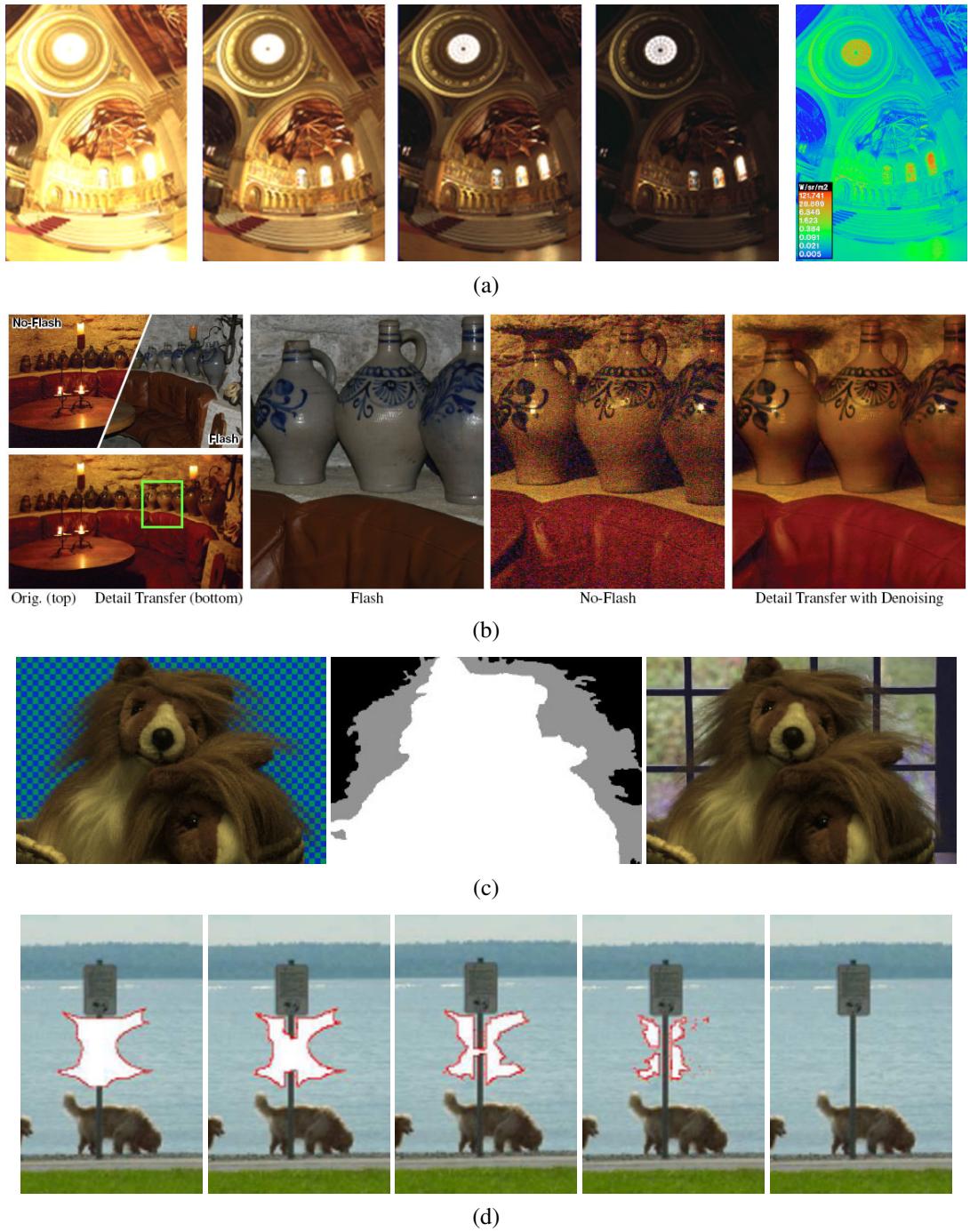


Figure 10.1 Computational photography: (a) merging multiple exposures to create high dynamic range images (Debevec and Malik 1997) © 1997 ACM; (b) merging flash and non-flash photographs; (Petschnigg, Agrawala, Hoppe *et al.* 2004) © 2004 ACM; (c) image matting and compositing; (Chuang, Curless, Salesin *et al.* 2001) © 2001 IEEE; (d) hole filling with inpainting (Criminisi, Pérez, and Toyama 2004) © 2004 IEEE.

Stitching multiple images into wide field of view panoramas, which we covered in Chapter 9, allows us to create photographs that could not be captured with a regular camera. This is just one instance of *computational photography*, where image analysis and processing algorithms are applied to one or more photographs to create images that go beyond the capabilities of traditional imaging systems. Some of these techniques are now being incorporated directly into digital still cameras. For example, some of the newer digital still cameras have sweep panorama modes and take multiple shots in low-light conditions to reduce image noise.

In this chapter, we cover a number of additional computational photography algorithms. We begin with a review of photometric image calibration (Section 10.1), i.e., the measurement of camera and lens responses, which is a prerequisite for many of the algorithms we describe later. We then discuss *high dynamic range imaging* (Section 10.2), which captures the full range of brightness in a scene through the use of multiple exposures (Figure 10.1a). We also discuss *tone mapping operators*, which map rich images back into regular display devices, such as screens and printers, as well as algorithms that merge flash and regular images to obtain better exposures (Figure 10.1b).

Next, we discuss how the resolution of images can be improved either by merging multiple photographs together or using sophisticated image priors (Section 10.3). This includes algorithms for extracting full-color images from the patterned Bayer mosaics present in most cameras.

In Section 10.4, we discuss algorithms for cutting pieces of images from one photograph and pasting them into others (Figure 10.1c). In Section 10.5, we describe how to generate novel textures from real-world samples for applications such as filling holes in images (Figure 10.1d). We close with a brief overview of *non-photorealistic rendering* (Section 10.5.2), which can turn regular photographs into artistic renderings that resemble traditional drawings and paintings.

One topic that we do not cover extensively in this book is novel computational sensors, optics, and cameras. A nice survey can be found in an article by Nayar (2006), a recently published book by Raskar and Tumblin (2010), and more recent research papers (Levin, Fergus, Durand *et al.* 2007). Some related discussion can also be found in Sections 10.2 and 13.3.

A good general-audience introduction to computational photography can be found in the article by Hayes (2008) as well as survey papers by Nayar (2006), Cohen and Szeliski (2006), Levoy (2006), and Debevec (2006).¹ Raskar and Tumblin (2010) give extensive coverage of topics in this area, with particular emphasis on computational cameras and sensors. The sub-field of high dynamic range imaging has its own book discussing research in this area (Reinhard, Ward, Pattanaik *et al.* 2005), as well as a wonderful book aimed more at professional photographers (Freeman 2008).² A good survey of image matting is provided by Wang and Cohen (2007a).

There are also several courses on computational photography where the instructors have provided extensive on-line materials, e.g., Frédo Durand's Computation Photography course at MIT,³ Alyosha Efros' class at Carnegie Mellon,⁴ Marc Levoy's class at Stanford,⁵ and a

¹ See also the two special issue journals edited by Bimber (2006) and Durand and Szeliski (2007).

² Gubins and Gubins (2009) discuss related photographic techniques.

³ MIT 6.815/6.865, <http://stellar.mit.edu/S/course/6/sp08/6.815/materials.html>.

⁴ CMU 15-463, <http://graphics.cs.cmu.edu/courses/15-463/>.

⁵ Stanford CS 448A, <http://graphics.stanford.edu/courses/cs448a-10/>.

series of SIGGRAPH courses on Computational Photography.⁶

10.1 Photometric calibration

Before we can successfully merge multiple photographs, we need to characterize the functions that map incoming irradiance into pixel values and also the amounts of noise present in each image. In this section, we examine three components of the imaging pipeline (Figure 10.2) that affect this mapping.

The first is the *radiometric response function* (Mitsunaga and Nayar 1999), which maps photons arriving at the lens into digital values stored in the image file (Section 10.1.1). The second is *vignetting*, which darkens pixel values near the periphery of images, especially at large apertures (Section 10.1.3). The third is the *point spread function*, which characterizes the blur induced by the lens, anti-aliasing filters, and finite sensor areas (Section 10.1.4).⁷ The material in this section builds on the image formation processes described in Sections 2.2.3 and 2.3.3, so if it has been a while since you looked at those sections, please go back and review them.

10.1.1 Radiometric response function

As we can see in Figure 10.2, a number of factors affect how the intensity of light arriving at the lens ends up being mapped into stored digital values. Let us ignore for now any non-uniform attenuation that may occur inside the lens, which we cover in Section 10.1.3.

The first factors to affect this mapping are the aperture and shutter speed (Section 2.3), which can be modeled as global multipliers on the incoming light, most conveniently measured in *exposure values* (\log_2 brightness ratios). Next, the analog to digital (A/D) converter on the sensing chip applies an electronic gain, usually controlled by the ISO setting on your camera. While in theory this gain is linear, as with any electronics non-linearities may be present (either unintentionally or by design). Ignoring, for now, photon noise, on-chip noise, amplifier noise, and quantization noise, which we discuss shortly, you can often assume that the mapping between incoming light and the values stored in a RAW camera file (if your camera supports this) is roughly linear.

If images are being stored in the more common JPEG format, the camera's digital signal processor (DSP) next performs Bayer pattern demosaicing (Sections 2.3.2 and 10.3.1), which is a mostly linear (but often non-stationary) process. Some sharpening is also often applied at this stage. Next, the color values are multiplied by different constants (or sometimes a 3×3 color twist matrix) to perform color balancing, i.e., to move the white point closer to pure white. Finally, a standard gamma is applied to the intensities in each color channel and the colors are converted into YCbCr format before being transformed by a DCT, quantized, and then compressed into the JPEG format (Section 2.3.3). Figure 10.2 shows all of these steps in pictorial form.

Given the complexity of all of this processing, it is difficult to model the camera response function (Figure 10.3a), i.e., the mapping between incoming irradiance and digital RGB val-

⁶ <http://web.media.mit.edu/~raskar/photo/>.

⁷ Additional photometric camera and lens effects include sensor glare, blooming, and chromatic aberration, which can also be thought of as a spectrally varying form of geometric aberration (Section 2.2.3).

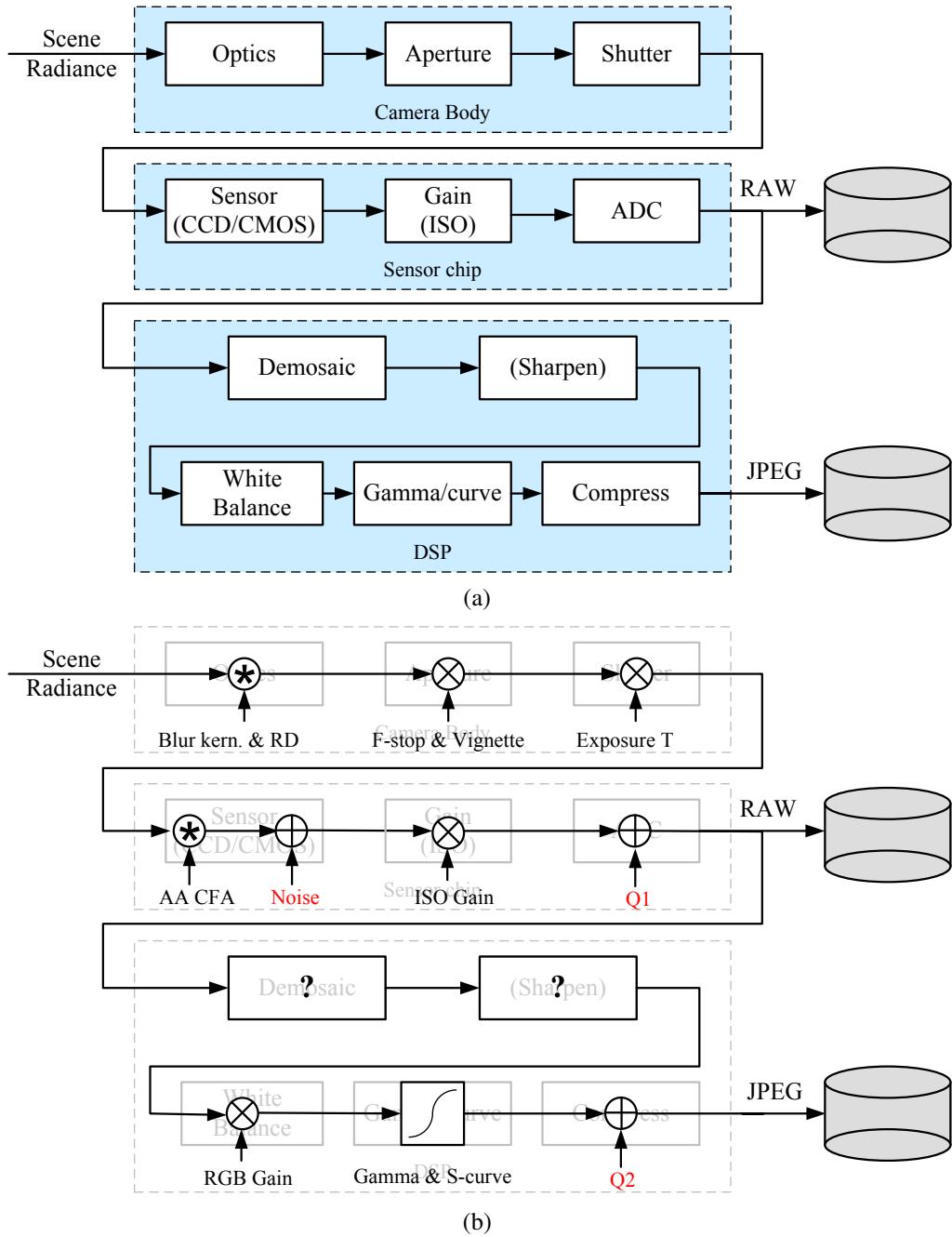


Figure 10.2 Image sensing pipeline: (a) block diagram showing the various sources of noise as well as the typical digital post-processing steps; (b) equivalent signal transforms, including convolution, gain, and noise injection. The abbreviations are: RD = radial distortion, AA = anti-aliasing filter, CFA = color filter array, Q1 and Q2 = quantization noise.

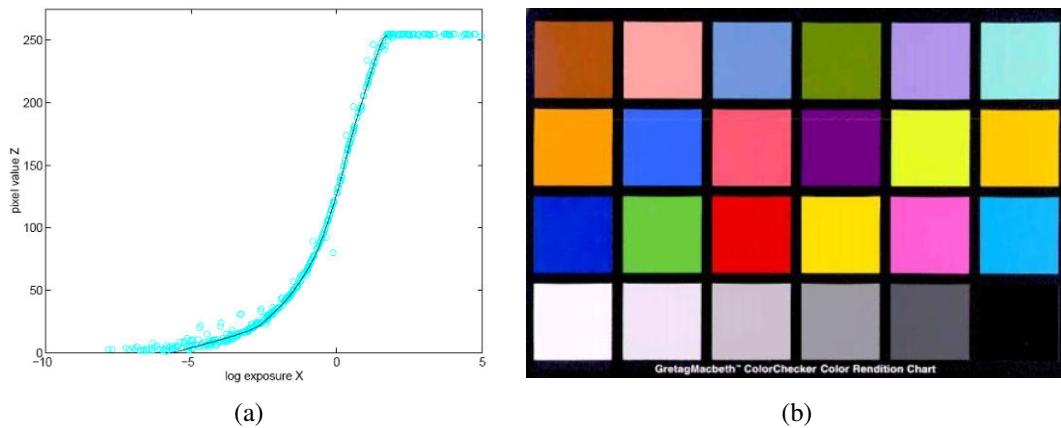


Figure 10.3 Radiometric response calibration: (a) typical camera response function, showing the mapping between incoming log irradiance (exposure) and output eight-bit pixel values, for one color channel (Debevec and Malik 1997) © 1997 ACM; (b) color checker chart.

ues, from first principles. A more practical approach is to calibrate the camera by measuring correspondences between incoming light and final values.

The most accurate, but most expensive, approach is to use an *integrating sphere*, which is a large (typically 1m diameter) sphere carefully painted on the inside with white matte paint. An accurately calibrated light at the top controls the amount of radiance inside the sphere (which is constant everywhere because of the sphere's radiometry) and a small opening at the side allows for a camera/lens combination to be mounted. By slowly varying the current going into the light, an accurate correspondence can be established between incoming radiance and measured pixel values. The vignetting and noise characteristics of the camera can also be simultaneously determined.

A more practical alternative is to use a calibration chart (Figure 10.3b) such as the Macbeth or Munsell ColorChecker Chart.⁸ The biggest problem with this approach is to ensure uniform lighting. One approach is to use a large dark room with a high-quality light source far away from (and perpendicular to) the chart. Another is to place the chart outdoors away from any shadows. (The results will differ under these two conditions, because the color of the illuminant will be different).

The easiest approach is probably to take multiple exposures of the same scene while the camera is on a tripod and to recover the response function by simultaneously estimating the incoming irradiance at each pixel and the response curve (Mann and Picard 1995;Debevec and Malik 1997; Mitsunaga and Nayar 1999). This approach is discussed in more detail in Section 10.2 on high dynamic range imaging.

If all else fails, i.e., you just have one or more unrelated photos, you can use an International Color Consortium (ICC) profile for the camera ([Fairchild 2005](#)).⁹ Even more simply, you can just assume that the response is linear if they are RAW files and that the images have a $\gamma = 2.2$ non-linearity (plus clipping) applied to each RGB channel if they are JPEG images.

⁸ <http://www.xrite.com>.

⁹ See also the ICC *Information on Profiles*, http://www.color.org/info_profiles2.xalter.

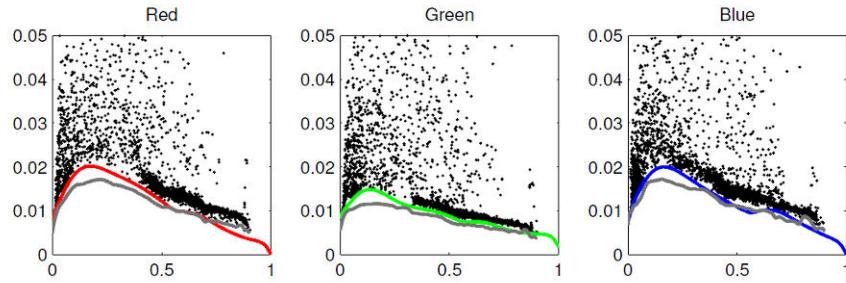


Figure 10.4 Noise level function estimates obtained from a single color photograph (Liu, Szeliski, Kang *et al.* 2008) © 2008 IEEE. The colored curves are the estimated NLF fit as the probabilistic lower envelope of the measured deviations between the noisy piecewise-smooth images. The ground truth NLFs obtained by averaging 29 images are shown in gray.

10.1.2 Noise level estimation

In addition to knowing the camera response function, it is also often important to know the amount of noise being injected under a particular camera setting (e.g., ISO/gain level). The simplest characterization of noise is a single standard deviation, usually measured in gray levels, independent of pixel value. A more accurate model can be obtained by estimating the noise level as a function of pixel value (Figure 10.4), which is known as the *noise level function* (Liu, Szeliski, Kang *et al.* 2008).

As with the camera response function, the simplest way to estimate these quantities is in the lab, using either an integrating sphere or a calibration chart. The noise can be estimated either at each pixel independently, by taking repeated exposures and computing the temporal variance in the measurements (Healey and Kondepudy 1994), or over regions, by assuming that pixel values should all be the same within some region (e.g., inside a color checker square) and computing a spatial variance.

This approach can be generalized to photos where there are regions of constant or slowly varying intensity (Liu, Szeliski, Kang *et al.* 2008). First, segment the image into such regions and fit a constant or linear function inside each region. Next, measure the (spatial) standard deviation of the differences between the noisy input pixels and the smooth fitted function away from large gradients and region boundaries. Plot these as a function of output level for each color channel, as shown in Figure 10.4. Finally, fit a lower envelope to this distribution in order to ignore pixels or deviations that are outliers. A fully Bayesian approach to this problem that models the statistical distribution of each quantity is presented by (Liu, Szeliski, Kang *et al.* 2008). A simpler approach, which should produce useful results in most cases, is to fit a low-dimensional function (e.g., positive valued B-spline) to the lower envelope (see Exercise 10.2).

In more recent work, Matsushita and Lin (2007) present a technique for simultaneously estimating a camera’s response and noise level functions based on skew (asymmetries) in level-dependent noise distributions. Their paper also contains extensive references to previous work in these areas.



Figure 10.5 Single image vignetting correction (Zheng, Yu, Kang *et al.* 2008) © 2008 IEEE: (a) original image with strong visible vignetting; (b) vignetting compensation as described by Zheng, Zhou, Georgescu *et al.* (2006); (c–d) vignetting compensation as described by Zheng, Yu, Kang *et al.* (2008).

10.1.3 Vignetting

A common problem with using wide-angle and wide-aperture lenses is that the image tends to darken in the corners (Figure 10.5a). This problem is generally known as *vignetting* and comes in several different forms, including natural, optical, and mechanical vignetting (Section 2.2.3) (Ray 2002). As with radiometric response function calibration, the most accurate way to calibrate vignetting is to use an integrating sphere or a picture of a uniformly colored and illuminated blank wall.

An alternative approach is to stitch a panoramic scene and to assume that the true radiance at each pixel comes from the central portion of each input image. This is easier to do if the radiometric response function is already known (e.g., by shooting in RAW mode) and if the exposure is kept constant. If the response function, image exposures, and vignetting function are unknown, they can still be recovered by optimizing a large least squares fitting problem (Litvinov and Schechner 2005; Goldman 2011). Figure 10.6 shows an example of simultaneously estimating the vignetting, exposure, and radiometric response function from a set of overlapping photographs (Goldman 2011). Note that unless vignetting is modeled and compensated, regular gradient-domain image blending (Section 9.3.4) will not create an attractive image.

If only a single image is available, vignetting can be estimated by looking for slow consistent intensity variations in the radial direction. The original algorithm proposed by Zheng, Lin, and Kang (2006) first pre-segmented the image into smoothly varying regions and then performed an analysis inside each region. Instead of pre-segmenting the image, Zheng, Yu, Kang *et al.* (2008) compute the radial gradients at all the pixels and use the asymmetry in this distribution (since gradients away from the center are, on average, slightly negative) to estimate the vignetting. Figure 10.5 shows the results of applying each of these algorithms to an image with a large amount of vignetting. Exercise 10.3 has you implement some of the above techniques.

10.1.4 Optical blur (spatial response) estimation

One final characteristic of imaging systems that you should calibrate is the spatial response function, which encodes the optical blur that gets convolved with the incoming image to produce the point-sampled image. The shape of the convolution kernel, which is also known as *point spread function (PSF)* or *optical transfer function*, depends on several factors, including lens blur and radial distortion (Section 2.2.3), anti-aliasing filters in front of the sensor, and



Figure 10.6 Simultaneous estimation of vignetting, exposure, and radiometric response (Goldman 2011) © 2011 IEEE: (a) original average of the input images; (b) after compensating for vignetting; (c) using gradient domain blending only (note the remaining mottled look); (d) after both vignetting compensation and blending.

the shape and extent of each active pixel area (Section 2.3) (Figure 10.2). A good estimate of this function is required for applications such as multi-image super-resolution and de-blurring (Section 10.3).

In theory, one could estimate the PSF by simply observing an infinitely small point light source everywhere in the image. Creating an array of samples by drilling through a dark plate and backlighting with a very bright light source is difficult in practice.

A more practical approach is to observe an image composed of long straight lines or bars, since these can be fitted to arbitrary precision. Because the location of a horizontal or vertical edge can be *aliased* during acquisition, slightly slanted edges are preferred. The profile and locations of such edges can be estimated to sub-pixel precision, which makes it possible to estimate the PSF at sub-pixel resolutions (Reichenbach, Park, and Narayanswamy 1991; Burns and Williams 1999; Williams and Burns 2001; Goesele, Fuchs, and Seidel 2003). The thesis by Murphy (2005) contains a nice survey of all aspects of camera calibration, including the spatial frequency response (SFR), spatial uniformity, tone reproduction, color reproduction, noise, dynamic range, color channel registration, and depth of field. It also includes a description of a slant-edge calibration algorithm called `sfrm2`.

The slant-edge technique can be used to recover a 1D projection of the 2D PSF, e.g., slightly vertical edges are used to recover the horizontal *line spread function* (LSF) (Williams 1999). The LSF is then often converted into the Fourier domain and its magnitude plotted as a one-dimensional *modulation transfer function* (MTF), which indicates which image frequencies are lost (blurred) and aliased during the acquisition process (Section 2.3.1). For most computational photography applications, it is preferable to directly estimate the full 2D PSF, since it can be hard to recover from its projections (Williams 1999).

Figure 10.7 shows a pattern containing edges at all orientations, which can be used to directly recover a two-dimensional PSF. First, corners in the pattern are located by extracting edges in the sensed image, linking them, and finding the intersections of the circular arcs. Next, the ideal pattern, whose analytic form is known, is warped (using a homography) to fit the central portion of the input image and its intensities are adjusted to fit the ones in

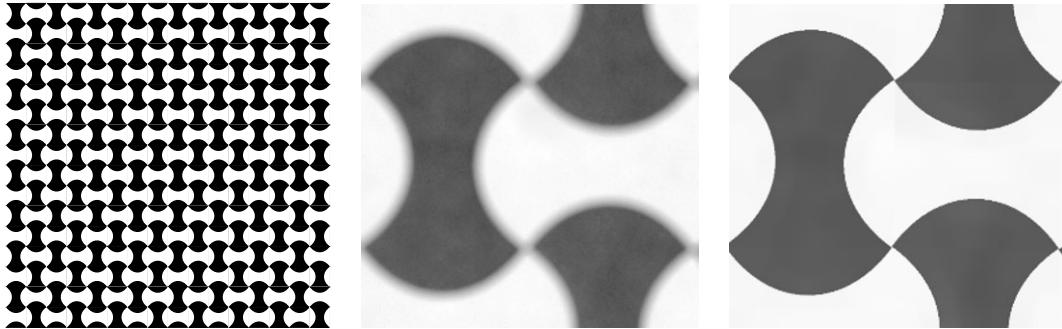


Figure 10.7 Calibration pattern with edges equally distributed at all orientations that can be used for PSF and radial distortion estimation (Joshi, Szeliski, and Kriegman 2008) © 2008 IEEE. A portion of an actual sensed image is shown in the middle and a close-up of the ideal pattern is on the right.

the sensed image. If desired, the pattern can be rendered at a higher resolution than the input image, which enables the estimation of the PSF to sub-pixel resolution (Figure 10.8a). Finally a large linear least squares system is solved to recover the unknown PSF kernel K ,

$$K = \arg \min_K \|B - D(I * K)\|^2, \quad (10.1)$$

where B is the sensed (blurred) image, I is the predicted (sharp) image, and D is an optional downsampling operator that matches the resolution of the ideal and sensed images (Joshi, Szeliski, and Kriegman 2008). In terms of the notation (3.75) introduced in Section 3.4.3, this could also be written as

$$b = \arg \min_b \|o - D(s * b)\|^2, \quad (10.2)$$

where o is the observed image, s is the sharp image, and b is the blur kernel.

If the process of estimating the PSF is done locally in overlapping patches of the image, it can also be used to estimate the radial distortion and chromatic aberration induced by the lens (Figure 10.8b). Because the homography mapping the ideal target to the sensed image is estimated in the central (undistorted) part of the image, any (per-channel) shifts induced by the optics manifest themselves as a displacement in the PSF centers.¹⁰ Compensating for these shifts eliminates both the achromatic radial distortion and the inter-channel shifts that result in visible chromatic aberration. The color-dependent blurring caused by chromatic aberration (Figure 2.21) can also be removed using the de-blurring techniques discussed in Section 10.3. Figure 10.8b shows how the radial distortion and chromatic aberration manifest themselves as elongated and displaced PSFs, along with the result of removing these effects in a region of the calibration target.

The local 2D PSF estimation technique can also be used to estimate vignetting. Figure 10.8c shows how the mechanical vignetting manifests itself as clipping of the PSF in the corners of the image. In order for the overall dimming associated with vignetting to be properly captured, the modified intensities of the ideal pattern need to be extrapolated from the center, which is best done with a uniformly illuminated target.

¹⁰ This process confounds the distinction between geometric and photometric calibration. In principle, any geometric distortion could be modeled by spatially varying displaced PSFs. In practice, it is easier to fold any large shifts into the geometric correction component.

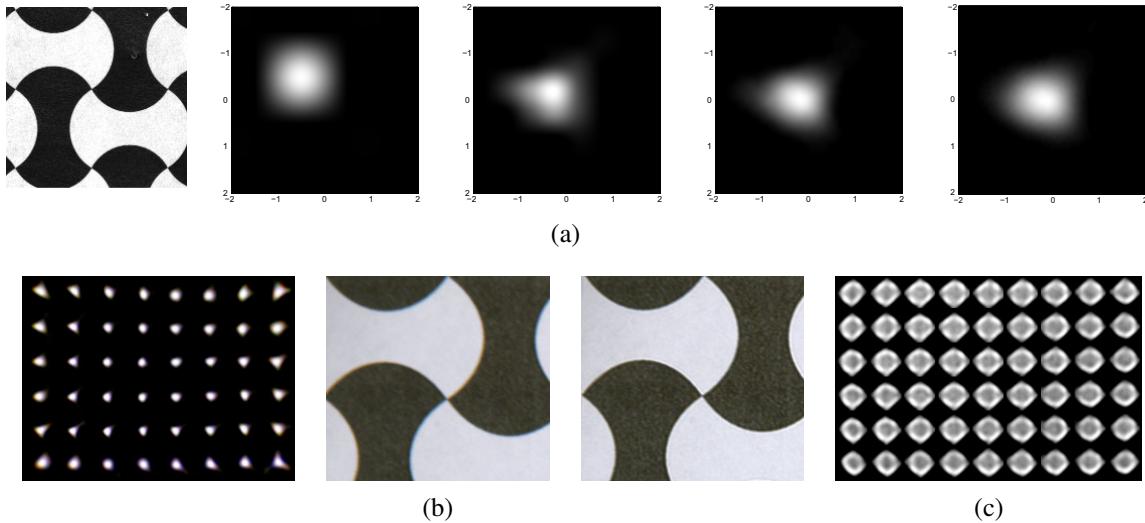


Figure 10.8 Point spread function estimation using a calibration target (Joshi, Szeliski, and Kriegman 2008) © 2008 IEEE. (a) Sub-pixel PSFs at successively higher resolutions (note the interaction between the square sensing area and the circular lens blur). (b) The radial distortion and chromatic aberration can also be estimated and removed. (c) PSF for a mis-focused (blurred) lens showing some diffraction and vignetting effects in the corners.

When working with RAW Bayer-pattern images, the correct way to estimate the PSF is to only evaluate the least squares terms in (10.1) at sensed pixel values, while interpolating the ideal image to all values. For JPEG images, you should linearize your intensities first, e.g., remove the gamma and any other non-linearities in your estimated radiometric response function.

What if you have an image that was taken with an uncalibrated camera? Can you still recover the PSF and use it to correct the image? In fact, with a slight modification, the previous algorithms still work.

Instead of assuming a known calibration image, you can detect strong elongated edges and fit ideal step edges in such regions (Figure 10.9b), resulting in the sharp image shown in Figure 10.9d. For every pixel that is surrounded by a complete set of valid estimated neighbors (green pixels in Figure 10.9c), apply the least squares formula (10.1) to estimate the kernel K . The resulting locally estimated PSFs can be used to correct for chromatic aberration (since the relative displacements between per-channel PSFs can be computed), as shown by Joshi, Szeliski, and Kriegman (2008).

Exercise 10.4 provides some more detailed instructions for implementing and testing edge-based PSF estimation algorithms. An alternative approach, which does not require the explicit detection of edges but uses image statistics (gradient distributions) instead, is presented by Fergus, Singh, Hertzmann *et al.* (2006).

10.2 High dynamic range imaging

As we mentioned earlier in this chapter, registered images taken at different exposures can be used to calibrate the radiometric response function of a camera. More importantly, they can

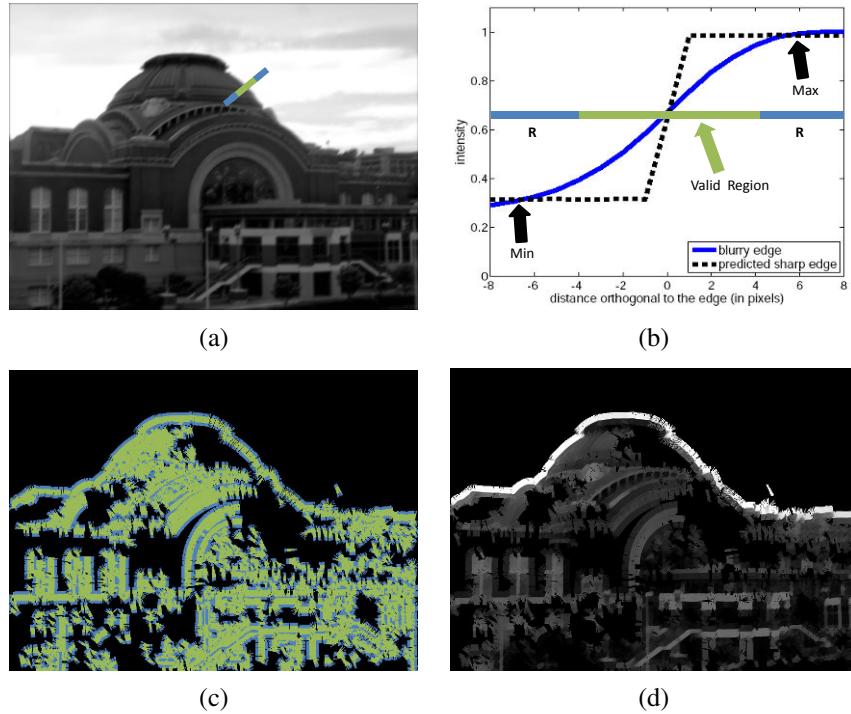


Figure 10.9 Estimating the PSF without using a calibration pattern (Joshi, Szeliski, and Kriegman 2008) © 2008 IEEE: (a) Input image with blue cross-section (profile) location, (b) Profile of sensed and predicted step edges, (c–d) Locations and values of the predicted colors near the edge locations.



Figure 10.10 Sample indoor image where the areas outside the window are overexposed and inside the room are too dark.



Figure 10.11 Relative brightness of different scenes, ranging from 1 inside a dark room lit by a monitor to 2,000,000 looking at the sun. Photos courtesy of Paul Debevec.



Figure 10.12 A bracketed set of shots (using the camera’s automatic exposure bracketing (AEB) mode) and the resulting high dynamic range (HDR) composite.

help you create well-exposed photographs under challenging conditions, such as brightly lit scenes where any single exposure contains saturated (overexposed) and dark (underexposed) regions (Figure 10.10). This problem is quite common, because the natural world contains a range of radiance values that is far greater than can be captured with any photographic sensor or film (Figure 10.11). Taking a set of *bracketed exposures* (exposures taken by a camera in automatic exposure bracketing (AEB) mode to deliberately under- and over-expose the image) gives you the material from which to create a properly exposed photograph, as shown in Figure 10.12 (Reinhard, Ward, Pattanaik *et al.* 2005; Freeman 2008; Gulbins and Gulbins 2009; Hasinoff, Durand, and Freeman 2010).

While it is possible to combine pixels from different exposures directly into a final composite (Burt and Kolczynski 1993; Mertens, Kautz, and Reeth 2007), this approach runs the risk of creating contrast reversals and halos. Instead, the more common approach is to proceed in three stages:

1. Estimate the radiometric response function from the aligned images.
2. Estimate a *radiance map* by selecting or blending pixels from different exposures.
3. Tone map the resulting high dynamic range (HDR) image back into a displayable gamut.

The idea behind estimating the radiometric response function is relatively straightforward (Mann and Picard 1995; Debevec and Malik 1997; Mitsunaga and Nayar 1999; Reinhard, Ward, Pattanaik *et al.* 2005). Suppose you take three sets of images at different exposures (shutter speeds), say at ± 2 exposure values.¹¹ If we were able to determine the irradiance

¹¹ Changing the shutter speed is preferable to changing the aperture, as the latter can modify the vignetting and focus. Using ± 2 “f-stops” (technically, exposure values, or EVs, since f-stops refer to apertures) is usually the right compromise between capturing a good dynamic range and having properly exposed pixels everywhere.

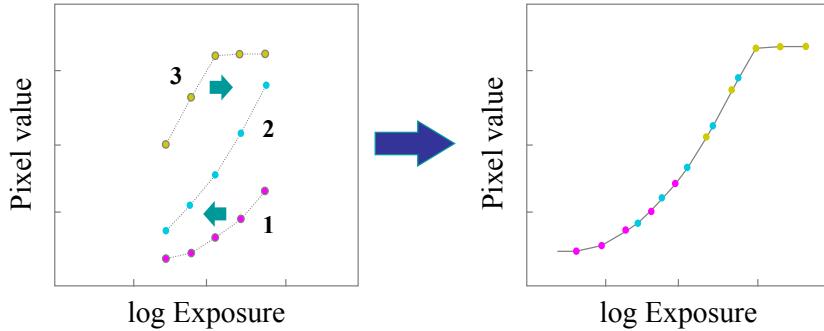


Figure 10.13 Radiometric calibration using multiple exposures (Debevec and Malik 1997). Corresponding pixel values are plotted as functions of log exposures (irradiance). The curves on the left are shifted to account for each pixel’s unknown radiance until they all line up into a single smooth curve.

(exposure) E_i at each pixel (2.101), we could plot it against the measured pixel value z_{ij} for each exposure time t_j , as shown in Figure 10.13.

Unfortunately, we do not know the irradiance values E_i , so these have to be estimated at the same time as the radiometric response function f , which can be written (Debevec and Malik 1997) as

$$z_{ij} = f(E_i t_j), \quad (10.3)$$

where t_j is the exposure time for the j th image. The inverse response curve f^{-1} is given by

$$f^{-1}(z_{ij}) = E_i t_j. \quad (10.4)$$

Taking logarithms of both sides (base 2 is convenient, as we can now measure quantities in EVs), we obtain

$$g(z_{ij}) = \log f^{-1}(z_{ij}) = \log E_i + \log t_j, \quad (10.5)$$

where $g = \log f^{-1}$ (which maps pixel values z_{ij} into log irradiance) is the curve we are estimating (Figure 10.13 turned on its side).

Debevec and Malik (1997) assume that the exposure times t_j are known. (Recall that these can be obtained from a camera’s EXIF tags, but that they actually follow a power of 2 progression $\dots, \frac{1}{128}, \frac{1}{64}, \frac{1}{32}, \frac{1}{16}, \frac{1}{8}, \dots$ instead of the marked $\dots, \frac{1}{125}, \frac{1}{60}, \frac{1}{30}, \frac{1}{15}, \frac{1}{8}, \dots$ values—see Exercise 2.5.) The unknowns are therefore the per-pixel exposures E_i and the response values $g_k = g(k)$, where g can be discretized according to the 256 pixel values commonly observed in eight-bit images. (The response curves are calibrated separately for each color channel.)

In order to make the response curve smooth, Debevec and Malik (1997) add a second-order smoothness constraint

$$\lambda \sum_k g''(k)^2 = \lambda \sum [g(k-1) - 2g(k) + g(k+1)]^2, \quad (10.6)$$

which is similar to the one used in snakes (5.3). Since pixel values are more reliable in the middle of their range (and the g function becomes singular near saturation values), they also

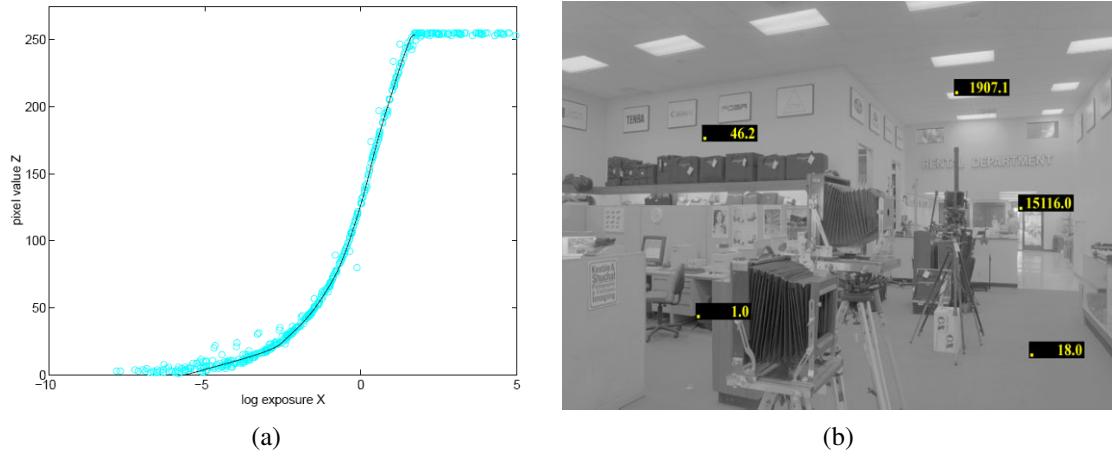


Figure 10.14 Recovered response function and radiance image for a real digital camera (DCS460) (Debevec and Malik 1997) © 1997 ACM.

add a weighting (hat) function $w(k)$ that decays to zero at both ends of the pixel value range,

$$w(z) = \begin{cases} z - z_{\min} & z \leq (z_{\min} + z_{\max})/2 \\ z_{\max} - z & z > (z_{\min} + z_{\max})/2. \end{cases} \quad (10.7)$$

Putting all of these terms together, they obtain a least squares problem in the unknowns $\{g_k\}$ and $\{E_i\}$,

$$E = \sum_i \sum_j w(z_{i,j}) [g(z_{i,j}) - \log E_i - \log t_j]^2 + \lambda \sum_k w(k) g''(k)^2. \quad (10.8)$$

(In order to remove the overall shift ambiguity in the response curve and irradiance values, the middle of the response curve is set to 0.) Debevec and Malik (1997) show how this can be implemented in 21 lines of MATLAB code, which partially accounts for the popularity of their technique.

While Debevec and Malik (1997) assume that the exposure times t_j are known exactly, there is no reason why these additional variables cannot be thrown into the least squares problem, constraining their final estimated values to lie close to their nominal values \hat{t}_j with an extra term $\eta \sum_j (t_j - \hat{t}_j)^2$.

Figure 10.14 shows the recovered radiometric response function for a digital camera along with select (relative) radiance values in the overall radiance map. Figure 10.15 shows the bracketed input images captured on color film and the corresponding radiance map.

While Debevec and Malik (1997) use a general second-order smooth curve g to parameterize their response curve, Mann and Picard (1995) use a three-parameter function

$$f(E) = \alpha + \beta E^\gamma, \quad (10.9)$$

while Mitsunaga and Nayar (1999) use a low-order ($N \leq 10$) polynomial for the inverse response function g . Pal, Szeliski, Uyttendaele *et al.* (2004) derive a Bayesian model that estimates an independent smooth response function for each image, which can better model

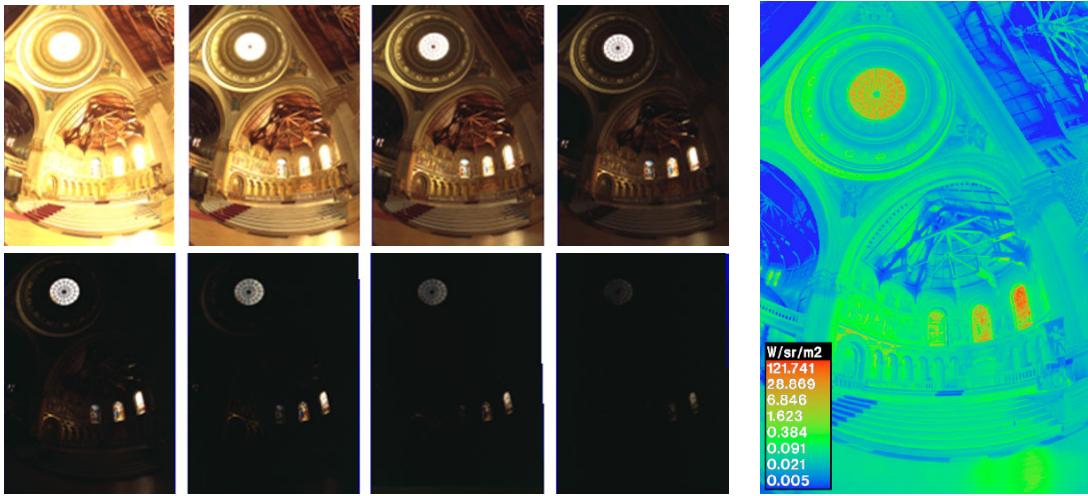


Figure 10.15 Bracketed set of exposures captured with a film camera and the resulting radiance image displayed in pseudocolor (Debevec and Malik 1997) © 1997 ACM.

the more sophisticated (and hence less predictable) automatic contrast and tone adjustment performed in today’s digital cameras.

Once the response function has been estimated, the second step in creating high dynamic range photographs is to merge the input images into a composite *radiance map*. If the response function and images were known exactly, i.e., if they were noise free, you could use any non-saturated pixel value to estimate the corresponding radiance by mapping it through the inverse response curve $E = g(z)$.

Unfortunately, pixels are noisy, especially under low-light conditions when fewer photons arrive at the sensor. To compensate for this, Mann and Picard (1995) use the derivative of the response function as a weight in determining the final radiance estimate, since “flatter” regions of the curve tell us less about the incoming irradiance. Debevec and Malik (1997) use a hat function (10.7) which accentuates mid-tone pixels while avoiding saturated values. Mitsunaga and Nayar (1999) show that in order to maximize the signal-to-noise ratio (SNR), the weighting function must emphasize both higher pixel values and larger gradients in the transfer function, i.e.,

$$w(z) = g(z)/g'(z), \quad (10.10)$$

where the weights w are used to form the final irradiance estimate

$$\log E_i = \frac{\sum_j w(z_{ij})[g(z_{ij}) - \log t_j]}{\sum_j w(z_{ij})}. \quad (10.11)$$

Exercise 10.1 has you implement one of the radiometric response function calibration techniques and then use it to create radiance maps.

Under real-world conditions, casually acquired images may not be perfectly registered and may contain moving objects. Ward (2003) uses a global (parametric) transform to align the input images, while Kang, Uyttendaele, Winder *et al.* (2003) present an algorithm that combines global registration with local motion estimation (optical flow) to accurately align

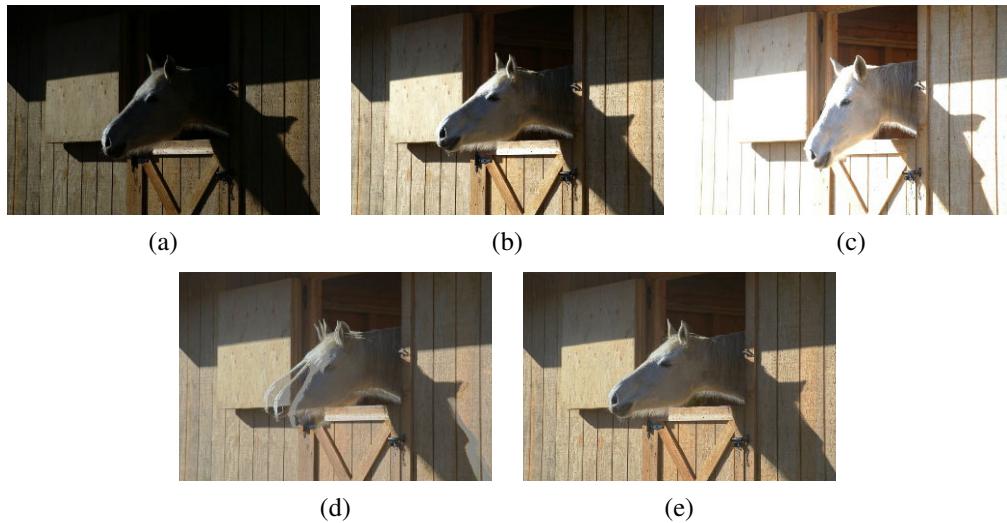


Figure 10.16 Merging multiple exposures to create a high dynamic range composite (Kang, Uyttendaele, Winder *et al.* 2003): (a–c) three different exposures; (d) merging the exposures using classic algorithms (note the ghosting due to the horse’s head movement); (e) merging the exposures with motion compensation.

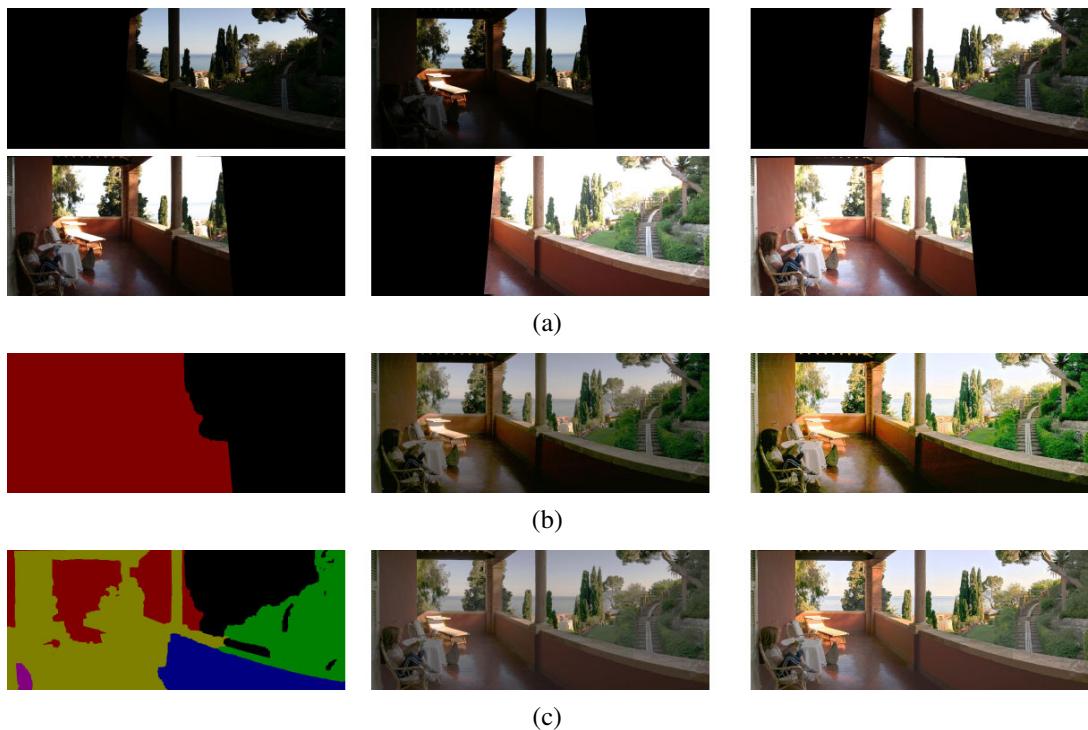


Figure 10.17 HDR merging with large amounts of motion (Eden, Uyttendaele, and Szeliski 2006) © 2006 IEEE: (a) registered bracketed input images; (b) results after the first pass of image selection: reference labels, image, and tone-mapped image; (c) results after the second pass of image selection: final labels, compressed HDR image, and tone-mapped image

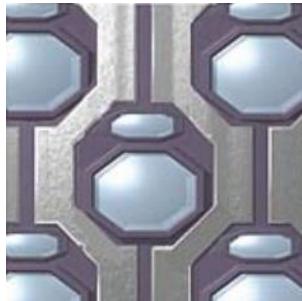


Figure 10.18 Fuji SuperCCD high dynamic range image sensor. The paired large and small active areas provide two different effective exposures.

the images before blending their radiance estimates (Figure 10.16). Since the images may have widely different exposures, care must be taken when estimating the motions, which must themselves be checked for consistency to avoid the creation of ghosts and object fragments.

Even this approach, however, may not work when the camera is simultaneously undergoing large panning motions and exposure changes, which is a common occurrence in casually acquired panoramas. Under such conditions, different parts of the image may be seen at one or more exposures. Devising a method to blend all of these different sources while avoiding sharp transitions and dealing with scene motion is a challenging problem. One approach is to first find a consensus mosaic and to then selectively compute radiances in under- and over-exposed regions (Eden, Uyttendaele, and Szeliski 2006), as shown in Figure 10.17.

Recently, some cameras, such as the Sony α550 and Pentax K-7, have started integrating multiple exposure merging and tone mapping directly into the camera body. In the future, the need to compute high dynamic range images from multiple exposures may be eliminated by advances in camera sensor technology (Figure 10.18) (Yang, El Gamal, Fowler *et al.* 1999; Nayar and Mitsunaga 2000; Nayar and Branzoi 2003; Kang, Uyttendaele, Winder *et al.* 2003; Narasimhan and Nayar 2005; Tumblin, Agrawal, and Raskar 2005). However, the need to blend such images and to tone map them to lower-gamut displays is likely to remain.

HDR image formats. Before we discuss techniques for mapping HDR images back to a displayable gamut, we should discuss the commonly used formats for storing HDR images.

If storage space is not an issue, storing each of the R, G, and B values as a 32-bit IEEE float is the best solution. The commonly used Portable PixMap (.ppm) format, which supports both uncompressed ASCII and raw binary encodings of values, can be extended to a Portable FloatMap (.pfm) format by modifying the header. TIFF also supports full floating point values.

A more compact representation is the Radiance format (.pic, .hdr) (Ward 1994), which uses a single common exponent and per-channel mantissas (10.19b). An intermediate encoding, OpenEXR from ILM,¹² uses 16-bit floats for each channel (10.19c), which is a format supported natively on most modern GPUs. Ward (2004) describes these and other data formats such as LogLuv (Larson 1998) in more detail, as do the books by Reinhard, Ward,

¹² <http://www.openexr.net/>.

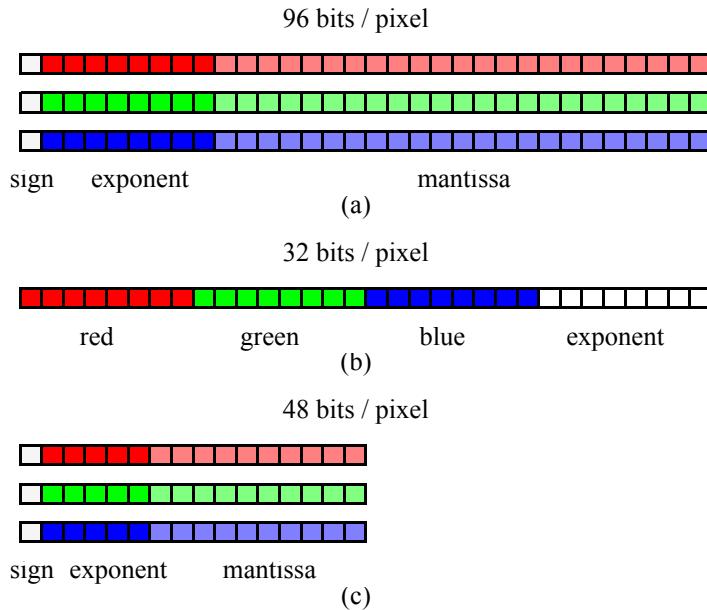


Figure 10.19 HDR image encoding formats: (a) Portable PixMap (.ppm); (b) Radiance (.pic, .hdr); (c) OpenEXR (.exr).

Pattanaik *et al.* (2005) and Freeman (2008). An even more recent HDR image format is the JPEG XR standard.¹³

10.2.1 Tone mapping

Once a radiance map has been computed, it is usually necessary to display it on a lower gamut (i.e., eight-bit) screen or printer. A variety of *tone mapping* techniques has been developed for this purpose, which involve either computing spatially varying transfer functions or reducing image gradients to fit the available dynamic range (Reinhard, Ward, Pattanaik *et al.* 2005).

The simplest way to compress a high dynamic range radiance image into a low dynamic range gamut is to use a global transfer curve (Larson, Rushmeier, and Piatko 1997). Figure 10.20 shows one such example, where a gamma curve is used to map an HDR image back into a displayable gamut. If gamma is applied separately to each channel (Figure 10.20b), the colors become muted (less saturated), since higher-valued color channels contribute less (proportionately) to the final color. Splitting the image up into its luminance and chrominance (say, L*a*b*) components (Section 2.3.2), applying the global mapping to the luminance channel, and then reconstituting a color image works better (Figure 10.20c).

Unfortunately, when the image has a really wide range of exposures, this global approach still fails to preserve details in regions with widely varying exposures. What is needed, instead, is something akin to the dodging and burning performed by photographers in the dark-room. Mathematically, this is similar to dividing each pixel by the *average* brightness in a region around that pixel.

¹³ <http://www.itu.int/rec/T-REC-T.832-200903-1/en>.



Figure 10.20 Global tone mapping: (a) input HDR image, linearly mapped; (b) gamma applied to each color channel independently; (c) gamma applied to intensity (colors are less washed out). Original HDR image courtesy of Paul Debevec, <http://ict.debevec.org/~debevec/Research/HDR/>. Processed images courtesy of Frédo Durand, MIT 6.815/6.865 course on Computational Photography.

Figure 10.21 shows how this process works. As before, the image is split into its luminance and chrominance channels. The log luminance image

$$H(x, y) = \log L(x, y) \quad (10.12)$$

is then low-pass filtered to produce a *base layer*

$$H_L(x, y) = B(x, y) * H(x, y), \quad (10.13)$$

and a high-pass *detail layer*

$$H_H(x, y) = H(x, y) - H_L(x, y). \quad (10.14)$$

The base layer is then contrast reduced by scaling to the desired log-luminance range,

$$H'_H(x, y) = s H_H(x, y) \quad (10.15)$$

and added to the detail layer to produce the new log-luminance image

$$I(x, y) = H'_H(x, y) + H_L(x, y), \quad (10.16)$$

which can then be exponentiated to produce the tone-mapped (compressed) luminance image. Note that this process is equivalent to dividing each luminance value by (a monotonic mapping of) the average log-luminance value in a region around that pixel.

Figure 10.21 shows the low-pass and high-pass log luminance image and the resulting tone-mapped color image. Note how the detail layer has visible *halos* around the high-contrast edges, which are visible in the final tone-mapped image. This is because linear filtering, which is not edge preserving, produces halos in the detail layer (Figure 10.23).

The solution to this problem is to use an edge-preserving filter to create the base layer. Durand and Dorsey (2002) study a number of such edge-preserving filters, including anisotropic and robust anisotropic diffusion, and select bilateral filtering (Section 3.3.1) as their edge-preserving filter. (A more recent paper by Farbman, Fattal, Lischinski *et al.* (2008) argues in favor of using a weighted least squares (WLF) filter as an alternative to the bilateral filter and Paris, Kornprobst, Tumblin *et al.* (2008) reviews bilateral filtering and its applications

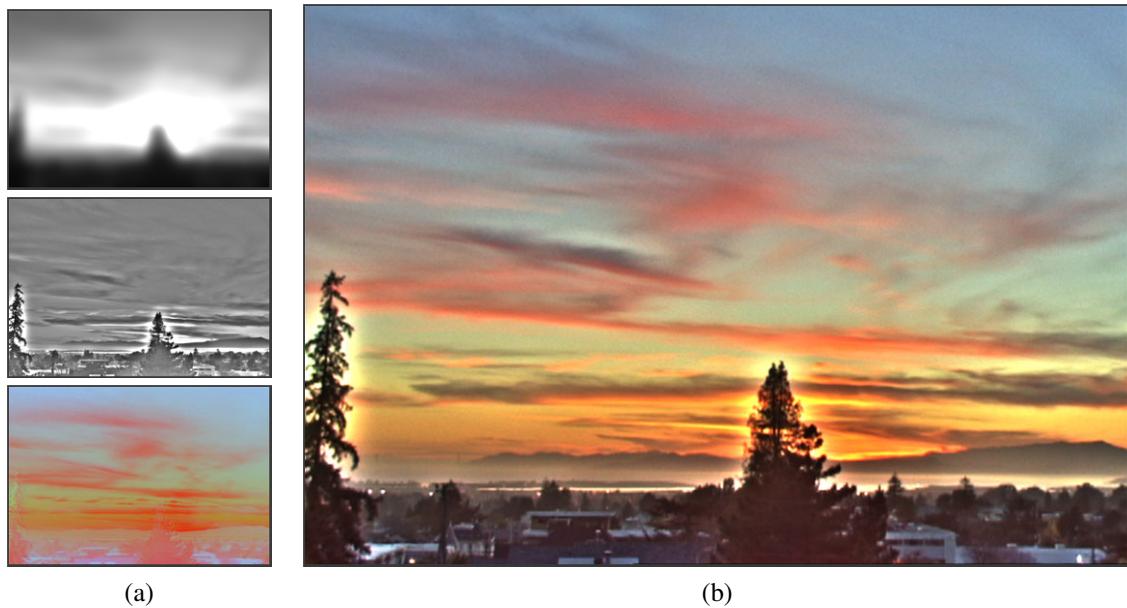


Figure 10.21 Local tone mapping using linear filters: (a) low-pass and high-pass filtered log luminance images and color (chrominance) image; (b) resulting tone-mapped image (after attenuating the low-pass log luminance image) shows visible halos around the trees. Processed images courtesy of Frédo Durand, MIT 6.815/6.865 course on Computational Photography.



Figure 10.22 Local tone mapping using bilateral filter (Durand and Dorsey 2002): (a) low-pass and high-pass bilateral filtered log luminance images and color (chrominance) image; (b) resulting tone-mapped image (after attenuating the low-pass log luminance image) shows no halos. Processed images courtesy of Frédo Durand, MIT 6.815/6.865 course on Computational Photography.

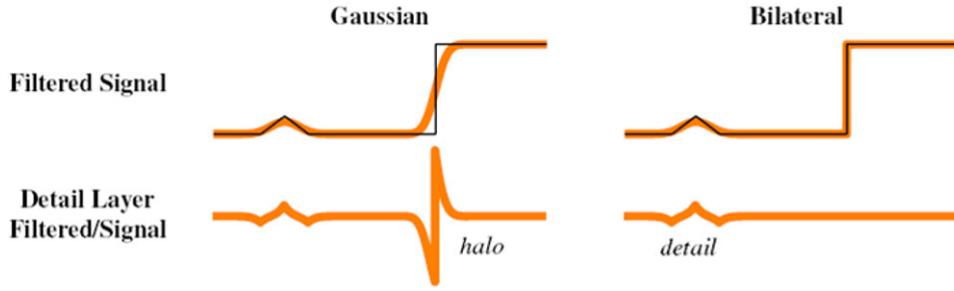


Figure 10.23 Gaussian vs. bilateral filtering (Petschnigg, Agrawala, Hoppe *et al.* 2004) © 2004 ACM: A Gaussian low-pass filter blurs across all edges and therefore creates strong peaks and valleys in the detail image that cause halos. The bilateral filter does not smooth across strong edges and thereby reduces halos while still capturing detail.

in computer vision and computational photography.) Figure 10.22 shows how replacing the linear low-pass filter with a bilateral filter produces tone-mapped images with no visible halos. Figure 10.24 summarizes the complete information flow in this process, starting with the decomposition into log luminance and chrominance images, bilateral filtering, contrast reduction, and re-composition into the final output image.

An alternative to compressing the base layer is to compress its *derivatives*, i.e., the gradient of the log-luminance image (Fattal, Lischinski, and Werman 2002). Figure 10.25 illustrates this process. The log-luminance image is differentiated to obtain a gradient image

$$H'(x, y) = \nabla H(x, y). \quad (10.17)$$

This gradient image is then attenuated by a spatially varying attenuation function $\Phi(x, y)$,

$$G(x, y) = H'(x, y) \Phi(x, y). \quad (10.18)$$

The attenuation function $I(x, y)$ is designed to attenuate large-scale brightness changes (Figure 10.26a) and is designed to take into account gradients at different spatial scales (Fattal, Lischinski, and Werman 2002).

After attenuation, the resulting gradient field is re-integrated by solving a first-order variational (least squares) problem,

$$\min \int \int \|\nabla I(x, y) - G(x, y)\|^2 dx dy \quad (10.19)$$

to obtain the compressed log-luminance image $I(x, y)$. This least squares problem is the same that was used for Poisson blending (Section 9.3.4) and was first introduced in our study of regularization (Section 3.7.1, 3.100). It can efficiently be solved using techniques such as multigrid and hierarchical basis preconditioning (Fattal, Lischinski, and Werman 2002; Szeliski 2006b; Farbman, Fattal, Lischinski *et al.* 2008). Once the new luminance image has been computed, it is combined with the original color image using

$$C_{\text{out}} = \left(\frac{C_{\text{in}}}{L_{\text{in}}} \right)^s L_{\text{out}}, \quad (10.20)$$

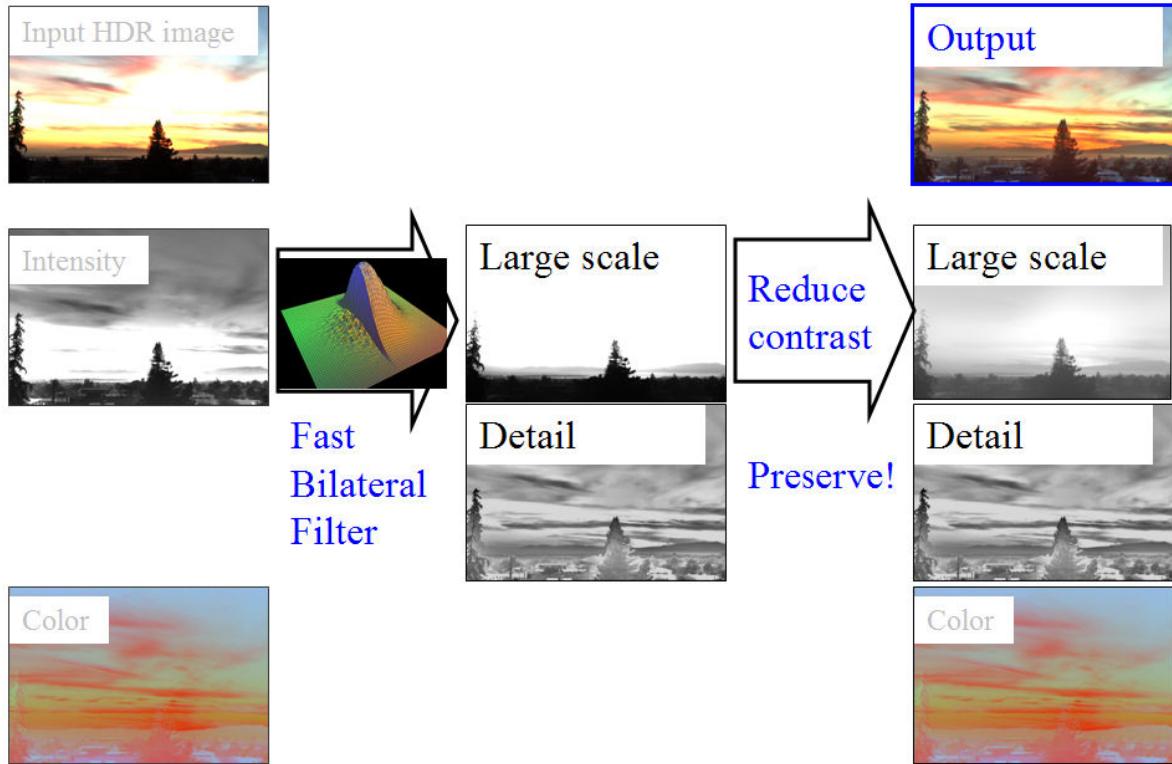


Figure 10.24 Local tone mapping using bilateral filter (Durand and Dorsey 2002): summary of algorithm workflow. Images courtesy of Frédo Durand, MIT 6.815/6.865 course on Computational Photography.

where $C = (R, G, B)$ and L_{in} and L_{out} are the original and compressed luminance images. The exponent s controls the saturation of the colors and is typically in the range $s \in [0.4, 0.6]$. Figure 10.26b shows the final tone-mapped color image, which shows no visible halos despite the extremely large variation in input radiance values.

Yet another alternative to these two approaches is to perform the local dodging and burning using a locally scale-selective operator (Reinhard, Stark, Shirley *et al.* 2002). Figure 10.27 shows how such a scale selection operator can determine a radius (scale) that only includes similar color values within the inner circle while avoiding much brighter values in the surrounding circle. In practice, a difference of Gaussians normalized by the inner Gaussian response is evaluated over a range of scales, and the largest scale whose metric is below a threshold is selected (Reinhard, Stark, Shirley *et al.* 2002).

What all of these techniques have in common is that they adaptively attenuate or brighten different regions of the image so that they can be displayed in a limited gamut without loss of contrast. Lischinski, Farbman, Uyttendaele *et al.* (2006b) introduce an *interactive* technique that performs this operation by interpolating a set of sparse user-drawn adjustments (strokes and associated exposure value corrections) to a piecewise-continuous exposure correction map (Figure 10.28). The interpolation is performed by minimizing a locally weighted least

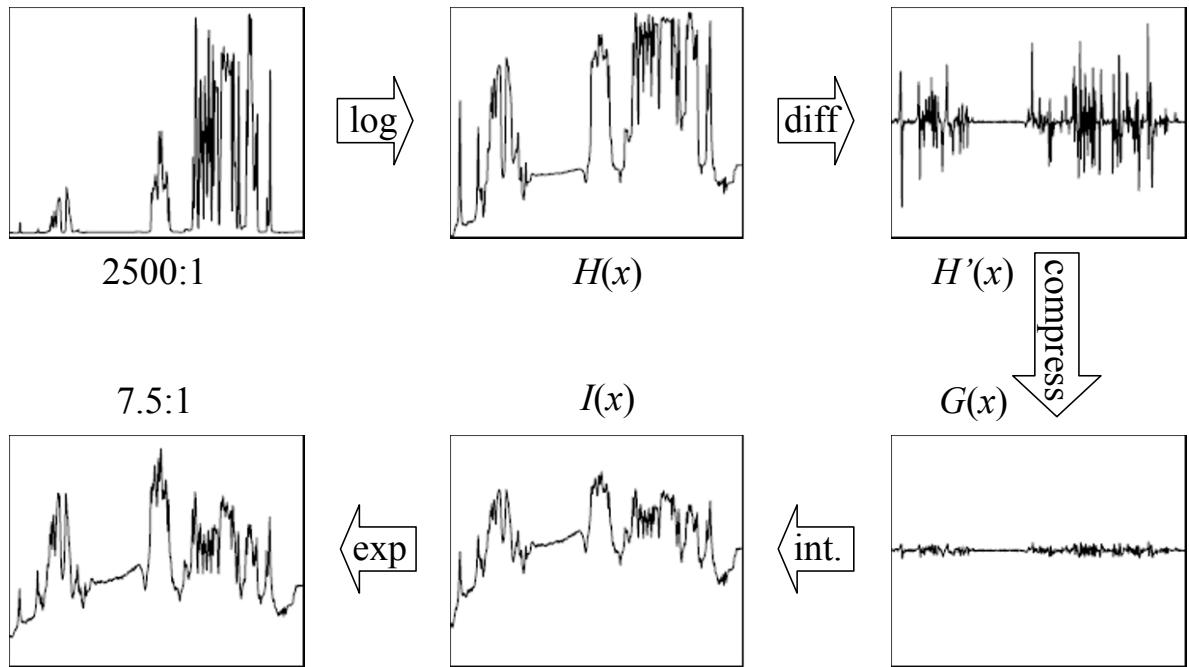


Figure 10.25 Gradient domain tone mapping (Fattal, Lischinski, and Werman 2002) © 2002 ACM. The original image with a dynamic range of 2415:1 is first converted into the log domain, $H(x)$, and its gradients are computed, $H'(x)$. These are attenuated (compressed) based on local contrast, $G(x)$, and integrated to produce the new logarithmic exposure image $I(x)$, which is exponentiated to produce the final intensity image, whose dynamic range is 7.5:1.



Figure 10.26 Gradient domain tone mapping (Fattal, Lischinski, and Werman 2002) © 2002 ACM: (a) attenuation map, with darker values corresponding to more attenuation; (b) final tone-mapped image.

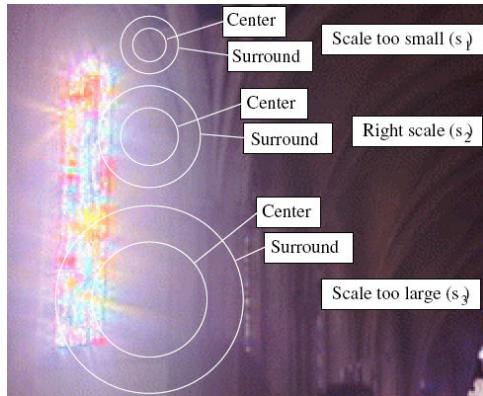


Figure 10.27 Scale selection for tone mapping (Reinhard, Stark, Shirley *et al.* 2002) © 2002 ACM.

square (WLS) variational problem,

$$\min \int \int w_d(x, y) \|f(x, y) - g(x, y)\|^2 dx dy + \lambda \int \int w_s(x, y) \|\nabla f(x, y)\|^2 dx dy, \quad (10.21)$$

where $g(x, y)$ and $f(x, y)$ are the input and output log exposure (attenuation) maps (Figure 10.28). The data weighting term $w_d(x, y)$ is 1 at stroke locations and 0 elsewhere. The smoothness weighting term $w_s(x, y)$ is inversely proportional to the log-luminance gradient,

$$w_s = \frac{1}{\|\nabla H\|^\alpha + \epsilon} \quad (10.22)$$

and hence encourages the $f(x, y)$ map to be smoother in low-gradient areas than along high-gradient discontinuities.¹⁴ The same approach can also be used for fully automated tone mapping by setting target exposure values at each pixel and allowing the weighted least squares to convert these into piecewise smooth adjustment maps.

The weighted least squares algorithm, which was originally developed for image colorization applications (Levin, Lischinski, and Weiss 2004), has recently been applied to general edge-preserving smoothing in applications such as contrast enhancement (Bae, Paris, and Durand 2006) and tone mapping (Farbman, Fattal, Lischinski *et al.* 2008) where the bilateral filtering was previously used. It can also be used to perform HDR merging and tone mapping simultaneously (Raman and Chaudhuri 2007, 2009).

Given the wide range of locally adaptive tone mapping algorithms that have been developed, which ones should be used in practice? Freeman (2008) provides a great discussion of commercially available algorithms, their artifacts, and the parameters that can be used to control them. He also has a wealth of tips for HDR photography and workflow. I highly recommend his book for anyone contemplating additional research (or personal photography) in this area.

¹⁴ In practice, the x and y discrete derivatives are weighted separately (Lischinski, Farbman, Uyttendaele *et al.* 2006b). Their default parameter settings are $\lambda = 0.2$, $\alpha = 1$, and $\epsilon = 0.0001$.

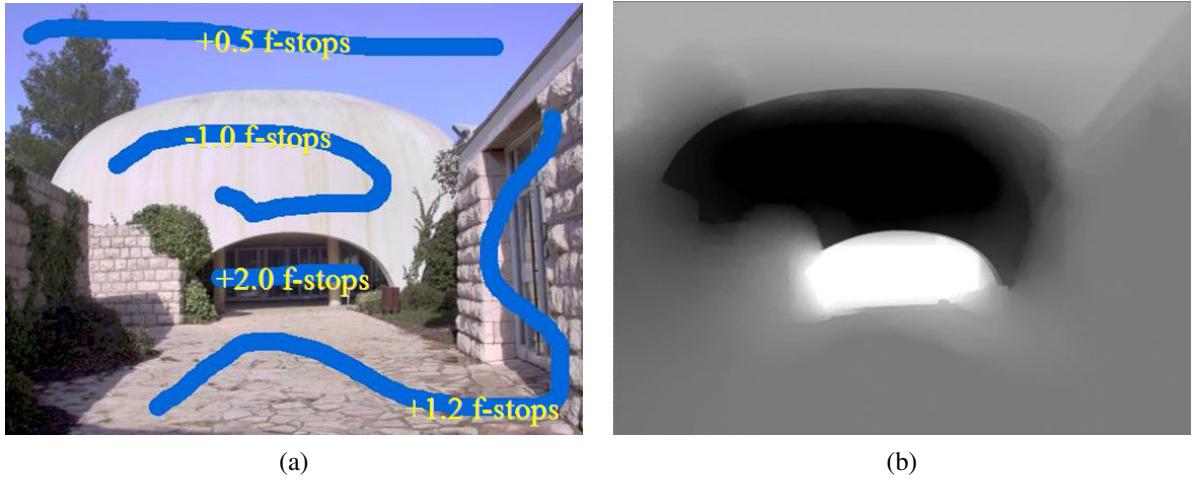


Figure 10.28 Interactive local tone mapping (Lischinski, Farbman, Uyttendaele *et al.* 2006b) © 2006 ACM: (a) user-drawn strokes with associated exposure values $g(x, y)$ (b) corresponding piecewise-smooth exposure adjustment map $f(x, y)$.

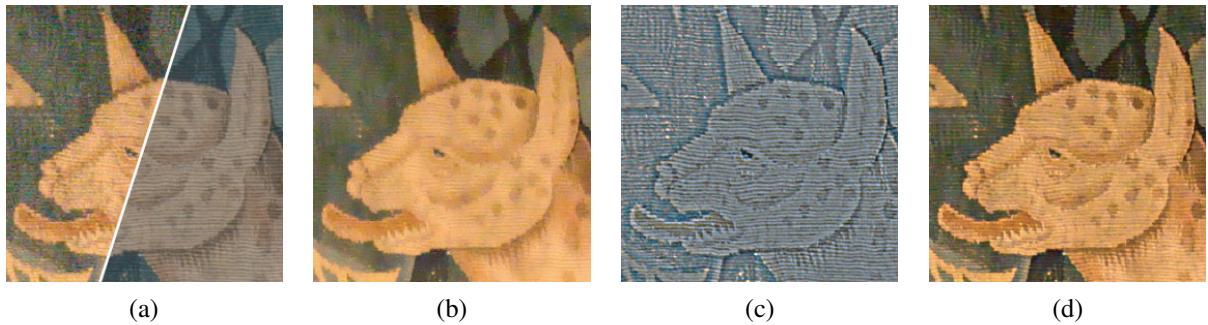


Figure 10.29 Detail transfer in flash/no-flash photography (Petschnigg, Agrawala, Hoppe *et al.* 2004) © 2004 ACM: (a) details of input ambient A and flash F images; (b) joint bilaterally filtered no-flash image A^{NR} ; (c) detail layer F^{Detail} computed from the flash image F ; (d) final merged image A^{Final} .

10.2.2 Application: Flash photography

While high dynamic range imaging combines images of a scene taken at different exposures, it is also possible to combine flash and non-flash images to achieve better exposure and color balance and to reduce noise (Eisemann and Durand 2004; Petschnigg, Agrawala, Hoppe *et al.* 2004).

The problem with flash images is that the color is often unnatural (it fails to capture the ambient illumination), there may be strong shadows or specularities, and there is a radial falloff in brightness away from the camera (Figures 10.1b and 10.29a). Non-flash photos taken under low light conditions often suffer from excessive noise (because of the high ISO gains and low photon counts) and blur (due to longer exposures). Is there some way to combine a non-flash photo taken just before the flash goes off with the flash photo to produce

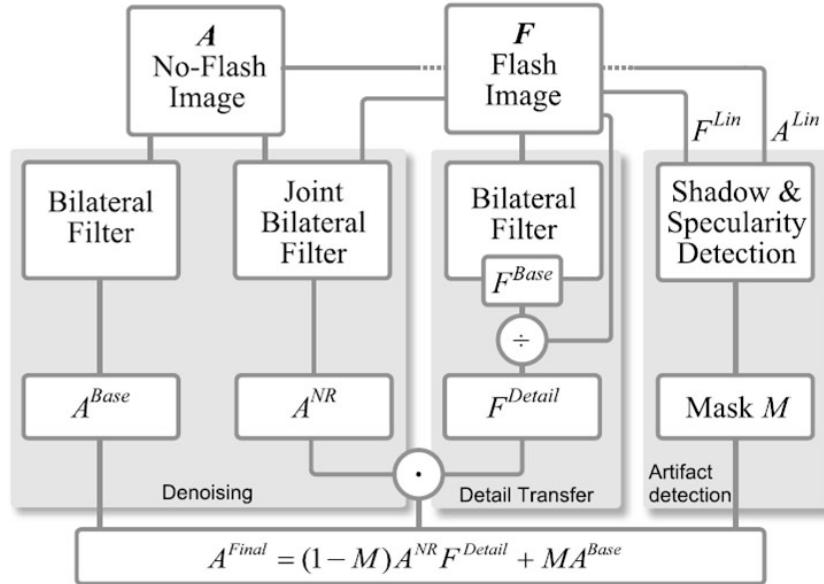


Figure 10.30 Flash/no-flash photography algorithm (Petschnigg, Agrawala, Hoppe *et al.* 2004) © 2004 ACM. The ambient (no-flash) image A is filtered with a regular bilateral filter to produce A^{Base} , which is used in shadow and specularity regions, and a joint bilaterally filtered noise reduced image A^{NR} . The flash image F is bilaterally filtered to produce a base image F^{Base} and a detail (ratio) image F^{Detail} , which is used to modulate the de-noised ambient image. The shadow/specularity mask M is computed by comparing linearized versions of the flash and no-flash images.

an image with good color values, sharpness, and low noise?¹⁵

Petschnigg, Agrawala, Hoppe *et al.* (2004) approach this problem by first filtering the no-flash (ambient) image A with a variant of the bilateral filter called the *joint bilateral filter*¹⁶ in which the range kernel (3.36)

$$r(i, j, k, l) = \exp\left(-\frac{\|f(i, j) - f(k, l)\|^2}{2\sigma_r^2}\right) \quad (10.23)$$

is evaluated on the flash image F instead of the ambient image A , since the flash image is less noisy and hence has more reliable edges (Figure 10.29b). Because the contents of the flash image can be unreliable inside and at the boundaries of shadows and specularities, these are detected and a regular bilaterally filtered image A^{Base} is used instead (Figure 10.30).

The second stage of their algorithm computes a flash detail image

$$F^{Detail} = \frac{F + \epsilon}{F^{Base} + \epsilon}, \quad (10.24)$$

where F^{Base} is a bilaterally filtered version of the flash image F and $\epsilon = 0.02$. This detail image (Figure 10.29c) encodes details that may have been filtered away from the noise-reduced

¹⁵ In fact, the discontinued FujiFilm FinePix F40fd camera takes a pair of flash and no flash images in quick succession; however, it only lets you decide to keep one of them.

¹⁶ Eisemann and Durand (2004) call this the *cross bilateral filter*.

no-flash image A^{NR} , as well as additional details created by the flash camera, which often add crispness. The detail image is used to modulate the noise-reduced ambient image A^{NR} to produce the final results

$$A^{Final} = (1 - M)A^{NR}F^{Detail} + MA^{Base} \quad (10.25)$$

shown in Figures 10.1b and 10.29d.

Eisemann and Durand (2004) present an alternative algorithm that shares some of the same basic concepts. Both papers are well worth reading and contrasting (Exercise 10.6).

Flash images can also be used for a variety of additional applications such as extracting more reliable foreground mattes of objects (Raskar, Tan, Feris *et al.* 2004; Sun, Li, Kang *et al.* 2006). Flash photography is just one instance of the more general topic of *active illumination*, which is discussed in more detail by Raskar and Tumblin (2010).

10.3 Super-resolution and blur removal

While high dynamic range imaging enables us to obtain an image with a larger dynamic range than a single regular image, super-resolution enables us to create images with higher *spatial* resolution and less noise than regular camera images (Chaudhuri 2001; Park, Park, and Kang 2003; Capel and Zisserman 2003; Capel 2004; van Ouwerkerk 2006). Most commonly, super-resolution refers to the process of aligning and combining several input images to produce such high-resolution composites (Irani and Peleg 1991; Cheeseman, Kanefsky, Hanson *et al.* 1993; Pickup, Capel, Roberts *et al.* 2009). However, some newer techniques can super-resolve a single image (Freeman, Jones, and Pasztor 2002; Baker and Kanade 2002; Fattal 2007) and are hence closely related to techniques for removing blur (Sections 3.4.3 and 3.4.4).

The most principled way to formulate the super-resolution problem is to write down the stochastic image formation equations and image priors and to then use Bayesian inference to recover the super-resolved (original) sharp image. We can do this by generalizing the image formation equations (3.75) used for image deblurring (Section 3.4.3), which we also used in (10.2) for blur kernel (PSF) estimation (Section 10.1.4). In this case, we have several observed images $\{o_k(\mathbf{x})\}$, as well as an image warping function $\hat{\mathbf{h}}_k(\mathbf{x})$ for each observed image (Figure 3.47). Combining all of these elements, we get the (noisy) observation equations¹⁷

$$o_k(\mathbf{x}) = D\{b(\mathbf{x}) * s(\hat{\mathbf{h}}_k(\mathbf{x}))\} + n_k(\mathbf{x}), \quad (10.26)$$

where D is the downsampling operator, which operates *after* the super-resolved (sharp) warped image $s(\hat{\mathbf{h}}_k(\mathbf{x}))$ has been convolved with the blur kernel $b(\mathbf{x})$. The above image formation equations lead to the following least squares problem,

$$\sum_k \|o_k(\mathbf{x}) - D\{b_k(\mathbf{x}) * s(\hat{\mathbf{h}}_k(\mathbf{x}))\}\|^2. \quad (10.27)$$

In most super-resolution algorithms, the alignment (warping) $\hat{\mathbf{h}}_k$ is estimated using one of the input frames as the *reference frame*; either feature-based (Section 6.1.3) or direct (image-based) (Section 8.2) parametric alignment techniques can be used. (A few algorithms, such

¹⁷ It is also possible to add an unknown bias–gain term to each observation (Capel 2004), as was done for motion estimation in (8.8).

as those described by Schultz and Stevenson (1996) or Capel (2004) use dense (per-pixel flow) estimates.) A better approach is to re-compute the alignment by directly minimizing (10.27) once an initial estimate of $s(\mathbf{x})$ has been computed (Hardie, Barnard, and Armstrong 1997) or to *marginalize* out the motion parameters altogether (Pickup, Capel, Roberts *et al.* 2007)—see also the work of Proter and Elad (2009) for some related video super-resolution work.

The point spread function (blur kernel) b_k is either inferred from knowledge of the image formation process (e.g., the amount of motion or defocus blur and the camera sensor optics) or calibrated from a test image or the observed images $\{o_k\}$ using one of the techniques described in Section 10.1.4. The problem of simultaneously inferring the blur kernel and the sharp image is known as *blind image deconvolution* (Kundur and Hatzinakos 1996; Levin 2006).¹⁸

Given an estimate of $\hat{\mathbf{h}}_k$ and $b_k(\mathbf{x})$, (10.27) can be re-written using matrix/vector notation as a large sparse least squares problem in the unknown values of the super-resolved pixels \mathbf{s} ,

$$\sum_k \|\mathbf{o}_k - \mathbf{D}\mathbf{B}_k \mathbf{W}_k \mathbf{s}\|^2. \quad (10.28)$$

(Recall from (3.89) that once the warping function $\hat{\mathbf{h}}_k$ is known, values of $s(\hat{\mathbf{h}}_k(\mathbf{x}))$ depend linearly on those in $s(\mathbf{x})$.) An efficient way to solve this least squares problem is to use preconditioned conjugate gradient descent (Capel 2004), although some earlier algorithms, such as the one developed by Irani and Peleg (1991), used regular gradient descent (also known as iterative back projection (IBP), in the computed tomography literature).

The above formulation assumes that warping can be expressed as a simple (sinc or bicubic) interpolated resampling of the super-resolved sharp image, followed by a stationary (spatially invariant) blurring (PSF) and area integration process. However, if the surface is severely foreshortened, we have to take into account the spatially varying filtering that occurs during the image warping (Section 3.6.1), before we can then model the PSF induced by the optics and camera sensor (Wang, Kang, Szeliski *et al.* 2001; Capel 2004).

How well does this least squares (MLE) approach to super-resolution work? In practice, this depends a lot on the amount of blur and aliasing in the camera optics, as well as the accuracy in the motion and PSF estimates (Baker and Kanade 2002; Jiang, Wong, and Bao 2003; Capel 2004). Less blurring and more aliasing means that there is more (aliased) high frequency information available to be recovered. However, because the least squares (maximum likelihood) formulation uses no image prior, a lot of high-frequency noise can be introduced into the solution (Figure 10.31c).

For this reason, most super-resolution algorithms assume some form of image prior. The simplest of these is to place a penalty on the image derivatives similar to Equations (3.105 and 3.113), e.g.,

$$\sum_{(i,j)} \rho_p(s(i,j) - s(i+1,j)) + \rho_p(s(i,j) - s(i,j+1)). \quad (10.29)$$

¹⁸ Notice that there is a chicken-and-egg problem if both the blur kernel and the super-resolved image are unknown. This can be “broken” either using structural assumptions about the sharp image, e.g., the presence of edges (Joshi, Szeliski, and Kriegman 2008) or prior models for the image, such as edge sparsity (Fergus, Singh, Hertzmann *et al.* 2006).

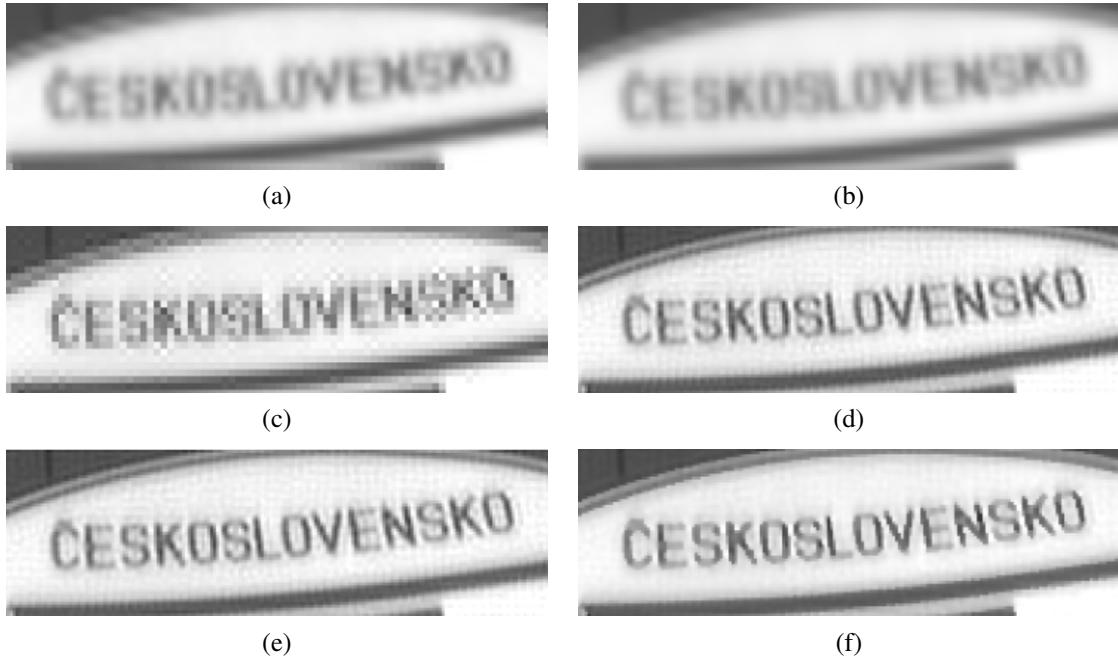


Figure 10.31 Super-resolution results using a variety of image priors (Capel 2001): (a) Low-res ROI (bicubic 3× zoom); (b) average image; (c) MLE @ 1.25× pixel-zoom; (d) simple $\|x\|^2$ prior ($\lambda = 0.004$); (e) GMRF ($\lambda = 0.003$); (f) HMRF ($\lambda = 0.01$, $\alpha = 0.04$). 10 images are used as input and a 3× super-resolved image is produced in each case, except for the MLE result in (c).

As discussed in Section 3.7.2, when ρ_p is quadratic, this is a form of Tikhonov regularization (Section 3.7.1), and the overall problem is still linear least squares. The resulting prior image model is a Gaussian Markov random field (GMRF), which can be extended to other (e.g., diagonal) differences, as in (Capel 2004) (Figure 10.31).

Unfortunately, GMRFs tend to produce solutions with visible ripples, which can also be interpreted as increased noise sensitivity in middle frequencies (Exercise 3.17). A better image prior is a robust prior that encourages piecewise continuous solutions (Black and Rangarajan 1996), see Appendix B.3. Examples of such priors include the Huber potential (Schultz and Stevenson 1996; Capel and Zisserman 2003), which is a blend of a Gaussian with a longer-tailed Laplacian, and the even sparser (heavier-tailed) hyper-Laplacians used by Levin, Fergus, Durand *et al.* (2007) and Krishnan and Fergus (2009). It is also possible to learn the parameters for such priors using cross-validation (Capel 2004; Pickup 2007).

While sparse (robust) derivative priors can reduce rippling effects and increase edge sharpness, they cannot *hallucinate* higher-frequency texture or details. To do this, a training set of sample images can be used to find plausible mappings between low-frequency originals and the missing higher frequencies. Inspired by some of the example-based texture synthesis algorithms we discuss in Section 10.5, the *example-based super-resolution* algorithm developed by Freeman, Jones, and Pasztor (2002) uses training images to *learn* the mapping between local texture patches and missing higher-frequency details. To ensure that overlapping patches are similar in appearance, a Markov random field is used and optimized

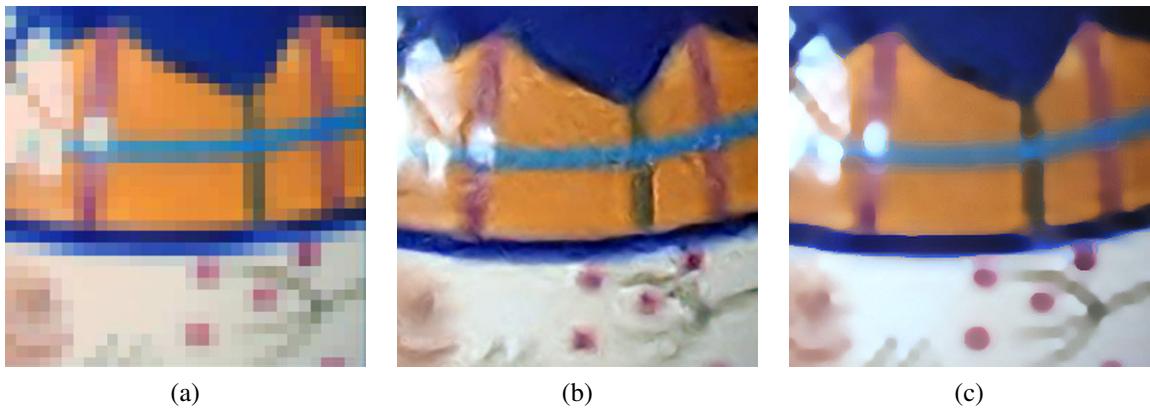


Figure 10.32 Example-based super-resolution: (a) original 32×32 low-resolution image; (b) example-based super-resolved 256×256 image (Freeman, Jones, and Pasztor 2002) © 2002 IEEE; (c) upsampling via imposed edge statistics (Fattal 2007) © 2007 ACM.

using either belief propagation (Freeman, Pasztor, and Carmichael 2000) or a raster-scan deterministic variant (Freeman, Jones, and Pasztor 2002). Figure 10.32 shows the results of hallucinating missing details using this approach and compares these results to a more recent algorithm by Fattal (2007). This latter algorithm learns to predict oriented gradient magnitudes in the finer resolution image based on a pixel’s location relative to the nearest detected edge along with the corresponding edge statistics (magnitude and width). It is also possible to combine sparse (robust) derivative priors with example-based super-resolution, as shown by Tappen, Russell, and Freeman (2003).

An alternative (but closely related) form of hallucination is to *recognize* the parts of a training database of images to which a low-resolution pixel might correspond. In their work, Baker and Kanade (2002) use local derivative-of-Gaussian filter responses as features and then match *parent structure* vectors in a manner similar to De Bonet (1997).¹⁹ The high-frequency gradient at each recognized training image location is then used as a constraint on the super-resolved image, along with the usual reconstruction (prediction) equation (10.27). Figure 10.33 shows the result of hallucinating higher-resolution faces from lower-resolution inputs; Baker and Kanade (2002) also show examples of super-resolving known-font text. Exercise 10.7 gives more details on how to implement and test one or more of these super-resolution techniques.

Under favorable conditions, super-resolution and related upsampling techniques can increase the resolution of a well-photographed image or image collection. When the input images are blurry to start with, the best one can often hope for is to reduce the amount of blur. This problem is closely related super-resolution, with the biggest differences being that the blur kernel b is usually much larger and the downsampling factor D is unity. A large literature on image deblurring exists; some of the more recent publications with nice literature reviews include those by Fergus, Singh, Hertzmann *et al.* (2006), Yuan, Sun, Quan *et al.* (2008), and Joshi, Zitnick, Szeliski *et al.* (2009). It is also possible to reduce blur by combining sharp (but noisy) images with blurrier (but cleaner) images (Yuan, Sun, Quan *et al.* 2007), take lots of

¹⁹ For face super-resolution, where all the images are pre-aligned, only corresponding pixels in different images are examined.

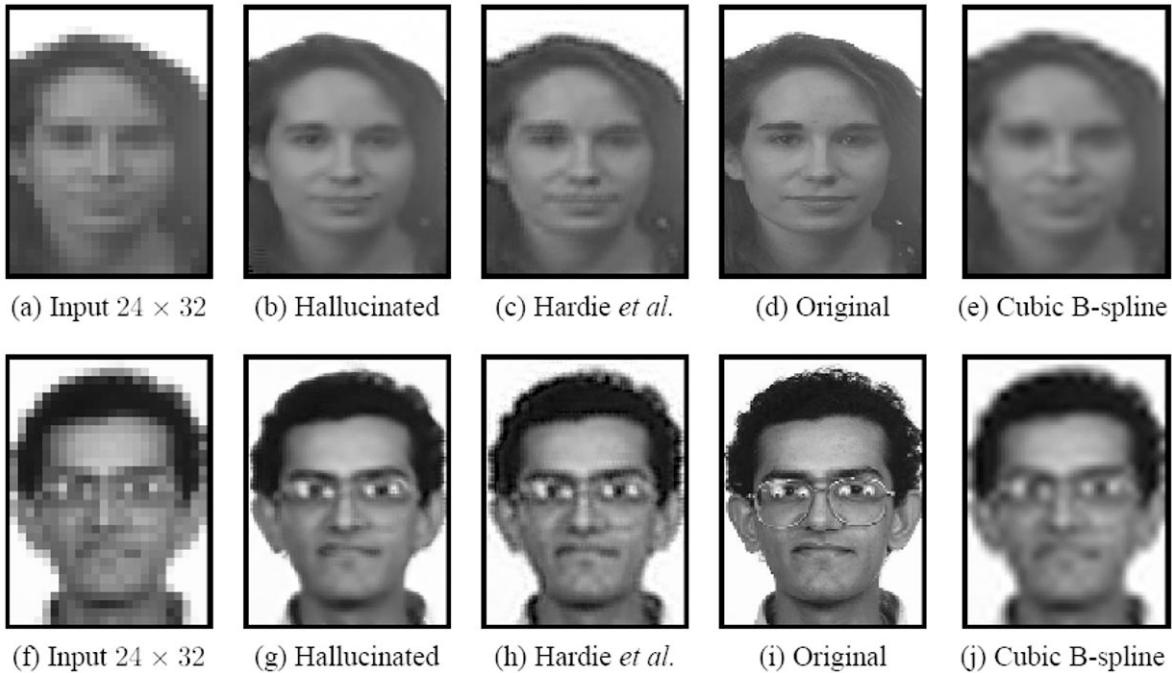


Figure 10.33 Recognition-based super-resolution (Baker and Kanade 2002) © 2002 IEEE. The *Hallucinated* column shows the results of the recognition-based algorithm compared to the regularization-based approach of Hardie, Barnard, and Armstrong (1997).

quick exposures²⁰ (Hasinoff and Kutulakos 2008; Hasinoff, Kutulakos, Durand *et al.* 2009; Hasinoff, Durand, and Freeman 2010), or use *coded aperture* techniques to simultaneously estimate depth and reduce blur (Levin, Fergus, Durand *et al.* 2007; Zhou, Lin, and Nayar 2009).

10.3.1 Color image demosaicing

A special case of super-resolution, which is used daily in most digital still cameras, is the process of *demosicing* samples from a color filter array (CFA) into a full-color RGB image. Figure 10.34 shows the most commonly used CFA known as the *Bayer pattern*, which has twice as many green (G) sensors as red and blue sensors.

The process of going from the known CFA pixels values to the full RGB image is quite challenging. Unlike regular super-resolution, where small errors in guessing unknown values usually show up as blur or aliasing, demosaicing artifacts often produce spurious colors or high-frequency patterned *zippering*, which are quite visible to the eye (Figure 10.35b).

Over the years, a variety of techniques have been developed for image demosaicing (Kimmel 1999). Bennett, Uyttendaele, Zitnick *et al.* (2006) present a recently developed algorithm along with some good references, while Longere, Delahunt, Zhang *et al.* (2002) and Tappen, Russell, and Freeman (2003) compare some previously developed techniques using perceptually motivated metrics. To reduce the zipper effect, most techniques use the edge or

²⁰ The SONY DSC-WX1 takes multiple shots to produce better low-light photos.

| | | | |
|---|---|---|---|
| G | R | G | R |
| B | G | B | G |
| G | R | G | R |
| B | G | B | G |

(a)

| | | | |
|-----|-----|-----|-----|
| rGb | Rgb | rGb | Rgb |
| rgB | rGb | rgB | rGb |
| rGb | Rgb | rGb | Rgb |
| rgB | rGb | rgB | rGb |

(b)

Figure 10.34 Bayer RGB pattern: (a) color filter array layout; (b) interpolated pixel values, with unknown (guessed) values shown as lower case.

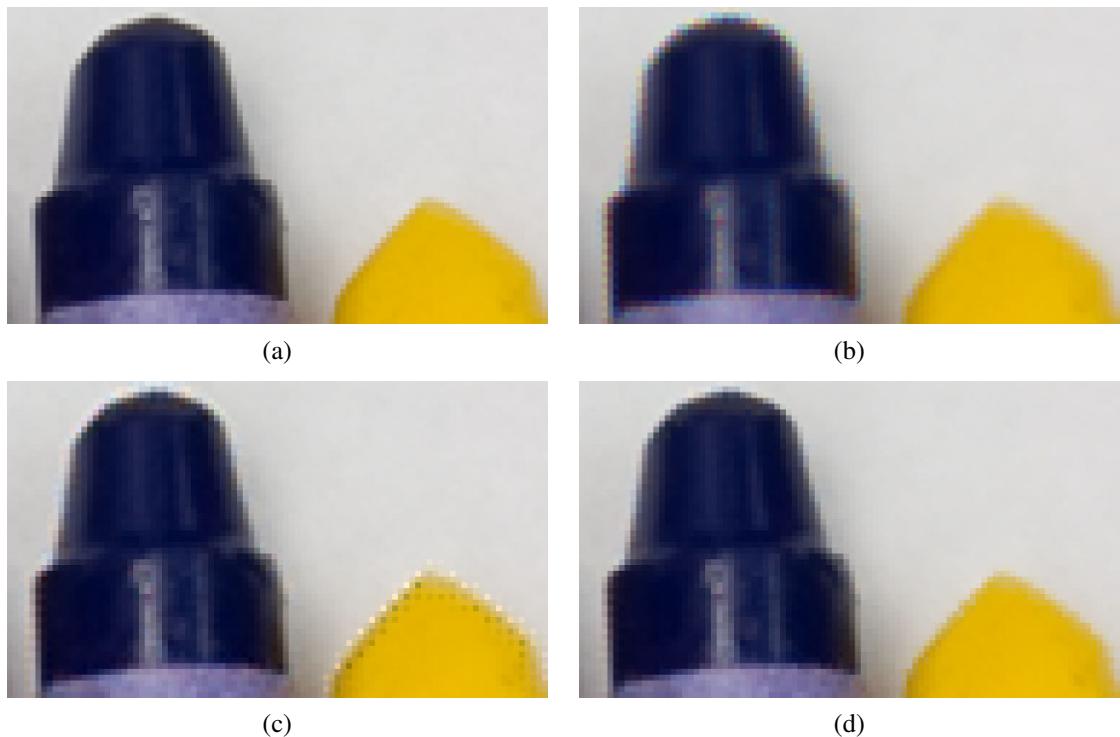


Figure 10.35 CFA demosaicing results (Bennett, Uyttendaele, Zitnick *et al.* 2006) © 2006 Springer: (a) original full-resolution image (a color subsampled version is used as the input to the algorithms); (b) bilinear interpolation results, showing color fringing near the tip of the blue crayon and zippering near its left (vertical) edge; (c) the high-quality linear interpolation results of Malvar, He, and Cutler (2004) (note the strong halo/checkerboard artifacts on the yellow crayon); (d) using the local two-color prior of Bennett, Uyttendaele, Zitnick *et al.* (2006).

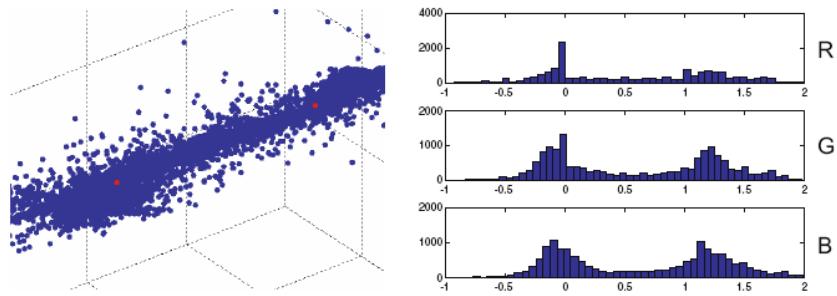


Figure 10.36 Two-color model computed from a collection of local 5×5 neighborhoods (Bennett, Uyttendaele, Zitnick *et al.* 2006) © 2006 Springer. After two-means clustering and reprojection along the line joining the two dominant colors (red dots), the majority of the pixels fall near the fitted line. The distribution along the line, projected along the RGB axes, is peaked at 0 and 1, the two dominant colors.

gradient information from the green channel, which is more reliable because it is sampled more densely, to infer plausible values for the red and blue channels, which are more sparsely sampled.

To reduce color fringing, some techniques perform a color space analysis, e.g., using median filtering on color opponent channels (Longere, Delahunt, Zhang *et al.* 2002). The approach of Bennett, Uyttendaele, Zitnick *et al.* (2006) locally forms a two-color model from an initial demosaicing result, using a moving 5×5 window to find the two dominant colors (Figure 10.36).²¹

Once the local color model has been estimated at each pixel, a Bayesian approach is then used to encourage pixel values to lie along each color line and to cluster around the dominant color values, which reduces halos (Figure 10.35d). The Bayesian approach also supports the simultaneous application of demosaicing and super-resolution, i.e., multiple CFA inputs can be merged into a higher-quality full-color image, which becomes more important as additional processing becomes incorporated into today’s cameras.

10.3.2 Application: Colorization

Although not strictly an example of super-resolution, the process of *colorization*, i.e., manually adding colors to a “black and white” (grayscale) image, is another example of a sparse interpolation problem. In most applications of colorization, the user draws some scribbles indicating the desired colors in certain regions (Figure 10.37a) and the system interpolates the specified chrominance (u, v) values to the whole image, which are then re-combined with the input luminance channel to produce a final colorized image, as shown in Figure 10.37b. In the system developed by Levin, Lischinski, and Weiss (2004), the interpolation is performed using locally weighted regularization (3.100), where the local smoothness weights are inversely proportional to luminance gradients. This approach to locally weighted regularization has inspired later algorithms for high dynamic range tone mapping (Lischinski, Farbman, Uyttendaele *et al.* 2006a), see Section 10.2.1, as well as other applications of the weighted least

²¹ Previous work on locally linear color models (Klinker, Shafer, and Kanade 1990; Omer and Werman 2004) focuses on color and illumination variation within a single material, whereas Bennett, Uyttendaele, Zitnick *et al.* (2006) use the two-color model to describe variations across color (material) edges.

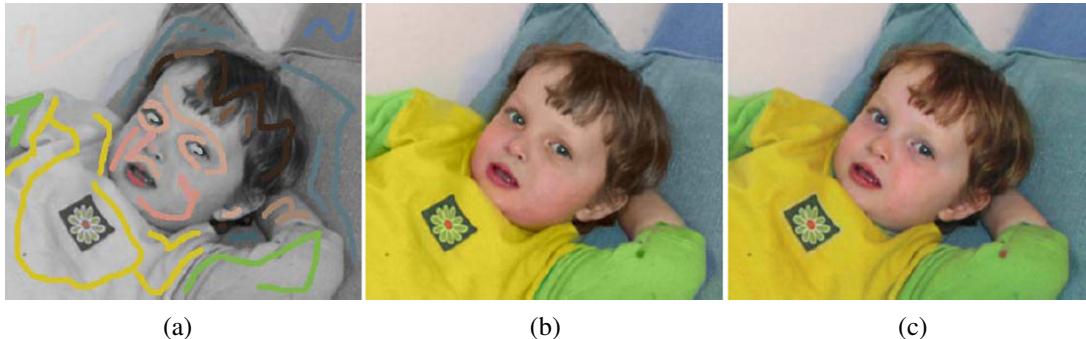


Figure 10.37 Colorization using optimization (Levin, Lischinski, and Weiss 2004) © 2004 ACM: (a) grayscale image some color scribbles overlaid; (b) resulting colorized image; (c) original color image from which the grayscale image and the chrominance values for the scribbles were derived. Original photograph by Rotem Weiss.

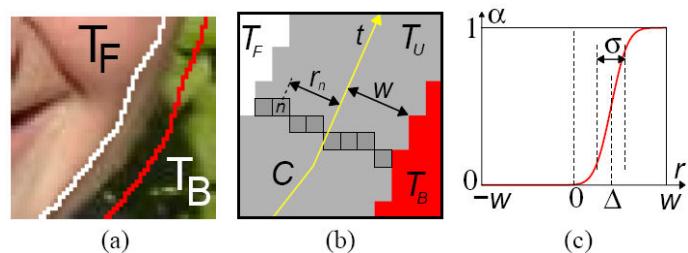


Figure 10.38 Softening a hard segmentation boundary (border matting) (Rother, Kolmogorov, and Blake 2004) © 2004 ACM: (a) the region surrounding a segmentation boundary where pixels of mixed foreground and background colors are visible; (b) pixel values along the boundary are used to compute a soft alpha matte; (c) at each point along the curve t , a displacement Δ and a width σ are estimated.

squares (WLS) formulation (Farbman, Fattal, Lischinski *et al.* 2008). An alternative approach to performing the sparse chrominance interpolation based on geodesic (edge-aware) distance functions has been developed by Yatziv and Sapiro (2006).

10.4 Image matting and compositing

Image matting and compositing is the process of cutting a foreground object out of one image and pasting it against a new background (Smith and Blinn 1996; Wang and Cohen 2007a). It is commonly used in television and film production to composite a live actor in front of computer-generated imagery such as weather maps or 3D virtual characters and scenery (Wright 2006; Brinkmann 2008).

We have already seen a number of tools for interactively segmenting objects in an image, including snakes (Section 5.1.1), scissors (Section 5.1.3), and GrabCut segmentation (Section 5.5). While these techniques can generate reasonable pixel-accurate segmentations, they fail to capture the subtle interplay of foreground and background colors at *mixed pixels* along the boundary (Szeliski and Golland 1999) (Figure 10.38a).

In order to successfully copy a foreground object from one image to another without

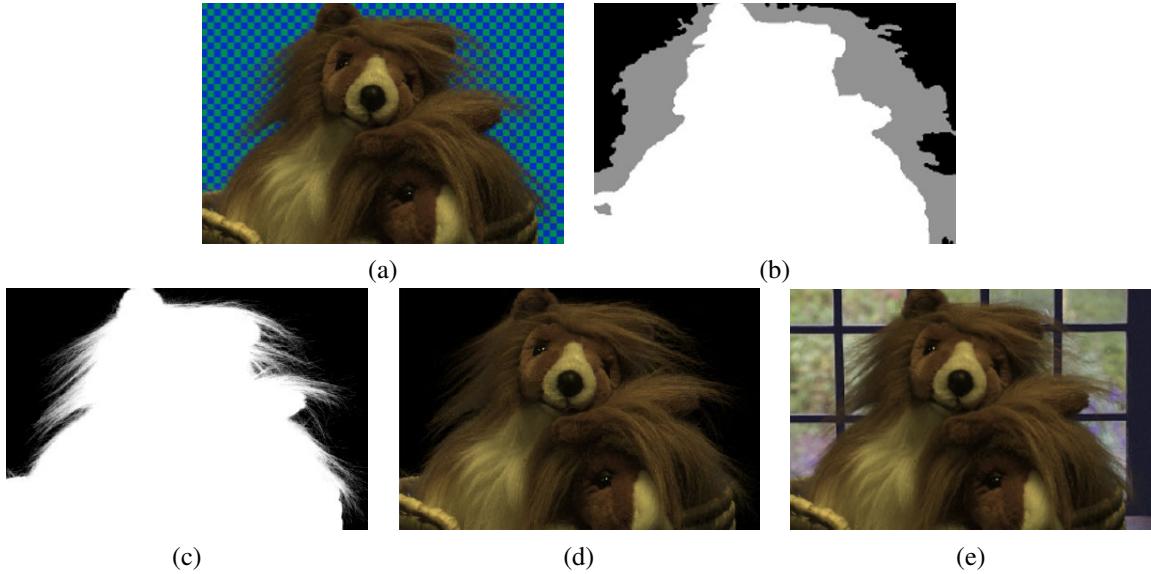


Figure 10.39 Natural image matting (Chuang, Curless, Salesin *et al.* 2001) © 2001 IEEE: (a) input image with a “natural” (non-constant) background; (b) hand-drawn trimap—gray indicates unknown regions; (c) extracted alpha map; (d) extracted (premultiplied) foreground colors; (e) composite over a new background.

visible discretization artifacts, we need to *pull a matte*, i.e., to estimate a soft opacity channel α and the uncontaminated foreground colors F from the input composite image C . Recall from Section 3.1.3 (Figure 3.4) that the compositing equation (3.8) can be written as

$$C = (1 - \alpha)B + \alpha F. \quad (10.30)$$

This operator attenuates the influence of the background image B by a factor $(1 - \alpha)$ and then adds in the (partial) color values corresponding to the foreground element F .

While the compositing operation is easy to implement, the reverse *matting* operation of estimating F , α , and B given an input image C is much more challenging (Figure 10.39). To see why, observe that while the composite pixel color C provides three measurements, the F , α , and B unknowns have a total of seven degrees of freedom. Devising techniques to estimate these unknowns despite the underconstrained nature of the problem is the essence of image matting.

In this section, we review a number of image matting techniques. We begin with *blue screen matting*, which assumes that the background is a constant known color, and discuss its variants, two-screen matting (when multiple backgrounds can be used) and difference matting (where the known background is arbitrary). We then discuss local variants of *natural image matting*, where both the foreground and background are unknown. In these applications, it is usual to first specify a *trimap*, i.e., a three-way labeling of the image into foreground, background, and unknown regions (Figure 10.39b). Next, we present some global optimization approaches to natural image matting. Finally, we discuss variants on the matting problem, including shadow matting, flash matting, and environment matting.

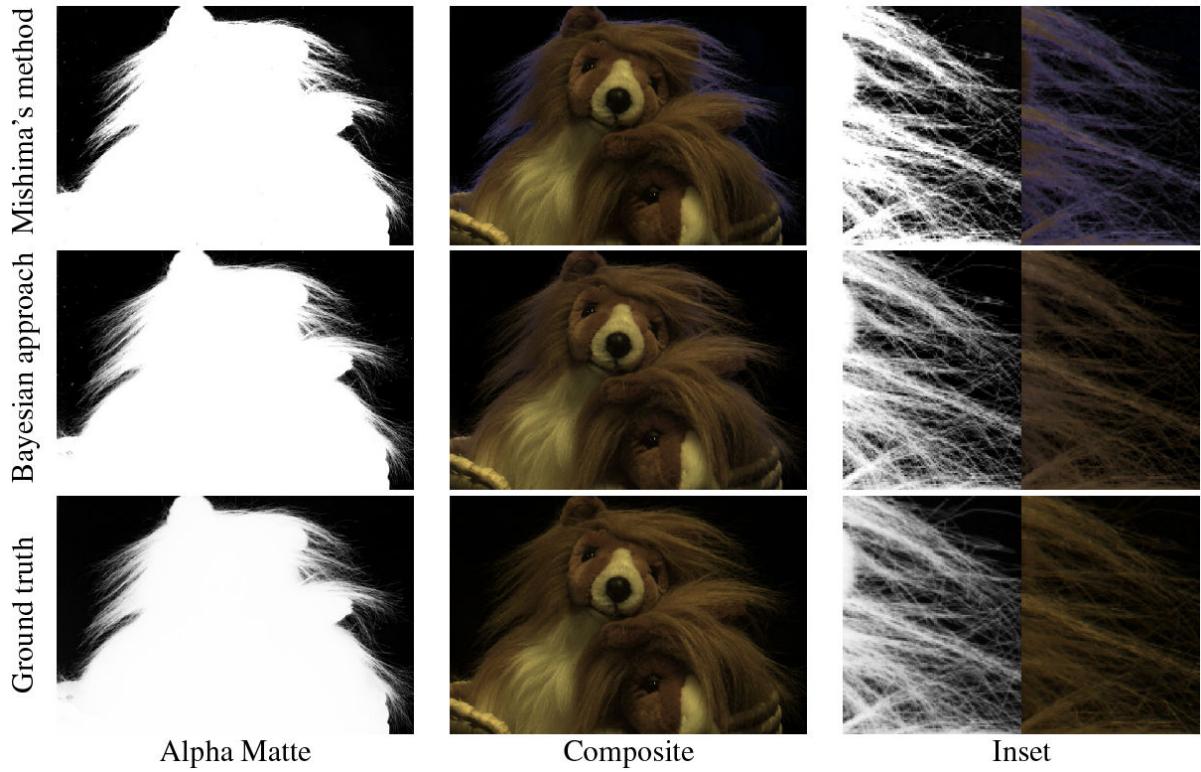


Figure 10.40 Blue-screen matting results (Chuang, Curless, Salesin *et al.* 2001) © 2001 IEEE. Mishima’s method produces visible blue spill (color fringing in the hair), while Chuang’s Bayesian matting approach produces accurate results.

10.4.1 Blue screen matting

Blue screen matting involves filming an actor (or object) in front of a constant colored background. While originally bright blue was the preferred color, bright green is now more commonly used (Wright 2006; Brinkmann 2008). Smith and Blinn (1996) discuss a number of techniques for blue screen matting, which are mostly described in patents rather than in the open research literature. Early techniques used linear combination of object color channels with user-tuned parameters to estimate the opacity α .

Chuang, Curless, Salesin *et al.* (2001) describe a newer technique called Mishima’s algorithm, which involves fitting two polyhedral surfaces (centered at the mean background color), separating the foreground and background color distributions and then measuring the relative distance of a novel color to these surfaces to estimate α (Figure 10.41e). While this technique works well in many studio settings, it can still suffer from *blue spill*, where translucent pixels around the edges of an object acquire some of the background blue coloration (Figure 10.40).

Two-screen matting. In their paper, Smith and Blinn (1996) also introduce an algorithm called *triangulation matting* that uses more than one known background color to over-constrain the equations required to estimate the opacity α and foreground color F .

For example, consider in the compositing equation (10.30) setting the background color to black, i.e., $B = 0$. The resulting composite image C is therefore equal to αF . Replacing the background color with a different known non-zero value B now results in

$$C - \alpha F = (1 - \alpha)B, \quad (10.31)$$

which is an overconstrained set of (color) equations for estimating α . In practice, B should be chosen so as not to saturate C and, for best accuracy, several values of B should be used. It is also important that colors be linearized before processing, which is the case for *all* image matting algorithms. Papers that generate ground truth alpha mattes for evaluation purposes normally use these techniques to obtain accurate matte estimates (Chuang, Curless, Salesin *et al.* 2001; Wang and Cohen 2007b; Levin, Acha, and Lischinski 2008; Rhemann, Rother, Rav-Acha *et al.* 2008; Rhemann, Rother, Wang *et al.* 2009).²² Exercise 10.8 has you do this as well.

Difference matting. A related approach when the background is irregular but known is called *difference matting* (Wright 2006; Brinkmann 2008). It is most commonly used when the actor or object is filmed against a static background, e.g., for office videoconferencing, person tracking applications (Toyama, Krumm, Brumitt *et al.* 1999), or to produce silhouettes for volumetric 3D reconstruction techniques (Section 11.6.2) (Szeliski 1993; Seitz and Dyer 1997; Seitz, Curless, Diebel *et al.* 2006). It can also be used with a panning camera where the background is composited from frames where the foreground has been removed using a *garbage matte* (Section 10.4.5) (Chuang, Agarwala, Curless *et al.* 2002). Another recent application is the detection of visual continuity errors in films, i.e., differences in the background when a shot is re-taken at later time (Pickup and Zisserman 2009).

In the case where the foreground and background motions can both be specified with parametric transforms, high-quality mattes can be extracted using a generalization of triangulation matting (Wexler, Fitzgibbon, and Zisserman 2002). When frames need to be processed independently, however, the results are often of poor quality (Figure 10.42). In such cases, using a pair of stereo cameras as input can dramatically improve the quality of the results (Criminisi, Cross, Blake *et al.* 2006; Yin, Criminisi, Winn *et al.* 2007).

10.4.2 Natural image matting

The most general version of image matting is when nothing is known about the background except, perhaps, for a rough segmentation of the scene into foreground, background, and unknown regions, which is known as the *trimap* (Figure 10.39b). Some recent techniques, however, relax this requirement and allow the user to just draw a few strokes or scribbles in the image, see Figures 10.45 and 10.46 (Wang and Cohen 2005; Wang, Agrawala, and Cohen 2007; Levin, Lischinski, and Weiss 2008; Rhemann, Rother, Rav-Acha *et al.* 2008; Rhemann, Rother, and Gelautz 2008). Fully automated single image matting results have also been reported (Levin, Acha, and Lischinski 2008; Singaraju, Rother, and Rhemann 2009). The survey paper by Wang and Cohen (2007a) has detailed descriptions and comparisons of all of these techniques, a selection of which are described briefly below.

²² See the alpha matting evaluation Web site at <http://alphamatting.com/>.

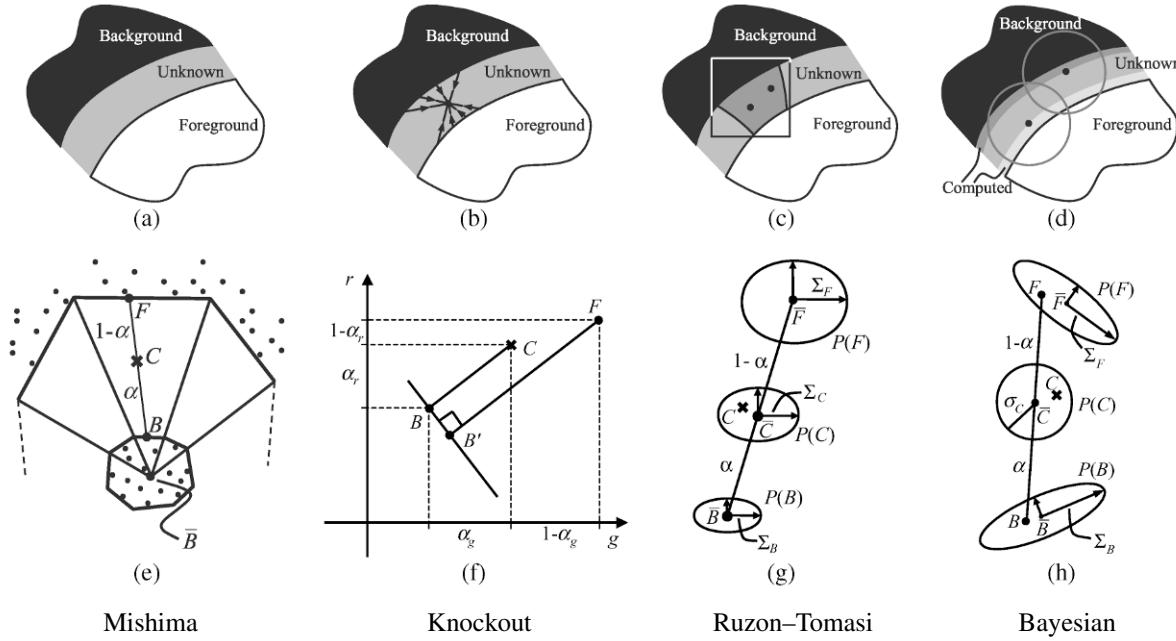


Figure 10.41 Image matting algorithms (Chuang, Curless, Salesin *et al.* 2001) © 2001 IEEE. Mishima’s algorithm models global foreground and background color distribution as polyhedral surfaces centered around the mean background (blue) color. Knockout uses a local color estimate of foreground and background for each pixel and computes α along each color axis. Ruzon and Tomasi’s algorithm locally models foreground and background colors and variances. Chuang *et al.*’s Bayesian matting approach computes a MAP estimate of (fractional) foreground color and opacity given the local foreground and background distributions.

A relatively simple algorithm for performing natural image matting is Knockout, as described by Chuang, Curless, Salesin *et al.* (2001) and illustrated in Figure 10.41f. In this algorithm, the nearest known foreground and background pixels (in image space) are determined and then blended with neighboring known pixels to produce a per-pixel foreground F and background B color estimate. The background color is then adjusted so that the measured color C lies on the line between F and B . Finally, opacity α is estimated on a per-channel basis, and the three estimates are combined based on per-channel color differences. (This is an approximation to the least squares solution for α .) Figure 10.42 shows that Knockout has problems when the background consists of more than one dominant local color.

More accurate matting results can be obtained if we treat the foreground and background colors as distributions sampled over some region (Figure 10.41g–h). Ruzon and Tomasi (2000) model local color distributions as mixtures of (uncorrelated) Gaussians and compute these models in strips. They then find the pairing of mixture components F and B that best describes the observed color C , compute the α as the relative distance between these means, and adjust the estimates of F and B so they are collinear with C .

Chuang, Curless, Salesin *et al.* (2001) and Hillman, Hannah, and Renshaw (2001) use full 3×3 color covariance matrices to model mixtures of correlated Gaussians, and compute estimates independently for each pixel. Matte extraction proceeds in strips starting from known color values growing into the unknown regions, so that recently computed F and B

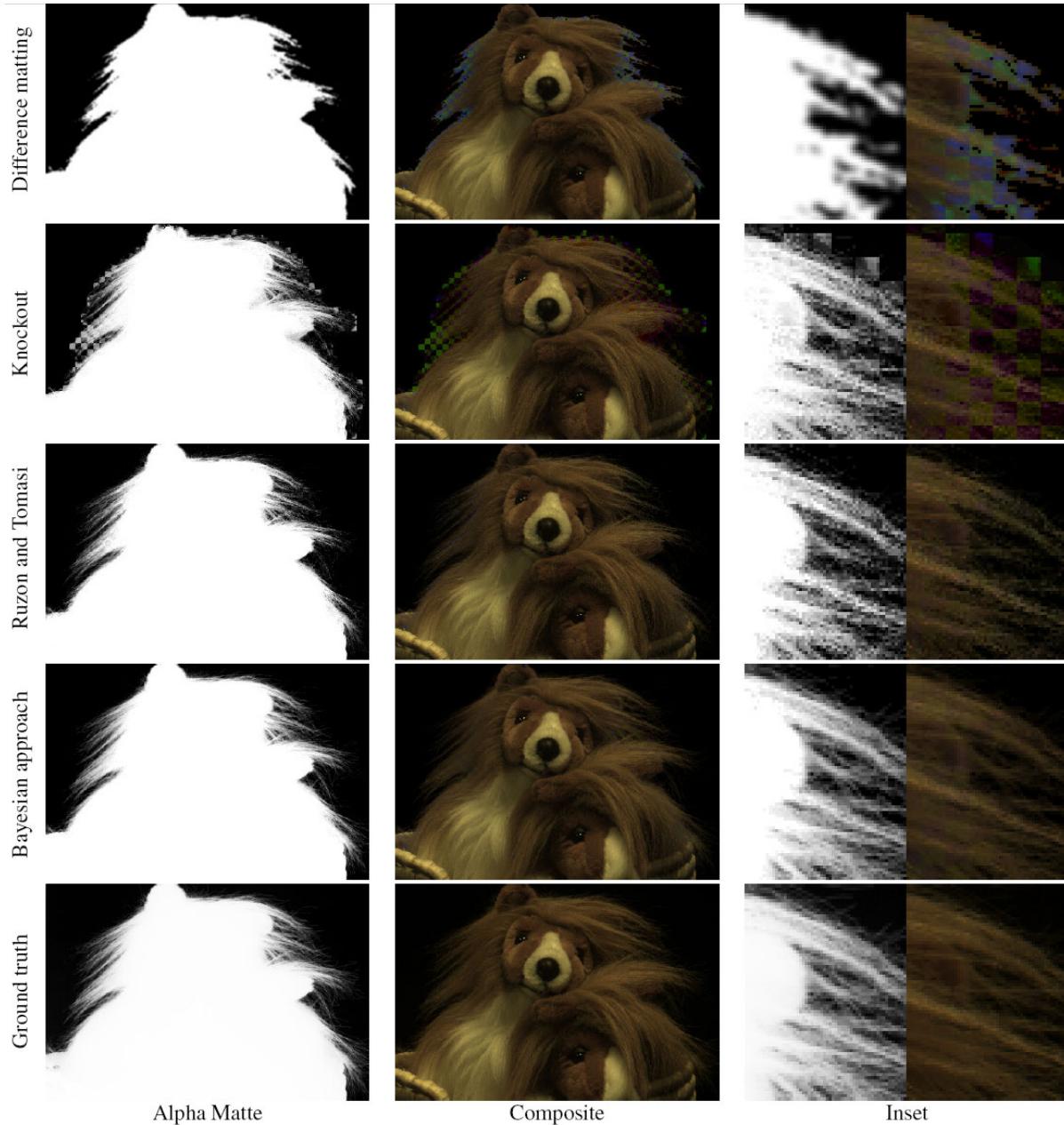


Figure 10.42 Natural image matting results (Chuang, Curless, Salesin *et al.* 2001) © 2001 IEEE. Difference matting and Knockout both perform poorly on this kind of background, while the more recent natural image matting techniques perform well. Chuang *et al.*'s results are slightly smoother and closer to the ground truth.

colors can be used in later stages.

To estimate the most likely value of an unknown pixel's opacity and (unmixed) foreground and background colors, Chuang *et al.* use a fully Bayesian formulation that maximizes

$$P(F, B, \alpha | C) = P(C|F, B, \alpha)P(F)P(B)P(\alpha)/P(C). \quad (10.32)$$

This is equivalent to minimizing the negative log likelihood

$$L(F, B, \alpha | C) = L(C|F, B, \alpha) + L(F) + L(B) + L(\alpha) \quad (10.33)$$

(dropping the $L(C)$ term since it is constant).

Let us examine each of these terms in turn. The first, $L(C|F, B, \alpha)$, is the likelihood that pixel color C was observed given values for the unknowns (F, B, α) . If we assume Gaussian noise in our observation with variance σ_C^2 , this negative log likelihood (data term) is

$$L(C) = 1/2 \|C - [\alpha F + (1 - \alpha)B]\|^2 / \sigma_C^2, \quad (10.34)$$

as illustrated in Figure 10.41h.

The second term, $L(F)$, corresponds to the likelihood that a particular foreground color F comes from the mixture of Gaussians distribution. After partitioning the sample foreground colors into clusters, a weighted mean and covariance is computed, where the weights are proportional to a given foreground pixel's opacity and distance from the unknown pixel. The negative log likelihood for each cluster is thus given by

$$L(F) = (F - \bar{F})^T \Sigma_F^{-1} (F - \bar{F}). \quad (10.35)$$

A similar method is used to estimate unknown background color distributions. If the background is already known, i.e., for blue screen or difference matting applications, its measured color value and variance are used instead.

An alternative to modeling the foreground and background color distributions as mixtures of Gaussians is to keep around the original color samples and to compute the most likely pairings that explain the observed color C (Wang and Cohen 2005, 2007b). These techniques are described in more detail in (Wang and Cohen 2007a).

In their Bayesian matting paper, Chuang, Curless, Salesin *et al.* (2001) assume a constant (non-informative) distribution for $L(\alpha)$. More recent papers assume this distribution to be more peaked around 0 and 1, or sometimes use Markov random fields (MRFs) to define a global correlated prior on $P(\alpha)$ (Wang and Cohen 2007a).

To compute the most likely estimates for (F, B, α) , the Bayesian matting algorithm alternates between computing (F, B) and α , since each of these problems is quadratic and hence can be solved as a small linear system. When several color clusters are estimated, the most likely pairing of foreground and background color clusters is used.

Bayesian image matting produces results that improve on the original natural image matting algorithm by Ruzon and Tomasi (2000), as can be seen in Figure 10.42. However, compared to more recent techniques (Wang and Cohen 2007a), its performance is not as good for complex background or inaccurate trimaps (Figure 10.44).

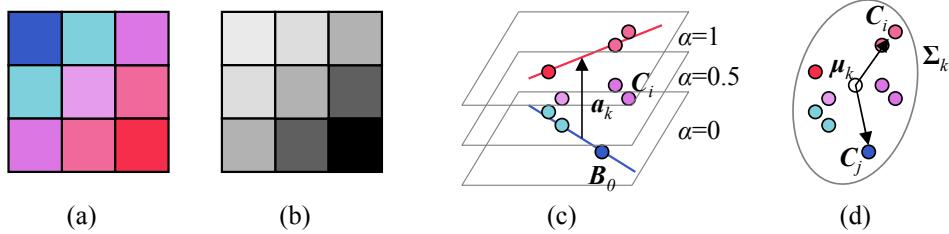


Figure 10.43 Color line matting (Levin, Lischinski, and Weiss 2008): (a) local 3×3 patch of colors; (b) potential assignment of α values; (c) foreground and background color lines, the vector \mathbf{a}_k joining their closest points of intersection, and the family of parallel planes of constant α values, $\alpha_i = \mathbf{a}_k \cdot (\mathbf{C}_i - \mathbf{B}_0)$; (d) a scatter plot of sample colors and the deviations from the mean μ_k for two sample colors \mathbf{C}_i and \mathbf{C}_j .

10.4.3 Optimization-based matting

An alternative to estimating each pixel’s opacity and foreground color independently is to use global optimization to compute a matte that takes into account correlations between neighboring α values. Two examples of this are border matting in the GrabCut interactive segmentation system (Rother, Kolmogorov, and Blake 2004) and Poisson Matting (Sun, Jia, Tang *et al.* 2004).

Border matting first dilates the region around the binary segmentation produced by GrabCut (Section 5.5) and then solves for a sub-pixel boundary location Δ and a blur width σ for every point along the boundary (Figure 10.38). Smoothness in these parameters along the boundary is enforced using regularization and the optimization is performed using dynamic programming. While this technique can obtain good results for smooth boundaries, such as a person’s face, it has difficulty with fine details, such as hair.

Poisson matting (Sun, Jia, Tang *et al.* 2004) assumes a known foreground and background color for each pixel in the trimap (as with Bayesian matting). However, instead of independently estimating each α value, it assumes that the gradient of the alpha matte and the gradient of the color image are related by

$$\nabla \alpha = \frac{F - B}{\|F - B\|^2} \cdot \nabla C, \quad (10.36)$$

which can be derived by taking gradients of both sides of (10.30) and assuming that the foreground and background vary slowly. The per-pixel gradient estimates are then integrated into a continuous $\alpha(\mathbf{x})$ field using the regularization (least squares) technique first described in Section 3.7.1 (3.100) and subsequently used in Poisson blending (Section 9.3.4, 9.44) and gradient-based dynamic range compression mapping (Section 10.2.1, 10.19). This technique works well when good foreground and background color estimates are available and these colors vary slowly.

Instead of computing per-pixel foreground and background colors, Levin, Lischinski, and Weiss (2008) assume only that these color distribution can locally be well approximated as mixtures of two colors, which is known as the *color line model* (Figure 10.43a–c). Under this assumption, a closed-form estimate for α at each pixel i in a (say, 3×3) window W_k is given by

$$\alpha_i = \mathbf{a}_k \cdot (\mathbf{C}_i - \mathbf{B}_0) = \mathbf{a}_k \cdot \mathbf{C} + b_k, \quad (10.37)$$

where \mathbf{C}_i is the pixel color treated as a three-vector, \mathbf{B}_0 is any pixel along the background color line, and \mathbf{a}_k is the vector joining the two closest points on the foreground and background color lines, as shown in Figure 10.43c. (Note that the geometric derivation shown in this figure is an alternative to the algebraic derivation presented by Levin, Lischinski, and Weiss (2008).) Minimizing the deviations of the alpha values α_i from their respective color line models (10.37) over all overlapping windows W_k in the image gives rise to the cost

$$E_\alpha = \sum_k \left(\sum_{i \in W_k} (\alpha_i - \mathbf{a}_k \cdot \mathbf{C}_i - b_k)^2 + \epsilon \|\mathbf{a}_k\| \right), \quad (10.38)$$

where the ϵ term is used to regularize the value of \mathbf{a}_k in the case where the two color distributions overlap (i.e., in constant α regions).

Because this formula is quadratic in the unknowns $\{(\mathbf{a}_k, b_k)\}$, they can be eliminated inside each window W_k , leading to a final energy

$$E_\alpha = \boldsymbol{\alpha}^T \mathbf{L} \boldsymbol{\alpha}, \quad (10.39)$$

where the entries in the \mathbf{L} matrix are given by

$$L_{ij} = \sum_{k:i \in W_k \wedge j \in W_k} \left(\delta_{ij} - \frac{1}{M} \left(1 + (\mathbf{C}_i - \boldsymbol{\mu}_k)^T \hat{\Sigma}_k^{-1} (\mathbf{C}_j - \boldsymbol{\mu}_k) \right) \right), \quad (10.40)$$

where $M = |W_k|$ is the number of pixels in each (overlapping) window, $\boldsymbol{\mu}_k$ is the mean color of the pixels in window W_k , and $\hat{\Sigma}_k$ is the 3×3 covariance of the pixel colors plus $\epsilon/M \mathbf{I}$.

Figure 10.43d shows the intuition behind the entries in this affinity matrix, which is called the *matting Laplacian*. Note how when two pixels \mathbf{C}_i and \mathbf{C}_j in W_k point in opposite directions away from the mean $\boldsymbol{\mu}_k$, their weighted dot product is close to -1 , and so their affinity becomes close to 0 . Pixels close to each other in color space (and hence with similar expected α values) will have affinities close to $-2/M$.

Minimizing the quadratic energy (10.39) constrained by the known values of $\alpha = \{0, 1\}$ at scribbles only requires the solution of a sparse set of linear equations, which is why the authors call their technique a *closed-form solution* to natural image matting. Once α has been computed, the foreground and background colors are estimated using a least squares minimization of the compositing equation (10.30) regularized with a spatially varying first-order smoothness,

$$E_{B,F} = \sum_i \|C_i - [\alpha + F_i + (1 - \alpha_i)B_i]\|^2 + \lambda |\nabla \alpha_i| (\|\nabla F_i\|^2 + \|\nabla B_i\|^2), \quad (10.41)$$

where the $|\nabla \alpha_i|$ weight is applied separately for the x and y components of the F and B derivatives (Levin, Lischinski, and Weiss 2008).

Laplacian (closed-form) matting is just one of many optimization-based techniques surveyed and compared by Wang and Cohen (2007a). Some of these techniques use alternative formulations for the affinities or smoothness terms on the α matte, alternative estimation techniques such as belief propagation, or alternative representations (e.g., local histograms) for modeling local foreground and background color distributions (Wang and Cohen 2005, 2007b,c). Some of these techniques also provide real-time results as the user draws a contour

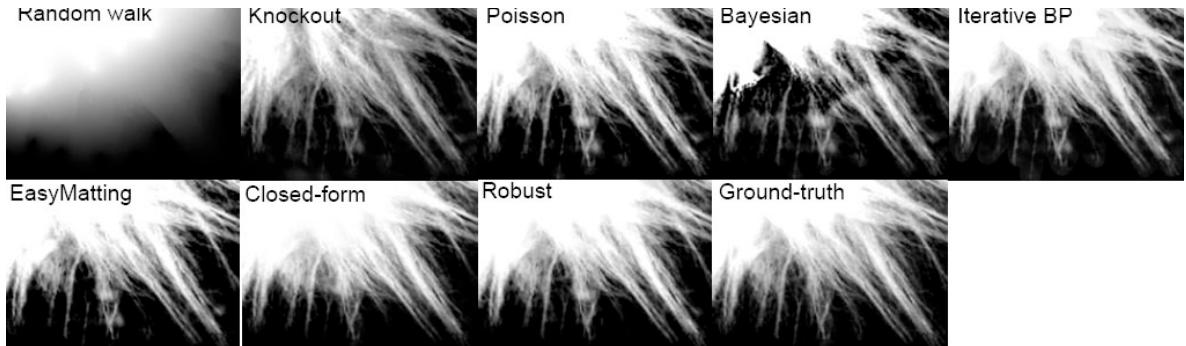


Figure 10.44 Comparative matting results for a medium accuracy trimap. Wang and Cohen (2007a) describe the individual techniques being compared.

line or sparse set of scribbles (Wang, Agrawala, and Cohen 2007; Rhemann, Rother, Rav-Acha *et al.* 2008) or even pre-segment the image into a small number of mattes that the user can select with simple clicks (Levin, Acha, and Lischinski 2008).

Figure 10.44 shows the results of running a number of the surveyed algorithms on a region of toy animal fur where a trimap has been specified, while Figure 10.45 shows results for techniques that can produce mattes with only a few scribbles as input. Figure 10.46 shows a result for an even more recent algorithm (Rhemann, Rother, Rav-Acha *et al.* 2008) that claims to outperform all of the techniques surveyed by Wang and Cohen (2007a).

Pasting. Once a matte has been pulled from an image, it is usually composited directly over the new background, unless the seams between the cutout and background regions are to be hidden, in which case Poisson blending (Pérez, Gangnet, and Blake 2003) can be used (Section 9.3.4).

In the latter case, it is helpful if the matte boundary passes through regions that either have little texture or look similar in the old and new images. Papers by Jia, Sun, Tang *et al.* (2006) and Wang and Cohen (2007c) explain how to do this.

10.4.4 Smoke, shadow, and flash matting

In addition to matting out solid objects with fractional boundaries, it is also possible to matte out translucent media such as smoke (Chuang, Agarwala, Curless *et al.* 2002). Starting with a video sequence, each pixel is modeled as a linear combination of its (unknown) background color and a constant foreground (smoke) color that is common to all pixels. Voting in color space is used to estimate this foreground color and the distance along each color line is used to estimate the per-pixel temporally varying alpha (Figure 10.47).

Extracting and re-inserting shadows is also possible using a related technique (Chuang, Goldman, Curless *et al.* 2003). Here, instead of assuming a constant foreground color, each pixel is assumed to vary between its fully lit and fully shadowed colors, which can be estimated by taking (robust) minimum and maximum values over time as a shadow passes over the scene (Exercise 10.9). The resulting fractional *shadow matte* can be used to re-project the shadow into a new scene. If the destination scene has a non-planar geometry, it can be

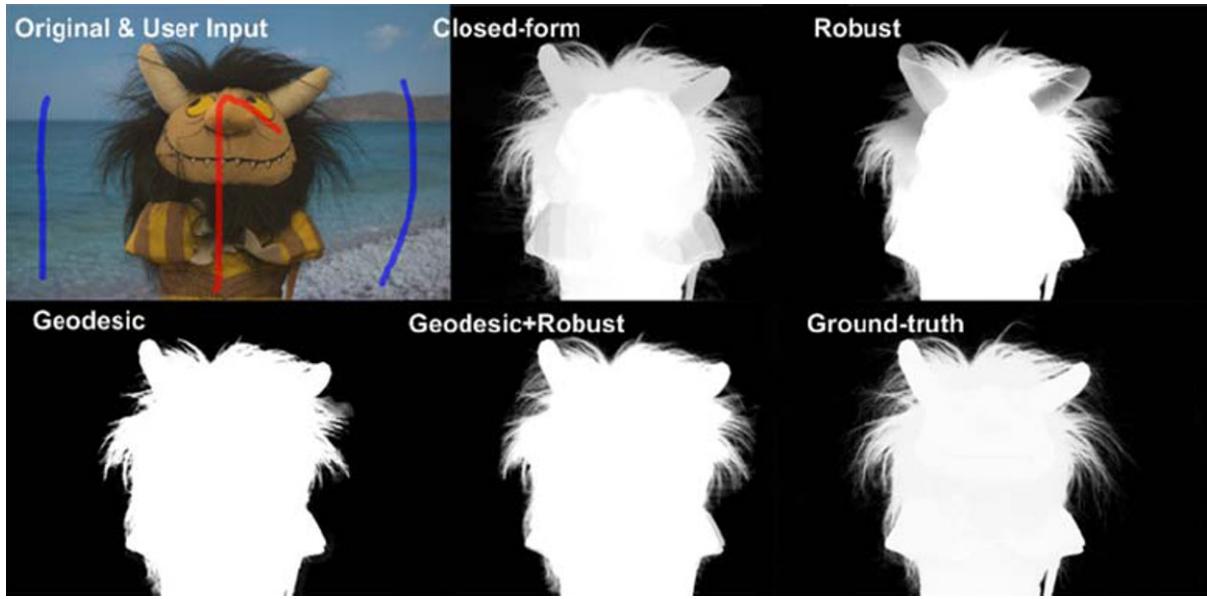


Figure 10.45 Comparative matting results with scribble-based inputs. Wang and Cohen (2007a) describe the individual techniques being compared.



Figure 10.46 Stroke-based segmentation result (Rhemann, Rother, Rav-Acha *et al.* 2008) © 2008 IEEE.



Figure 10.47 Smoke matting (Chuang, Agarwala, Curless *et al.* 2002) © 2002 ACM: (a) input video frame; (b) after removing the foreground object; (c) estimated alpha matte; (d) insertion of new objects into the background.

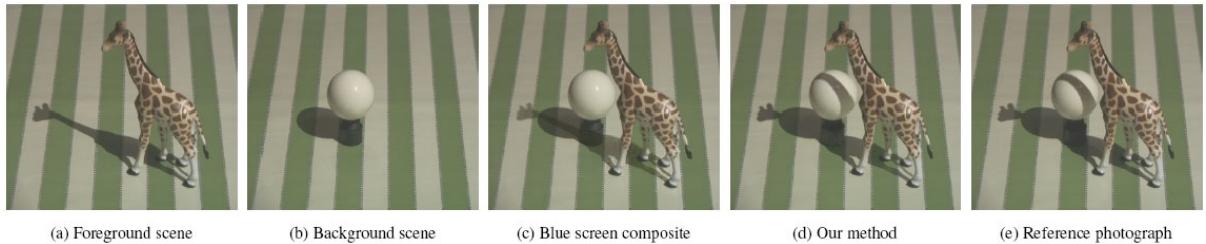


Figure 10.48 Shadow matting (Chuang, Goldman, Curless *et al.* 2003) © 2003 ACM. Instead of simply darkening the new scene with the shadow (c), shadow matting correctly dims the lit scene with the new shadow and drapes the shadow over 3D geometry (d).

scanned by waving a straight stick shadow across the scene. The new shadow matte can then be warped with the computed deformation field to have it drape correctly over the new scene (Figure 10.48).

The quality and reliability of matting algorithms can also be enhanced using more sophisticated acquisition systems. For example, taking a flash and non-flash image pair supports the reliable extraction of foreground mattes, which show up as regions of large illumination change between the two images (Sun, Li, Kang *et al.* 2006). Taking simultaneous video streams focused at different distances (McGuire, Matusik, Pfister *et al.* 2005) or using multi-camera arrays (Joshi, Matusik, and Avidan 2006) are also good approaches to producing high-quality mattes. These techniques are described in more detail in (Wang and Cohen 2007a).

Lastly, photographing a refractive object in front of a number of patterned backgrounds allows the object to be placed in novel 3D environments. These environment matting techniques (Zongker, Werner, Curless *et al.* 1999; Chuang, Zongker, Hindorff *et al.* 2000) are discussed in Section 13.4.

10.4.5 Video matting

While regular single-frame matting techniques such as blue or green screen matting (Smith and Blinn 1996; Wright 2006; Brinkmann 2008) can be applied to video sequences, the presence of moving objects can sometimes make the matting process easier, as portions of the background may get revealed in preceding or subsequent frames.

Chuang, Agarwala, Curless *et al.* (2002) describe a nice approach to this *video matting* problem, where foreground objects are first removed using a conservative *garbage matte* and the resulting *background plates* are aligned and composited to yield a high-quality background estimate. They also describe how trimaps drawn at sparse keyframes can be interpolated to in-between frames using bi-direction optic flow. Alternative approaches to video matting, such as rotoscoping, which involves drawing and tracking curves in video sequences (Agarwala, Hertzmann, Seitz *et al.* 2004), are discussed in the matting survey paper by Wang and Cohen (2007a).

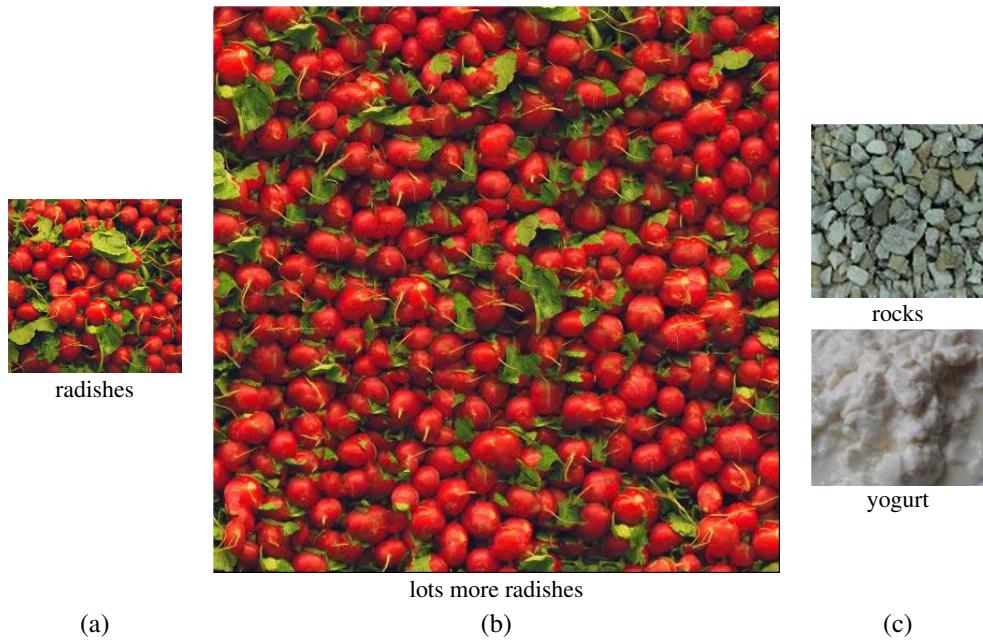


Figure 10.49 Texture synthesis: (a) given a small patch of texture, the task is to synthesize (b) a similar-looking larger patch; (c) other semi-structured textures that are challenging to synthesize. (Images courtesy of Alyosha Efros.)

10.5 Texture analysis and synthesis

While texture analysis and synthesis may not at first seem like computational photography techniques, they are, in fact, widely used to repair defects, such as small holes, in images or to create non-photorealistic painterly renderings from regular photographs.

The problem of texture synthesis can be formulated as follows: given a small sample of a “texture” (Figure 10.49a), generate a larger similar-looking image (Figure 10.49b). As you can imagine, for certain sample textures, this problem can be quite challenging.

Traditional approaches to texture analysis and synthesis try to match the spectrum of the source image while generating shaped noise. Matching the frequency characteristics, which is equivalent to matching spatial correlations, is in itself not sufficient. The distributions of the responses at different frequencies must also match. Heeger and Bergen (1995) develop an algorithm that alternates between matching the histograms of multi-scale (steerable pyramid) responses and matching the final image histogram. Portilla and Simoncelli (2000) improve on this technique by also matching pairwise statistics across scale and orientations. De Bonet (1997) uses a coarse-to-fine strategy to find locations in the source texture with a similar *parent structure*, i.e., similar multi-scale oriented filter responses, and then randomly chooses one of these matching locations as the current sample value.

More recent texture synthesis algorithms sequentially generate texture pixels by looking for neighborhoods in the source texture that are similar to the currently synthesized image (Efros and Leung 1999). Consider the (as yet) unknown pixel p in the partially constructed texture on the left side of Figure 10.50. Since some of its neighboring pixels have been

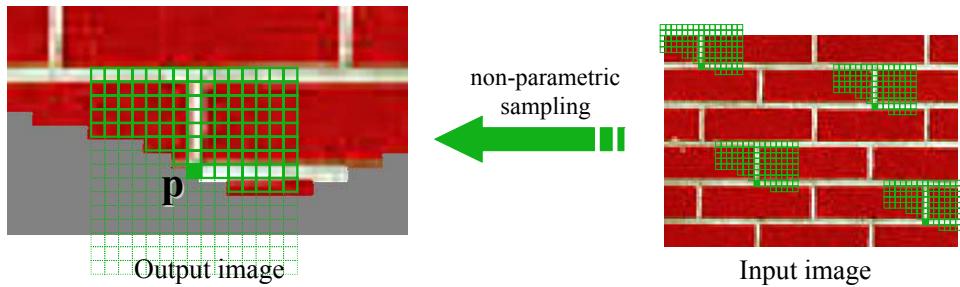


Figure 10.50 Texture synthesis using non-parametric sampling (Efros and Leung 1999). The value of the newest pixel p is randomly chosen from similar local (partial) patches in the source texture (input image). (Figure courtesy of Alyosha Efros.)

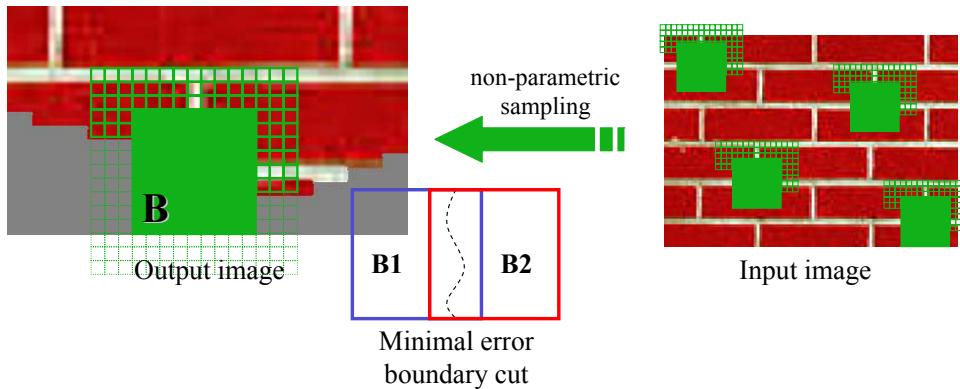


Figure 10.51 Texture synthesis by image quilting (Efros and Freeman 2001). Instead of generating a single pixel at a time, larger blocks are copied from the source texture. The transitions in the overlap regions between the selected blocks are then optimized using dynamic programming. (Figure courtesy of Alyosha Efros.)

already been synthesized, we can look for similar partial neighborhoods in the sample texture image on the right and randomly select one of these as the new value of p . This process can be repeated down the new image either in a raster fashion or by scanning around the periphery (“onion peeling”) when filling holes, as discussed in (Section 10.5.1). In their actual implementation, Efros and Leung (1999) find the most similar neighborhood and then include all other neighborhoods within a $d = (1 + \epsilon)$ distance, with $\epsilon = 0.1$. They also optionally weight the random pixel selections by the similarity metric d .

To accelerate this process and improve its visual quality, Wei and Levoy (2000) extend this technique using a coarse-to-fine generation process, where coarser levels of the pyramid, which have already been synthesized, are also considered during the matching (De Bonet 1997). To accelerate the nearest neighbor finding, tree-structured vector quantization is used.

Efros and Freeman (2001) propose an alternative acceleration and visual quality improvement technique. Instead of synthesizing a single pixel at a time, overlapping square blocks are selected using similarity with previously synthesized regions (Figure 10.51). Once the appropriate blocks have been selected, the seam between newly overlapping blocks is determined using dynamic programming. (Full graph cut seam selection is not required, since only

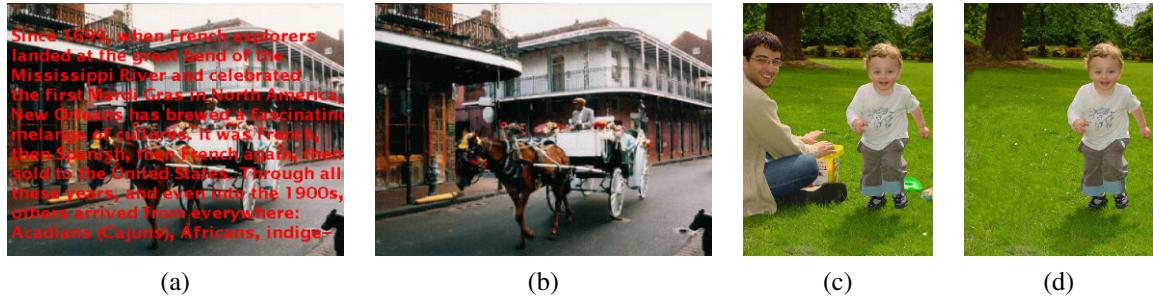


Figure 10.52 Image inpainting (hole filling): (a–b) propagation along isophote directions (Bertalmio, Sapiro, Caselles *et al.* 2000) © 2000 ACM; (c–d) exemplar-based inpainting with confidence-based filling order (Criminisi, Pérez, and Toyama 2004).

one seam location per row is needed for a vertical boundary.) Because this process involves selecting small patches and then stitching them together, Efros and Freeman (2001) call their system *image quilting*. Komodakis and Tziritas (2007b) present an MRF-based version of this block synthesis algorithm that uses a new, efficient version of loopy belief propagation they call “Priority-BP”.

10.5.1 Application: Hole filling and inpainting

Filling holes left behind when objects or defects are excised from photographs, which is known as *inpainting*, is one of the most common applications of texture synthesis. Such techniques are used not only to remove unwanted people or interlopers from photographs (King 1997) but also to fix small defects in old photos and movies (*scratch removal*) or to remove wires holding props or actors in mid-air during filming (*wire removal*). Bertalmio, Sapiro, Caselles *et al.* (2000) solve the problem by propagating pixel values along isophote (constant-value) directions interleaved with some anisotropic diffusion steps (Figure 10.52a–b). Telea (2004) develops a faster technique that uses the fast marching method from level sets (Section 5.1.4). However, these techniques will not hallucinate texture in the missing regions. Bertalmio, Vese, Sapiro *et al.* (2003) augment their earlier technique by adding synthetic texture to the infilled regions.

The example-based (non-parametric) texture generation techniques discussed in the previous section can also be used by filling the holes from the outside in (the “onion-peel” ordering). However, this approach may fail to propagate strong oriented structures. Criminisi, Pérez, and Toyama (2004) use exemplar-based texture synthesis where the order of synthesis is determined by the strength of the gradient along the region boundary (Figures 10.1d and 10.52c–d). Sun, Yuan, Jia *et al.* (2004) present a related approach where the user draws interactive lines to indicate where structures should be preferentially propagated. Additional techniques related to these approaches include those developed by Drori, Cohen-Or, and Yeshurun (2003), Kwatra, Schödl, Essa *et al.* (2003), Kwatra, Essa, Bobick *et al.* (2005), Wilczkowiak, Brostow, Tordoff *et al.* (2005), Komodakis and Tziritas (2007b), and Wexler, Shechtman, and Irani (2007).

Most hole filling algorithms borrow small pieces of the original image to fill in the holes. When a large database of source images is available, e.g., when images are taken from a

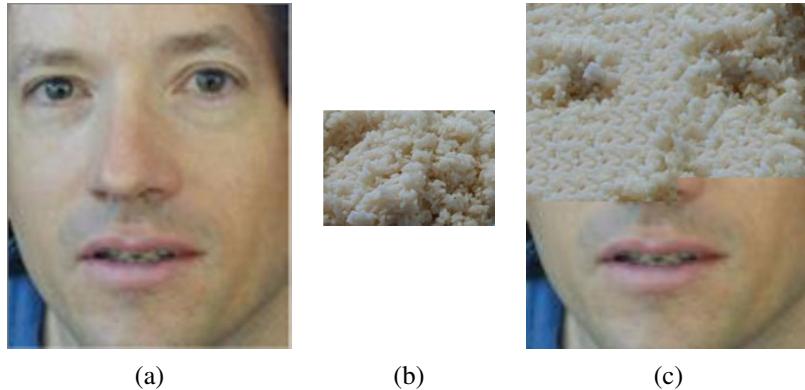


Figure 10.53 Texture transfer (Efros and Freeman 2001) © 2001 ACM: (a) reference (target) image; (b) source texture; (c) image (partially) rendered using the texture.

photo sharing site or the Internet, it is sometimes possible to copy a single contiguous image region to fill the hole. Hays and Efros (2007) present such a technique, which uses image context and boundary compatibility to select the source image, which is then blended with the original (holey) image using graph cuts and Poisson blending. This technique is discussed in more detail in Section 14.4.4 and Figure 14.46.

10.5.2 Application: Non-photorealistic rendering

Two more applications of the exemplar-based texture synthesis ideas are texture transfer (Efros and Freeman 2001) and image analogies (Hertzmann, Jacobs, Oliver *et al.* 2001), which are both examples of non-photorealistic rendering (Gooch and Gooch 2001).

In addition to using a source texture image, texture transfer also takes a reference (or target) image, and tries to match certain characteristics of the target image with the newly synthesized image. For example, the new image being rendered in Figure 10.53c not only tries to satisfy the usual similarity constraints with the source texture in Figure 10.53b, but it also tries to match the luminance characteristics of the reference image. Efros and Freeman (2001) mention that blurred image intensities or local image orientation angles are alternative quantities that could be matched.

Hertzmann, Jacobs, Oliver *et al.* (2001) formulate the following problem:

Given a pair of images A and A' (the unfiltered and filtered source images, respectively), along with some additional unfiltered target image B , synthesize a new filtered target image B' such that

$$A : A' :: B : B'.$$

Instead of having the user program a certain non-photorealistic rendering effect, it is sufficient to supply the system with examples of before and after images, and let the system synthesize the novel image using exemplar-based synthesis, as shown in Figure 10.54.

The algorithm used to solve image analogies proceeds in a manner analogous to the texture synthesis algorithms of (Efros and Leung 1999; Wei and Levoy 2000). Once Gaussian pyramids have been computed for all of the source and reference images, the algorithm



Figure 10.54 Image analogies (Hertzmann, Jacobs, Oliver *et al.* 2001) © 2001 ACM. Given an example pair of a source image *A* and its rendered (filtered) version *A'*, generate the rendered version *B'* from another unfiltered source image *B*.

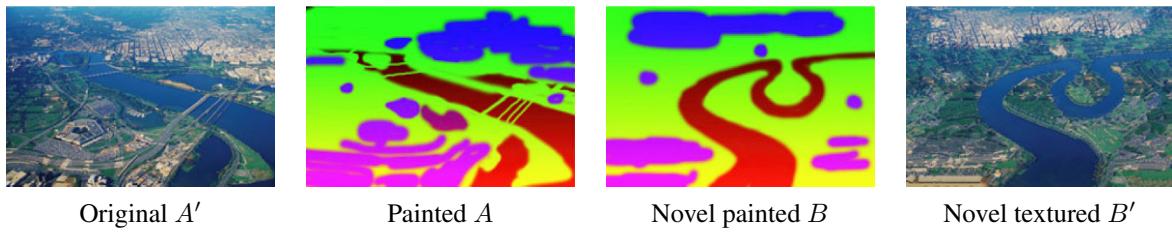


Figure 10.55 Texture-by-numbers (Hertzmann, Jacobs, Oliver *et al.* 2001) © 2001 ACM. Given a textured image *A'* and a hand-labeled (painted) version *A*, synthesize a new image *B'* given just the painted version *B*.

looks for neighborhoods in the source filtered pyramids generated from *A'* that are similar to the partially constructed neighborhood in *B'*, while at the same time having similar multi-resolution appearances at corresponding locations in *A* and *B*. As with texture transfer, appearance characteristics can include not only (blurred) color or luminance values but also orientations.

This general framework allows image analogies to be applied to a variety of rendering tasks. In addition to exemplar-based non-photorealistic rendering, image analogies can be used for traditional texture synthesis, super-resolution, and texture transfer (using the same textured image for both *A* and *A'*). If only the filtered (rendered) image *A'* is available, as is the case with paintings, the missing reference image *A* can be hallucinated using a smart (edge preserving) blur operator. Finally, it is possible to train a system to perform *texture-by-numbers* by manually painting over a natural image with pseudocolors corresponding to pixels' semantic meanings, e.g., water, trees, and grass (Figure 10.55a–b). The resulting system can then convert a novel sketch into a fully rendered synthetic photograph (Figure 10.55c–d). In more recent work, Cheng, Vishwanathan, and Zhang (2008) add ideas from image quilting (Efros and Freeman 2001) and MRF inference (Komodakis, Tziritas, and Paragios 2008) to the basic image analogies algorithm, while Ramanarayanan and Bala (2007) recast this process as energy minimization, which means it can also be viewed as a conditional random field (Section 3.7.2), and devise an efficient algorithm to find a good minimum.

More traditional filtering and feature detection techniques can also be used for non-photorealistic rendering.²³ For example, pen-and-ink illustration (Winkenbach and Salesin

²³ For a good selection of papers, see the Symposia on Non-Photorealistic Animation and Rendering (NPAR) at <http://www.npar.org/>.



Figure 10.56 Non-photorealistic abstraction of photographs: (a) DeCarlo and Santella (2002) © 2002 ACM and (b) Farbman, Fattal, Lischinski *et al.* (2008) © 2008 ACM.

1994) and painterly rendering techniques (Litwinowicz 1997) use local color, intensity, and orientation estimates as an input to their procedural rendering algorithms. Techniques for stylizing and simplifying photographs and video (DeCarlo and Santella 2002; Winnemöller, Olsen, and Gooch 2006; Farbman, Fattal, Lischinski *et al.* 2008), as in Figure 10.56, use combinations of edge-preserving blurring (Section 3.3.1) and edge detection and enhancement (Section 4.2.3).

10.6 Additional reading

A good overview of computational photography can be found in the book by Raskar and Tumblin (2010), survey articles by Nayar (2006), Cohen and Szeliski (2006), Levoy (2006),Debevec (2006), and Hayes (2008), as well as two special journal issues edited by Bimber (2006) and Durand and Szeliski (2007). Notes from the courses on computational photography mentioned at the beginning of this chapter are another great source of material and references.²⁴

The sub-field of high dynamic range imaging has its own book discussing research in this area (Reinhard, Ward, Pattanaik *et al.* 2005), as well as some books describing related photographic techniques (Freeman 2008; Gulbins and Gulbins 2009). Algorithms for calibrating the radiometric response function of a camera can be found in articles by Mann and Picard (1995), Debevec and Malik (1997), and Mitsunaga and Nayar (1999).

The subject of tone mapping is treated extensively in (Reinhard, Ward, Pattanaik *et al.* 2005). Representative papers from the large volume of literature on this topic include those by Tumblin and Rushmeier (1993), Larson, Rushmeier, and Piatko (1997), Pattanaik, Ferwerda, Fairchild *et al.* (1998), Tumblin and Turk (1999), Durand and Dorsey (2002), Fattal, Lischinski, and Werman (2002), Reinhard, Stark, Shirley *et al.* (2002), Lischinski, Farbman, Uyttendaele *et al.* (2006b), and Farbman, Fattal, Lischinski *et al.* (2008).

²⁴ MIT 6.815/6.865, <http://stellar.mit.edu/S/course/6/sp08/6.815/materials.html>, CMU 15-463, http://graphics.cs.cmu.edu/courses/15-463/2008_fall/, Stanford CS 448A, <http://graphics.stanford.edu/courses/cs448a-08-spring/>, and SIGGRAPH courses, <http://web.media.mit.edu/~raskar/photo/>.

The literature on super-resolution is quite extensive (Chaudhuri 2001; Park, Park, and Kang 2003; Capel and Zisserman 2003; Capel 2004; van Ouwerkerk 2006). The term super-resolution usually describes techniques for aligning and merging multiple images to produce higher-resolution composites (Keren, Peleg, and Brada 1988; Irani and Peleg 1991; Cheeseman, Kanefsky, Hanson *et al.* 1993; Mann and Picard 1994; Chiang and Boult 1996; Bascle, Blake, and Zisserman 1996; Capel and Zisserman 1998; Smelyanskiy, Cheeseman, Maluf *et al.* 2000; Capel and Zisserman 2000; Pickup, Capel, Roberts *et al.* 2009; Gulbins and Gulbins 2009). However, single-image super-resolution techniques have also been developed (Freeman, Jones, and Pasztor 2002; Baker and Kanade 2002; Fattal 2007).

A good survey on image matting is given by Wang and Cohen (2007a). Representative papers, which include extensive comparisons with previous work, include those by Chuang, Curless, Salesin *et al.* (2001), Wang and Cohen (2007b), Levin, Acha, and Lischinski (2008), Rhemann, Rother, Rav-Acha *et al.* (2008), and Rhemann, Rother, Wang *et al.* (2009).

The literature on texture synthesis and hole filling includes traditional approaches to texture synthesis, which try to match image statistics between source and destination images (Heeger and Bergen 1995; De Bonet 1997; Portilla and Simoncelli 2000), as well as newer approaches, which search for matching neighborhoods or patches inside the source sample (Efros and Leung 1999; Wei and Levoy 2000; Efros and Freeman 2001). In a similar vein, traditional approaches to hole filling involve the solution of local variational (smooth continuation) problems (Bertalmio, Sapiro, Caselles *et al.* 2000; Bertalmio, Vese, Sapiro *et al.* 2003; Telea 2004). More recent techniques use data-driven texture synthesis approaches (Drori, Cohen-Or, and Yeshurun 2003; Kwatra, Schödl, Essa *et al.* 2003; Criminisi, Pérez, and Toyama 2004; Sun, Yuan, Jia *et al.* 2004; Kwatra, Essa, Bobick *et al.* 2005; Wilczkowiak, Brostow, Tordoff *et al.* 2005; Komodakis and Tziritas 2007b; Wexler, Shechtman, and Irani 2007).

10.7 Exercises

Ex 10.1: Radiometric calibration Implement one of the multi-exposure radiometric calibration algorithms described in Section 10.2 (Debevec and Malik 1997; Mitsunaga and Nayar 1999; Reinhard, Ward, Pattanaik *et al.* 2005). This calibration will be useful in a number of different applications, such as stitching images or stereo matching with different exposures and shape from shading.

1. Take a series of bracketed images with your camera on a tripod. If your camera has an automatic exposure bracketing (AEB) mode, taking three images may be sufficient to calibrate most of your camera's dynamic range, especially if your scene has a lot of bright and dark regions. (Shooting outdoors or through a window on a sunny day is best.)
2. If your images are not taken on a tripod, first perform a global alignment (similarity transform).
3. Estimate the radiometric response function using one of the techniques cited above.
4. Estimate the high dynamic range radiance image by selecting or blending pixels from

different exposures (Debevec and Malik 1997; Mitsunaga and Nayar 1999; Eden, Uyttendaele, and Szeliski 2006).

5. Repeat your calibration experiments under different conditions, e.g., indoors under incandescent light, to get a sense for the range of color balancing effects that your camera imposes.
6. If your camera supports RAW and JPEG mode, calibrate both sets of images simultaneously and to each other (the radiance at each pixel will correspond). See if you can come up with a model for what your camera does, e.g., whether it treats color balance as a diagonal or full 3×3 matrix multiply, whether it uses non-linearities in addition to gamma, whether it sharpens the image while “developing” the JPEG image, etc.
7. Develop an interactive viewer to change the exposure of an image based on the average exposure of a region around the mouse. (One variant is to show the adjusted image inside a window around the mouse. Another is to adjust the complete image based on the mouse position.)
8. Implement a tone mapping operator (Exercise 10.5) and use this to map your radiance image to a displayable gamut.

Ex 10.2: Noise level function Determine your camera’s noise level function using either multiple shots or by analyzing smooth regions.

1. Set up your camera on a tripod looking at a calibration target or a static scene with a good variation in input levels and colors. (Check your camera’s histogram to ensure that all values are being sampled.)
2. Take repeated images of the same scene (ideally with a remote shutter release) and average them to compute the variance at each pixel. Discarding pixels near high gradients (which are affected by camera motion), plot for each color channel the standard deviation at each pixel as a function of its output value.
3. Fit a lower envelope to these measurements and use this as your noise level function. How much variation do you see in the noise as a function of input level? How much of this is significant, i.e., away from flat regions in your camera response function where you do not want to be sampling anyway?
4. (Optional) Using the same images, develop a technique that segments the image into near-constant regions (Liu, Szeliski, Kang *et al.* 2008). (This is easier if you are photographing a calibration chart.) Compute the deviations for each region from a *single* image and use them to estimate the NLF. How does this compare to the multi-image technique, and how stable are your estimates from image to image?

Ex 10.3: Vignetting Estimate the amount of vignetting in some of your lenses using one of the following three techniques (or devise one of your choosing):

1. Take an image of a large uniform intensity region (well-illuminated wall or blue sky—but be careful of brightness gradients) and fit a radial polynomial curve to estimate the vignetting.

2. Construct a center-weighted panorama and compare these pixel values to the input image values to estimate the vignetting function. Weight pixels in slowly varying regions more highly, as small misalignments will give large errors at high gradients. Optionally estimate the radiometric response function as well (Litvinov and Schechner 2005; Goldman 2011).
3. Analyze the radial gradients (especially in low-gradient regions) and fit the robust means of these gradients to the derivative of the vignetting function, as described by Zheng, Yu, Kang *et al.* (2008).

For the parametric form of your vignetting function, you can either use a simple radial function, e.g.,

$$f(r) = 1 + \alpha_1 r + \alpha_2 r^2 + \dots \quad (10.42)$$

or one of the specialized equations developed by Kang and Weiss (2000) and Zheng, Lin, and Kang (2006).

In all of these cases, be sure that you are using linearized intensity measurements, by using either RAW images or images linearized through a radiometric response function, or at least images where the gamma curve has been removed.

(Optional) What happens if you forget to undo the gamma before fitting a (multiplicative) vignetting function?

Ex 10.4: Optical blur (PSF) estimation Compute the optical PSF either using a known target (Figure 10.7) or by detecting and fitting step edges (Section 10.1.4) (Joshi, Szeliski, and Kriegman 2008).

1. Detect strong edges to sub-pixel precision.
2. Fit a local profile to each oriented edge and fill these pixels into an ideal target image, either at image resolution or at a higher resolution (Figure 10.9c–d).
3. Use least squares (10.1) at valid pixels to estimate the PSF kernel K , either globally or in locally overlapping sub-regions of the image.
4. Visualize the recovered PSFs and use them to remove chromatic aberration or de-blur the image.

Ex 10.5: Tone mapping Implement one of the tone mapping algorithms discussed in Section 10.2.1 (Durand and Dorsey 2002; Fattal, Lischinski, and Werman 2002; Reinhard, Stark, Shirley *et al.* 2002; Lischinski, Farbman, Uyttendaele *et al.* 2006b) or any of the numerous additional algorithms discussed by Reinhard, Ward, Pattanaik *et al.* (2005) and <http://stellar.mit.edu/S/course/6/sp08/6.815/materials.html>.

(Optional) Compare your algorithm to local histogram equalization (Section 3.1.4).

Ex 10.6: Flash enhancement Develop an algorithm to combine flash and non-flash photographs to best effect. You can use ideas from Eisemann and Durand (2004) and Petschnigg, Agrawala, Hoppe *et al.* (2004) or anything else you think might work well.

Ex 10.7: Super-resolution Implement one or more super-resolution algorithms and compare their performance.

1. Take a set of photographs of the same scene using a hand-held camera (to ensure that there is some jitter between the photographs).
2. Determine the PSF for the images you are trying to super-resolve using one of the techniques in Exercise 10.4.
3. Alternatively, simulate a collection of lower-resolution images by taking a high-quality photograph (avoid those with compression artifacts) and applying your own pre-filter kernel and downsampling.
4. Estimate the relative motion between the images using a parametric translation and rotation motion estimation algorithm (Sections 6.1.3 or 8.2).
5. Implement a basic least squares super-resolution algorithm by minimizing the difference between the observed and downsampled images (10.27–10.28).
6. Add in a gradient image prior, either as another least squares term or as a robust term that can be minimized using iteratively reweighted least squares (Appendix A.3).
7. (Optional) Implement one of the example-based super-resolution techniques, where matching against a set of exemplar images is used either to infer higher-frequency information to be added to the reconstruction (Freeman, Jones, and Pasztor 2002) or higher-frequency gradients to be matched in the super-resolved image (Baker and Kanade 2002).
8. (Optional) Use local edge statistic information to improve the quality of the super-resolved image (Fattal 2007).

Ex 10.8: Image matting Develop an algorithm for pulling a foreground matte from natural images, as described in Section 10.4.

1. Make sure that the images you are taking are linearized (Exercise 10.1 and Section 10.1) and that your camera exposure is fixed (full manual mode), at least when taking multiple shots of the same scene.
2. To acquire ground truth data, place your object in front of a computer monitor and display a variety of solid background colors as well as some natural imagery.
3. Remove your object and re-display the same images to acquire known background colors.
4. Use triangulation matting (Smith and Blinn 1996) to estimate the ground truth opacities α and pre-multiplied foreground colors αF for your objects.
5. Implement one or more of the natural image matting algorithms described in Section 10.4 and compare your results to the ground truth values you computed. Alternatively, use the matting test images published on <http://alphamatting.com/>.
6. (Optional) Run your algorithms on other images taken with the same calibrated camera (or other images you find interesting).

Ex 10.9: Smoke and shadow matting Extract smoke or shadow mattes from one scene and insert them into another (Chuang, Agarwala, Curless *et al.* 2002; Chuang, Goldman, Curless *et al.* 2003).

1. Take a still or video sequence of images with and without some intermittent smoke and shadows. (Remember to linearize your images before proceeding with any computations.)
2. For each pixel, fit a line to the observed color values.
3. If performing smoke matting, robustly compute the intersection of these lines to obtain the smoke color estimate. Then, estimate the background color as the other extremum (unless you already took a smoke-free background image).

If performing shadow matting, compute robust shadow (minimum) and lit (maximum) values for each pixel.

4. Extract the smoke or shadow mattes from each frame as the fraction between these two values (background and smoke or shadowed and lit).
5. Scan a new (destination) scene or modify the original background with an image editor.
6. Re-insert the smoke or shadow matte, along with any other foreground objects you may have extracted.
7. (Optional) Using a series of cast stick shadows, estimate the deformation field for the destination scene in order to correctly warp (drape) the shadows across the new geometry. (This is related to the shadow scanning technique developed by Bouguet and Perona (1999) and implemented in Exercise 12.2.)
8. (Optional) Chuang, Goldman, Curless *et al.* (2003) only demonstrated their technique for planar source geometries. Can you extend their technique to capture shadows acquired from an irregular source geometry?
9. (Optional) Can you change the direction of the shadow, i.e., simulate the effect of changing the light source direction?

Ex 10.10: Texture synthesis Implement one of the texture synthesis or hole filling algorithms presented in Section 10.5. Here is one possible procedure:

1. Implement the basic Efros and Leung (1999) algorithm, i.e., starting from the outside (for hole filling) or in raster order (for texture synthesis), search for a similar neighborhood in the source texture image, and copy that pixel.
2. Add in the Wei and Levoy (2000) extension of generating the pixels in a coarse-to-fine fashion, i.e., generate a lower-resolution synthetic texture (or filled image), and use this as a guide for matching regions in the finer resolution version.
3. Add in the Criminisi, Pérez, and Toyama (2004) idea of prioritizing pixels to be filled by some function of the local structure (gradient or orientation strength).

4. Extend any of the above algorithms by selecting sub-blocks in the source texture and using optimization to determine the seam between the new block and the existing image that it overlaps (Efros and Freeman 2001).
5. (Optional) Implement one of the isophote (smooth continuation) inpainting algorithms (Bertalmio, Sapiro, Caselles *et al.* 2000; Telea 2004).
6. (Optional) Add the ability to supply a target (reference) image (Efros and Freeman 2001) or to provide sample filtered or unfiltered (reference and rendered) images (Hertzmann, Jacobs, Oliver *et al.* 2001), see Section 10.5.2.

Ex 10.11: Colorization Implement the Levin, Lischinski, and Weiss (2004) colorization algorithm that is sketched out in Section 10.3.2 and Figure 10.37. Find some historic monochrome photographs and some modern color ones. Write an interactive tool that lets you “pick” colors from a modern photo and paint over the old one. Tune the algorithm parameters to give you good results. Are you pleased with the results? Can you think of ways to make them look more “antique”, e.g., with softer (less saturated and edgy) colors?

Chapter 11

Stereo correspondence

| | | |
|--------|---|-----|
| 11.1 | Epipolar geometry | 471 |
| 11.1.1 | Rectification | 472 |
| 11.1.2 | Plane sweep | 474 |
| 11.2 | Sparse correspondence | 475 |
| 11.2.1 | 3D curves and profiles | 476 |
| 11.3 | Dense correspondence | 477 |
| 11.3.1 | Similarity measures | 479 |
| 11.4 | Local methods | 480 |
| 11.4.1 | Sub-pixel estimation and uncertainty | 482 |
| 11.4.2 | <i>Application:</i> Stereo-based head tracking | 483 |
| 11.5 | Global optimization | 484 |
| 11.5.1 | Dynamic programming | 485 |
| 11.5.2 | Segmentation-based techniques | 487 |
| 11.5.3 | <i>Application:</i> Z-keying and background replacement | 489 |
| 11.6 | Multi-view stereo | 489 |
| 11.6.1 | Volumetric and 3D surface reconstruction | 492 |
| 11.6.2 | Shape from silhouettes | 497 |
| 11.7 | Additional reading | 499 |
| 11.8 | Exercises | 500 |

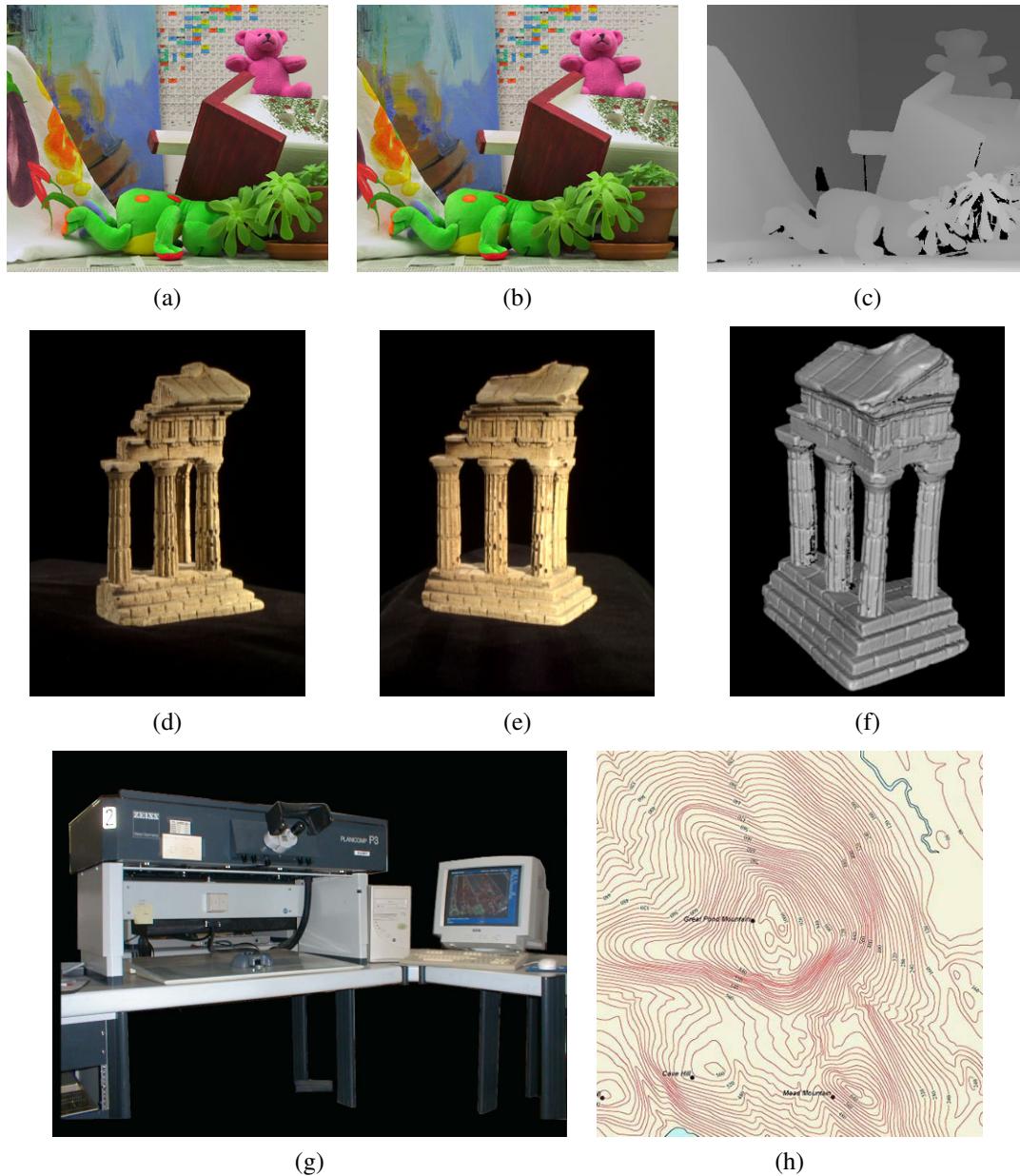


Figure 11.1 Stereo reconstruction techniques can convert (a–b) a pair of images into (c) a depth map (<http://vision.middlebury.edu/stereo/data/scenes2003/>) or (d–e) a sequence of images into (f) a 3D model (<http://vision.middlebury.edu/mview/data/>). (g) An analytical stereo plotter, courtesy of Kenney Aerial Mapping, Inc., can generate (h) contour plots.

Stereo matching is the process of taking two or more images and estimating a 3D model of the scene by finding matching pixels in the images and converting their 2D positions into 3D depths. In Chapters 6–7, we described techniques for recovering camera positions and building sparse 3D models of scenes or objects. In this chapter, we address the question of how to build a more complete 3D model, e.g., a sparse or dense *depth map* that assigns relative depths to pixels in the input images. We also look at the topic of *multi-view stereo* algorithms that produce complete 3D volumetric or surface-based object models.

Why are people interested in stereo matching? From the earliest inquiries into visual perception, it was known that we perceive depth based on the differences in appearance between the left and right eye.¹ As a simple experiment, hold your finger vertically in front of your eyes and close each eye alternately. You will notice that the finger jumps left and right relative to the background of the scene. The same phenomenon is visible in the image pair shown in Figure 11.1a–b, in which the foreground objects shift left and right relative to the background.

As we will shortly see, under simple imaging configurations (both eyes or cameras looking straight ahead), the amount of horizontal motion or *disparity* is inversely proportional to the distance from the observer. While the basic physics and geometry relating visual disparity to scene structure are well understood (Section 11.1), automatically measuring this disparity by establishing dense and accurate inter-image *correspondences* is a challenging task.

The earliest stereo matching algorithms were developed in the field of *photogrammetry* for automatically constructing topographic elevation maps from overlapping aerial images. Prior to this, operators would use photogrammetric stereo plotters, which displayed shifted versions of such images to each eye and allowed the operator to float a dot cursor around constant elevation contours (Figure 11.1g). The development of fully automated stereo matching algorithms was a major advance in this field, enabling much more rapid and less expensive processing of aerial imagery (Hannah 1974; Hsieh, McKeown, and Perlant 1992).

In computer vision, the topic of stereo matching has been one of the most widely studied and fundamental problems (Marr and Poggio 1976; Barnard and Fischler 1982; Dhond and Aggarwal 1989; Scharstein and Szeliski 2002; Brown, Burschka, and Hager 2003; Seitz, Curless, Diebel *et al.* 2006), and continues to be one of the most active research areas. While photogrammetric matching concentrated mainly on aerial imagery, computer vision applications include modeling the human visual system (Marr 1982), robotic navigation and manipulation (Moravec 1983; Konolige 1997; Thrun, Montemerlo, Dahlkamp *et al.* 2006), as well as view interpolation and image-based rendering (Figure 11.2a–d), 3D model building (Figure 11.2e–f and h–j), and mixing live action with computer-generated imagery (Figure 11.2g).

In this chapter, we describe the fundamental principles behind stereo matching, following the general taxonomy proposed by Scharstein and Szeliski (2002). We begin in Section 11.1 with a review of the *geometry* of stereo image matching, i.e., how to compute for a given pixel in one image the range of possible locations the pixel might appear at in the other image, i.e., its *epipolar line*. We describe how to pre-warp images so that corresponding epipolar lines are coincident (*rectification*). We also describe a general resampling algorithm called *plane sweep* that can be used to perform multi-image stereo matching with arbitrary camera configurations.

¹ The word *stereo* comes from the Greek for *solid*; stereo vision is how we perceive solid shape (Koenderink 1990).

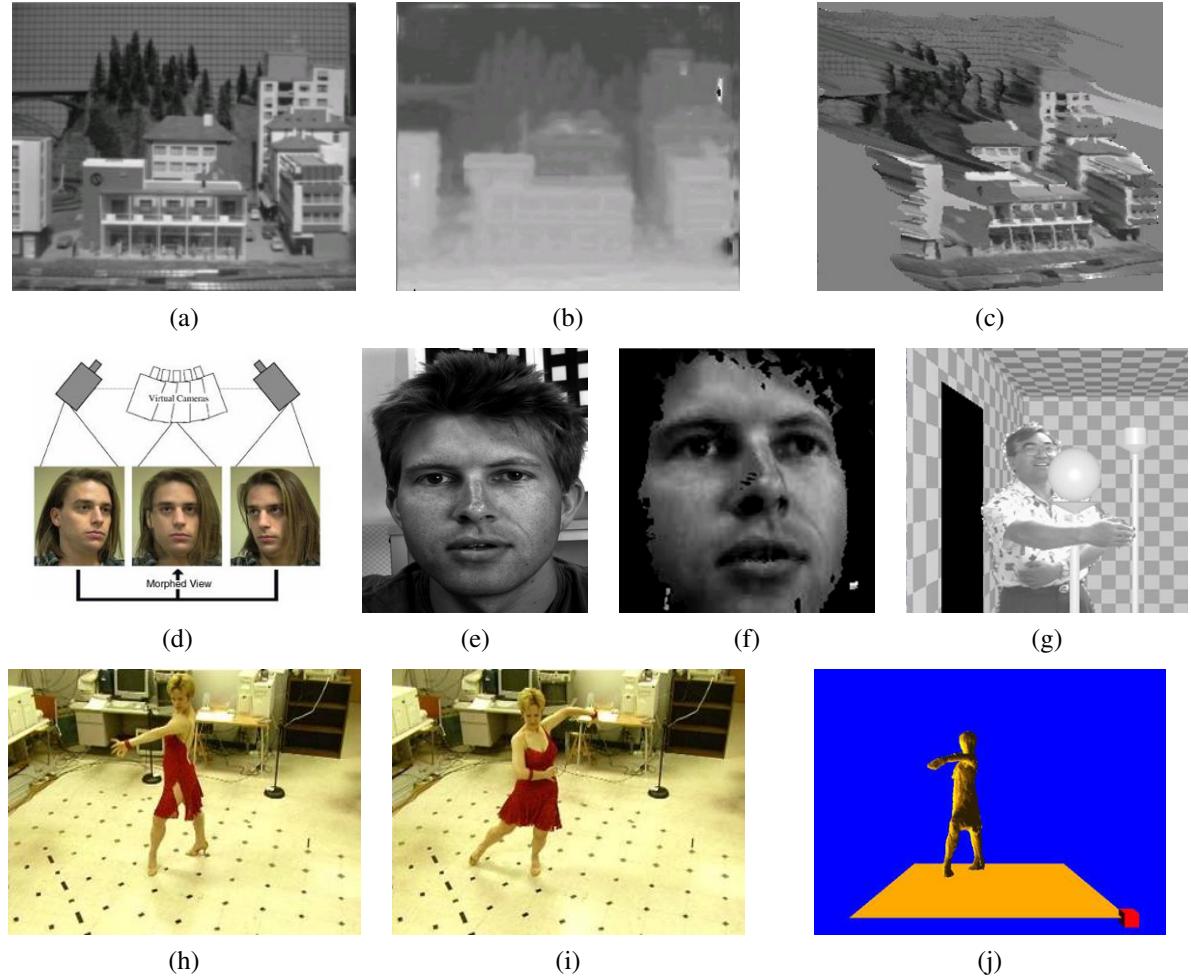


Figure 11.2 Applications of stereo vision: (a) input image, (b) computed depth map, and (c) new view generation from multi-view stereo (Matthies, Kanade, and Szeliski 1989) © 1989 Springer; (d) view morphing between two images (Seitz and Dyer 1996) © 1996 ACM; (e–f) 3D face modeling (images courtesy of Frédéric Devernay); (g) *z-keying* live and computer-generated imagery (Kanade, Yoshida, Oda *et al.* 1996) © 1996 IEEE; (h–j) building 3D surface models from multiple video streams in Virtualized Reality (Kanade, Rander, and Narayanan 1997).

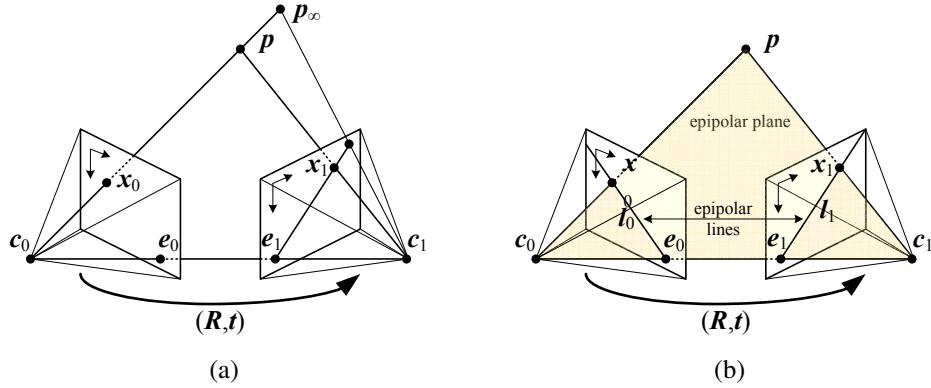


Figure 11.3 Epipolar geometry: (a) epipolar line segment corresponding to one ray; (b) corresponding set of epipolar lines and their epipolar plane.

Next, we briefly survey techniques for the *sparse* stereo matching of interest points and edge-like features (Section 11.2). We then turn to the main topic of this chapter, namely the estimation of a *dense* set of pixel-wise correspondences in the form of a *disparity map* (Figure 11.1c). This involves first selecting a pixel matching criterion (Section 11.3) and then using either local area-based aggregation (Section 11.4) or global optimization (Section 11.5) to help disambiguate potential matches. In Section 11.6, we discuss *multi-view stereo* methods that aim to reconstruct a complete 3D model instead of just a single disparity image (Figure 11.1d–f).

11.1 Epipolar geometry

Given a pixel in one image, how can we compute its correspondence in the other image? In Chapter 8, we saw that a variety of search techniques can be used to match pixels based on their local appearance as well as the motions of neighboring pixels. In the case of stereo matching, however, we have some additional information available, namely the positions and calibration data for the cameras that took the pictures of the same static scene (Section 7.2).

How can we exploit this information to reduce the number of potential correspondences, and hence both speed up the matching and increase its reliability? Figure 11.3a shows how a pixel in one image x_0 projects to an *epipolar line segment* in the other image. The segment is bounded at one end by the projection of the original viewing ray at infinity p_∞ and at the other end by the projection of the original camera center c_0 into the second camera, which is known as the *epipole* e_1 . If we project the epipolar line in the second image back into the first, we get another line (segment), this time bounded by the other corresponding epipole e_0 . Extending both line segments to infinity, we get a pair of corresponding *epipolar lines* (Figure 11.3b), which are the intersection of the two image planes with the *epipolar plane* that passes through both camera centers c_0 and c_1 as well as the point of interest p (Faugeras and Luong 2001; Hartley and Zisserman 2004).

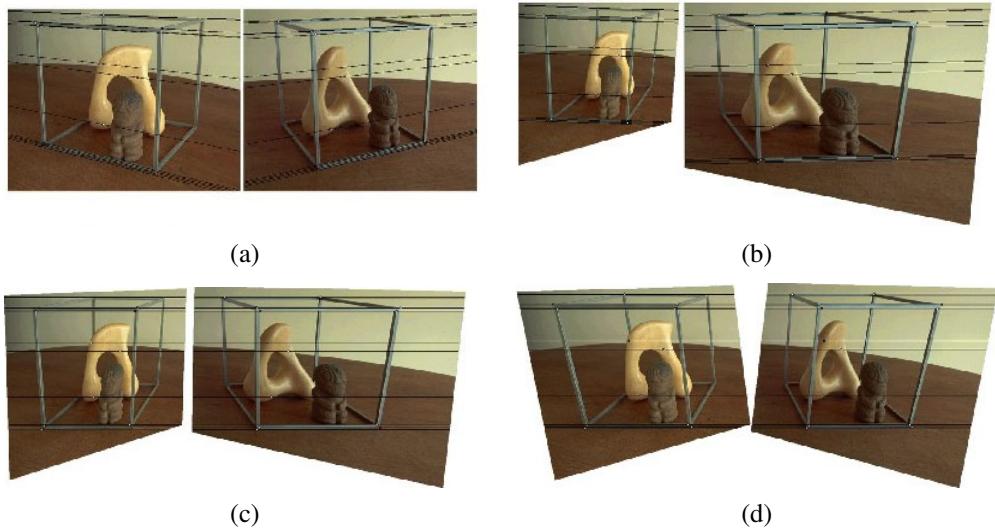


Figure 11.4 The multi-stage stereo rectification algorithm of Loop and Zhang (1999) © 1999 IEEE. (a) Original image pair overlaid with several epipolar lines; (b) images transformed so that epipolar lines are parallel; (c) images rectified so that epipolar lines are horizontal and in vertical correspondence; (d) final rectification that minimizes horizontal distortions.

11.1.1 Rectification

As we saw in Section 7.2, the epipolar geometry for a pair of cameras is implicit in the relative pose and calibrations of the cameras, and can easily be computed from seven or more point matches using the fundamental matrix (or five or more points for the calibrated essential matrix) (Zhang 1998a,b; Faugeras and Luong 2001; Hartley and Zisserman 2004). Once this geometry has been computed, we can use the epipolar line corresponding to a pixel in one image to constrain the search for corresponding pixels in the other image. One way to do this is to use a general correspondence algorithm, such as optical flow (Section 8.4), but to only consider locations along the epipolar line (or to project any flow vectors that fall off back onto the line).

A more efficient algorithm can be obtained by first *rectifying* (i.e., warping) the input images so that corresponding horizontal scanlines are epipolar lines (Loop and Zhang 1999; Faugeras and Luong 2001; Hartley and Zisserman 2004).² Afterwards, it is possible to match horizontal scanlines independently or to shift images horizontally while computing matching scores (Figure 11.4).

A simple way to rectify the two images is to first rotate both cameras so that they are looking perpendicular to the line joining the camera centers c_0 and c_1 . Since there is a degree of freedom in the *tilt*, the smallest rotations that achieve this should be used. Next, to determine the desired twist around the optical axes, make the *up vector* (the camera y axis) perpendicular to the camera center line. This ensures that corresponding epipolar lines are

² This makes most sense if the cameras are next to each other, although by rotating the cameras, rectification can be performed on any pair that is not *verged* too much or has too much of a scale change. In those latter cases, using plane sweep (below) or hypothesizing small planar patch locations in 3D (Goesele, Snavely, Curless *et al.* 2007) may be preferable.

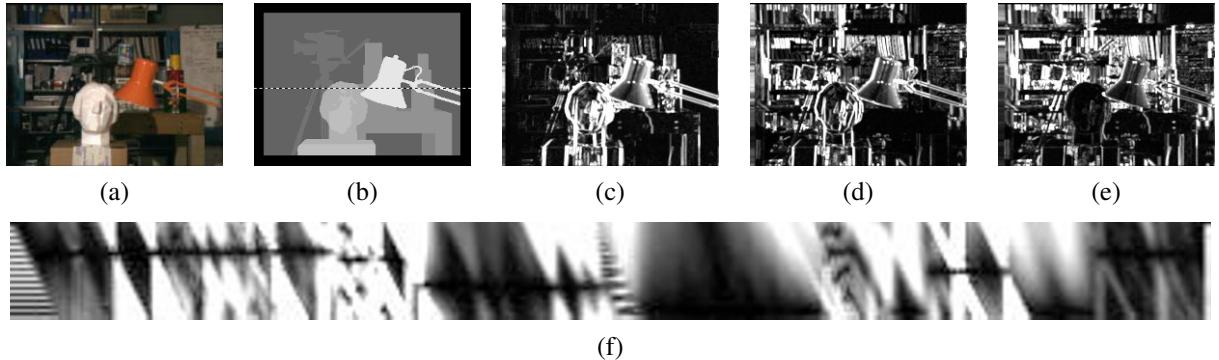


Figure 11.5 Slices through a typical disparity space image (DSI) (Scharstein and Szeliski 2002) © 2002 Springer: (a) original color image; (b) ground truth disparities; (c–e) three (x, y) slices for $d = 10, 16, 21$; (f) an (x, d) slice for $y = 151$ (the dashed line in (b)). Various dark (matching) regions are visible in (c–e), e.g., the bookshelves, table and cans, and head statue, and three disparity levels can be seen as horizontal lines in (f). The dark bands in the DSIs indicate regions that match at this disparity. (Smaller dark regions are often the result of textureless regions.) Additional examples of DSIs are discussed by Bobick and Intille (1999).

horizontal and that the disparity for points at infinity is 0. Finally, re-scale the images, if necessary, to account for different focal lengths, magnifying the smaller image to avoid aliasing. (The full details of this procedure can be found in Fusiello, Trucco, and Verri (2000) and Exercise 11.1.) Note that in general, it is not possible to rectify an arbitrary collection of images simultaneously unless their optical centers are collinear, although rotating the cameras so that they all point in the same direction reduces the inter-camera pixel movements to scalings and translations.

The resulting *standard rectified geometry* is employed in a lot of stereo camera setups and stereo algorithms, and leads to a very simple inverse relationship between 3D depths Z and disparities d ,

$$d = f \frac{B}{Z}, \quad (11.1)$$

where f is the focal length (measured in pixels), B is the baseline, and

$$x' = x + d(x, y), \quad y' = y \quad (11.2)$$

describes the relationship between corresponding pixel coordinates in the left and right images (Bolles, Baker, and Marimont 1987; Okutomi and Kanade 1993; Scharstein and Szeliski 2002).³ The task of extracting depth from a set of images then becomes one of estimating the *disparity map* $d(x, y)$.

After rectification, we can easily compare the similarity of pixels at corresponding locations (x, y) and $(x', y') = (x + d, y)$ and store them in a *disparity space image* (DSI) $C(x, y, d)$ for further processing (Figure 11.5). The concept of the disparity space (x, y, d) dates back to early work in stereo matching (Marr and Poggio 1976), while the concept of a disparity space image (volume) is generally associated with Yang, Yuille, and Lu (1993) and Intille and Bobick (1994).

³ The term *disparity* was first introduced in the human vision literature to describe the difference in location of corresponding features seen by the left and right eyes (Marr 1982). Horizontal disparity is the most commonly studied phenomenon, but vertical disparity is possible if the eyes are verged.

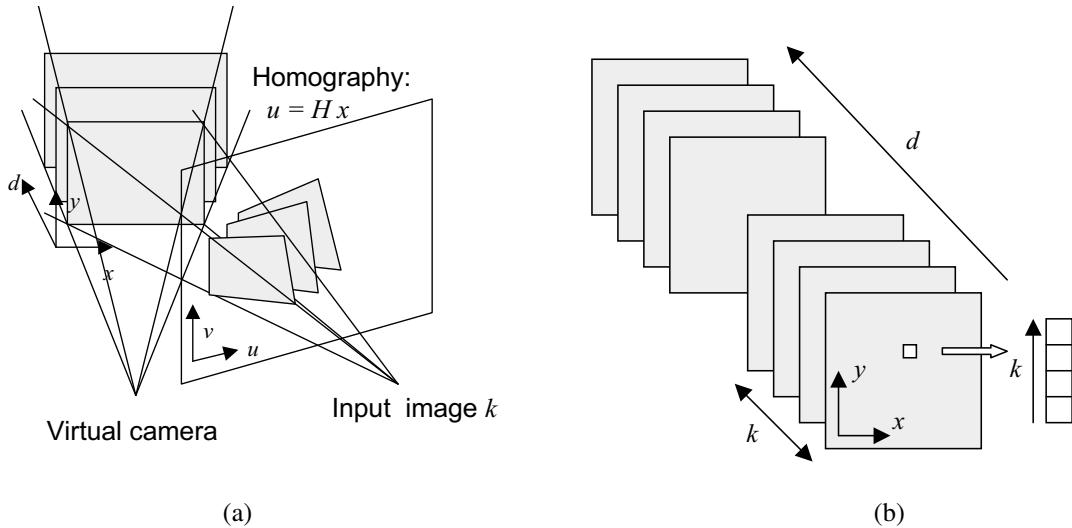


Figure 11.6 Sweeping a set of planes through a scene (Szeliski and Golland 1999) © 1999 Springer: (a) The set of planes seen from a virtual camera induces a set of homographies in any other source (input) camera image. (b) The warped images from all the other cameras can be stacked into a generalized disparity space volume $\tilde{I}(x, y, d, k)$ indexed by pixel location (x, y) , disparity d , and camera k .

11.1.2 Plane sweep

An alternative to pre-rectifying the images before matching is to sweep a set of planes through the scene and to measure the *photoconsistency* of different images as they are re-projected onto these planes (Figure 11.6). This process is commonly known as the *plane sweep* algorithm (Collins 1996; Szeliski and Golland 1999; Saito and Kanade 1999).

As we saw in Section 2.1.5, where we introduced projective depth (also known as *plane plus parallax* (Kumar, Anandan, and Hanna 1994; Sawhney 1994; Szeliski and Coughlan 1997)), the last row of a full-rank 4×4 projection matrix $\tilde{\mathbf{P}}$ can be set to an arbitrary plane equation $\mathbf{p}_3 = s_3[\hat{\mathbf{n}}_0 | c_0]$. The resulting four-dimensional projective transform (*collineation*) (2.68) maps 3D world points $\mathbf{p} = (X, Y, Z, 1)$ into screen coordinates $\mathbf{x}_s = (x_s, y_s, 1, d)$, where the *projective depth* (or *parallax*) d (2.66) is 0 on the reference plane (Figure 2.11).

Sweeping d through a series of disparity hypotheses, as shown in Figure 11.6a, corresponds to mapping each input image into the *virtual camera* $\tilde{\mathbf{P}}$ defining the disparity space through a series of homographies (2.68–2.71),

$$\tilde{\mathbf{x}}_k \sim \tilde{\mathbf{P}}_k \tilde{\mathbf{P}}^{-1} \mathbf{x}_s = \tilde{\mathbf{H}}_k \tilde{\mathbf{x}} + \mathbf{t}_k d = (\tilde{\mathbf{H}}_k + \mathbf{t}_k [0 \ 0 \ d]) \tilde{\mathbf{x}}, \quad (11.3)$$

as shown in Figure 2.12b, where $\tilde{\mathbf{x}}_k$ and $\tilde{\mathbf{x}}$ are the homogeneous pixel coordinates in the source and virtual (reference) images (Szeliski and Golland 1999). The members of the family of homographies $\tilde{\mathbf{H}}_k(d) = \tilde{\mathbf{H}}_k + \mathbf{t}_k [0 \ 0 \ d]$, which are parameterized by the addition of a rank-1 matrix, are related to each other through a *planar homology* (Hartley and Zisserman 2004, A5.2).

The choice of virtual camera and parameterization is application dependent and is what gives this framework a lot of its flexibility. In many applications, one of the input cameras (the *reference camera*) is used, thus computing a depth map that is registered with one of the

input images and which can later be used for image-based rendering (Sections 13.1 and 13.2). In other applications, such as view interpolation for gaze correction in video-conferencing (Section 11.4.2) (Ott, Lewis, and Cox 1993; Criminisi, Shotton, Blake *et al.* 2003), a camera centrally located between the two input cameras is preferable, since it provides the needed per-pixel disparities to hallucinate the virtual middle image.

The choice of disparity sampling, i.e., the setting of the zero parallax plane and the scaling of integer disparities, is also application dependent, and is usually set to bracket the range of interest, i.e., the *working volume*, while scaling disparities to sample the image in pixel (or sub-pixel) shifts. For example, when using stereo vision for obstacle avoidance in robot navigation, it is most convenient to set up disparity to measure per-pixel elevation above the ground (Ivanchenko, Shen, and Coughlan 2009).

As each input image is warped onto the current planes parameterized by disparity d , it can be stacked into a *generalized disparity space image* $\tilde{I}(x, y, d, k)$ for further processing (Figure 11.6b) (Szeliski and Golland 1999). In most stereo algorithms, the photoconsistency (e.g., sum of squared or robust differences) with respect to the reference image I_r is calculated and stored in the DSI

$$C(x, y, d) = \sum_k \rho(\tilde{I}(x, y, d, k) - I_r(x, y)). \quad (11.4)$$

However, it is also possible to compute alternative statistics such as robust variance, focus, or entropy (Section 11.3.1) (Vaish, Szeliski, Zitnick *et al.* 2006) or to use this representation to reason about occlusions (Szeliski and Golland 1999; Kang and Szeliski 2004). The generalized DSI will come in particularly handy when we come back to the topic of multi-view stereo in Section 11.6.

Of course, planes are not the only surfaces that can be used to define a 3D sweep through the space of interest. Cylindrical surfaces, especially when coupled with panoramic photography (Chapter 9), are often used (Ishiguro, Yamamoto, and Tsuji 1992; Kang and Szeliski 1997; Shum and Szeliski 1999; Li, Shum, Tang *et al.* 2004; Zheng, Kang, Cohen *et al.* 2007). It is also possible to define other manifold topologies, e.g., ones where the camera rotates around a fixed axis (Seitz 2001).

Once the DSI has been computed, the next step in most stereo correspondence algorithms is to produce a univalued function in disparity space $d(x, y)$ that best describes the shape of the surfaces in the scene. This can be viewed as finding a surface embedded in the disparity space image that has some optimality property, such as lowest cost and best (piecewise) smoothness (Yang, Yuille, and Lu 1993). Figure 11.5 shows examples of slices through a typical DSI. More figures of this kind can be found in the paper by Bobick and Intille (1999).

11.2 Sparse correspondence

Early stereo matching algorithms were *feature-based*, i.e., they first extracted a set of potentially matchable image locations, using either interest operators or edge detectors, and then searched for corresponding locations in other images using a patch-based metric (Hannah 1974; Marr and Poggio 1979; Mayhew and Frisby 1980; Baker and Binford 1981; Arnold 1983; Grimson 1985; Ohta and Kanade 1985; Bolles, Baker, and Marimont 1987; Matthies, Kanade, and Szeliski 1989; Hsieh, McKeown, and Perlant 1992; Bolles, Baker, and Hannah

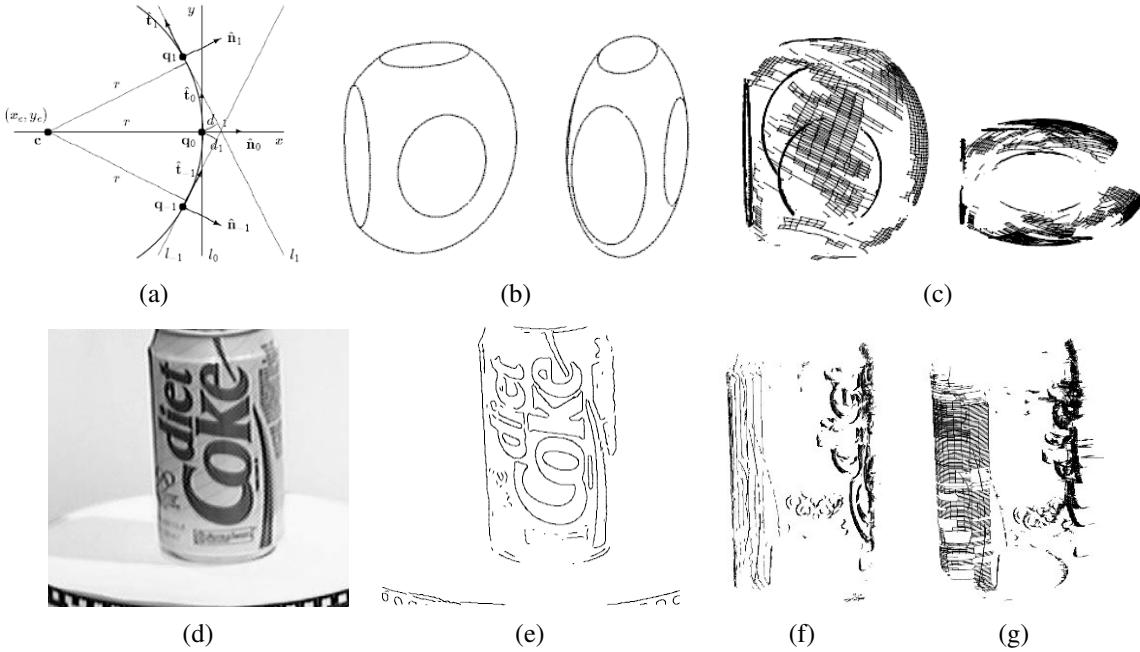


Figure 11.7 Surface reconstruction from occluding contours (Szeliski and Weiss 1998) © 2002 Springer: (a) circular arc fitting in the epipolar plane; (b) synthetic example of an ellipsoid with a truncated side and elliptic surface markings; (c) partially reconstructed surface mesh seen from an oblique and top-down view; (d) real-world image sequence of a soda can on a turntable; (e) extracted edges; (f) partially reconstructed profile curves; (g) partially reconstructed surface mesh. (Partial reconstructions are shown so as not to clutter the images.)

1993). This limitation to sparse correspondences was partially due to computational resource limitations, but was also driven by a desire to limit the answers produced by stereo algorithms to matches with high certainty. In some applications, there was also a desire to match scenes with potentially very different illuminations, where edges might be the only stable features (Collins 1996). Such sparse 3D reconstructions could later be interpolated using surface fitting algorithms such as those discussed in Sections 3.7.1 and 12.3.1.

More recent work in this area has focused on first extracting highly reliable features and then using these as *seeds* to grow additional matches (Zhang and Shan 2000; Lhuillier and Quan 2002). Similar approaches have also been extended to wide baseline multi-view stereo problems and combined with 3D surface reconstruction (Lhuillier and Quan 2005; Strecha, Tuytelaars, and Van Gool 2003; Goesele, Snavely, Curless *et al.* 2007) or free-space reasoning (Taylor 2003), as described in more detail in Section 11.6.

11.2.1 3D curves and profiles

Another example of sparse correspondence is the matching of *profile curves* (or *occluding contours*), which occur at the boundaries of objects (Figure 11.7) and at interior self occlusions, where the surface curves away from the camera viewpoint.

The difficulty in matching profile curves is that in general, the locations of profile curves vary as a function of camera viewpoint. Therefore, matching curves directly in two images

and then triangulating these matches can lead to erroneous shape measurements. Fortunately, if three or more closely spaced frames are available, it is possible to fit a local circular arc to the locations of corresponding edgels (Figure 11.7a) and therefore obtain semi-dense curved surface meshes directly from the matches (Figures 11.7c and g). Another advantage of matching such curves is that they can be used to reconstruct surface shape for untextured surfaces, so long as there is a visible difference between foreground and background colors.

Over the years, a number of different techniques have been developed for reconstructing surface shape from profile curves (Giblin and Weiss 1987; Cipolla and Blake 1992; Vaillant and Faugeras 1992; Zheng 1994; Boyer and Berger 1997; Szeliski and Weiss 1998). Cipolla and Giblin (2000) describe many of these techniques, as well as related topics such as inferring camera motion from profile curve sequences. Below, we summarize the approach developed by Szeliski and Weiss (1998), which assumes a discrete set of images, rather than formulating the problem in a continuous differential framework.

Let us assume that the camera is moving smoothly enough that the local epipolar geometry varies slowly, i.e., the epipolar planes induced by the successive camera centers and an edgel under consideration are nearly co-planar. The first step in the processing pipeline is to extract and link edges in each of the input images (Figures 11.7b and e). Next, edgels in successive images are matched using pairwise epipolar geometry, proximity and (optionally) appearance. This provides a linked set of edges in the spatio-temporal volume, which is sometimes called the *weaving wall* (Baker 1989).

To reconstruct the 3D location of an individual edgel, along with its local in-plane normal and curvature, we project the viewing rays corresponding to its neighbors onto the instantaneous epipolar plane defined by the camera center, the viewing ray, and the camera velocity, as shown in Figure 11.7a. We then fit an *osculating circle* to the projected lines, parameterizing the circle by its centerpoint $c = (x_c, y_c)$ and radius r ,

$$c_i x_c + s_i y_c + r = d_i, \quad (11.5)$$

where $c_i = \hat{t}_i \cdot \hat{t}_0$ and $s_i = -\hat{t}_i \cdot \hat{n}_0$ are the cosine and sine of the angle between viewing ray i and the central viewing ray 0, and $d_i = (\mathbf{q}_i - \mathbf{q}_0) \cdot \hat{n}_0$ is the perpendicular distance between viewing ray i and the local origin \mathbf{q}_0 , which is a point chosen on the central viewing ray close to the line intersections (Szeliski and Weiss 1998). The resulting set of linear equations can be solved using least squares, and the quality of the solution (residual error) can be used to check for erroneous correspondences.

The resulting set of 3D points, along with their spatial (in-image) and temporal (between-image) neighbors, form a 3D surface mesh with local normal and curvature estimates (Figures 11.7c and g). Note that whenever a curve is due to a surface marking or a sharp crease edge, rather than a smooth surface profile curve, this shows up as a 0 or small radius of curvature. Such curves result in isolated 3D space curves, rather than elements of smooth surface meshes, but can still be incorporated into the 3D surface model during a later stage of surface interpolation (Section 12.3.1).

11.3 Dense correspondence

While sparse matching algorithms are still occasionally used, most stereo matching algorithms today focus on dense correspondence, since this is required for applications such as

image-based rendering or modeling. This problem is more challenging than sparse correspondence, since inferring depth values in textureless regions requires a certain amount of guesswork. (Think of a solid colored background seen through a picket fence. What depth should it be?)

In this section, we review the taxonomy and categorization scheme for dense correspondence algorithms first proposed by Scharstein and Szeliski (2002). The taxonomy consists of a set of algorithmic “building blocks” from which a large set of algorithms can be constructed. It is based on the observation that stereo algorithms generally perform some subset of the following four steps:

1. matching cost computation;
2. cost (support) aggregation;
3. disparity computation and optimization; and
4. disparity refinement.

For example, *local* (window-based) algorithms (Section 11.4), where the disparity computation at a given point depends only on intensity values within a finite window, usually make implicit smoothness assumptions by aggregating support. Some of these algorithms can cleanly be broken down into steps 1, 2, 3. For example, the traditional sum-of-squared-differences (SSD) algorithm can be described as:

1. The matching cost is the squared difference of intensity values at a given disparity.
2. Aggregation is done by summing the matching cost over square windows with constant disparity.
3. Disparities are computed by selecting the minimal (winning) aggregated value at each pixel.

Some local algorithms, however, combine steps 1 and 2 and use a matching cost that is based on a support region, e.g. normalized cross-correlation (Hannah 1974; Bolles, Baker, and Hannah 1993) and the rank transform (Zabih and Woodfill 1994) and other ordinal measures (Bhat and Nayar 1998). (This can also be viewed as a preprocessing step; see (Section 11.3.1).)

Global algorithms, on the other hand, make explicit smoothness assumptions and then solve a global optimization problem (Section 11.5). Such algorithms typically do not perform an aggregation step, but rather seek a disparity assignment (step 3) that minimizes a global cost function that consists of data (step 1) terms and smoothness terms. The main distinctions among these algorithms is the minimization procedure used, e.g., simulated annealing (Marroquin, Mitter, and Poggio 1987; Barnard 1989), probabilistic (mean-field) diffusion (Scharstein and Szeliski 1998), expectation maximization (EM) (Birchfield, Natarajan, and Tomasi 2007), graph cuts (Boykov, Veksler, and Zabih 2001), or loopy belief propagation (Sun, Zheng, and Shum 2003), to name just a few.

In between these two broad classes are certain iterative algorithms that do not explicitly specify a global function to be minimized, but whose behavior mimics closely that of iterative optimization algorithms (Marr and Poggio 1976; Zitnick and Kanade 2000). Hierarchical (coarse-to-fine) algorithms resemble such iterative algorithms, but typically operate on an

image pyramid where results from coarser levels are used to constrain a more local search at finer levels (Witkin, Terzopoulos, and Kass 1987; Quam 1984; Bergen, Anandan, Hanna *et al.* 1992).

11.3.1 Similarity measures

The first component of any dense stereo matching algorithm is a similarity measure that compares pixel values in order to determine how likely they are to be in correspondence. In this section, we briefly review the similarity measures introduced in Section 8.1 and mention a few others that have been developed specifically for stereo matching (Scharstein and Szeliski 2002; Hirschmüller and Scharstein 2009).

The most common pixel-based matching costs include sums of *squared intensity differences* (SSD) (Hannah 1974) and *absolute intensity differences* (SAD) (Kanade 1994). In the video processing community, these matching criteria are referred to as the *mean-squared error* (MSE) and *mean absolute difference* (MAD) measures; the term *displaced frame difference* is also often used (Tekalp 1995).

More recently, robust measures (8.2), including truncated quadratics and contaminated Gaussians, have been proposed (Black and Anandan 1996; Black and Rangarajan 1996; Scharstein and Szeliski 1998). These measures are useful because they limit the influence of mismatches during aggregation. Vaish, Szeliski, Zitnick *et al.* (2006) compare a number of such robust measures, including a new one based on the entropy of the pixel values at a particular disparity hypothesis (Zitnick, Kang, Uyttendaele *et al.* 2004), which is particularly useful in multi-view stereo.

Other traditional matching costs include normalized cross-correlation (8.11) (Hannah 1974; Bolles, Baker, and Hannah 1993; Evangelidis and Psarakis 2008), which behaves similarly to sum-of-squared-differences (SSD), and binary matching costs (i.e., match or no match) (Marr and Poggio 1976), based on binary features such as edges (Baker and Binford 1981; Grimson 1985) or the sign of the Laplacian (Nishihara 1984). Because of their poor discriminability, simple binary matching costs are no longer used in dense stereo matching.

Some costs are insensitive to differences in camera gain or bias, for example gradient-based measures (Seitz 1989; Scharstein 1994), phase and filter-bank responses (Marr and Poggio 1979; Kass 1988; Jenkin, Jepson, and Tsotsos 1991; Jones and Malik 1992), filters that remove regular or robust (bilaterally filtered) means (Ansar, Castano, and Matthies 2004; Hirschmüller and Scharstein 2009), dense feature descriptor (Tola, Lepetit, and Fua 2010), and non-parametric measures such as rank and census transforms (Zabih and Woodfill 1994), ordinal measures (Bhat and Nayar 1998), or entropy (Zitnick, Kang, Uyttendaele *et al.* 2004; Zitnick and Kang 2007). The census transform, which converts each pixel inside a moving window into a bit vector representing which neighbors are above or below the central pixel, was found by Hirschmüller and Scharstein (2009) to be quite robust against large-scale, non-stationary exposure and illumination changes.

It is also possible to correct for differing global camera characteristics by performing a preprocessing or iterative refinement step that estimates inter-image bias–gain variations using global regression (Gennert 1988), histogram equalization (Cox, Roy, and Hingorani 1995), or mutual information (Kim, Kolmogorov, and Zabih 2003; Hirschmüller 2008). Local, smoothly varying compensation fields have also been proposed (Strecha, Tuytelaars, and Van Gool 2003; Zhang, McMillan, and Yu 2006).

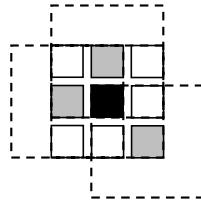


Figure 11.8 Shiftable window (Scharstein and Szeliski 2002) © 2002 Springer. The effect of trying all 3×3 shifted windows around the black pixel is the same as taking the minimum matching score across all *centered* (non-shifted) windows in the same neighborhood. (For clarity, only three of the neighboring shifted windows are shown here.)

In order to compensate for sampling issues, i.e., dramatically different pixel values in high-frequency areas, Birchfield and Tomasi (1998) proposed a matching cost that is less sensitive to shifts in image sampling. Rather than just comparing pixel values shifted by integral amounts (which may miss a valid match), they compare each pixel in the reference image against a linearly interpolated function of the other image. More detailed studies of these and additional matching costs are explored in (Szeliski and Scharstein 2004; Hirschmüller and Scharstein 2009). In particular, if you expect there to be significant exposure or appearance variation between images that you are matching, some of the more robust measures that performed well in the evaluation by Hirschmüller and Scharstein (2009), such as the census transform (Zabih and Woodfill 1994), ordinal measures (Bhat and Nayar 1998), bilateral subtraction (Ansar, Castano, and Matthies 2004), or hierarchical mutual information (Hirschmüller 2008), should be used.

11.4 Local methods

Local and window-based methods aggregate the matching cost by summing or averaging over a *support region* in the DSI $C(x, y, d)$.⁴ A support region can be either two-dimensional at a fixed disparity (favoring fronto-parallel surfaces), or three-dimensional in x - y - d space (supporting slanted surfaces). Two-dimensional evidence aggregation has been implemented using square windows or Gaussian convolution (traditional), multiple windows anchored at different points, i.e., shiftable windows (Arnold 1983; Fusiello, Roberto, and Trucco 1997; Bobick and Intille 1999), windows with adaptive sizes (Okutomi and Kanade 1992; Kanade and Okutomi 1994; Kang, Szeliski, and Chai 2001; Veksler 2001, 2003), windows based on connected components of constant disparity (Boykov, Veksler, and Zabih 1998), or the results of color-based segmentation (Yoon and Kweon 2006; Tombari, Mattoccia, Di Stefano *et al.* 2008). Three-dimensional support functions that have been proposed include limited disparity difference (Grimson 1985), limited disparity gradient (Pollard, Mayhew, and Frisby 1985), Prazdny's coherence principle (Prazdny 1985), and the more recent work (which includes visibility and occlusion reasoning) by Zitnick and Kanade (2000).

⁴ For two recent surveys and comparisons of such techniques, please see the work of Gong, Yang, Wang *et al.* (2007) and Tombari, Mattoccia, Di Stefano *et al.* (2008).



Figure 11.9 Aggregation window sizes and weights adapted to image content (Tombari, Mattoccia, Di Stefano *et al.* 2008) © 2008 IEEE: (a) original image with selected evaluation points; (b) variable windows (Veksler 2003); (c) adaptive weights (Yoon and Kweon 2006); (d) segmentation-based (Tombari, Mattoccia, and Di Stefano 2007). Notice how the adaptive weights and segmentation-based techniques adapt their support to similarly colored pixels.

Aggregation with a fixed support region can be performed using 2D or 3D convolution,

$$C(x, y, d) = w(x, y, d) * C_0(x, y, d), \quad (11.6)$$

or, in the case of rectangular windows, using efficient moving average box-filters (Section 3.2.2) (Kanade, Yoshida, Oda *et al.* 1996; Kimura, Shinbo, Yamaguchi *et al.* 1999). Shiftable windows can also be implemented efficiently using a separable sliding min-filter (Figure 11.8) (Scharstein and Szeliski 2002, Section 4.2). Selecting among windows of different shapes and sizes can be performed more efficiently by first computing a *summed area table* (Section 3.2.3, 3.30–3.32) (Veksler 2003). Selecting the right window is important, since windows must be large enough to contain sufficient texture and yet small enough so that they do not straddle depth discontinuities (Figure 11.9). An alternative method for aggregation is *iterative diffusion*, i.e., repeatedly adding to each pixel’s cost the weighted values of its neighboring pixels’ costs (Szeliski and Hinton 1985; Shah 1993; Scharstein and Szeliski 1998).

Of the local aggregation methods compared by Gong, Yang, Wang *et al.* (2007) and Tombari, Mattoccia, Di Stefano *et al.* (2008), the fast variable window approach of Veksler (2003) and the locally weighting approach developed by Yoon and Kweon (2006) consistently stood out as having the best tradeoff between performance and speed.⁵ The local weighting technique, in particular, is interesting because, instead of using square windows with uniform weighting, each pixel within an aggregation window influences the final matching cost based on its color similarity and spatial distance, just as in bilinear filtering (Figure 11.9c). (In fact, their aggregation step is closely related to doing a joint bilateral filter on the color/disparity image, except that it is done symmetrically in both reference and target images.) The segmentation-based aggregation method of Tombari, Mattoccia, and Di Stefano (2007) did even better, although a fast implementation of this algorithm does not yet exist.

In local methods, the emphasis is on the matching cost computation and cost aggregation steps. Computing the final disparities is trivial: simply choose at each pixel the disparity associated with the minimum cost value. Thus, these methods perform a local “winner-take-all” (WTA) optimization at each pixel. A limitation of this approach (and many other

⁵ More recent and extensive results from Tombari, Mattoccia, Di Stefano *et al.* (2008) can be found at <http://www.vision.deis.unibo.it/spe/SPEHome.aspx>.

correspondence algorithms) is that uniqueness of matches is only enforced for one image (the *reference image*), while points in the other image might match multiple points, unless cross-checking and subsequent hole filling is used (Fua 1993; Hirschmüller and Scharstein 2009).

11.4.1 Sub-pixel estimation and uncertainty

Most stereo correspondence algorithms compute a set of disparity estimates in some discretized space, e.g., for integer disparities (exceptions include continuous optimization techniques such as optical flow (Bergen, Anandan, Hanna *et al.* 1992) or splines (Szeliski and Coughlan 1997)). For applications such as robot navigation or people tracking, these may be perfectly adequate. However for image-based rendering, such quantized maps lead to very unappealing view synthesis results, i.e., the scene appears to be made up of many thin shearing layers. To remedy this situation, many algorithms apply a sub-pixel refinement stage after the initial discrete correspondence stage. (An alternative is to simply start with more discrete disparity levels (Szeliski and Scharstein 2004).)

Sub-pixel disparity estimates can be computed in a variety of ways, including iterative gradient descent and fitting a curve to the matching costs at discrete disparity levels (Ryan, Gray, and Hunt 1980; Lucas and Kanade 1981; Tian and Huhns 1986; Matthies, Kanade, and Szeliski 1989; Kanade and Okutomi 1994). This provides an easy way to increase the resolution of a stereo algorithm with little additional computation. However, to work well, the intensities being matched must vary smoothly, and the regions over which these estimates are computed must be on the same (correct) surface.

Recently, some questions have been raised about the advisability of fitting correlation curves to integer-sampled matching costs (Shimizu and Okutomi 2001). This situation may even be worse when sampling-insensitive dissimilarity measures are used (Birchfield and Tomasi 1998). These issues are explored in more depth by Szeliski and Scharstein (2004).

Besides sub-pixel computations, there are other ways of post-processing the computed disparities. Occluded areas can be detected using cross-checking, i.e., comparing left-to-right and right-to-left disparity maps (Fua 1993). A median filter can be applied to clean up spurious mismatches, and holes due to occlusion can be filled by surface fitting or by distributing neighboring disparity estimates (Birchfield and Tomasi 1999; Scharstein 1999; Hirschmüller and Scharstein 2009).

Another kind of post-processing, which can be useful in later processing stages, is to associate *confidences* with per-pixel depth estimates (Figure 11.10), which can be done by looking at the curvature of the correlation surface, i.e., how strong the minimum in the DSI image is at the winning disparity. Matthies, Kanade, and Szeliski (1989) show that under the assumption of small noise, photometrically calibrated images, and densely sampled disparities, the variance of a local depth estimate can be estimated as

$$\text{Var}(d) = \frac{\sigma_I^2}{a}, \quad (11.7)$$

where a is the curvature of the DSI as a function of d , which can be measured using a local parabolic fit or by squaring all the horizontal gradients in the window, and σ_I^2 is the variance of the image noise, which can be estimated from the minimum SSD score. (See also Section 6.1.4, (8.44), and Appendix B.6.)

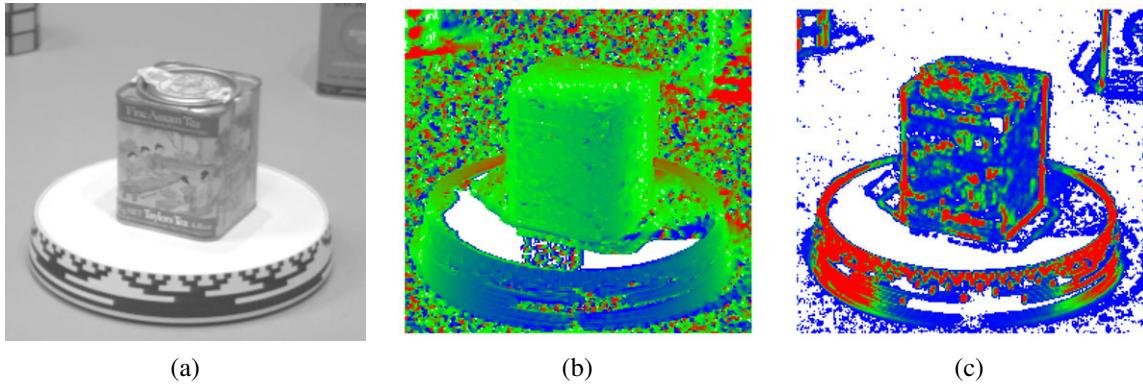


Figure 11.10 Uncertainty in stereo depth estimation (Szeliski 1991b): (a) input image; (b) estimated depth map (blue is closer); (c) estimated confidence(red is higher). As you can see, more textured areas have higher confidence.

11.4.2 Application: Stereo-based head tracking

A common application of real-time stereo algorithms is for tracking the position of a user interacting with a computer or game system. The use of stereo can dramatically improve the reliability of such a system compared to trying to use monocular color and intensity information (Darrell, Gordon, Harville *et al.* 2000). Once recovered, this information can be used in a variety of applications, including controlling a virtual environment or game, correcting the apparent gaze during video conferencing, and background replacement. We discuss the first two applications below and defer the discussion of background replacement to Section 11.5.3.

The use of head tracking to control a user’s virtual viewpoint while viewing a 3D object or environment on a computer monitor is sometimes called *fish tank virtual reality*, since the user is observing a 3D world as if it were contained inside a fish tank (Ware, Arthur, and Booth 1993). Early versions of these systems used mechanical head tracking devices and stereo glasses. Today, such systems can be controlled using stereo-based head tracking and stereo glasses can be replaced with autostereoscopic displays. Head tracking can also be used to construct a “virtual mirror”, where the user’s head can be modified in real-time using a variety of visual effects (Darrell, Baker, Crow *et al.* 1997).

Another application of stereo head tracking and 3D reconstruction is in gaze correction (Ott, Lewis, and Cox 1993). When a user participates in a desktop video-conference or video chat, the camera is usually placed on top of the monitor. Since the person is gazing at a window somewhere on the screen, it appears as if they are looking down and away from the other participants, instead of straight at them. Replacing the single camera with two or more cameras enables a virtual view to be constructed right at the position where they are looking resulting in virtual eye contact. Real-time stereo matching is used to construct an accurate 3D head model and view interpolation (Section 13.1) is used to synthesize the novel in-between view (Criminisi, Shotton, Blake *et al.* 2003).

11.5 Global optimization

Global stereo matching methods perform some optimization or iteration steps after the disparity computation phase and often skip the aggregation step altogether, because the global smoothness constraints perform a similar function. Many global methods are formulated in an energy-minimization framework, where, as we saw in Sections 3.7 (3.100–3.102) and 8.4, the objective is to find a solution d that minimizes a global energy,

$$E(d) = E_d(d) + \lambda E_s(d). \quad (11.8)$$

The data term, $E_d(d)$, measures how well the disparity function d agrees with the input image pair. Using our previously defined disparity space image, we define this energy as

$$E_d(d) = \sum_{(x,y)} C(x, y, d(x, y)), \quad (11.9)$$

where C is the (initial or aggregated) matching cost DSI.

The smoothness term $E_s(d)$ encodes the smoothness assumptions made by the algorithm. To make the optimization computationally tractable, the smoothness term is often restricted to measuring only the differences between neighboring pixels' disparities,

$$E_s(d) = \sum_{(x,y)} \rho(d(x, y) - d(x + 1, y)) + \rho(d(x, y) - d(x, y + 1)), \quad (11.10)$$

where ρ is some monotonically increasing function of disparity difference. It is also possible to use larger neighborhoods, such as \mathcal{N}_8 , which can lead to better boundaries (Boykov and Kolmogorov 2003), or to use second-order smoothness terms (Woodford, Reid, Torr *et al.* 2008), but such terms require more complex optimization techniques. An alternative to smoothness functionals is to use a lower-dimensional representation such as splines (Szeliski and Coughlan 1997).

In standard regularization (Section 3.7.1), ρ is a quadratic function, which makes d smooth everywhere and may lead to poor results at object boundaries. Energy functions that do not have this problem are called *discontinuity-preserving* and are based on robust ρ functions (Terzopoulos 1986b; Black and Rangarajan 1996). The seminal paper by Geman and Geman (1984) gave a Bayesian interpretation of these kinds of energy functions and proposed a discontinuity-preserving energy function based on Markov random fields (MRFs) and additional *line processes*, which are additional binary variables that control whether smoothness penalties are enforced or not. Black and Rangarajan (1996) show how independent line process variables can be replaced by robust pairwise disparity terms.

The terms in E_s can also be made to depend on the intensity differences, e.g.,

$$\rho_d(d(x, y) - d(x + 1, y)) \cdot \rho_I(\|I(x, y) - I(x + 1, y)\|), \quad (11.11)$$

where ρ_I is some monotonically decreasing function of intensity differences that lowers smoothness costs at high-intensity gradients. This idea (Gamble and Poggio 1987; Fua 1993; Bobick and Intille 1999; Boykov, Veksler, and Zabih 2001) encourages disparity discontinuities to coincide with intensity or color edges and appears to account for some of the good performance of global optimization approaches. While most researchers set these functions

heuristically, Scharstein and Pal (2007) show how the free parameters in such *conditional random fields* (Section 3.7.2, (3.118)) can be learned from ground truth disparity maps.

Once the global energy has been defined, a variety of algorithms can be used to find a (local) minimum. Traditional approaches associated with regularization and Markov random fields include continuation (Blake and Zisserman 1987), simulated annealing (Geman and Geman 1984; Marroquin, Mitter, and Poggio 1987; Barnard 1989), highest confidence first (Chou and Brown 1990), and mean-field annealing (Geiger and Girosi 1991).

More recently, *max-flow* and *graph cut* methods have been proposed to solve a special class of global optimization problems (Roy and Cox 1998; Boykov, Veksler, and Zabih 2001; Ishikawa 2003). Such methods are more efficient than simulated annealing and have produced good results, as have techniques based on loopy belief propagation (Sun, Zheng, and Shum 2003; Tappen and Freeman 2003). Appendix B.5 and a recent survey paper on MRF inference (Szeliski, Zabih, Scharstein *et al.* 2008) discuss and compare such techniques in more detail.

While global optimization techniques currently produce the best stereo matching results, there are some alternative approaches worth studying.

Cooperative algorithms. Cooperative algorithms, inspired by computational models of human stereo vision, were among the earliest methods proposed for disparity computation (Dev 1974; Marr and Poggio 1976; Marroquin 1983; Szeliski and Hinton 1985; Zitnick and Kanade 2000). Such algorithms iteratively update disparity estimates using non-linear operations that result in an overall behavior similar to global optimization algorithms. In fact, for some of these algorithms, it is possible to explicitly state a global function that is being minimized (Scharstein and Szeliski 1998).

Coarse-to-fine and incremental warping. Most of today’s best algorithms first enumerate all possible matches at all possible disparities and then select the best set of matches in some way. Faster approaches can sometimes be obtained using methods inspired by classic (infinitesimal) optical flow computation. Here, images are successively warped and disparity estimates incrementally updated until a satisfactory registration is achieved. These techniques are most often implemented within a coarse-to-fine hierarchical refinement framework (Quam 1984; Bergen, Anandan, Hanna *et al.* 1992; Barron, Fleet, and Beauchemin 1994; Szeliski and Coughlan 1997).

11.5.1 Dynamic programming

A different class of global optimization algorithm is based on *dynamic programming*. While the 2D optimization of Equation (11.8) can be shown to be NP-hard for common classes of smoothness functions (Veksler 1999), dynamic programming can find the global minimum for independent scanlines in polynomial time. Dynamic programming was first used for stereo vision in sparse, edge-based methods (Baker and Binford 1981; Ohta and Kanade 1985). More recent approaches have focused on the dense (intensity-based) scanline matching problem (Belhumeur 1996; Geiger, Ladendorf, and Yuille 1992; Cox, Hingorani, Rao *et al.* 1996; Bobick and Intille 1999; Birchfield and Tomasi 1999). These approaches work by computing the minimum-cost path through the matrix of all pairwise matching costs between two corresponding scanlines, i.e., through a horizontal slice of the DSI. Partial occlusion is

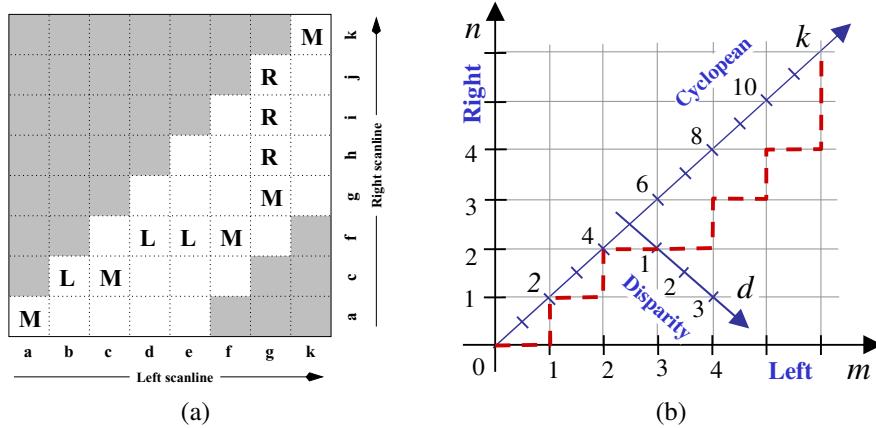


Figure 11.11 Stereo matching using dynamic programming, as illustrated by (a) Scharstein and Szeliski (2002) © 2002 Springer and (b) Kolmogorov, Criminisi, Blake *et al.* (2006). © 2006 IEEE. For each pair of corresponding scanlines, a minimizing path through the matrix of all pairwise matching costs (DSI) is selected. Lowercase letters (a–k) symbolize the intensities along each scanline. Uppercase letters represent the selected path through the matrix. Matches are indicated by M, while partially occluded points (which have a fixed cost) are indicated by L or R, corresponding to points only visible in the left or right images, respectively. Usually, only a limited disparity range is considered (0–4 in the figure, indicated by the non-shaded squares). The representation in (a) allows for diagonal moves while the representation in (b) does not. Note that these diagrams, which use the *Cyclopean* representation of depth, i.e., depth relative to a camera between the two input cameras, show an “unskewed” x - d slice through the DSI.

handled explicitly by assigning a group of pixels in one image to a single pixel in the other image. Figure 11.11 schematically shows how DP works, while Figure 11.5f shows a real DSI slice over which the DP is applied.

To implement dynamic programming for a scanline y , each entry (state) in a 2D cost matrix $D(m, n)$ is computed by combining its DSI value

$$C'(m, n) = C(m + n, m - n, y) \quad (11.12)$$

with one of its predecessor cost values. Using the representation shown in Figure 11.11a, which allows for “diagonal” moves, the aggregated match costs can be recursively computed as

$$\begin{aligned}
D(m, n, M) &= \min(D(m-1, n-1, M), D(m-1, n, L), D(m-1, n-1, R)) \\
&\quad + C'(m, n) \\
D(m, n, L) &= \min(D(m-1, n-1, M), D(m-1, n, L)) + O \\
D(m, n, R) &= \min(D(m, n-1, M), D(m, n-1, R)) + O,
\end{aligned} \tag{11.13}$$

where O is a per-pixel occlusion cost. The aggregation rules corresponding to Figure 11.11b are given by Kolmogorov, Criminisi, Blake *et al.* (2006), who also use a two-state foreground–background model for bi-layer segmentation.

Problems with dynamic programming stereo include the selection of the right cost for occluded pixels and the difficulty of enforcing inter-scanline consistency, although several

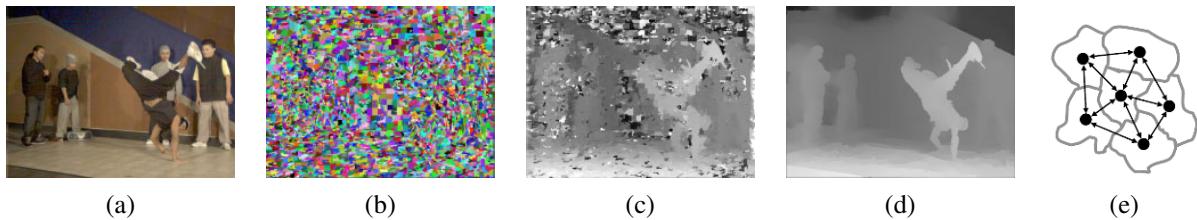


Figure 11.12 Segmentation-based stereo matching (Zitnick, Kang, Uyttendaele *et al.* 2004) © 2004 ACM: (a) input color image; (b) color-based segmentation; (c) initial disparity estimates; (d) final piecewise-smoothed disparities; (e) MRF neighborhood defined over the segments in the disparity space distribution (Zitnick and Kang 2007) © 2007 Springer.

methods propose ways of addressing the latter (Ohta and Kanade 1985; Belhumeur 1996; Cox, Hingorani, Rao *et al.* 1996; Bobick and Intille 1999; Birchfield and Tomasi 1999; Kolmogorov, Criminisi, Blake *et al.* 2006). Another problem is that the dynamic programming approach requires enforcing the *monotonicity* or *ordering constraint* (Yuille and Poggio 1984). This constraint requires that the relative ordering of pixels on a scanline remain the same between the two views, which may not be the case in scenes containing narrow foreground objects.

An alternative to traditional dynamic programming, introduced by Scharstein and Szeliski (2002), is to neglect the vertical smoothness constraints in (11.10) and simply optimize independent scanlines in the global energy function (11.8), which can easily be done using a recursive algorithm,

$$D(x, y, d) = C(x, y, d) + \min_{d'} \{ D(x - 1, y, d') + \rho_d(d - d') \}. \quad (11.14)$$

The advantage of this *scanline optimization* algorithm is that it computes the same representation and minimizes a reduced version of the same energy function as the full 2D energy function (11.8). Unfortunately, it still suffers from the same streaking artifacts as dynamic programming.

A much better approach is to evaluate the cumulative cost function (11.14) from multiple directions, e.g., from the eight cardinal directions, N, E, W, S, NE, SE, SW, NW (Hirschmüller 2008). The resulting *semi-global* optimization performs quite well and is extremely efficient to implement.

Even though dynamic programming and scanline optimization algorithms do not generally produce *the* most accurate stereo reconstructions, when combined with sophisticated aggregation strategies, they can produce very fast and high-quality results.

11.5.2 Segmentation-based techniques

While most stereo matching algorithms perform their computations on a per-pixel basis, some of the more recent techniques first segment the images into regions and then try to label each region with a disparity.

For example, Tao, Sawhney, and Kumar (2001) segment the reference image, estimate per-pixel disparities using a local technique, and then do local plane fits inside each segment



Figure 11.13 Stereo matching with adaptive over-segmentation and matting (Taguchi, Wilburn, and Zitnick 2008) © 2008 IEEE: (a) segment boundaries are refined during the optimization, leading to more accurate results (e.g., the thin green leaf in the bottom row); (b) alpha mattes are extracted at segment boundaries, which leads to visually better compositing results (middle column).

before applying smoothness constraints between neighboring segments. Zitnick, Kang, Uyttendaele *et al.* (2004) and Zitnick and Kang (2007) use over-segmentation to mitigate initial bad segmentations. After a set of initial cost values for each segment has been stored into a *disparity space distribution* (DSD), iterative relaxation (or loopy belief propagation, in the more recent work of Zitnick and Kang (2007)) is used to adjust the disparity estimates for each segment, as shown in Figure 11.12. Taguchi, Wilburn, and Zitnick (2008) refine the segment shapes as part of the optimization process, which leads to much improved results, as shown in Figure 11.13.

Even more accurate results are obtained by Klaus, Sormann, and Karner (2006), who first segment the reference image using mean shift, run a small (3×3) SAD plus gradient SAD (weighted by cross-checking) to get initial disparity estimates, fit local planes, re-fit with global planes, and then run a final MRF on plane assignments with loopy belief propagation. When the algorithm was first introduced in 2006, it was the top ranked algorithm on the evaluation site at <http://vision.middlebury.edu/stereo>; in early 2010, it still had the top rank on the new evaluation datasets.

The highest ranked algorithm, by Wang and Zheng (2008), follows a similar approach of segmenting the image, doing local plane fits, and then performing cooperative optimization of neighboring plane fit parameters. Another highly ranked algorithm, by Yang, Wang, Yang *et al.* (2009), uses the color correlation approach of Yoon and Kweon (2006) and hierarchical belief propagation to obtain an initial set of disparity estimates. After left-right consistency checking to detect occluded pixels, the data terms for low-confidence and occluded pixels are recomputed using segmentation-based plane fits and one or more rounds of hierarchical belief propagation are used to obtain the final disparity estimates.

Another important ability of segmentation-based stereo algorithms, which they share with algorithms that use explicit layers (Baker, Szeliski, and Anandan 1998; Szeliski and Golland 1999) or boundary extraction (Hasinoff, Kang, and Szeliski 2006), is the ability to extract fractional pixel alpha mattes at depth discontinuities (Bleyer, Gelautz, Rother *et al.* 2009). This ability is crucial when attempting to create virtual view interpolation without clinging boundary or tearing artifacts (Zitnick, Kang, Uyttendaele *et al.* 2004) and also to seamlessly insert virtual objects (Taguchi, Wilburn, and Zitnick 2008), as shown in Figure 11.13b.



Figure 11.14 Background replacement using *z-keying* with a bi-layer segmentation algorithm (Kolmogorov, Criminisi, Blake *et al.* 2006) © 2006 IEEE.

Since new stereo matching algorithms continue to be introduced every year, it is a good idea to periodically check the Middlebury evaluation site at <http://vision.middlebury.edu/stereo> for a listing of the most recent algorithms to be evaluated.

11.5.3 Application: Z-keying and background replacement

Another application of real-time stereo matching is *z-keying*, which is the process of segmenting a foreground actor from the background using depth information, usually for the purpose of replacing the background with some computer-generated imagery, as shown in Figure 11.2g.

Originally, *z-keying* systems required expensive custom-built hardware to produce the desired depth maps in real time and were, therefore, restricted to broadcast studio applications (Kanade, Yoshida, Oda *et al.* 1996; Iddan and Yahav 2001). Off-line systems were also developed for estimating 3D multi-viewpoint geometry from video streams (Section 13.5.4) (Kanade, Rander, and Narayanan 1997; Carranza, Theobalt, Magnor *et al.* 2003; Zitnick, Kang, Uyttendaele *et al.* 2004; Vedula, Baker, and Kanade 2005). Recent advances in highly accurate real-time stereo matching, however, now make it possible to perform *z-keying* on regular PCs, enabling desktop videoconferencing applications such as those shown in Figure 11.14 (Kolmogorov, Criminisi, Blake *et al.* 2006).

11.6 Multi-view stereo

While matching pairs of images is a useful way of obtaining depth information, matching more images can lead to even better results. In this section, we review not only techniques for creating complete 3D object models, but also simpler techniques for improving the quality of depth maps using multiple source images.

As we saw in our discussion of plane sweep (Section 11.1.2), it is possible to resample all neighboring k images at each disparity hypothesis d into a generalized disparity space

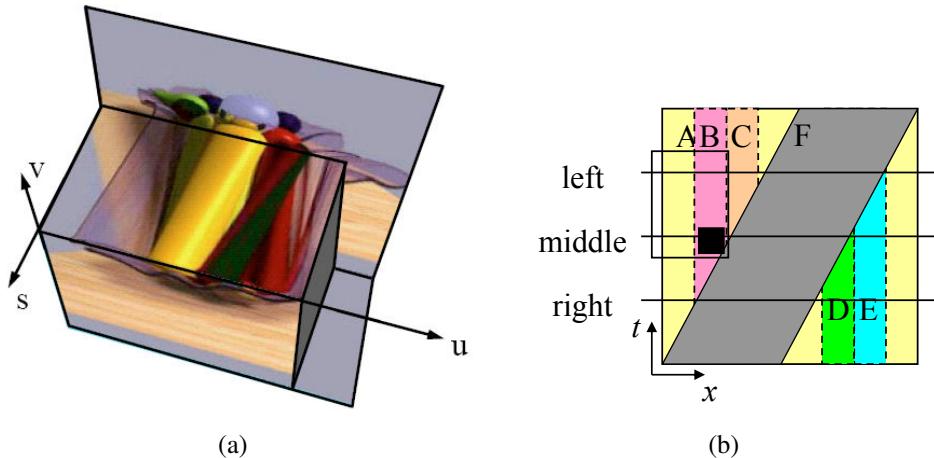


Figure 11.15 Epipolar plane image (EPI) (Gortler, Grzeszczuk, Szeliski *et al.* 1996) © 1996 ACM and a schematic EPI (Kang, Szeliski, and Chai 2001) © 2001 IEEE. (a) The Lumigraph (light field) (Section 13.3) is the 4D space of all light rays passing through a volume of space. Taking a 2D slice results in all of the light rays embedded in a plane and is equivalent to a scanline taken from a stacked EPI volume. Objects at different depths move sideways with velocities (slopes) proportional to their inverse depth. Occlusion (and translucency) effects can easily be seen in this representation. (b) The EPI corresponding to Figure 11.16 showing the three images (middle, left, and right) as slices through the EPI volume. The spatially and temporally shifted window around the black pixel is indicated by the rectangle, showing the right image is not being used in matching.

volume $\tilde{I}(x, y, d, k)$. The simplest way to take advantage of these additional images is to sum up their differences from the reference image I_r as in (11.4),

$$C(x, y, d) = \sum_k \rho(\tilde{I}(x, y, d, k) - I_r(x, y)). \quad (11.15)$$

This is the basis of the well-known sum of summed-squared-difference (SSSD) and SSAD approaches (Okutomi and Kanade 1993; Kang, Webb, Zitnick *et al.* 1995), which can be extended to reason about likely patterns of occlusion (Nakamura, Matsuura, Satoh *et al.* 1996). More recent work by Gallup, Frahm, Mordohai *et al.* (2008) show how to adapt the baselines used to the expected depth in order to get the best tradeoff between geometric accuracy (wide baseline) and robustness to occlusion (narrow baseline). Alternative multi-view cost metrics include measures such as synthetic focus sharpness and the entropy of the pixel color distribution (Vaish, Szeliski, Zitnick *et al.* 2006).

A useful way to visualize the multi-frame stereo estimation problem is to examine the *epipolar plane image* (EPI) formed by stacking corresponding scanlines from all the images, as shown in Figures 8.13c and 11.15 (Bolles, Baker, and Marimont 1987; Baker and Bolles 1989; Baker 1989). As you can see in Figure 11.15, as a camera translates horizontally (in a standard horizontally rectified geometry), objects at different depths move sideways at a rate inversely proportional to their depth (11.1).⁶ Foreground objects occlude background objects,

⁶ The four-dimensional generalization of the EPI is the *light field*, which we study in Section 13.3. In principle, there is enough information in a light field to recover both the shape and the BRDF of objects (Soatto, Yezzi, and Jin 2003), although relatively little progress has been made to date on this topic.

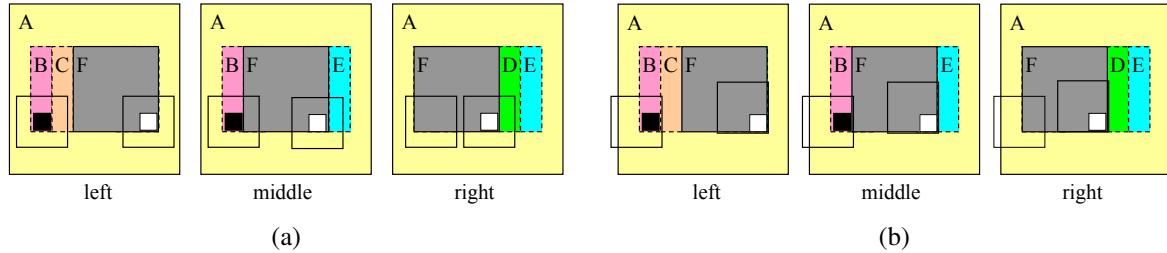


Figure 11.16 Spatio-temporally shiftable windows (Kang, Szeliski, and Chai 2001) © 2001 IEEE: A simple three-image sequence (the middle image is the reference image), which has a moving frontal gray square (marked F) and a stationary background. Regions B, C, D, and E are partially occluded. (a) A regular SSD algorithm will make mistakes when matching pixels in these regions (e.g. the window centered on the black pixel in region B) and in windows straddling depth discontinuities (the window centered on the white pixel in region F). (b) Shiftable windows help mitigate the problems in partially occluded regions and near depth discontinuities. The shifted window centered on the white pixel in region F matches correctly in all frames. The shifted window centered on the black pixel in region B matches correctly in the left image, but requires temporal selection to disable matching the right image. Figure 11.15b shows an EPI corresponding to this sequence and describes in more detail how temporal selection works.

which can be seen as *EPI-strips* (Criminisi, Kang, Swaminathan *et al.* 2005) occluding other strips in the EPI. If we are given a dense enough set of images, we can find such strips and reason about their relationships in order to both reconstruct the 3D scene and make inferences about translucent objects (Tsin, Kang, and Szeliski 2006) and specular reflections (Swaminathan, Kang, Szeliski *et al.* 2002; Criminisi, Kang, Swaminathan *et al.* 2005). Alternatively, we can treat the series of images as a set of sequential observations and merge them using Kalman filtering (Matthies, Kanade, and Szeliski 1989) or maximum likelihood inference (Cox 1994).

When fewer images are available, it becomes necessary to fall back on aggregation techniques such as sliding windows or global optimization. With additional input images, however, the likelihood of occlusions increases. It is therefore prudent to adjust not only the best window locations using a shiftable window approach, as shown in Figure 11.16a, but also to optionally select a subset of neighboring frames in order to discount those images where the region of interest is occluded, as shown in Figure 11.16b (Kang, Szeliski, and Chai 2001). Figure 11.15b shows how such spatio-temporal selection or shifting of windows corresponds to selecting the most likely un-occluded volumetric region in the epipolar plane image volume.

The results of applying these techniques to the multi-frame *flower garden* image sequence are shown in Figure 11.17, which compares the results of using regular (non-shifted) SSSD with spatially shifted windows and full spatio-temporal window selection. (The task of applying stereo to a rigid scene filmed with a moving camera is sometimes called *motion stereo*). Similar improvements from using spatio-temporal selection are reported by (Kang and Szeliski 2004) and are evident even when local measurements are combined with global optimization.

While computing a depth map from multiple inputs outperforms pairwise stereo matching, even more dramatic improvements can be obtained by estimating multiple depth maps

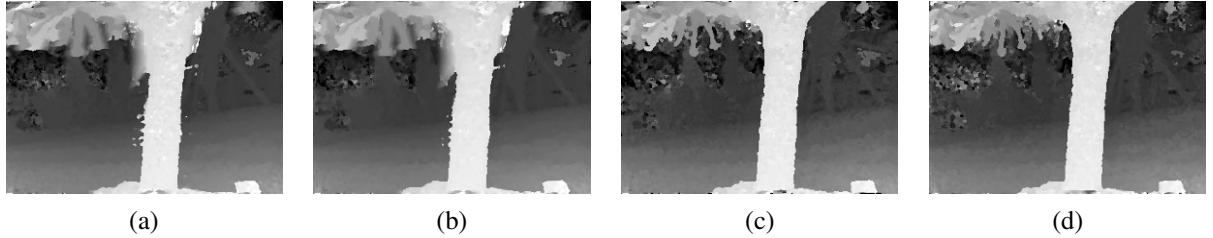


Figure 11.17 Local (5×5 window-based) matching results (Kang, Szeliski, and Chai 2001) © 2001 IEEE: (a) window that is not spatially perturbed (centered); (b) spatially perturbed window; (c) using the best five of 10 neighboring frames; (d) using the better half sequence. Notice how the results near the tree trunk are improved using temporal selection.

simultaneously (Szeliski 1999; Kang and Szeliski 2004). The existence of multiple depth maps enables more accurate reasoning about occlusions, as regions which are occluded in one image may be visible (and matchable) in others. The multi-view reconstruction problem can be formulated as the simultaneous estimation of depth maps at key frames (Figure 8.13c) while maximizing not only photoconsistency and piecewise disparity smoothness but also the consistency between disparity estimates at different frames. While Szeliski (1999) and Kang and Szeliski (2004) use soft (penalty-based) constraints to encourage multiple disparity maps to be consistent, Kolmogorov and Zabih (2002) show how such consistency measures can be encoded as hard constraints, which guarantee that the multiple depth maps are not only similar but actually identical in overlapping regions. Newer algorithms that simultaneously estimate multiple disparity maps include papers by Maitre, Shinagawa, and Do (2008) and Zhang, Jia, Wong *et al.* (2008).

A closely related topic to multi-frame stereo estimation is *scene flow*, in which multiple cameras are used to capture a dynamic scene. The task is then to simultaneously recover the 3D shape of the object at every instant in time and to estimate the full 3D motion of every surface point between frames. Representative papers in this area include those by Vedula, Baker, Rander *et al.* (2005), Zhang and Kambhamettu (2003), Pons, Keriven, and Faugeras (2007), Huguet and Devernay (2007), and Wedel, Rabe, Vaudrey *et al.* (2008). Figure 11.18a shows an image of the 3D scene flow for the tango dancer shown in Figure 11.2h–j, while Figure 11.18b shows 3D scene flows captured from a moving vehicle for the purpose of obstacle avoidance. In addition to supporting mensuration and safety applications, scene flow can be used to support both spatial and temporal view interpolation (Section 13.5.4), as demonstrated by Vedula, Baker, and Kanade (2005).

11.6.1 Volumetric and 3D surface reconstruction

According to Seitz, Curless, Diebel *et al.* (2006):

The goal of multi-view stereo is to reconstruct a complete 3D object model from a collection of images taken from known camera viewpoints.

The most challenging but potentially most useful variant of multi-view stereo reconstruction is to create globally consistent 3D models. This topic has a long history in computer vision, starting with surface mesh reconstruction techniques such as the one developed by

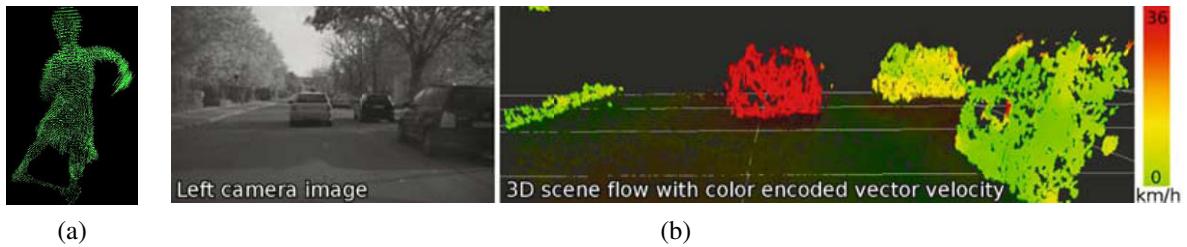


Figure 11.18 Three-dimensional scene flow: (a) computed from a multi-camera dome surrounding the dancer shown in Figure 11.2h–j (Vedula, Baker, Rander *et al.* 2005) © 2005 IEEE; (b) computed from stereo cameras mounted on a moving vehicle (Wedel, Rabe, Vaudrey *et al.* 2008) © 2008 Springer.

Fua and Leclerc (1995) (Figure 11.19a). A variety of approaches and representations have been used to solve this problem, including 3D voxel representations (Seitz and Dyer 1999; Szeliski and Golland 1999; De Bonet and Viola 1999; Kutulakos and Seitz 2000; Eisert, Steinbach, and Giros 2000; Slabaugh, Culbertson, Slabaugh *et al.* 2004; Sinha and Pollefeys 2005; Vogiatzis, Hernandez, Torr *et al.* 2007; Hiep, Keriven, Pons *et al.* 2009), level sets (Faugeras and Keriven 1998; Pons, Keriven, and Faugeras 2007), polygonal meshes (Fua and Leclerc 1995; Narayanan, Rander, and Kanade 1998; Hernandez and Schmitt 2004; Furukawa and Ponce 2009), and multiple depth maps (Kolmogorov and Zabih 2002). Figure 11.19 shows representative examples of 3D object models reconstructed using some of these techniques.

In order to organize and compare all these techniques, Seitz, Curless, Diebel *et al.* (2006) developed a six-point taxonomy that can help classify algorithms according to the *scene representation*, *photoconsistency measure*, *visibility model*, *shape priors*, *reconstruction algorithm*, and *initialization requirements* they use. Below, we summarize some of these choices and list a few representative papers. For more details, please consult the full survey paper (Seitz, Curless, Diebel *et al.* 2006) and the evaluation Web site, <http://vision.middlebury.edu/mview/>, which contains pointers to even more recent papers and results.

Scene representation. One of the more popular 3D representations is a uniform grid of 3D voxels,⁷ which can be reconstructed using a variety of carving (Seitz and Dyer 1999; Kutulakos and Seitz 2000) or optimization (Sinha and Pollefeys 2005; Vogiatzis, Hernandez, Torr *et al.* 2007; Hiep, Keriven, Pons *et al.* 2009) techniques. Level set techniques (Section 5.1.4) also operate on a uniform grid but, instead of representing a binary occupancy map, they represent the signed distance to the surface (Faugeras and Keriven 1998; Pons, Keriven, and Faugeras 2007), which can encode a finer level of detail. Polygonal meshes are another popular representation (Fua and Leclerc 1995; Narayanan, Rander, and Kanade 1998; Isidoro and Sclaroff 2003; Hernandez and Schmitt 2004; Furukawa and Ponce 2009; Hiep, Keriven, Pons *et al.* 2009). Meshes are the standard representation used in computer graphics and also readily support the computation of visibility and occlusions. Finally, as we discussed in the previous section, multiple depth maps can also be used (Szeliski 1999; Kolmogorov and Zabih 2002; Kang and Szeliski 2004). Many algorithms also use more than a single representation, e.g., they may start by computing multiple depth maps and then merge

⁷ For outdoor scenes that go to infinity, a non-uniform gridding of space may be preferable (Slabaugh, Culbertson, Slabaugh *et al.* 2004).

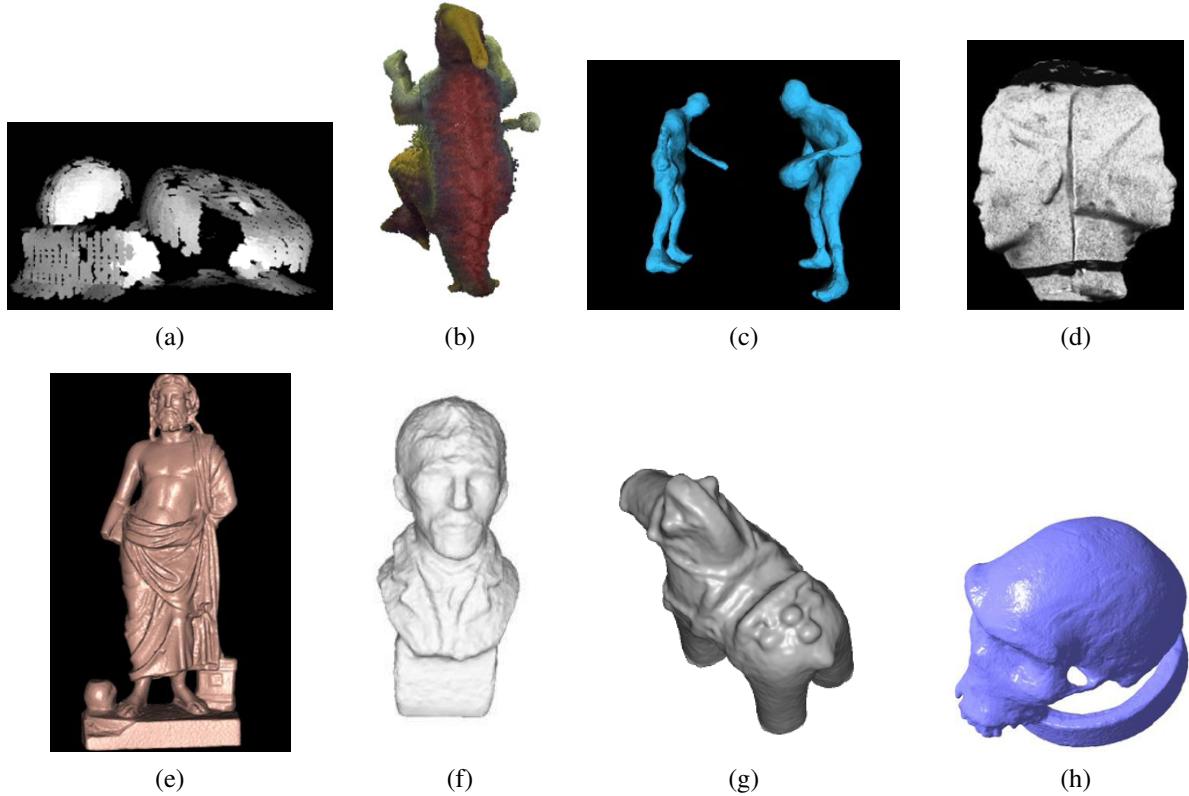


Figure 11.19 Multi-view stereo algorithms: (a) surface-based stereo (Fua and Leclerc 1995); (b) voxel coloring (Seitz and Dyer 1999) © 1999 Springer; (c) depth map merging (Narayanan, Rander, and Kanade 1998); (d) level set evolution (Faugeras and Keriven 1998) © 1998 IEEE; (e) silhouette and stereo fusion (Hernandez and Schmitt 2004) © 2004 Elsevier; (f) multi-view image matching (Pons, Keriven, and Faugeras 2005) © 2005 IEEE; (g) volumetric graph cut (Vogiatzis, Torr, and Cipolla 2005) © 2005 IEEE; (h) carved visual hulls (Furukawa and Ponce 2009) © 2009 Springer.

them into a 3D object model (Narayanan, Rander, and Kanade 1998; Furukawa and Ponce 2009; Goesele, Curless, and Seitz 2006; Goesele, Snavely, Curless *et al.* 2007; Furukawa, Curless, Seitz *et al.* 2010).

Photoconsistency measure. As we discussed in (Section 11.3.1), a variety of similarity measures can be used to compare pixel values in different images, including measures that try to discount illumination effects or be less sensitive to outliers. In multi-view stereo, algorithms have a choice of computing these measures directly on the surface of the model, i.e., in *scene space*, or projecting pixel values from one image (or from a textured model) back into another image, i.e., in *image space*. (The latter corresponds more closely to a Bayesian approach, since input images are noisy measurements of the colored 3D model.) The geometry of the object, i.e., its distance to each camera and its local surface normal, when available, can be used to adjust the matching windows used in the computation to account for foreshortening and scale change (Goesele, Snavely, Curless *et al.* 2007).

Visibility model. A big advantage that multi-view stereo algorithms have over single-depth-map approaches is their ability to reason in a principled manner about visibility and occlusions. Techniques that use the current state of the 3D model to predict which surface pixels are visible in each image (Kutulakos and Seitz 2000; Faugeras and Keriven 1998; Vogiatzis, Hernandez, Torr *et al.* 2007; Hiep, Keriven, Pons *et al.* 2009) are classified as using *geometric visibility models* in the taxonomy of Seitz, Curless, Diebel *et al.* (2006). Techniques that select a neighboring subset of image to match are called *quasi-geometric* (Narayanan, Rander, and Kanade 1998; Kang and Szeliski 2004; Hernandez and Schmitt 2004), while techniques that use traditional robust similarity measures are called *outlier-based*. While full geometric reasoning is the most principled and accurate approach, it can be very slow to evaluate and depends on the evolving quality of the current surface estimate to predict visibility, which can be a bit of a chicken-and-egg problem, unless conservative assumptions are used, as they are by Kutulakos and Seitz (2000).

Shape priors. Because stereo matching is often underconstrained, especially in textureless regions, most matching algorithms adopt (either explicitly or implicitly) some form of prior model for the expected shape. Many of the techniques that rely on optimization use a 3D smoothness or area-based photoconsistency constraint, which, because of the natural tendency of smooth surfaces to shrink inwards, often results in a *minimal surface* prior (Faugeras and Keriven 1998; Sinha and Pollefeys 2005; Vogiatzis, Hernandez, Torr *et al.* 2007). Approaches that carve away the volume of space often stop once a photoconsistent solution is found (Seitz and Dyer 1999; Kutulakos and Seitz 2000), which corresponds to a *maximal surface* bias, i.e., these techniques tend to over-estimate the true shape. Finally, multiple depth map approaches often adopt traditional *image-based* smoothness (regularization) constraints.

Reconstruction algorithm. The details of how the actual reconstruction algorithm proceeds is where the largest variety—and greatest innovation—in multi-view stereo algorithms can be found.

Some approaches use global optimization defined over a three-dimensional photoconsistency volume to recover a complete surface. Approaches based on graph cuts use polynomial complexity binary segmentation algorithms to recover the object model defined on the voxel grid (Sinha and Pollefeys 2005; Vogiatzis, Hernandez, Torr *et al.* 2007; Hiep, Keriven, Pons *et al.* 2009). Level set approaches use a continuous surface evolution to find a good minimum in the configuration space of potential surfaces and therefore require a reasonably good initialization (Faugeras and Keriven 1998; Pons, Keriven, and Faugeras 2007). In order for the photoconsistency volume to be meaningful, matching costs need to be computed in some robust fashion, e.g., using sets of limited views or by aggregating multiple depth maps.

An alternative approach to global optimization is to sweep through the 3D volume while computing both photoconsistency and visibility simultaneously. The *voxel coloring* algorithm of Seitz and Dyer (1999) performs a front-to-back plane sweep. On every plane, any voxels that are sufficiently photoconsistent are labeled as part of the object. The corresponding pixels in the source images can then be “erased”, since they are already accounted for, and therefore do not contribute to further photoconsistency computations. (A similar approach, albeit without the front-to-back sweep order, is used by Szeliski and Golland (1999).) The resulting 3D volume, under noise- and resampling-free conditions, is guaranteed to produce

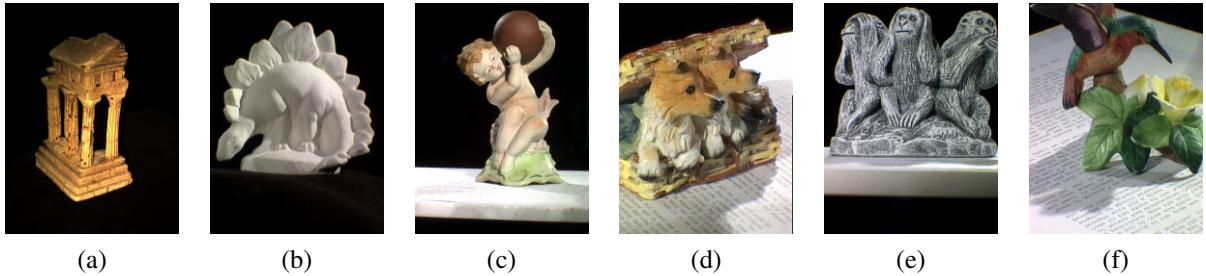


Figure 11.20 The multi-view stereo data sets captured by Seitz, Curless, Diebel *et al.* (2006) © 2006 Springer. Only (a) and (b) are currently used for evaluation.

both a photoconsistent 3D model and to enclose whatever true 3D object model generated the images.

Unfortunately, voxel coloring is only guaranteed to work if all of the cameras lie on the same side of the sweep planes, which is not possible in general ring configurations of cameras. Kutulakos and Seitz (2000) generalize voxel coloring to *space carving*, where subsets of cameras that satisfy the voxel coloring constraint are iteratively selected and the 3D voxel grid is alternately carved away along different axes.

Another popular approach to multi-view stereo is to first independently compute multiple depth maps and then merge these partial maps into a complete 3D model. Approaches to depth map merging, which are discussed in more detail in Section 12.2.1, include signed distance functions (Curless and Levoy 1996), used by Goesele, Curless, and Seitz (2006), and Poisson surface reconstruction (Kazhdan, Bolitho, and Hoppe 2006), used by Goesele, Snavely, Curless *et al.* (2007). It is also possible to reconstruct sparser representations, such as 3D points and lines, and to interpolate them to full 3D surfaces (Section 12.3.1) (Taylor 2003).

Initialization requirements. One final element discussed by Seitz, Curless, Diebel *et al.* (2006) is the varying degrees of initialization required by different algorithms. Because some algorithms refine or evolve a rough 3D model, they require a reasonably accurate (or overcomplete) initial model, which can often be obtained by reconstructing a volume from object silhouettes, as discussed in Section 11.6.2. However, if the algorithm performs a global optimization (Kolev, Klodt, Brox *et al.* 2009; Kolev and Cremers 2009), this dependence on initialization is not an issue.

Empirical evaluation. In order to evaluate the large number of design alternatives in multi-view stereo, Seitz, Curless, Diebel *et al.* (2006) collected a dataset of calibrated images using a spherical gantry. A representative image from each of the six datasets is shown in Figure 11.20, although only the first two datasets have as yet been fully processed and used for evaluation. Figure 11.21 shows the results of running seven different algorithms on the *temple* dataset. As you can see, most of the techniques do an impressive job of capturing the fine details in the columns, although it is also clear that the techniques employ differing amounts of smoothing to achieve these results.

Since the publication of the survey by Seitz, Curless, Diebel *et al.* (2006), the field of

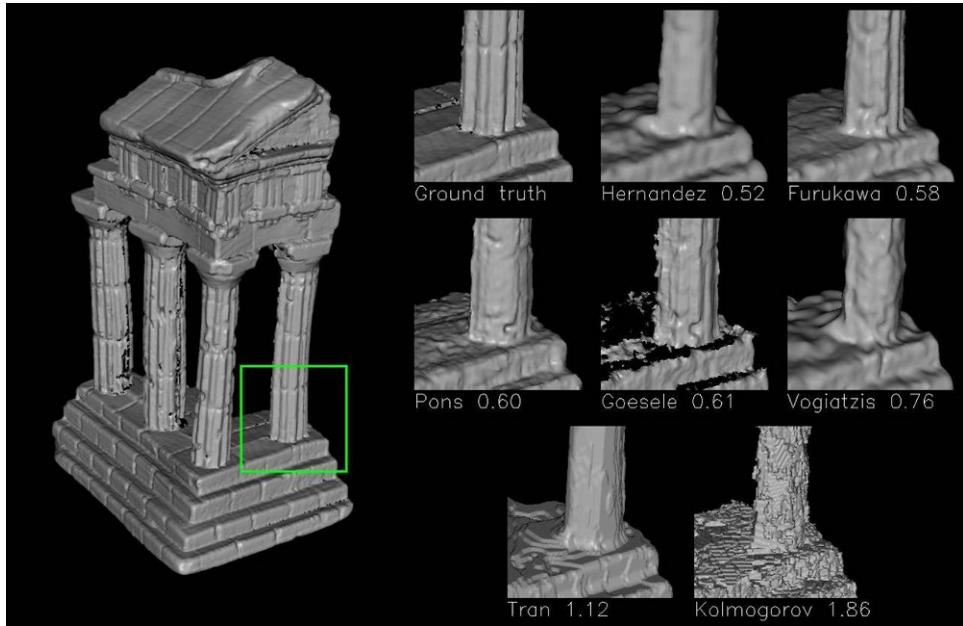


Figure 11.21 Reconstruction results (details) for seven algorithms (Hernandez and Schmitt 2004; Furukawa and Ponce 2009; Pons, Keriven, and Faugeras 2005; Goesele, Curless, and Seitz 2006; Vogiatzis, Torr, and Cipolla 2005; Tran and Davis 2002; Kolmogorov and Zabih 2002) evaluated by Seitz, Curless, Diebel *et al.* (2006) on the 47-image Temple Ring dataset. The numbers underneath each detail image are the accuracy of each of these techniques measured in millimeters.

multi-view stereo has continued to advance at a rapid pace (Strecha, Fransens, and Van Gool 2006; Hernandez, Vogiatzis, and Cipolla 2007; Habbecke and Kobbelt 2007; Furukawa and Ponce 2007; Vogiatzis, Hernandez, Torr *et al.* 2007; Goesele, Snavely, Curless *et al.* 2007; Sinha, Mordohai, and Pollefeys 2007; Gargallo, Prados, and Sturm 2007; Merrell, Akbarzadeh, Wang *et al.* 2007; Zach, Pock, and Bischof 2007b; Furukawa and Ponce 2008; Hornung, Zeng, and Kobbelt 2008; Bradley, Boubekeur, and Heidrich 2008; Zach 2008; Campbell, Vogiatzis, Hernández *et al.* 2008; Kolev, Klodt, Brox *et al.* 2009; Hiep, Keriven, Pons *et al.* 2009; Furukawa, Curless, Seitz *et al.* 2010). The multi-view stereo evaluation site, <http://vision.middlebury.edu/mview/>, provides quantitative results for these algorithms along with pointers to where to find these papers.

11.6.2 Shape from silhouettes

In many situations, performing a foreground–background segmentation of the object of interest is a good way to initialize or fit a 3D model (Grauman, Shakhnarovich, and Darrell 2003; Vlasic, Baran, Matusik *et al.* 2008) or to impose a convex set of constraints on multi-view stereo (Kolev and Cremers 2008). Over the years, a number of techniques have been developed to reconstruct a 3D volumetric model from the intersection of the binary silhouettes projected into 3D. The resulting model is called a *visual hull* (or sometimes a *line hull*), analogous with the convex hull of a set of points, since the volume is maximal with respect

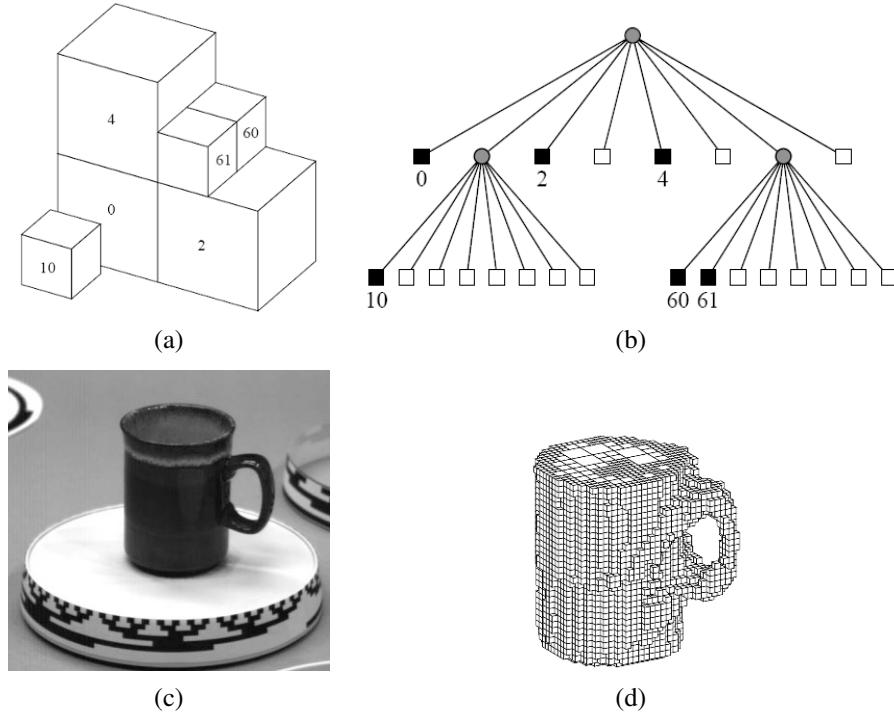


Figure 11.22 Volumetric octree reconstruction from binary silhouettes (Szeliski 1993) © 1993 Elsevier: (a) octree representation and its corresponding (b) tree structure; (c) input image of an object on a turntable; (d) computed 3D volumetric octree model.

to the visual silhouettes and surface elements are tangent to the viewing rays (lines) along the silhouette boundaries (Laurentini 1994). It is also possible to carve away a more accurate reconstruction using multi-view stereo (Sinha and Pollefeys 2005) or by analyzing cast shadows (Savarese, Andreetto, Rushmeier *et al.* 2007).

Some techniques first approximate each silhouette with a polygonal representation and then intersect the resulting faceted conical regions in three-space to produce polyhedral models (Baumgart 1974; Martin and Aggarwal 1983; Matusik, Buehler, and McMillan 2001), which can later be refined using triangular splines (Sullivan and Ponce 1998). Other approaches use voxel-based representations, usually encoded as octrees (Samet 1989), because of the resulting space-time efficiency. Figures 11.22a–b show an example of a 3D octree model and its associated colored tree, where black nodes are interior, white nodes are exterior, and gray nodes are of mixed occupancy. Examples of octree-based reconstruction approaches include those by Potmesil (1987), Noborio, Fukada, and Arimoto (1988), Srivasan, Liang, and Hackwood (1990), and Szeliski (1993).

The approach of Szeliski (1993) first converts each binary silhouette into a one-sided variant of a distance map, where each pixel in the map indicates the largest square that is completely inside (or outside) the silhouette. This makes it fast to project an octree cell into the silhouette to confirm whether it is completely inside or outside the object, so that it can be colored black, white, or left as gray (mixed) for further refinement on a smaller grid. The octree construction algorithm proceeds in a coarse-to-fine manner, first building an

octree at a relatively coarse resolution, and then refining it by revisiting and subdividing all the input images for the gray (mixed) cells whose occupancy has not yet been determined. Figure 11.22d shows the resulting octree model computed from a coffee cup rotating on a turntable.

More recent work on visual hull computation borrows ideas from image-based rendering, and is hence called an *image-based visual hull* (Matusik, Buehler, Raskar *et al.* 2000). Instead of precomputing a global 3D model, an image-based visual hull is recomputed for each new viewpoint, by successively intersecting viewing ray segments with the binary silhouettes in each image. This not only leads to a fast computation algorithm but also enables fast texturing of the recovered model with color values from the input images. This approach can also be combined with high-quality deformable templates to capture and re-animate whole body motion (Vlasic, Baran, Matusik *et al.* 2008).

11.7 Additional reading

The field of stereo correspondence and depth estimation is one of the oldest and most widely studied topics in computer vision. A number of good surveys have been written over the years (Marr and Poggio 1976; Barnard and Fischler 1982; Dhond and Aggarwal 1989; Scharstein and Szeliski 2002; Brown, Burschka, and Hager 2003; Seitz, Curless, Diebel *et al.* 2006) and they can serve as good guides to this extensive literature.

Because of computational limitations and the desire to find appearance-invariant correspondences, early algorithms often focused on finding *sparse correspondences* (Hannah 1974; Marr and Poggio 1979; Mayhew and Frisby 1980; Baker and Binford 1981; Arnold 1983; Grimson 1985; Ohta and Kanade 1985; Bolles, Baker, and Marimont 1987; Matthies, Kanade, and Szeliski 1989; Hsieh, McKeown, and Perlant 1992; Bolles, Baker, and Hannah 1993).

The topic of computing epipolar geometry and pre-rectifying images is covered in Sections 7.2 and 11.1 and is also treated in textbooks on multi-view geometry (Faugeras and Luong 2001; Hartley and Zisserman 2004) and articles specifically on this topic (Torr and Murray 1997; Zhang 1998a,b). The concepts of the *disparity space* and *disparity space image* are often associated with the seminal work by Marr (1982) and the papers of Yang, Yuille, and Lu (1993) and Intille and Bobick (1994). The plane sweep algorithm was first popularized by Collins (1996) and then generalized to a full arbitrary projective setting by Szeliski and Golland (1999) and Saito and Kanade (1999). Plane sweeps can also be formulated using cylindrical surfaces (Ishiguro, Yamamoto, and Tsuji 1992; Kang and Szeliski 1997; Shum and Szeliski 1999; Li, Shum, Tang *et al.* 2004; Zheng, Kang, Cohen *et al.* 2007) or even more general topologies (Seitz 2001).

Once the topology for the cost volume or DSI has been set up, we need to compute the actual photoconsistency measures for each pixel and potential depth. A wide range of such measures have been proposed, as discussed in Section 11.3.1. Some of these are compared in recent surveys and evaluations of matching costs (Scharstein and Szeliski 2002; Hirschmüller and Scharstein 2009).

To compute an actual depth map from these costs, some form of optimization or selection criterion must be used. The simplest of these are sliding windows of various kinds, which are discussed in Section 11.4 and surveyed by Gong, Yang, Wang *et al.* (2007) and Tombari,

Mattoccia, Di Stefano *et al.* (2008). More commonly, global optimization frameworks are used to compute the best disparity field, as described in Section 11.5. These techniques include dynamic programming and truly global optimization algorithms, such as graph cuts and loopy belief propagation. Because the literature on this is so extensive, it is described in more detail in Section 11.5. A good place to find pointers to the latest results in this field is the Middlebury Stereo Vision Page at <http://vision.middlebury.edu/stereo>.

Algorithms for multi-view stereo typically fall into two categories. The first include algorithms that compute traditional depth maps using several images for computing photoconsistency measures (Okutomi and Kanade 1993; Kang, Webb, Zitnick *et al.* 1995; Nakamura, Matsuura, Satoh *et al.* 1996; Szeliski and Golland 1999; Kang, Szeliski, and Chai 2001; Vaish, Szeliski, Zitnick *et al.* 2006; Gallup, Frahm, Mordohai *et al.* 2008). Optionally, some of these techniques compute multiple depth maps and use additional constraints to encourage the different depth maps to be consistent (Szeliski 1999; Kolmogorov and Zabih 2002; Kang and Szeliski 2004; Maitre, Shinagawa, and Do 2008; Zhang, Jia, Wong *et al.* 2008).

The second category consists of papers that compute true 3D volumetric or surface-based object models. Again, because of the large number of papers published on this topic, rather than citing them here, we refer you to the material in Section 11.6.1, the survey by Seitz, Curless, Diebel *et al.* (2006), and the on-line evaluation Web site at <http://vision.middlebury.edu/mview/>.

11.8 Exercises

Ex 11.1: Stereo pair rectification Implement the following simple algorithm (Section 11.1.1):

1. Rotate both cameras so that they are looking perpendicular to the line joining the two camera centers c_0 and c_1 . The smallest rotation can be computed from the cross product between the original and desired optical axes.
2. Twist the optical axes so that the horizontal axis of each camera looks in the direction of the other camera. (Again, the cross product between the current x -axis after the first rotation and the line joining the cameras gives the rotation.)
3. If needed, scale up the smaller (less detailed) image so that it has the same resolution (and hence line-to-line correspondence) as the other image.

Now compare your results to the algorithm proposed by Loop and Zhang (1999). Can you think of situations where their approach may be preferable?

Ex 11.2: Rigid direct alignment Modify your spline-based or optical flow motion estimator from Exercise 8.4 to use epipolar geometry, i.e. to only estimate disparity.

(Optional) Extend your algorithm to simultaneously estimate the epipolar geometry (without first using point correspondences) by estimating a base homography corresponding to a reference plane for the dominant motion and then an epipole for the residual parallax (motion).

Ex 11.3: Shape from profiles Reconstruct a surface model from a series of edge images (Section 11.2.1).

1. Extract edges and link them (Exercises 4.7–4.8).
2. Based on previously computed epipolar geometry, match up edges in triplets (or longer sets) of images.
3. Reconstruct the 3D locations of the curves using osculating circles (11.5).
4. Render the resulting 3D surface model as a sparse mesh, i.e., drawing the reconstructed 3D profile curves and links between 3D points in neighboring images with similar osculating circles.

Ex 11.4: Plane sweep Implement a plane sweep algorithm (Section 11.1.2).

If the images are already pre-rectified, this consists simply of shifting images relative to each other and comparing pixels. If the images are not pre-rectified, compute the homography that resamples the target image into the reference image’s coordinate system for each plane.

Evaluate a subset of the following similarity measures (Section 11.3.1) and compare their performance by visualizing the disparity space image (DSI), which should be dark for pixels at correct depths:

- squared difference (SD);
- absolute difference (AD);
- truncated or robust measures;
- gradient differences;
- rank or census transform (the latter usually performs better);
- mutual information from a pre-computed joint density function.

Consider using the Birchfield and Tomasi (1998) technique of comparing ranges between neighboring pixels (different shifted or warped images). Also, try pre-compensating images for bias or gain variations using one or more of the techniques discussed in Section 11.3.1.

Ex 11.5: Aggregation and window-based stereo Implement one or more of the matching cost aggregation strategies described in Section 11.4:

- convolution with a box or Gaussian kernel;
- shifting window locations by applying a min filter (Scharstein and Szeliski 2002);
- picking a window that maximizes some match-reliability metric (Veksler 2001, 2003);
- weighting pixels by their similarity to the central pixel (Yoon and Kweon 2006).

Once you have aggregated the costs in the DSI, pick the winner at each pixel (winner-take-all), and then optionally perform one or more of the following post-processing steps:

1. compute matches both ways and pick only the reliable matches (draw the others in another color);
2. tag matches that are unsure (whose confidence is too low);

3. fill in the matches that are unsure from neighboring values;
4. refine your matches to sub-pixel disparity by either fitting a parabola to the DSI values around the winner or by using an iteration of Lukas–Kanade.

Ex 11.6: Optimization-based stereo Compute the disparity space image (DSI) volume using one of the techniques you implemented in Exercise 11.4 and then implement one (or more) of the global optimization techniques described in Section 11.5 to compute the depth map. Potential choices include:

- dynamic programming or scanline optimization (relatively easy);
- semi-global optimization (Hirschmüller 2008), which is a simple extension of scanline optimization and performs well;
- graph cuts using alpha expansions (Boykov, Veksler, and Zabih 2001), for which you will need to find a max-flow or min-cut algorithm (<http://vision.middlebury.edu/stereo>);
- loopy belief propagation (Appendix B.5.3).

Evaluate your algorithm by running it on the Middlebury stereo data sets.

How well does your algorithm do against local aggregation (Yoon and Kweon 2006)? Can you think of some extensions or modifications to make it even better?

Ex 11.7: View interpolation, revisited Compute a dense depth map using one of the techniques you developed above and use it (or, better yet, a depth map for each source image) to generate smooth in-between views from a stereo data set.

Compare your results against using the ground truth depth data (if available).

What kinds of artifacts do you see? Can you think of ways to reduce them?

More details on implementing such algorithms can be found in Section 13.1 and Exercises 13.1–13.4.

Ex 11.8: Multi-frame stereo Extend one of your previous techniques to use multiple input frames (Section 11.6) and try to improve the results you obtained with just two views.

If helpful, try using temporal selection (Kang and Szeliski 2004) to deal with the increased number of occlusions in multi-frame data sets.

You can also try to simultaneously estimate multiple depth maps and make them consistent (Kolmogorov and Zabih 2002; Kang and Szeliski 2004).

Test your algorithms out on some standard multi-view data sets.

Ex 11.9: Volumetric stereo Implement voxel coloring (Seitz and Dyer 1999) as a simple extension to the plane sweep algorithm you implemented in Exercise 11.4.

1. Instead of computing the complete DSI all at once, evaluate each plane one at a time from front to back.
2. Tag every voxel whose photoconsistency is below a certain threshold as being part of the object and remember its average (or robust) color (Seitz and Dyer 1999; Eisert, Steinbach, and Girod 2000; Kutulakos 2000; Slabaugh, Culbertson, Slabaugh *et al.* 2004).

3. Erase the input pixels corresponding to tagged voxels in the input images, e.g., by setting their alpha value to 0 (or to some reduced number, depending on occupancy).
4. As you evaluate the next plane, use the source image alpha values to modify your photoconsistency score, e.g., only consider pixels that have full alpha or weight pixels by their alpha values.
5. If the cameras are not all on the same side of your plane sweeps, use space carving (Kutulakos and Seitz 2000) to cycle through different subsets of source images while carving away the volume from different directions.

Ex 11.10: Depth map merging Use the technique you developed for multi-frame stereo in Exercise 11.8 or a different technique, such as the one described by Goesele, Snavely, Curless *et al.* (2007), to compute a depth map for every input image.

Merge these depth maps into a coherent 3D model, e.g., using Poisson surface reconstruction (Kazhdan, Bolitho, and Hoppe 2006).

Ex 11.11: Shape from silhouettes Build a silhouette-based volume reconstruction algorithm (Section 11.6.2). Use an octree or some other representation of your choosing.

Chapter 12

3D reconstruction

| | |
|--|-----|
| 12.1 Shape from X | 508 |
| 12.1.1 Shape from shading and photometric stereo | 508 |
| 12.1.2 Shape from texture | 510 |
| 12.1.3 Shape from focus | 511 |
| 12.2 Active rangefinding | 512 |
| 12.2.1 Range data merging | 515 |
| 12.2.2 Application: Digital heritage | 517 |
| 12.3 Surface representations | 518 |
| 12.3.1 Surface interpolation | 518 |
| 12.3.2 Surface simplification | 520 |
| 12.3.3 Geometry images | 520 |
| 12.4 Point-based representations | 521 |
| 12.5 Volumetric representations | 522 |
| 12.5.1 Implicit surfaces and level sets | 522 |
| 12.6 Model-based reconstruction | 523 |
| 12.6.1 Architecture | 524 |
| 12.6.2 Heads and faces | 526 |
| 12.6.3 Application: Facial animation | 528 |
| 12.6.4 Whole body modeling and tracking | 530 |
| 12.7 Recovering texture maps and albedos | 534 |
| 12.7.1 Estimating BRDFs | 536 |
| 12.7.2 Application: 3D photography | 537 |
| 12.8 Additional reading | 538 |
| 12.9 Exercises | 539 |

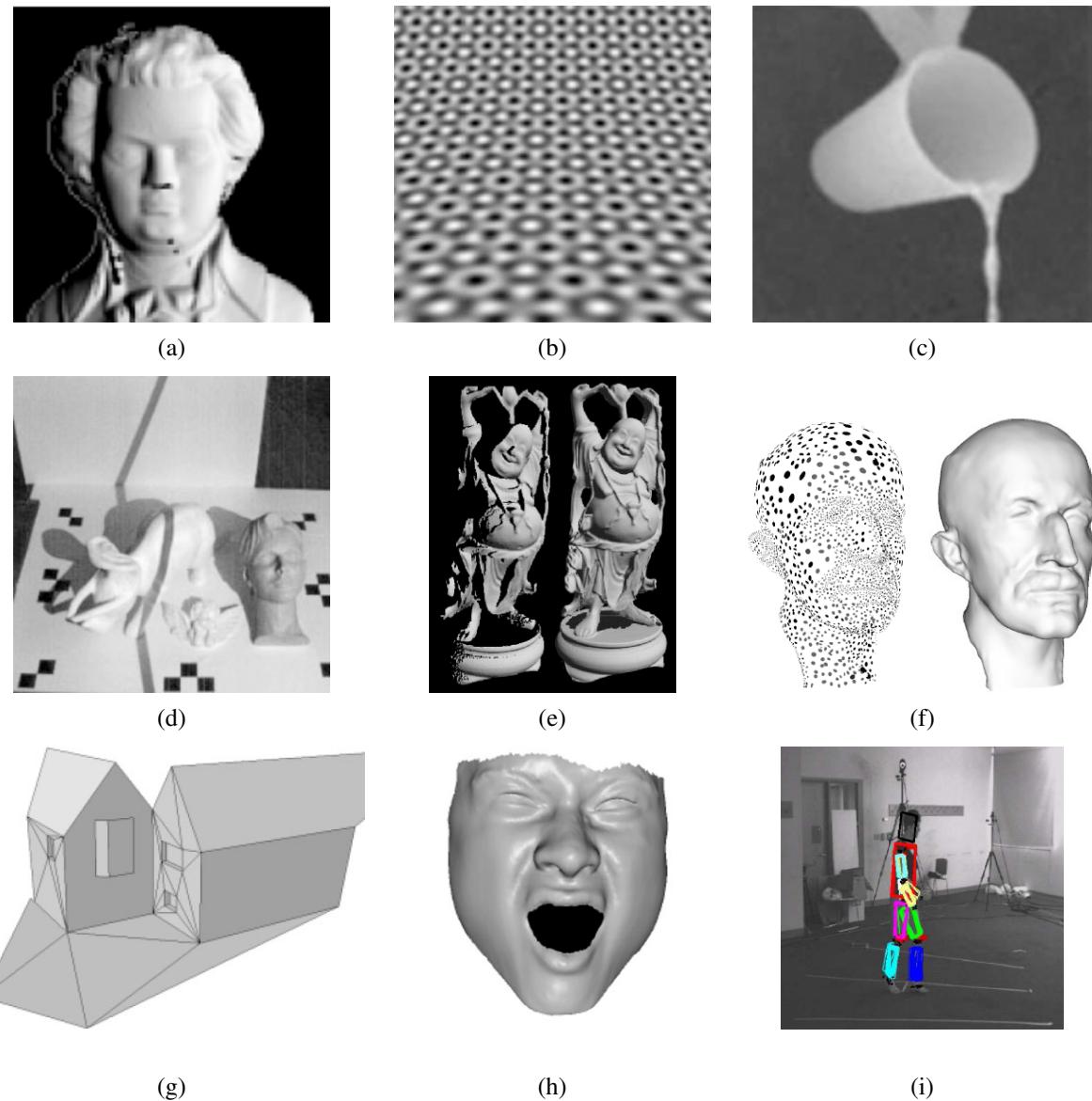


Figure 12.1 3D shape acquisition and modeling techniques: (a) shaded image (Zhang, Tsai, Cryer *et al.* 1999) © 1999 IEEE; (b) texture gradient (Garding 1992) © 1992 Springer; (c) real-time depth from focus (Nayar, Watanabe, and Noguchi 1996) © 1996 IEEE; (d) scanning a scene with a stick shadow (Bouguet and Perona 1999) © 1999 Springer; (e) merging range maps into a 3D model (Curless and Levoy 1996) © 1996 ACM; (f) point-based surface modeling (Pauly, Keiser, Kobbel *et al.* 2003) © 2003 ACM; (g) automated modeling of a 3D building using lines and planes (Werner and Zisserman 2002) © 2002 Springer; (h) 3D face model from spacetime stereo (Zhang, Snavely, Curless *et al.* 2004) © 2004 ACM; (i) person tracking (Sigal, Bhatia, Roth *et al.* 2004) © 2004 IEEE.

As we saw in the previous chapter, a variety of stereo matching techniques have been developed to reconstruct high quality 3D models from two or more images. However, stereo is just one of the many potential cues that can be used to infer shape from images. In this chapter, we investigate a number of such techniques, which include not only visual cues such as shading and focus, but also techniques for merging multiple range or depth images into 3D models, as well as techniques for reconstructing specialized models, such as heads, bodies, or architecture.

Among the various cues that can be used to infer shape, the shading on a surface (Figure 12.1a) can provide a lot of information about local surface orientations and hence overall surface shape (Section 12.1.1). This approach becomes even more powerful when lights shining from different directions can be turned on and off separately (*photometric stereo*). Texture gradients (Figure 12.1b), i.e., the foreshortening of regular patterns as the surface slants or bends away from the camera, can provide similar cues on local surface orientation (Section 12.1.2). Focus is another powerful cue to scene depth, especially when two or more images with different focus settings are used (Section 12.1.3).

3D shape can also be estimated using active illumination techniques such as light stripes (Figure 12.1d) or time of flight range finders (Section 12.2). The partial surface models obtained using such techniques (or passive image-based stereo) can then be merged into more coherent 3D surface models (Figure 12.1e), as discussed in Section 12.2.1. Such techniques have been used to construct highly detailed and accurate models of cultural heritage such as historic sites (Section 12.2.2). The resulting surface models can then be simplified to support viewing at different resolutions and streaming across the Web (Section 12.3.2). An alternative to working with continuous surfaces is to represent 3D surfaces as dense collections of 3D oriented points (Section 12.4) or as volumetric primitives (Section 12.5).

3D modeling can be more efficient and effective if we know something about the objects we are trying to reconstruct. In Section 12.6, we look at three specialized but commonly occurring examples, namely architecture (Figure 12.1g), heads and faces (Figure 12.1h), and whole bodies (Figure 12.1i). In addition to modeling people, we also discuss techniques for tracking them.

The last stage of shape and appearance modeling is to extract some textures to paint onto our 3D models (Section 12.7). Some techniques go beyond this and actually estimate full BRDFs (Section 12.7.1).

Because there exists such a large variety of techniques to perform 3D modeling, this chapter does not go into detail on any one of these. Readers are encouraged to find more information in the cited references or more specialized publications and conferences devoted to these topics, e.g., the International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT), the International Conference on 3D Digital Imaging and Modeling (3DIM), the International Conference on Automatic Face and Gesture Recognition (FG), the IEEE Workshop on Analysis and Modeling of Faces and Gestures, and the International Workshop on Tracking Humans for the Evaluation of their Motion in Image Sequences (THEMIS).

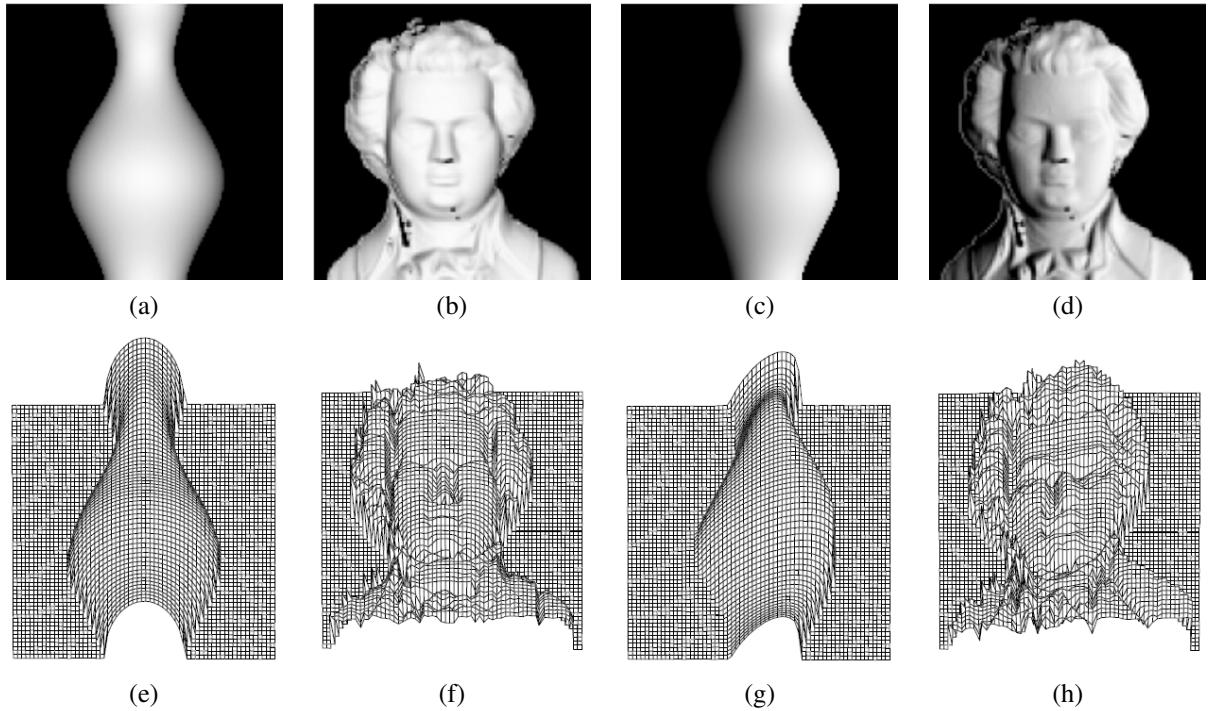


Figure 12.2 Synthetic shape from shading (Zhang, Tsai, Cryer *et al.* 1999) © 1999 IEEE: shaded images, (a–b) with light from in front ($0, 0, 1$) and (c–d) with light the front right ($1, 0, 1$); (e–f) corresponding shape from shading reconstructions using the technique of Tsai and Shah (1994).

12.1 Shape from X

In addition to binocular disparity, shading, texture, and focus all play a role in how we perceive shape. The study of how shape can be inferred from such cues is sometimes called *shape from X*, since the individual instances are called *shape from shading*, *shape from texture*, and *shape from focus*.¹ In this section, we look at these three cues and how they can be used to reconstruct 3D geometry. A good overview of all these topics can be found in the collection of papers on physics-based shape inference edited by Wolff, Shafer, and Healey (1992b).

12.1.1 Shape from shading and photometric stereo

When you look at images of smooth shaded objects, such as the ones shown in Figure 12.2, you can clearly see the shape of the object from just the shading variation. How is this possible? The answer is that as the surface normal changes across the object, the apparent brightness changes as a function of the angle between the local surface orientation and the incident illumination, as shown in Figure 2.15 (Section 2.2.2).

The problem of recovering the shape of a surface from this intensity variation is known as

¹ We have already seen examples of shape from stereo, shape from profiles, and shape from silhouettes in Chapter 11.

shape from shading and is one of the classic problems in computer vision (Horn 1975). The collection of papers edited by Horn and Brooks (1989) is a great source of information on this topic, especially the chapter on variational approaches. The survey by Zhang, Tsai, Cryer *et al.* (1999) not only reviews more recent techniques, but also provides some comparative results.

Most shape from shading algorithms assume that the surface under consideration is of a uniform albedo and reflectance, and that the light source directions are either known or can be calibrated by the use of a reference object. Under the assumptions of distant light sources and observer, the variation in intensity (*irradiance equation*) become purely a function of the local surface orientation,

$$I(x, y) = R(p(x, y), q(x, y)), \quad (12.1)$$

where $(p, q) = (z_x, z_y)$ are the depth map derivatives and $R(p, q)$ is called the *reflectance map*. For example, a diffuse (Lambertian) surface has a reflectance map that is the (non-negative) dot product (2.88) between the surface normal $\hat{n} = (p, q, 1)/\sqrt{1 + p^2 + q^2}$ and the light source direction $v = (v_x, v_y, v_z)$,

$$R(p, q) = \max \left(0, \rho \frac{pv_x + qv_y + v_z}{\sqrt{1 + p^2 + q^2}} \right), \quad (12.2)$$

where ρ is the surface reflectance factor (albedo).

In principle, Equations (12.1–12.2) can be used to estimate (p, q) using non-linear least squares or some other method. Unfortunately, unless additional constraints are imposed, there are more unknowns per pixel (p, q) than there are measurements (I). One commonly used constraint is the smoothness constraint,

$$\mathcal{E}_s = \int p_x^2 + p_y^2 + q_x^2 + q_y^2 dx dy = \int \|\nabla p\|^2 + \|\nabla q\|^2 dx dy, \quad (12.3)$$

which we already saw in Section 3.7.1 (3.94). The other is the *integrability constraint*,

$$\mathcal{E}_i = \int (p_y - q_x)^2 dx dy, \quad (12.4)$$

which arises naturally, since for a valid depth map $z(x, y)$ with $(p, q) = (z_x, z_y)$, we have $p_y = z_{xy} = z_{yx} = q_x$.

Instead of first recovering the orientation fields (p, q) and integrating them to obtain a surface, it is also possible to directly minimize the discrepancy in the image formation equation (12.1) while finding the optimal depth map $z(x, y)$ (Horn 1990). Unfortunately, shape from shading is susceptible to local minima in the search space and, like other variational problems that involve the simultaneous estimation of many variables, can also suffer from slow convergence. Using multi-resolution techniques (Szeliski 1991a) can help accelerate the convergence, while using more sophisticated optimization techniques (Dupuis and Olien-sis 1994) can help avoid local minima.

In practice, surfaces other than plaster casts are rarely of a single uniform albedo. Shape from shading therefore needs to be combined with some other technique or extended in some way to make it useful. One way to do this is to combine it with stereo matching (Fua and Leclerc 1995) or known texture (surface patterns) (White and Forsyth 2006). The stereo and texture components provide information in textured regions, while shape from shading helps fill in the information across uniformly colored regions and also provides finer information about surface shape.

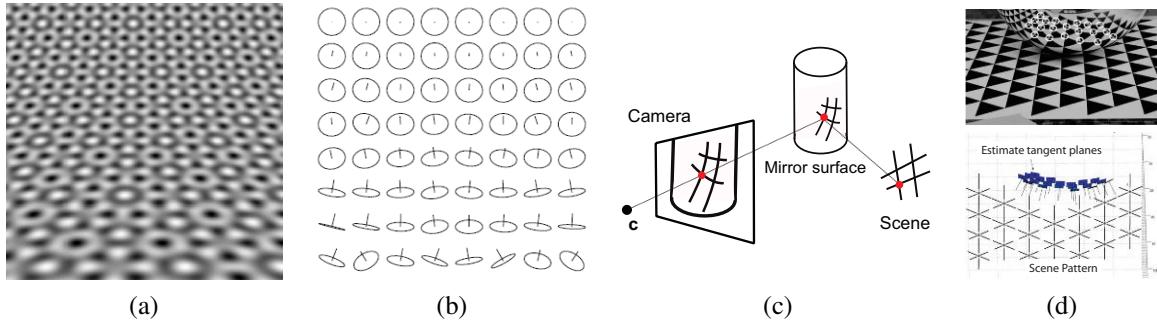


Figure 12.3 Synthetic shape from texture (Garding 1992) © 1992 Springer: (a) regular texture wrapped onto a curved surface and (b) the corresponding surface normal estimates. Shape from mirror reflections (Savarese, Chen, and Perona 2005) © 2005 Springer: (c) a regular pattern reflecting off a curved mirror gives rise to (d) curved lines, from which 3D point locations and normals can be inferred.

Photometric stereo. Another way to make shape from shading more reliable is to use multiple light sources that can be selectively turned on and off. This technique is called *photometric stereo*, since the light sources play a role analogous to the cameras located at different locations in traditional stereo (Woodham 1981).² For each light source, we have a different reflectance map, $R_1(p, q)$, $R_2(p, q)$, etc. Given the corresponding intensities I_1 , I_2 , etc. at a pixel, we can in principle recover both an unknown albedo ρ and a surface orientation estimate (p, q) .

For diffuse surfaces (12.2), if we parameterize the local orientation by $\hat{\mathbf{n}}$, we get (for non-shadowed pixels) a set of linear equations of the form

$$I_k = \rho \hat{\mathbf{n}} \cdot \mathbf{v}_k, \quad (12.5)$$

from which we can recover $\rho \hat{\mathbf{n}}$ using linear least squares. These equations are well conditioned as long as the (three or more) vectors \mathbf{v}_k are linearly independent, i.e., they are not along the same azimuth (direction away from the viewer).

Once the surface normals or gradients have been recovered at each pixel, they can be integrated into a depth map using a variant of regularized surface fitting (3.100). (Nehab, Rusinkiewicz, Davis *et al.* (2005) and Harker and O’Leary (2008) have produced some recent work in this area.)

When surfaces are specular, more than three light directions may be required. In fact, the irradiance equation given in (12.1) not only requires that the light sources and camera be distant from the surface, it also neglects inter-reflections, which can be a significant source of the shading observed on object surfaces, e.g., the darkening seen inside concave structures such as grooves and crevasses (Nayar, Ikeuchi, and Kanade 1991).

12.1.2 Shape from texture

The variation in foreshortening observed in regular textures can also provide useful information about local surface orientation. Figure 12.3 shows an example of such a pattern, along

² An alternative to turning lights on-and-off is to use three colored lights (Woodham 1994; Hernandez, Vogiatzis, Brostow *et al.* 2007; Hernandez and Vogiatzis 2010).

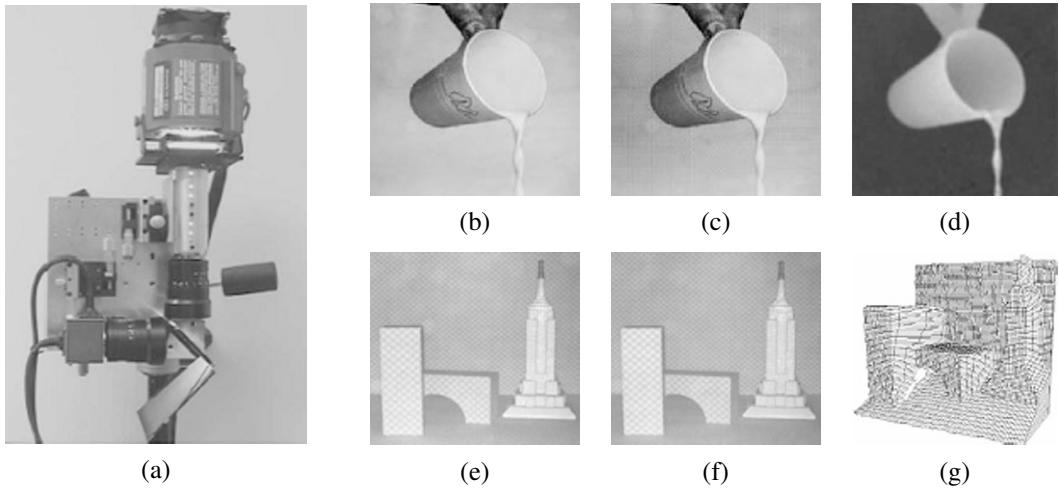


Figure 12.4 Real time depth from defocus (Nayar, Watanabe, and Noguchi 1996) © 1996 IEEE: (a) the real-time focus range sensor, which includes a half-silvered mirror between the two telecentric lenses (lower right), a prism that splits the image into two CCD sensors (lower left), and an edged checkerboard pattern illuminated by a Xenon lamp (top); (b–c) input video frames from the two cameras along with (d) the corresponding depth map; (e–f) two frames (you can see the texture if you zoom in) and (g) the corresponding 3D mesh model.

with the estimated local surface orientations. Shape from texture algorithms require a number of processing steps, including the extraction of repeated patterns or the measurement of local frequencies in order to compute local affine deformations, and a subsequent stage to infer local surface orientation. Details on these various stages can be found in the research literature (Witkin 1981; Ikeuchi 1981; Blostein and Ahuja 1987; Garding 1992; Malik and Rosenholtz 1997; Lobay and Forsyth 2006).

When the original pattern is regular, it is possible to fit a regular but slightly deformed grid to the image and use this grid for a variety of image replacement or analysis tasks (Liu, Collins, and Tsin 2004; Liu, Lin, and Hays 2004; Hays, Leordeanu, Efros *et al.* 2006; Lin, Hays, Wu *et al.* 2006; Park, Brocklehurst, Collins *et al.* 2009). This process becomes even easier if specially printed textured cloth patterns are used (White and Forsyth 2006; White, Crane, and Forsyth 2007).

The deformations induced in a regular pattern when it is viewed in the reflection of a curved mirror, as shown in Figure 12.3c–d, can be used to recover the shape of the surface (Savarese, Chen, and Perona 2005; Rozenfeld, Shimshoni, and Lindenbaum 2007). It is also possible to infer local shape information from *specular flow*, i.e., the motion of specularities when viewed from a moving camera (Oren and Nayar 1997; Zisserman, Giblin, and Blake 1989; Swaminathan, Kang, Szeliski *et al.* 2002).

12.1.3 Shape from focus

A strong cue for object depth is the amount of blur, which increases as the object's surface moves away from the camera's focusing distance. As shown in Figure 2.19, moving the object surface away from the focus plane increases the circle of confusion, according to a formula that is easy to establish using similar triangles (Exercise 2.4).

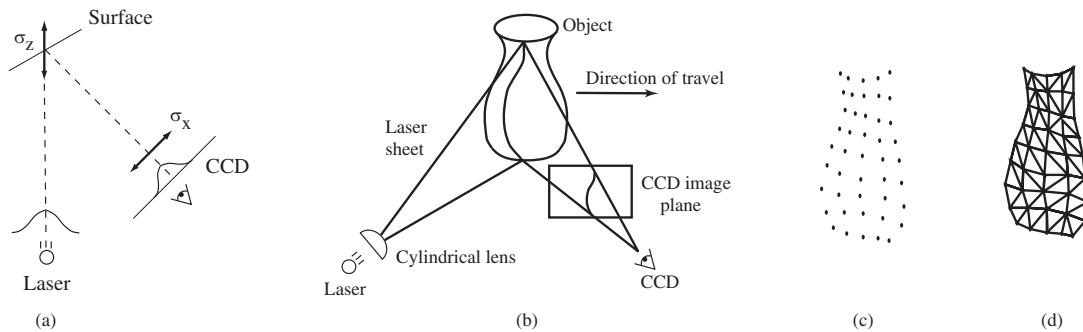


Figure 12.5 Range data scanning (Curless and Levoy 1996) © 1996 ACM: (a) a laser dot on a surface is imaged by a CCD sensor; (b) a laser stripe (sheet) is imaged by the sensor (the deformation of the stripe encodes the distance to the object); (c) the resulting set of 3D points are turned into (d) a triangulated mesh.

A number of techniques have been developed to estimate depth from the amount of defocus (*depth from defocus*) (Pentland 1987; Nayar and Nakagawa 1994; Nayar, Watanabe, and Noguchi 1996; Watanabe and Nayar 1998; Chaudhuri and Rajagopalan 1999; Favaro and Soatto 2006). In order to make such a technique practical, a number issues need to be addressed:

- The amount of blur increase in *both* directions as you move away from the focus plane. Therefore, it is necessary to use two or more images captured with different focus distance settings (Pentland 1987; Nayar, Watanabe, and Noguchi 1996) or to translate the object in depth and look for the point of maximum sharpness (Nayar and Nakagawa 1994).
- The magnification of the object can vary as the focus distance is changed or the object is moved. This can be modeled either explicitly (making correspondence more difficult) or using *telecentric optics*, which approximate an orthographic camera and require an aperture in front of the lens (Nayar, Watanabe, and Noguchi 1996).
- The amount of defocus must be reliably estimated. A simple approach is to average the squared gradient in a region but this suffers from several problems, including the image magnification problem mentioned above. A better solution is to use carefully designed *rational filters* (Watanabe and Nayar 1998).

Figure 12.4 shows an example of a real-time depth from defocus sensor, which employs two imaging chips at slightly different depths sharing a common optical path, as well as an active illumination system that projects a checkerboard pattern from the same direction. As you can see in Figure 12.4b–g, the system produces high-accuracy real-time depth maps for both static and dynamic scenes.

12.2 Active rangefinding

As we have seen in the previous section, actively lighting a scene, whether for the purpose of estimating normals using photometric stereo or for adding artificial texture for shape from

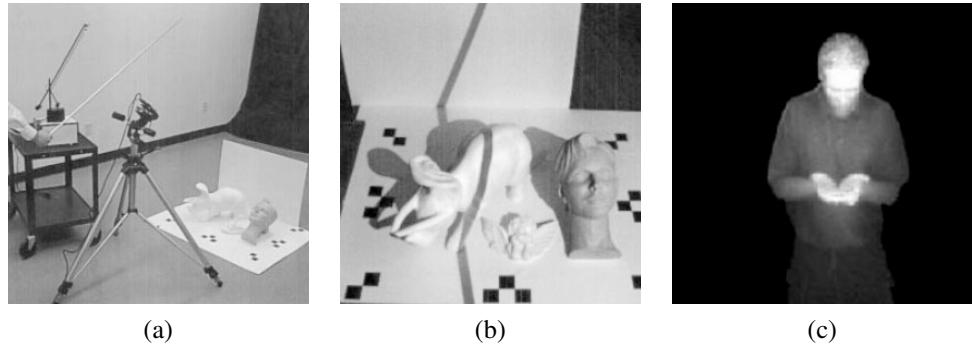


Figure 12.6 Shape scanning using cast shadows (Bouguet and Perona 1999) © 1999 Springer: (a) camera setup with a point light source (a desk lamp without its reflector), a hand-held stick casting a shadow, and (b) the objects being scanned in front of two planar backgrounds. (c) Real-time depth map using a pulsed illumination system (Iddan and Yahav 2001) © 2001 SPIE.

defocus, can greatly improve the performance of vision systems. This kind of *active illumination* has been used from the earliest days of machine vision to construct highly reliable sensors for estimating 3D depth images using a variety of *rangefinding* (or *range sensing*) techniques (Besl 1989; Curless 1999; Hebert 2000).

One of the most popular active illumination sensors is a laser or light stripe sensor, which sweeps a plane of light across the scene or object while observing it from an offset viewpoint, as shown in Figure 12.5b (Rioux and Bird 1993; Curless and Levoy 1995). As the stripe falls across the object, it deforms its shape according to the shape of the surface it is illuminating. It is then a simple matter of using *optical triangulation* to estimate the 3D locations of all the points seen in a particular stripe. In more detail, knowledge of the 3D plane equation of the light stripe allows us to infer the 3D location corresponding to each illuminated pixel, as previously discussed in (2.70–2.71). The accuracy of light striping techniques can be improved by finding the exact temporal peak in illumination for each pixel (Curless and Levoy 1995). The final accuracy of a scanner can be determined using slant edge modulation techniques, i.e., by imaging sharp creases in a calibration object (Goesele, Fuchs, and Seidel 2003).

An interesting variant on light stripe rangefinding is presented by Bouguet and Perona (1999). Instead of projecting a light stripe, they simply wave a stick casting a shadow over a scene or object illuminated by a point light source such as a lamp or the sun (Figure 12.6a). As the shadow falls across two background planes whose orientation relative to the camera is known (or inferred during pre-calibration), the plane equation for each stripe can be inferred from the two projected lines, whose 3D equations are known (Figure 12.6b). The deformation of the shadow as it crosses the object being scanned then reveals its 3D shape, as with regular light stripe rangefinding (Exercise 12.2). This technique can also be used to estimate the 3D geometry of a background scene and how its appearance varies as it moves into shadow, in order to cast new shadows onto the scene (Chuang, Goldman, Curless *et al.* 2003) (Section 10.4.3).

The time it takes to scan an object using a light stripe technique is proportional to the number of depth planes used, which is usually comparable to the number of pixels across an image. A much faster scanner can be constructed by turning different projector pixels on

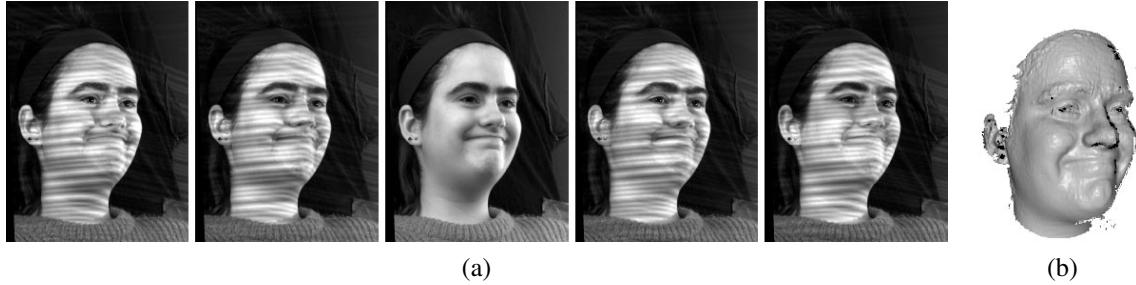


Figure 12.7 Real-time dense 3D face capture using spacetime stereo (Zhang, Snavely, Curless *et al.* 2004) © 2004 ACM: (a) set of five consecutive video frames from one of two stereo cameras (every fifth frame is free of stripe patterns, in order to extract texture); (b) resulting high-quality 3D surface model (depth map visualized as a shaded rendering).

and off in a structured manner, e.g., using a binary or Gray code (Besl 1989). For example, let us assume that the LCD projector we are using has 1024 columns of pixels. Taking the 10-bit binary code corresponding to each column’s address ($0 \dots 1023$), we project the first bit, then the second, etc. After 10 projections (e.g., a third of a second for a synchronized 30Hz camera-projector system), each pixel in the camera knows which of the 1024 columns of projector light it is seeing. A similar approach can also be used to estimate the refractive properties of an object by placing a monitor behind the object (Zongker, Werner, Curless *et al.* 1999; Chuang, Zongker, Hindorff *et al.* 2000) (Section 13.4). Very fast scanners can also be constructed with a single laser beam, i.e., a real-time *flying spot* optical triangulation scanner (RiouxB, Bechthold, Taylor *et al.* 1987).

If even faster, i.e., frame-rate, scanning is required, we can project a single textured pattern into the scene. Proesmans, Van Gool, and Defoort (1998) describe a system where a checkerboard grid is projected onto an object (e.g., a person’s face) and the deformation of the grid is used to infer 3D shape. Unfortunately, such a technique only works if the surface is continuous enough to link all of the grid points.

A much better system can be constructed using high-speed custom illumination and sensing hardware. Iddan and Yahav (2001) describe the construction of their 3DV Zcam video-rate depth sensing camera, which projects a pulsed plane of light onto the scene and then integrates the returning light for a short interval, essentially obtaining time-of-flight measurement for the distance to individual pixels in the scene. A good description of earlier time-of-flight systems, including amplitude and frequency modulation schemes for LIDAR, can be found in (Besl 1989).

Instead of using a single camera, it is also possible to construct an active illumination range sensor using stereo imaging setups. The simplest way to do this is to just project random stripe patterns onto the scene to create synthetic texture, which helps match textureless surfaces (Kang, Webb, Zitnick *et al.* 1995). Projecting a known series of stripes, just as in coded pattern single-camera rangefinding, makes the correspondence between pixels unambiguous and allows for the recovery of depth estimates at pixels only seen in a single camera (Scharstein and Szeliski 2003). This technique has been used to produce large numbers of highly accurate registered multi-image stereo pairs and depth maps for the purpose of evaluating stereo correspondence algorithms (Scharstein and Szeliski 2002; Hirschmüller and

Scharstein 2009) and learning depth map priors and parameters (Scharstein and Pal 2007).

While projecting multiple patterns usually requires the scene or object to remain still, additional processing can enable the production of real-time depth maps for dynamic scenes. The basic idea (Davis, Ramamoorthi, and Rusinkiewicz 2003; Zhang, Curless, and Seitz 2003) is to assume that depth is nearly constant within a 3D space–time window around each pixel and to use the 3D window for matching and reconstruction. Depending on the surface shape and motion, this assumption may be error-prone, as shown in (Davis, Nahab, Ramamoorthi *et al.* 2005). To model shapes more accurately, Zhang, Curless, and Seitz (2003) model the linear disparity variation within the space–time window and show that better results can be obtained by globally optimizing disparity and disparity gradient estimates over video volumes (Zhang, Snavely, Curless *et al.* 2004). Figure 12.7 shows the results of applying this system to a person’s face; the frame-rate 3D surface model can then be used for further model-based fitting and computer graphics manipulation (Section 12.6.2).

12.2.1 Range data merging

While individual range images can be useful for applications such as real-time z-keying or facial motion capture, they are often used as building blocks for more complete 3D object modeling. In such applications, the next two steps in processing are the registration (alignment) of partial 3D surface models and their integration into coherent 3D surfaces (Curless 1999). If desired, this can be followed by a model fitting stage using either parametric representations such as generalized cylinders (Agin and Binford 1976; Nevatia and Binford 1977; Marr and Nishihara 1978; Brooks 1981), superquadrics (Pentland 1986; Solina and Bajcsy 1990; Terzopoulos and Metaxas 1991), or non-parametric models such as triangular meshes (Boissonat 1984) or physically-based models (Terzopoulos, Witkin, and Kass 1988; Delingette, Hebert, and Ikeuchi 1992; Terzopoulos and Metaxas 1991; McInerney and Terzopoulos 1993; Terzopoulos 1999). A number of techniques have also been developed for segmenting range images into simpler constituent surfaces (Hoover, Jean-Baptiste, Jiang *et al.* 1996).

The most widely used 3D registration technique is the *iterated closest point* (ICP) algorithm, which alternates between finding the closest point matches between the two surfaces being aligned and then solving a 3D *absolute orientation* problem (Section 6.1.5, (6.31–6.32) (Besl and McKay 1992; Chen and Medioni 1992; Zhang 1994; Szeliski and Lavallée 1996; Gold, Rangarajan, Lu *et al.* 1998; David, DeMenthon, Duraiswami *et al.* 2004; Li and Hartley 2007; Enqvist, Josephson, and Kahl 2009).³ Since the two surfaces being aligned usually only have partial overlap and may also have outliers, robust matching criteria (Section 6.1.4 and Appendix B.3) are typically used. In order to speed up the determination of the closest point, and also to make the distance-to-surface computation more accurate, one of the two point sets (e.g., the current merged model) can be converted into a *signed distance function*, optionally represented using an *octree spline* for compactness (Lavallée and Szeliski 1995). Variants on the basic ICP algorithm can be used to register 3D point sets under non-rigid deformations, e.g., for medical applications (Feldmar and Ayache 1996; Szeliski and Lavallée 1996). Color values associated with the points or range measurements can also be used as part of the registration process to improve robustness (Johnson and Kang 1997; Pulli 1999).

³ Some techniques, such as the one developed by Chen and Medioni (1992), use local surface tangent planes to make this computation more accurate and to accelerate convergence.

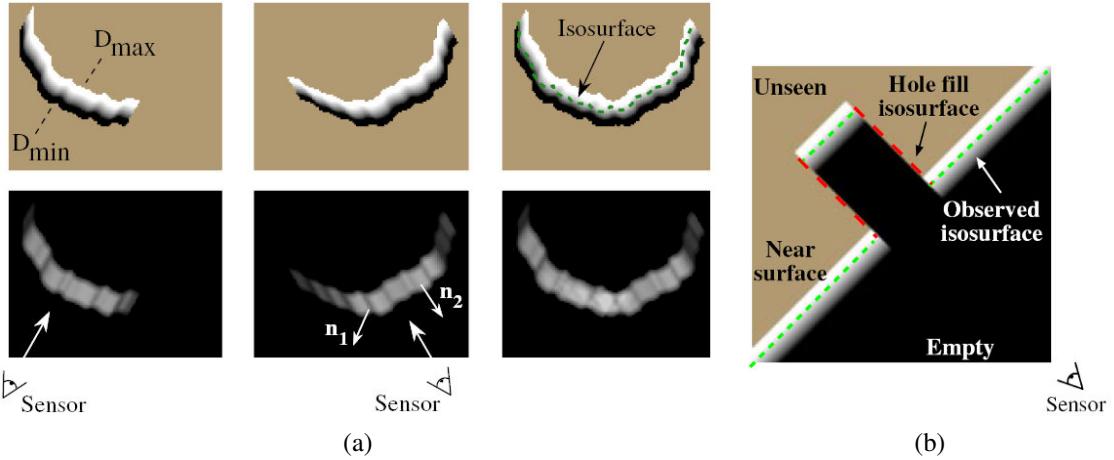


Figure 12.8 Range data merging (Curless and Levoy 1996) © 1996 ACM: (a) two signed distance functions (top left) are merged with their (weights) bottom left to produce a combined set of functions (right column) from which an isosurface can be extracted (green dashed line); (b) the signed distance functions are combined with empty and unseen space labels to fill holes in the isosurface.

Unfortunately, the ICP algorithm and its variants can only find a locally optimal alignment between 3D surfaces. If this is not known *a priori*, more global correspondence or search techniques, based on local descriptors invariant to 3D rigid transformations, need to be used. An example of such a descriptor is the *spin image*, which is a local circular projection of a 3D surface patch around the local normal axis (Johnson and Hebert 1999). Another (earlier) example is the *splash* representation introduced by Stein and Medioni (1992).

Once two or more 3D surfaces have been aligned, they can be merged into a single model. One approach is to represent each surface using a triangulated mesh and combine these meshes using a process that is sometimes called *zippering* (Soucy and Laurendeau 1992; Turk and Levoy 1994). Another, now more widely used, approach is to compute a signed distance function that fits all of the 3D data points (Hoppe, DeRose, Duchamp *et al.* 1992; Curless and Levoy 1996; Hilton, Stoddart, Illingworth *et al.* 1996; Wheeler, Sato, and Ikeuchi 1998).

Figure 12.8 shows one such approach, the *volumetric range image processing* (VRIP) technique developed by Curless and Levoy (1996), which first computes a weighted signed distance function from each range image and then merges them using a weighted averaging process. To make the representation more compact, run-length coding is used to encode the empty, seen, and varying (signed distance) voxels, and only the signed distance values near each surface are stored.⁴ Once the merged signed distance function has been computed, a zero-crossing surface extraction algorithm, such as *marching cubes* (Lorensen and Cline 1987), can be used to recover a meshed surface model. Figure 12.9 shows an example of the complete range data merging and isosurface extraction pipeline.

Volumetric range data merging techniques based on signed distance or characteristic (inside–outside) functions are also widely used to extract smooth well-behaved surfaces from oriented or unoriented sets of points (Hoppe, DeRose, Duchamp *et al.* 1992; Ohtake, Belyaev,

⁴ An alternative, even more compact, representation could be to use octrees (Lavallée and Szeliski 1995).

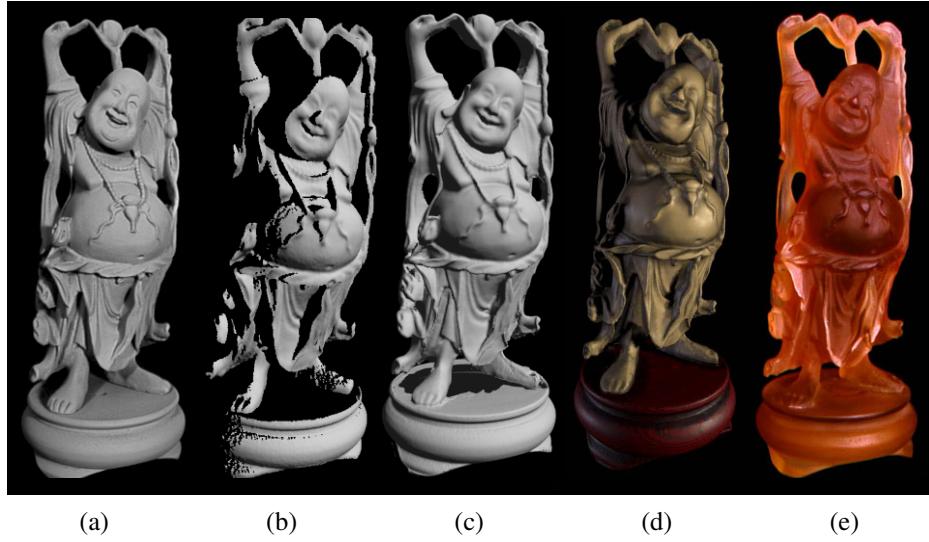


Figure 12.9 Reconstruction and hardcopy of the “Happy Buddha” statuette (Curless and Levoy 1996) © 1996 ACM: (a) photograph of the original statue after spray painting with matte gray; (b) partial range scan; (c) merged range scans; (d) colored rendering of the reconstructed model; (e) hardcopy of the model constructed using stereolithography.

Alexa *et al.* 2003; Kazhdan, Bolitho, and Hoppe 2006; Lempitsky and Boykov 2007; Zach, Pock, and Bischof 2007b; Zach 2008), as discussed in more detail in Section 12.5.1.

12.2.2 Application: Digital heritage

Active rangefinding technologies, combined with surface modeling and appearance modeling techniques (Section 12.7), are widely used in the fields of archeological and historical preservation, which often also goes under the name *digital heritage* (MacDonald 2006). In such applications, detailed 3D models of cultural objects are acquired and later used for applications such as analysis, preservation, restoration, and the production of duplicate artwork (RiouxBird 1993).

A more recent example of such an endeavor is the Digital Michelangelo project of Levoy, Pulli, Curless *et al.* (2000), which used Cyberware laser stripe scanners and high-quality digital SLR cameras mounted on a large gantry to obtain detailed scans of Michelangelo’s David and other sculptures in Florence. The project also took scans of the *Forma Urbis Romae*, an ancient stone map of Rome that had shattered into pieces, for which new matches were obtained using digital techniques. The whole process, from initial planning, to software development, acquisition, and post-processing, took several years (and many volunteers), and produced a wealth of 3D shape and appearance modeling techniques as a result.

Even larger-scale projects are now being attempted, for example, the scanning of complete temple sites such as Angkor-Thom (Ikeuchi and Sato 2001; Ikeuchi and Miyazaki 2007; Banno, Masuda, Oishi *et al.* 2008). Figure 12.10 shows details from this project, including a sample photograph, a detailed 3D (sculptural) head model scanned from ground level, and an aerial overview of the final merged 3D site model, which was acquired using a balloon.

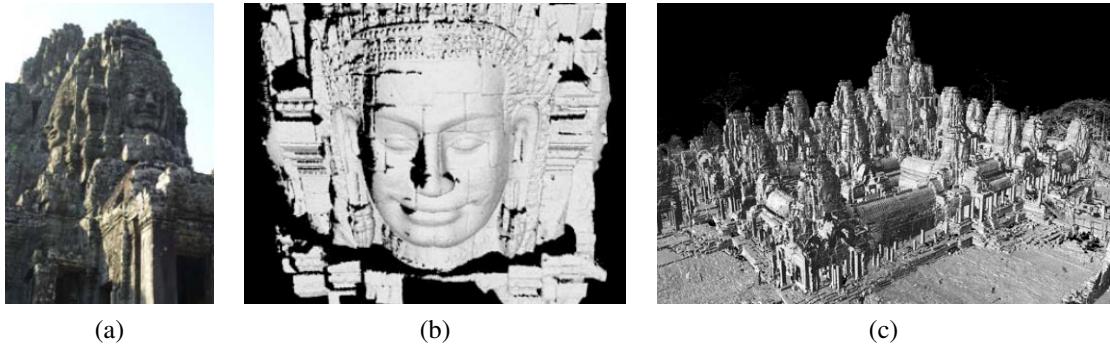


Figure 12.10 Laser range modeling of the Bayon temple at Angkor-Thom (Banno, Masuda, Oishi *et al.* 2008) © 2008 Springer: (a) sample photograph from the site; (b) a detailed head model scanned from the ground; (c) final merged 3D model of the temple scanned using a laser range sensor mounted on a balloon.

12.3 Surface representations

In previous sections, we have seen different representations being used to integrate 3D range scans. We now look at several of these representations in more detail. Explicit surface representations, such as triangle meshes, splines (Farin 1992, 1996), and subdivision surfaces (Stollnitz, DeRose, and Salesin 1996; Zorin, Schröder, and Sweldens 1996; Warren and Weimer 2001; Peters and Reif 2008), enable not only the creation of highly detailed models but also processing operations, such as interpolation (Section 12.3.1), fairing or smoothing, and decimation and simplification (Section 12.3.2). We also examine discrete point-based representations (Section 12.4) and volumetric representations (Section 12.5).

12.3.1 Surface interpolation

One of the most common operations on surfaces is their reconstruction from a set of sparse data constraints, i.e. *scattered data interpolation*. When formulating such problems, surfaces may be parameterized as height fields $f(\mathbf{x})$, as 3D parametric surfaces $\mathbf{f}(\mathbf{x})$, or as non-parametric models such as collections of triangles.

In the section on image processing, we saw how two-dimensional function interpolation and approximation problems $\{d_i\} \rightarrow f(\mathbf{x})$ could be cast as energy minimization problems using regularization (Section 3.7.1 (3.94–3.98)).⁵ Such problems can also specify the locations of discontinuities in the surface as well as local orientation constraints (Terzopoulos 1986b; Zhang, Dugas-Phocion, Samson *et al.* 2002).

One approach to solving such problems is to discretize both the surface and the energy on a discrete grid or mesh using finite element analysis (3.100–3.102) (Terzopoulos 1986b). Such problems can then be solved using sparse system solving techniques, such as multigrid (Briggs, Henson, and McCormick 2000) or hierarchically preconditioned conjugate gradient (Szeliski 2006b). The surface can also be represented using a hierarchical combination of multilevel B-splines (Lee, Wolberg, and Shin 1996).

⁵ The difference between interpolation and approximation is that the former requires the surface or function to pass through the data while the latter allows the function to pass near the data, and can therefore be used for surface smoothing as well.

An alternative approach is to use *radial basis* (or *kernel*) functions (Boult and Kender 1986; Nielson 1993). To interpolate a field $\mathbf{f}(\mathbf{x})$ through (or near) a number of data values \mathbf{d}_i located at \mathbf{x}_i , the *radial basis function* approach uses

$$\mathbf{f}(\mathbf{x}) = \frac{\sum_i w_i(\mathbf{x}) \mathbf{d}_i}{\sum_i w_i(\mathbf{x})}, \quad (12.6)$$

where the weights,

$$w_i(\mathbf{x}) = K(\|\mathbf{x} - \mathbf{x}_i\|), \quad (12.7)$$

are computed using a *radial basis* (spherically symmetrical) function $K(r)$.

If we want the function $\mathbf{f}(\mathbf{x})$ to exactly interpolate the data points, the kernel functions must either be singular at the origin, $\lim_{r \rightarrow 0} K(r) \rightarrow \infty$ (Nielson 1993), or a dense linear system must be solved to determine the magnitude associated with each basis function (Boult and Kender 1986). It turns out that, for certain regularized problems, e.g., (3.94–3.96), there exist radial basis functions (kernels) that give the same results as a full analytical solution (Boult and Kender 1986). Unfortunately, because the dense system solving is cubic in the number of data points, basis function approaches can only be used for small problems such as feature-based image morphing (Beier and Neely 1992).

When a three-dimensional *parametric surface* is being modeled, the vector-valued function \mathbf{f} in (12.6) or (3.94–3.102) encodes 3D coordinates (x, y, z) on the surface and the domain $\mathbf{x} = (s, t)$ encodes the surface parameterization. One example of such surfaces are symmetry-seeking parametric models, which are elastically deformable versions of *generalized cylinders*⁶ (Terzopoulos, Witkin, and Kass 1987). In these models, s is the parameter *along* the spine of the deformable tube and t is the parameter *around* the tube. A variety of smoothness and radial symmetry forces are used to constrain the model while it is fitted to image-based silhouette curves.

It is also possible to define *non-parametric* surface models such as general triangulated meshes and to equip such meshes (using finite element analysis) with both internal smoothness metrics and external data fitting metrics (Sander and Zucker 1990; Fua and Sander 1992; Delingette, Hebert, and Ikeuchi 1992; McInerney and Terzopoulos 1993). While most of these approaches assume a standard *elastic* deformation model, which uses quadratic internal smoothness terms, it is also possible to use sub-linear energy models in order to better preserve surface creases (Diebel, Thrun, and Brünig 2006). Triangle meshes can also be augmented with either spline elements (Sullivan and Ponce 1998) or subdivision surfaces (Stollnitz, DeRose, and Salesin 1996; Zorin, Schröder, and Sweldens 1996; Warren and Weimer 2001; Peters and Reif 2008) to produce surfaces with better smoothness control.

Both parametric and non-parametric surface models assume that the topology of the surface is known and fixed ahead of time. For more flexible surface modeling, we can either represent the surface as a collection of oriented points (Section 12.4) or use 3D implicit functions (Section 12.5.1), which can also be combined with elastic 3D surface models (McInerney and Terzopoulos 1993).

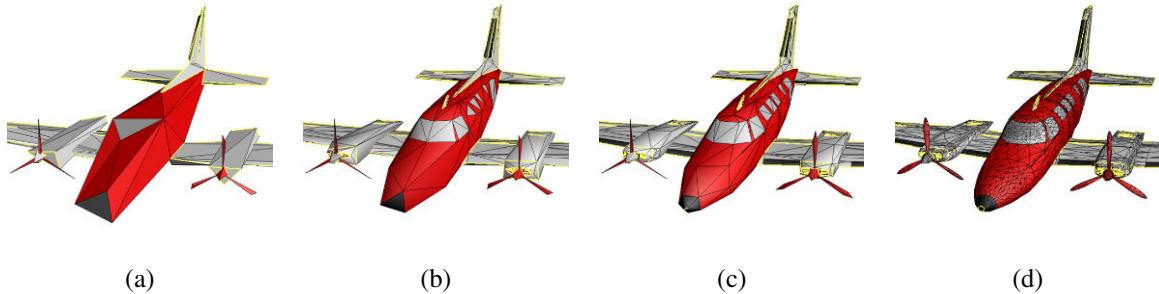


Figure 12.11 Progressive mesh representation of an airplane model (Hoppe 1996) © 1996 ACM: (a) base mesh M^0 (150 faces); (b) mesh M^{175} (500 faces); (c) mesh M^{425} (1000 faces); (d) original mesh $M = M^n$ (13,546 faces).

12.3.2 Surface simplification

Once a triangle mesh has been created from 3D data, it is often desirable to create a hierarchy of mesh models, for example, to control the displayed *level of detail* (LOD) in a computer graphics application. (In essence, this is a 3D analog to image pyramids (Section 3.5).) One approach to doing this is to approximate a given mesh with one that has *subdivision connectivity*, over which a set of triangular wavelet coefficients can then be computed (Eck, DeRose, Duchamp *et al.* 1995). A more continuous approach is to use sequential *edge collapse* operations to go from the original fine-resolution mesh to a coarse base-level mesh (Hoppe 1996). The resulting *progressive mesh* (PM) representation can be used to render the 3D model at arbitrary levels of detail, as shown in Figure 12.11.

12.3.3 Geometry images

While multi-resolution surface representations such as (Eck, DeRose, Duchamp *et al.* 1995; Hoppe 1996) support level of detail operations, they still consist of an irregular collection of triangles, which makes them more difficult to compress and store in a cache-efficient manner.⁷

To make the triangulation completely regular (uniform and gridded), Gu, Gortler, and Hoppe (2002) describe how to create *geometry images* by cutting surface meshes along well-chosen lines and “flattening” the resulting representation into a square. Figure 12.12a shows the resulting (x, y, z) values of the surface mesh mapped over the unit square, while Figure 12.12b shows the associated (n_x, n_y, n_z) *normal map*, i.e., the surface normals associated with each mesh vertex, which can be used to compensate for loss in visual fidelity if the original geometry image is heavily compressed.

⁶ A generalized cylinder (Brooks 1981) is a *solid of revolution*, i.e., the result of rotating a (usually smooth) curve around an axis. It can also be generated by sweeping a slowly varying circular cross-section along the axis. (These two interpretations are equivalent.)

⁷ Subdivision triangulations, such as those in (Eck, DeRose, Duchamp *et al.* 1995), are *semi-regular*, i.e., regular (ordered and nested) within each subdivided base triangle.

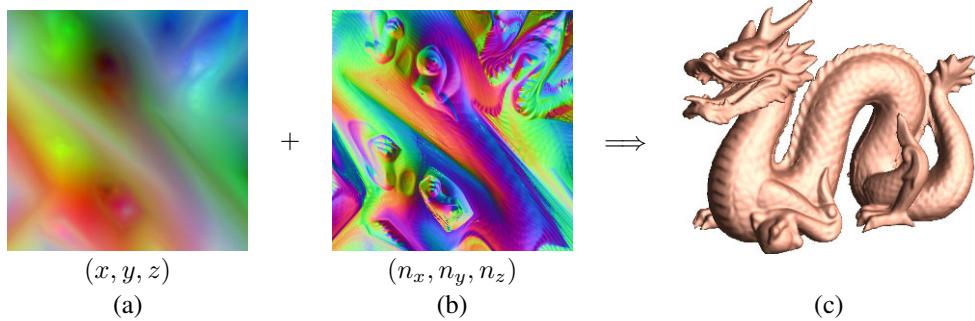


Figure 12.12 Geometry images (Gu, Gortler, and Hoppe 2002) © 2002 ACM: (a) the 257×257 geometry image defines a mesh over the surface; (b) the 512×512 normal map defines vertex normals; (c) final lit 3D model.

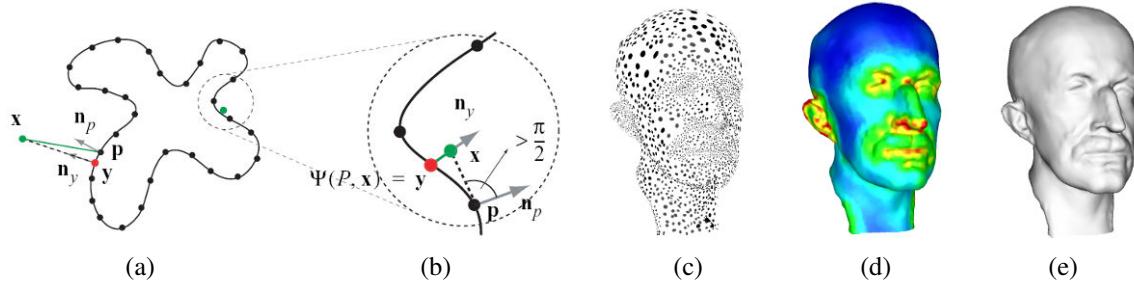


Figure 12.13 Point-based surface modeling with moving least squares (MLS) (Pauly, Keiser, Kobelt *et al.* 2003) © 2003 ACM: (a) a set of points (black dots) is turned into an implicit inside–outside function (black curve); (b) the signed distance to the nearest oriented point can serve as an approximation to the inside–outside distance; (c) a set of oriented points with variable sampling density representing a 3D surface (head model); (d) local estimate of sampling density, which is used in the moving least squares; (e) reconstructed continuous 3D surface.

12.4 Point-based representations

As we mentioned previously, triangle-based surface models assume that the topology (and often the rough shape) of the 3D model is known ahead of time. While it is possible to re-mesh a model as it is being deformed or fitted, a simpler solution is to dispense with an explicit triangle mesh altogether and to have triangle vertices behave as *oriented points*, or *particles*, or *surface elements* (surfels) (Szeliski and Tonnesen 1992).

In order to endow the resulting particle system with internal smoothness constraints, pairwise interaction potentials can be defined that approximate the equivalent elastic bending energies that would be obtained using local finite-element analysis.⁸ Instead of defining the finite element neighborhood for each particle (vertex) ahead of time, a soft influence function is used to couple nearby particles. The resulting 3D model can change both topology and particle density as it evolves and can therefore be used to interpolate partial 3D data with holes (Szeliski, Tonnesen, and Terzopoulos 1993b). Discontinuities in both the surface orientation and crease curves can also be modeled (Szeliski, Tonnesen, and Terzopoulos 1993a).

⁸ As mentioned before, an alternative is to use *sub-linear* interaction potentials, which encourage the preservation of surface creases (Diebel, Thrun, and Brüning 2006).

To render the particle system as a continuous surface, local dynamic triangulation heuristics (Szeliski and Tonnesen 1992) or direct surface element *splatting* (Pfister, Zwicker, van Baar *et al.* 2000) can be used. Another alternative is to first convert the point cloud into an implicit signed distance or inside–outside function, using either minimum signed distances to the oriented points (Hoppe, DeRose, Duchamp *et al.* 1992) or by interpolating a characteristic (inside–outside) function using radial basis functions (Turk and O’Brien 2002; Dinh, Turk, and Slabaugh 2002). Even greater precision over the implicit function fitting, including the ability to handle irregular point densities, can be obtained by computing a *moving least squares* (MLS) estimate of the signed distance function (Alexa, Behr, Cohen-Or *et al.* 2003; Pauly, Keiser, Kobbett *et al.* 2003), as shown in Figure 12.13. Further improvements can be obtained using local sphere fitting (Guennebaud and Gross 2007), faster and more accurate re-sampling (Guennebaud, Germann, and Gross 2008), and kernel regression to better tolerate outliers (Oztireli, Guennebaud, and Gross 2008).

12.5 Volumetric representations

A third alternative for modeling 3D surfaces is to construct 3D volumetric inside–outside functions. We already saw examples of this in Section 11.6.1, where we looked at voxel coloring (Seitz and Dyer 1999), space carving (Kutulakos and Seitz 2000), and level set (Faugeras and Keriven 1998; Pons, Keriven, and Faugeras 2007) techniques for stereo matching, and Section 11.6.2, where we discussed using binary silhouette images to reconstruct volumes.

In this section, we look at continuous *implicit* (inside–outside) functions to represent 3D shape.

12.5.1 Implicit surfaces and level sets

While polyhedral and voxel-based representations can represent three-dimensional shapes to an arbitrary precision, they lack some of the intrinsic smoothness properties available with continuous implicit surfaces, which use an *indicator function (characteristic function)* $F(x, y, z)$ to indicate which 3D points are inside $F(x, y, z) < 0$ or outside $F(x, y, z) > 0$ the object.

An early example of using implicit functions to model 3D objects in computer vision are superquadrics, which are a generalization of quadric (e.g., ellipsoidal) parametric volumetric models,

$$F(x, y, z) = \left(\left(\frac{x}{a_1} \right)^{2/\epsilon_2} + \left(\frac{y}{a_2} \right)^{2/\epsilon_2} \right)^{\epsilon_2/\epsilon_1} + \left(\frac{z}{a_3} \right)^{2/\epsilon_1} - 1 = 0 \quad (12.8)$$

(Pentland 1986; Solina and Bajcsy 1990; Waithe and Ferrie 1991; Leonardis, Jaklič, and Solina 1997). The values of (a_1, a_2, a_3) control the extent of model along each (x, y, z) axis, while the values of (ϵ_1, ϵ_2) control how “square” it is. To model a wider variety of shapes, superquadrics are usually combined with either rigid or non-rigid deformations (Terzopoulos and Metaxas 1991; Metaxas and Terzopoulos 2002). Superquadric models can either be fit to range data or used directly for stereo matching.

A different kind of implicit shape model can be constructed by defining a *signed distance function* over a regular three-dimensional grid, optionally using an octree spline to represent

this function more coarsely away from its surface (zero-set) (Lavallée and Szeliski 1995; Szeliski and Lavallée 1996; Frisken, Perry, Rockwood *et al.* 2000; Ohtake, Belyaev, Alexa *et al.* 2003). We have already seen examples of signed distance functions being used to represent distance transforms (Section 3.3.3), level sets for 2D contour fitting and tracking (Section 5.1.4), volumetric stereo (Section 11.6.1), range data merging (Section 12.2.1), and point-based modeling (Section 12.4). The advantage of representing such functions directly on a grid is that it is quick and easy to look up distance function values for any (x, y, z) location and also easy to extract the isosurface using the marching cubes algorithm (Lorensen and Cline 1987). The work of Ohtake, Belyaev, Alexa *et al.* (2003) is particularly notable since it allows for several distance functions to be used simultaneously and then combined locally to produce sharp features such as creases.

Poisson surface reconstruction (Kazhdan, Bolitho, and Hoppe 2006) uses a closely related volumetric function, namely a smoothed 0/1 inside–outside (characteristic) function, which can be thought of as a clipped signed distance function. The gradients for this function are set to lie along oriented surface normals near known surface points and 0 elsewhere. The function itself is represented using a quadratic tensor-product B-spline over an octree, which provides a compact representation with larger cells away from the surface or in regions of lower point density, and also admits the efficient solution of the related Poisson equations (3.100–3.102), see Section 9.3.4 (Pérez, Gangnet, and Blake 2003).

It is also possible to replace the quadratic penalties used in the Poisson equations with L_1 (total variation) constraints and still obtain a convex optimization problem, which can be solved using either continuous (Zach, Pock, and Bischof 2007b; Zach 2008) or discrete graph cut (Lempitsky and Boykov 2007) techniques.

Signed distance functions also play an integral role in level-set evolution equations ((Sections 5.1.4 and 11.6.1), where the values of distance transforms on the mesh are updated as the surface evolves to fit multi-view stereo photoconsistency measures (Faugeras and Keriven 1998).

12.6 Model-based reconstruction

When we know something ahead of time about the objects we are trying to model, we can construct more detailed and reliable 3D models using specialized techniques and representations. For example, architecture is usually made up of large planar regions and other parametric forms (such as surfaces of revolution), usually oriented perpendicular to gravity and to each other (Section 12.6.1). Heads and faces can be represented using low-dimensional, non-rigid shape models, since the variability in shape and appearance of human faces, while extremely large, is still bounded (Section 12.6.2). Human bodies or parts, such as hands, form highly articulated structures, which can be represented using kinematic chains of piecewise rigid skeletal elements linked by joints (Section 12.6.4).

In this section, we highlight some of the main ideas, representations, and modeling algorithms used for these three cases. Additional details and references can be found in specialized conferences and workshops devoted to these topics, e.g., the International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT), the International Conference on 3D Digital Imaging and Modeling (3DIM), the International Conference on Automatic Face and Gesture Recognition (FG), the IEEE Workshop on Analysis and Modeling of Faces



Figure 12.14 Interactive architectural modeling using the Façade system (Debevec, Taylor, and Malik 1996) © 1996 ACM: (a) input image with user-drawn edges shown in green; (b) shaded 3D solid model; (c) geometric primitives overlaid onto the input image; (d) final view-dependent, texture-mapped 3D model.

and Gestures, and the International Workshop on Tracking Humans for the Evaluation of their Motion in Image Sequences (THEMIS).

12.6.1 Architecture

Architectural modeling, especially from aerial photography, has been one of the longest studied problems in both photogrammetry and computer vision (Walker and Herman 1988). Recently, the development of reliable image-based modeling techniques, as well as the prevalence of digital cameras and 3D computer games, has spurred renewed interest in this area.

The work by Debevec, Taylor, and Malik (1996) was one of the earliest hybrid geometry- and image-based modeling and rendering systems. Their Façade system combines an interactive image-guided geometric modeling tool with model-based (local plane plus parallax) stereo matching and view-dependent texture mapping. During the interactive photogrammetric modeling phase, the user selects block elements and aligns their edges with visible edges in the input images (Figure 12.14a). The system then automatically computes the dimensions and locations of the blocks along with the camera positions using constrained optimization (Figure 12.14b–c). This approach is intrinsically more reliable than general feature-based structure from motion, because it exploits the strong geometry available in the block primitives. Related work by Becker and Bove (1995), Horry, Anjyo, and Arai (1997), and Criminisi, Reid, and Zisserman (2000) exploits similar information available from vanishing points. In the interactive, image-based modeling system of Sinha, Steedly, Szeliski *et al.* (2008), vanishing point directions are used to guide the user drawing of polygons, which are then automatically fitted to sparse 3D points recovered using structure from motion.

Once the rough geometry has been estimated, more detailed offset maps can be computed for each planar face using a local plane sweep, which Debevec, Taylor, and Malik (1996) call *model-based stereo*. Finally, during rendering, images from different viewpoints are warped and blended together as the camera moves around the scene, using a process (related to light field and Lumigraph rendering, see Section 13.3) called *view-dependent texture mapping* (Figure 12.14d).

For interior modeling, instead of working with single pictures, it is more useful to work

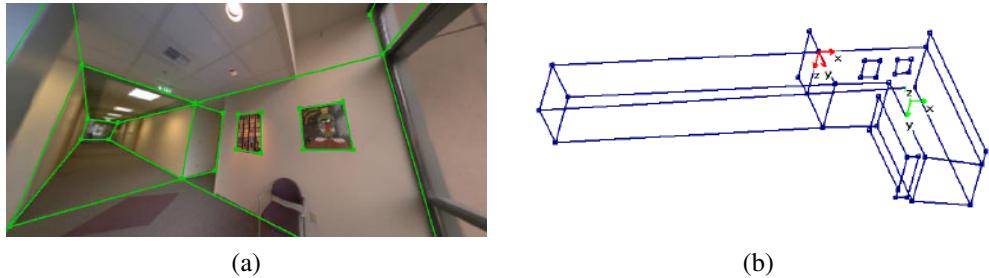


Figure 12.15 Interactive 3D modeling from panoramas (Shum, Han, and Szeliski 1998) © 1998 IEEE: (a) wide-angle view of a panorama with user-drawn vertical and horizontal (axis-aligned) lines; (b) single-view reconstruction of the corridors.

with panoramas, since you can see larger extents of walls and other structures. The 3D modeling system developed by Shum, Han, and Szeliski (1998) first constructs calibrated panoramas from multiple images (Section 7.4) and then has the user draw vertical and horizontal lines in the image to demarcate the boundaries of planar regions. The lines are initially used to establish an absolute rotation for each panorama and are later used (along with the inferred vertices and planes) to optimize the 3D structure, which can be recovered up to scale from one or more images (Figure 12.15). 360° high dynamic range panoramas can also be used for outdoor modeling, since they provide highly reliable estimates of relative camera orientations as well as vanishing point directions (Antone and Teller 2002; Teller, Antone, Bodnar *et al.* 2003).

While earlier image-based modeling systems required some user authoring, Werner and Zisserman (2002) present a fully automated line-based reconstruction system. As described in Section 7.5.1, they first detect lines and vanishing points and use them to calibrate the camera; then they establish line correspondences using both appearance matching and trifocal tensors, which enables them to reconstruct families of 3D line segments, as shown in Figure 12.16a. They then generate plane hypotheses, using both co-planar 3D lines and a plane sweep (Section 11.1.2) based on cross-correlation scores evaluated at interest points. Intersections of planes are used to determine the extent of each plane, i.e., an initial coarse geometry, which is then refined with the addition of rectangular or wedge-shaped indentations and extrusions (Figure 12.16c). Note that when top-down maps of the buildings being modeled are available, these can be used to further constrain the 3D modeling process (Robertson and Cipolla 2002, 2009). The idea of using matched 3D lines for estimating vanishing point directions and dominant planes continues to be used in a number of recent fully automated image-based architectural modeling systems (Zebelin, Bauer, Karner *et al.* 2008; Mičušík and Košecká 2009; Furukawa, Curless, Seitz *et al.* 2009b; Sinha, Steedly, and Szeliski 2009).

Another common characteristic of architecture is the repeated use of primitives such as windows, doors, and colonnades. Architectural modeling systems can be designed to search for such repeated elements and to use them as part of the structure inference process (Dick, Torr, and Cipolla 2004; Mueller, Zeng, Wonka *et al.* 2007; Schindler, Krishnamurthy, Lubliner *et al.* 2008; Sinha, Steedly, Szeliski *et al.* 2008).

The combination of all these techniques now makes it possible to reconstruct the structure of large 3D scenes (Zhu and Kanade 2008). For example, the *Urbanscan* system of Polle-

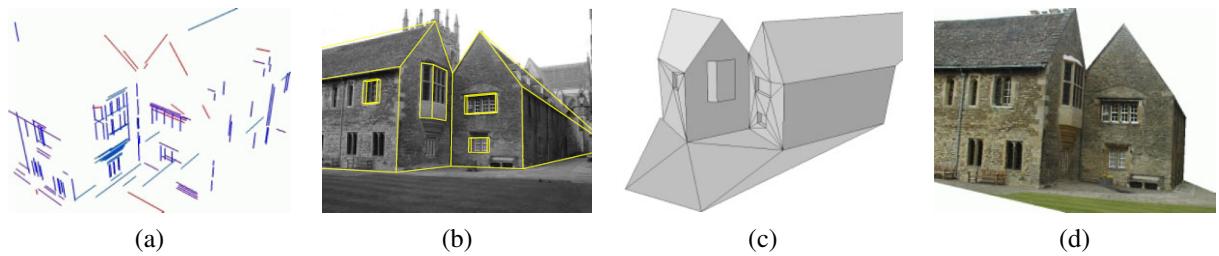


Figure 12.16 Automated architectural reconstruction using 3D lines and planes (Werner and Zisserman 2002) © 2002 Springer: (a) reconstructed 3D lines, color coded by their vanishing directions; (b) wire-frame model superimposed onto an input image; (c) triangulated piecewise-planar model with windows; (d) final texture-mapped model.

feys, Nistér, Frahm *et al.* (2008) reconstructs texture-mapped 3D models of city streets from videos acquired with a GPS-equipped vehicle. To obtain real-time performance, they use both optimized on-line structure-from-motion algorithms, as well as GPU implementations of plane-sweep stereo aligned to dominant planes and depth map fusion. Cornelis, Leibe, Cornelis *et al.* (2008) present a related system that also uses plane-sweep stereo (aligned to vertical building façades) combined with object recognition and segmentation for vehicles. Mičušík and Košecká (2009) build on these results using omni-directional images and super-pixel-based stereo matching along dominant plane orientations. Reconstruction directly from active range scanning data combined with color imagery that has been compensated for exposure and lighting variations is also possible (Chen and Chen 2008; Stamos, Liu, Chen *et al.* 2008; Troccoli and Allen 2008).

12.6.2 Heads and faces

Another area in which specialized shape and appearance models are extremely helpful is in the modeling of heads and faces. Even though the appearance of people seems at first glance to be infinitely variable, the actual shape of a person's head and face can be described reasonably well using a few dozen parameters (Pighin, Hecker, Lischinski *et al.* 1998; Guenter, Grimm, Wood *et al.* 1998; DeCarlo, Metaxas, and Stone 1998; Blanz and Vetter 1999; Shan, Liu, and Zhang 2001).

Figure 12.17 shows an example of an image-based modeling system, where user-specified keypoints in several images are used to fit a generic head model to a person's face. As you can see in Figure 12.17c, after specifying just over 100 keypoints, the shape of the face has become quite adapted and recognizable. Extracting a texture map from the original images and then applying it to the head model results in an animatable model with striking visual fidelity (Figure 12.18a).

A more powerful system can be built by applying *principal component analysis* (PCA) to a collection of 3D scanned faces, which is a topic we discuss in Section 12.6.3. As you can see in Figure 12.19, it is then possible to fit morphable 3D models to single images and to use such models for a variety of animation and visual effects (Blanz and Vetter 1999). It is also possible to design stereo matching algorithms that optimize directly for the head model parameters (Shan, Liu, and Zhang 2001; Kang and Jones 2002) or to use the output of real-

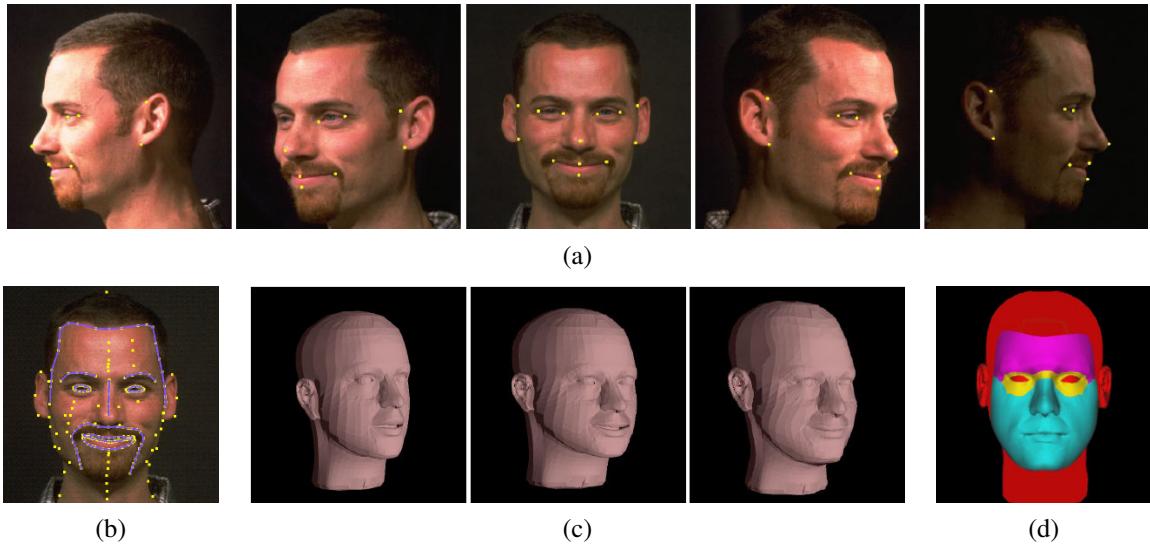


Figure 12.17 3D model fitting to a collection of images: (Pighin, Hecker, Lischinski *et al.* 1998) © 1998 ACM: (a) set of five input images along with user-selected keypoints; (b) the complete set of keypoints and curves; (c) three meshes—the original, adapted after 13 keypoints, and after an additional 99 keypoints; (d) the partition of the image into separately animatable regions.

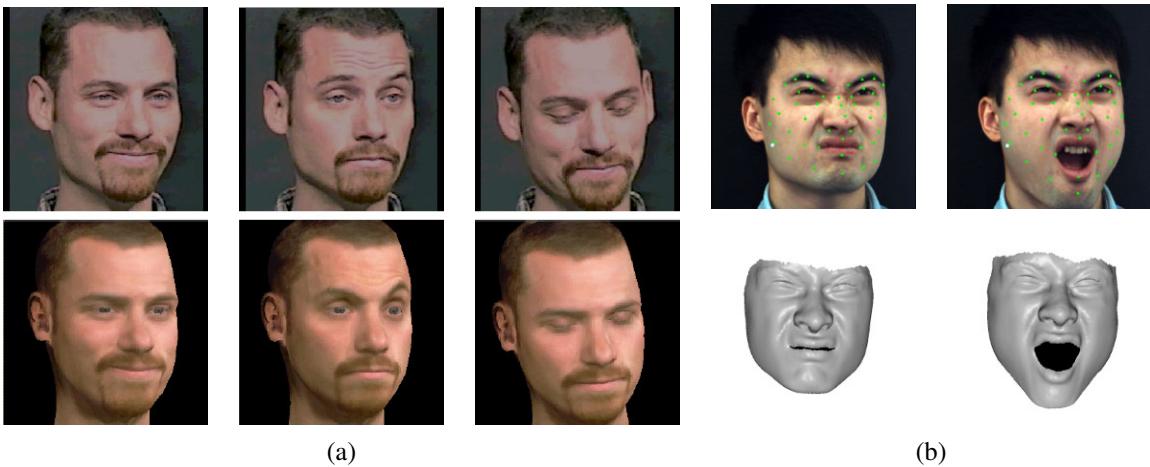


Figure 12.18 Head and expression tracking and re-animation using deformable 3D models. (a) Models fit directly to five input video streams (Pighin, Szeliski, and Salesin 2002) © 2002 Springer: The bottom row shows the results of re-animating a synthetic texture-mapped 3D model with pose and expression parameters fitted to the input images in the top row. (b) Models fit to frame-rate spacetime stereo surface models (Zhang, Snavely, Curless *et al.* 2004) © 2004 ACM: The top row shows the input images with synthetic green markers overlaid, while the bottom row shows the fitted 3D surface model.

time stereo with active illumination (Zhang, Snavely, Curless *et al.* 2004) (Figures 12.7 and 12.18b).

As the sophistication of 3D facial capture systems evolves, so does the detail and realism in the reconstructed models. Newer systems can capture (in real-time) not only surface details such as wrinkles and creases, but also accurate models of skin reflection, translucency, and sub-surface scattering (Weyrich, Matusik, Pfister *et al.* 2006; Golovinskiy, Matusik, ster *et al.* 2006; Bickel, Botsch, Angst *et al.* 2007; Igarashi, Nishino, and Nayar 2007).

Once a 3D head model has been constructed, it can be used in a variety of applications, such as head tracking (Toyama 1998; Lepetit, Pilet, and Fua 2004; Matthews, Xiao, and Baker 2007), as shown in Figures 4.29 and 14.24, and face transfer, i.e., replacing one person’s face with another in a video (Bregler, Covell, and Slaney 1997; Vlasic, Brand, Pfister *et al.* 2005). Additional applications include face beautification by warping face images toward a more attractive “standard” (Leyvand, Cohen-Or, Dror *et al.* 2008), face de-identification for privacy protection (Gross, Sweeney, De la Torre *et al.* 2008), and face swapping (Bitouk, Kumar, Dhillon *et al.* 2008).

12.6.3 Application: Facial animation

Perhaps the most widely used application of 3D head modeling is facial animation. Once a parameterized 3D model of shape and appearance (surface texture) has been constructed, it can be used directly to track a person’s facial motions (Figure 12.18a) and to animate a different character with these same motions and expressions (Pighin, Szeliski, and Salesin 2002).

An improved version of such a system can be constructed by first applying principal component analysis (PCA) to the space of possible head shapes and facial appearances. Blanz and Vetter (1999) describe a system where they first capture a set of 200 colored range scans of faces (Figure 12.19a), which can be represented as a large collection of (X, Y, Z, R, G, B) samples (vertices).⁹ In order for 3D morphing to be meaningful, corresponding vertices in different people’s scans must first be put into correspondence (Pighin, Hecker, Lischinski *et al.* 1998). Once this is done, PCA can be applied to more naturally parameterize the 3D morphable model. The flexibility of this model can be increased by performing separate analyses in different subregions, such as the eyes, nose, and mouth, just as in modular eigenspaces (Moghaddam and Pentland 1997).

After computing a subspace representation, different directions in this space can be associated with different characteristics such as gender, facial expressions, or facial features (Figure 12.19a). As in the work of Rowland and Perrett (1995), faces can be turned into caricatures by exaggerating their displacement from the mean image.

3D morphable models can be fitted to a single image using gradient descent on the error between the input image and the re-synthesized model image, after an initial manual placement of the model in an approximately correct pose, scale, and location (Figures 12.19b–c). The efficiency of this fitting process can be increased using inverse compositional image alignment (8.64–8.65), as described by Romdhani and Vetter (2003).

The resulting texture-mapped 3D model can then be modified to produce a variety of vi-

⁹ A cylindrical coordinate system provides a natural two-dimensional embedding for this collection, but such an embedding is not necessary to perform PCA.



Figure 12.19 3D morphable face model (Blanz and Vetter 1999) © 1999 ACM: (a) original 3D face model with the addition of shape and texture variations in specific directions: deviation from the mean (caricature), gender, expression, weight, and nose shape; (b) a 3D morphable model is fit to a single image, after which its weight or expression can be manipulated; (c) another example of a 3D reconstruction along with a different set of 3D manipulations such as lighting and pose change.

sual effects, including changing a person’s weight or expression, or three-dimensional effects such as re-lighting or 3D video-based animation (Section 13.5.1). Such models can also be used for video compression, e.g., by only transmitting a small number of facial expression and pose parameters to drive a synthetic avatar (Eisert, Wiegand, and Girod 2000; Gao, Chen, Wang *et al.* 2003).

3D facial animation is often matched to the performance of an actor, in what is known as *performance-driven animation* (Section 4.1.5) (Williams 1990). Traditional performance-driven animation systems use marker-based motion capture (Ma, Jones, Chiang *et al.* 2008), while some newer systems use video footage to control the animation (Buck, Finkelstein, Jacobs *et al.* 2000; Pighin, Szeliski, and Salesin 2002; Zhang, Snavely, Curless *et al.* 2004; Vlasic, Brand, Pfister *et al.* 2005).

An example of the latter approach is the system developed for the film *Benjamin Button*, in which Digital Domain used the CONTOUR system from Mova¹⁰ to capture actor Brad Pitt’s facial motions and expressions (Roble and Zafar 2009). CONTOUR uses a combination of phosphorescent paint and multiple high-resolution video cameras to capture real-time 3D range scans of the actor. These 3D models were then translated into Facial Action Coding System (FACS) shape and expression parameters (Ekman and Friesen 1978) to drive a different (older) synthetically animated computer-generated imagery (CGI) character.

12.6.4 Whole body modeling and tracking

The topics of tracking humans, modeling their shape and appearance, and recognizing their activities, are some of the most actively studied areas of computer vision. Annual conferences¹¹ and special journal issues (Hilton, Fua, and Ronfard 2006) are devoted to this subject, and two recent surveys (Forsyth, Arikhan, Ikemoto *et al.* 2006; Moeslund, Hilton, and Krüger 2006) each list over 400 papers devoted to these topics.¹² The HumanEva database of articulated human motions¹³ contains multi-view video sequences of human actions along with corresponding motion capture data, evaluation code, and a reference 3D tracker based on particle filtering. The companion paper by Sigal, Balan, and Black (2010) not only describes the database and evaluation but also has a nice survey of important work in this field.

Given the breadth of this area, it is difficult to categorize all of this research, especially since different techniques usually build on each other. Moeslund, Hilton, and Krüger (2006) divide their survey into initialization, tracking (which includes background modeling and segmentation), pose estimation, and action (activity) recognition. Forsyth, Arikhan, Ikemoto *et al.* (2006) divide their survey into sections on tracking (background subtraction, deformable templates, flow, and probabilistic models), recovering 3D pose from 2D observations, and data association and body parts. They also include a section on motion synthesis, which is more widely studied in computer graphics (Arikhan and Forsyth 2002; Kovar, Gleicher, and Pighin 2002; Lee, Chai, Reitsma *et al.* 2002; Li, Wang, and Shum 2002; Pullen and Bregler

¹⁰ <http://www.mova.com>.

¹¹ International Conference on Automatic Face and Gesture Recognition (FG), IEEE Workshop on Analysis and Modeling of Faces and Gestures, and International Workshop on Tracking Humans for the Evaluation of their Motion in Image Sequences (THEMIS).

¹² Older surveys include those by Gavrila (1999) and Moeslund and Granum (2001). Some surveys on gesture recognition, which we do not cover in this book, include those by Pavlović, Sharma, and Huang (1997) and Yang, Ahuja, and Tabb (2002).

¹³ <http://vision.cs.brown.edu/humaneva/>.

2002), see Section 13.5.2. Another potential taxonomy for work in this field would be along the lines of whether 2D or 3D (or multi-view) images are used as input and whether 2D or 3D kinematic models are used.

In this section, we briefly review some of the more seminal and widely cited papers in the areas of background subtraction, initialization and detection, tracking with flow, 3D kinematic models, probabilistic models, adaptive shape modeling, and activity recognition. We refer the reader to the previously mentioned surveys for other topics and more details.

Background subtraction. One of the first steps in many (but certainly not all) human tracking systems is to model the background in order to extract the moving foreground objects (silhouettes) corresponding to people. Toyama, Krumm, Brumitt *et al.* (1999) review several *difference matting* and *background maintenance* (modeling) techniques and provide a good introduction to this topic. Stauffer and Grimson (1999) describe some techniques based on mixture models, while Sidenbladh and Black (2003) develop a more comprehensive treatment, which models not only the background image statistics but also the appearance of the foreground objects, e.g., their edge and motion (frame difference) statistics.

Once silhouettes have been extracted from one or more cameras, they can then be modeled using deformable templates or other contour models (Baumberg and Hogg 1996; Wren, Azarbayejani, Darrell *et al.* 1997). Tracking such silhouettes over time supports the analysis of multiple people moving around a scene, including building shape and appearance models and detecting if they are carrying objects (Haritaoglu, Harwood, and Davis 2000; Mittal and Davis 2003; Dimitrijevic, Lepetit, and Fua 2006).

Initialization and detection. In order to track people in a fully automated manner, it is necessary to first detect (or re-acquire) their presence in individual video frames. This topic is closely related to *pedestrian detection*, which is often considered as a kind of object recognition (Mori, Ren, Efros *et al.* 2004; Felzenszwalb and Huttenlocher 2005; Felzenszwalb, McAllester, and Ramanan 2008), and is therefore treated in more depth in Section 14.1.2. Additional techniques for initializing 3D trackers based on 2D images include those described by Howe, Leventon, and Freeman (2000), Rosales and Sclaroff (2000), Shakhnarovich, Viola, and Darrell (2003), Sminchisescu, Kanaujia, Li *et al.* (2005), Agarwal and Triggs (2006), Lee and Cohen (2006), Sigal and Black (2006), and Stenger, Thayananthan, Torr *et al.* (2006).

Single-frame human detection and pose estimation algorithms can sometimes be used by themselves to perform tracking (Ramanan, Forsyth, and Zisserman 2005; Rogez, Rihan, Ramalingam *et al.* 2008; Bourdev and Malik 2009), as described in Section 4.1.4. More often, however, they are combined with frame-to-frame tracking techniques to provide better reliability (Fossati, Dimitrijevic, Lepetit *et al.* 2007; Andriluka, Roth, and Schiele 2008; Ferrari, Marin-Jimenez, and Zisserman 2008).

Tracking with flow. The tracking of people and their pose from frame to frame can be enhanced by computing optic flow or matching the appearance of their limbs from one frame to another. For example, the *cardboard people* model of Ju, Black, and Yacoob (1996) models the appearance of each leg portion (upper and lower) as a moving rectangle, and uses optic flow to estimate their location in each subsequent frame. Cham and Rehg (1999) and Sidenbladh, Black, and Fleet (2000) track limbs using optical flow and templates, along with

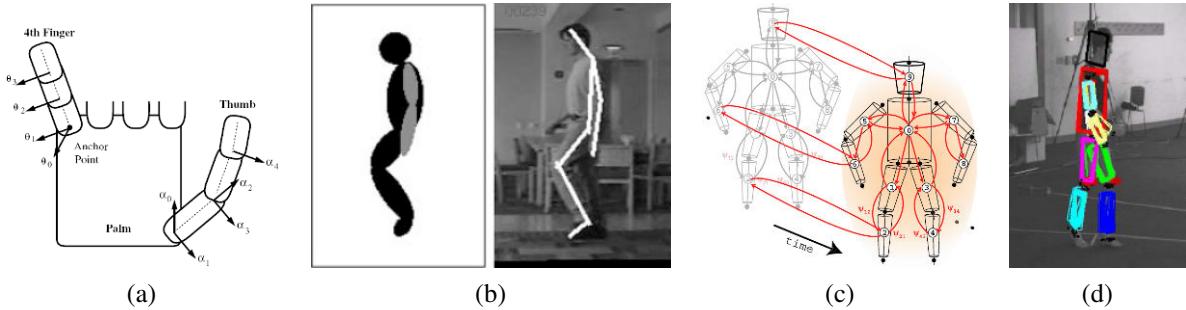


Figure 12.20 Tracking 3D human motion: (a) kinematic chain model for a human hand (Rehg, Morris, and Kanade 2003) © 2003, reprinted by permission of SAGE; (b) tracking a kinematic chain blob model in a video sequence (Bregler, Malik, and Pullen 2004) © 2004 Springer; (c-d) probabilistic loose-limbed collection of body parts (Sigal, Bhatia, Roth *et al.* 2004)

techniques for dealing with multiple hypotheses and uncertainty. Bregler, Malik, and Pullen (2004) use a full 3D model of limb and body motion, as described below. It is also possible to match the estimated motion field itself to some prototypes in order to identify the particular phase of a running motion or to match two low-resolution video portions in order to perform video replacement (Efros, Berg, Mori *et al.* 2003).

3D kinematic models. The effectiveness of human modeling and tracking can be greatly enhanced using a more accurate 3D model of a person’s shape and motion. Underlying such representations, which are ubiquitous in 3D computer animation in games and special effects, is a *kinematic model* or *kinematic chain*, which specifies the length of each limb in a skeleton as well as the 2D or 3D rotation angles between the limbs or segments (Figure 12.20a–b). Inferring the values of the joint angles from the locations of the visible surface points is called *inverse kinematics* (IK) and is widely studied in computer graphics.

Figure 12.20a shows the kinematic model for a human hand used by Rehg, Morris, and Kanade (2003) to track hand motion in a video. As you can see, the attachment points between the fingers and the thumb have two degrees of freedom, while the finger joints themselves have only one. Using this kind of model can greatly enhance the ability of an edge-based tracker to cope with rapid motion, ambiguities in 3D pose, and partial occlusions.

Kinematic chain models are even more widely used for whole body modeling and tracking (O’Rourke and Badler 1980; Hogg 1983; Rohr 1994). One popular approach is to associate an ellipsoid or superquadric with each rigid limb in the kinematic model, as shown in Figure 12.20b. This model can then be fitted to each frame in one or more video streams either by matching silhouettes extracted from known backgrounds or by matching and tracking the locations of occluding edges (Gavrila and Davis 1996; Kakadiaris and Metaxas 2000; Bregler, Malik, and Pullen 2004; Kehl and Van Gool 2006). Note that some techniques use 2D models coupled to 2D measurements, some use 3D measurements (range data or multi-view video) with 3D models, and some use monocular video to infer and track 3D models directly.

It is also possible to use temporal models to improve the tracking of periodic motions, such as walking, by analyzing the joint angles as functions of time (Polana and Nelson 1997; Seitz and Dyer 1997; Cutler and Davis 2000). The generality and applicability of such tech-

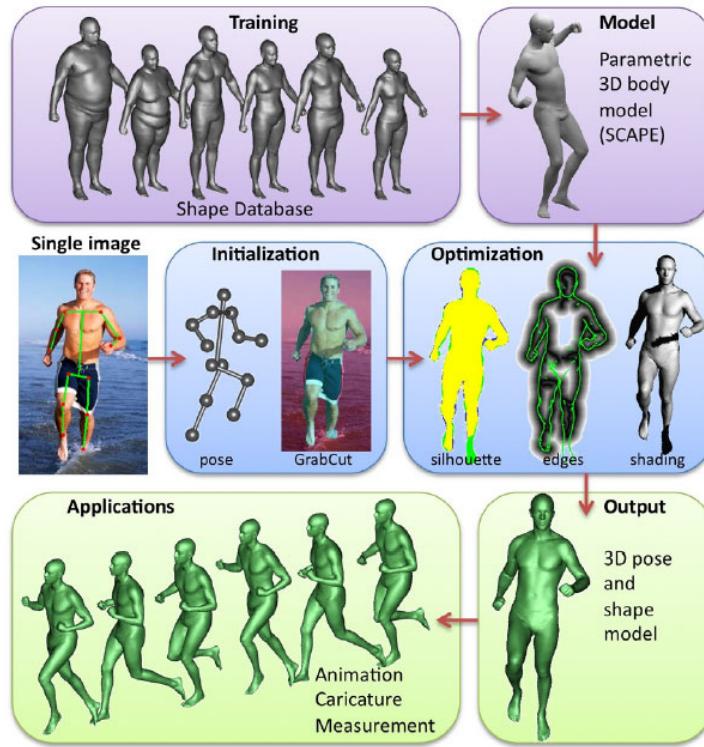


Figure 12.21 Estimating human shape and pose from a single image using a parametric 3D model (Guan, Weiss, Bălan *et al.* 2009) © 2009 IEEE.

niques can be improved by learning typical motion patterns using principal component analysis (Sidenbladh, Black, and Fleet 2000; Urtasun, Fleet, and Fua 2006).

Probabilistic models. Because tracking can be such a difficult task, sophisticated probabilistic inference techniques are often used to estimate the likely states of the person being tracked. One popular approach, called *particle filtering* (Isard and Blake 1998), was originally developed for tracking the outlines of people and hands, as described in Section 5.1.2 (Figures 5.6–5.8). It was subsequently applied to whole-body tracking (Deutscher, Blake, and Reid 2000; Sidenbladh, Black, and Fleet 2000; Deutscher and Reid 2005) and continues to be used in modern trackers (Ong, Micilotta, Bowden *et al.* 2006). Alternative approaches to handling the uncertainty inherent in tracking include multiple hypothesis tracking (Cham and Rehg 1999) and inflated covariances (Sminchisescu and Triggs 2001).

Figure 12.20c–d shows an example of a sophisticated spatio-temporal probabilistic graphical model called *loose-limbed people*, which models not only the geometric relationship between various limbs, but also their likely temporal dynamics (Sigal, Bhatia, Roth *et al.* 2004). The conditional probabilities relating various limbs and time instances are learned from training data, and particle filtering is used to perform the final pose inference.

Adaptive shape modeling. Another essential component of whole body modeling and tracking is the fitting of parameterized shape models to visual data. As we saw in Section 12.6.3 (Figure 12.19), the availability of large numbers of registered 3D range scans can be used to create *morphable models* of shape and appearance (Allen, Curless, and Popović 2003). Building on this work, Anguelov, Srinivasan, Koller *et al.* (2005) develop a sophisticated system called SCAPE (Shape Completion and Animation for PEople), which first acquires a large number of range scans of different people and of one person in different poses, and then registers these scans using semi-automated marker placement. The registered datasets are used to model the variation in shape as a function of personal characteristics and skeletal pose, e.g., the bulging of muscles as certain joints are flexed (Figure 12.21, top row). The resulting system can then be used for *shape completion*, i.e., the recovery of a full 3D mesh model from a small number of captured markers, by finding the best model parameters in both shape and pose space that fit the measured data.

Because it is constructed completely from scans of people in close-fitting clothing and uses a parametric shape model, the SCAPE system cannot cope with people wearing loose-fitting clothing. Bălan and Black (2008) overcome this limitation by estimating the body shape that fits within the visual hull of the same person observed in multiple poses, while Vlasic, Baran, Matusik *et al.* (2008) adapt an initial surface mesh fitted with a parametric shape model to better match the visual hull.

While the preceding body fitting and pose estimation systems use multiple views to estimate body shape, even more recent work by Guan, Weiss, Bălan *et al.* (2009) can fit a human shape and pose model to a single image of a person on a natural background. Manual initialization is used to estimate a rough pose (skeleton) and height model, and this is then used to segment the person’s outline using the Grab Cut segmentation algorithm (Section 5.5). The shape and pose estimate are then refined using a combination of silhouette edge cues and shading information (Figure 12.21). The resulting 3D model can be used to create novel animations.

Activity recognition. The final widely studied topic in human modeling is motion, activity, and action recognition (Bobick 1997; Hu, Tan, Wang *et al.* 2004; Hilton, Fua, and Ronfard 2006). Examples of actions that are commonly recognized include walking and running, jumping, dancing, picking up objects, sitting down and standing up, and waving. Recent representative papers on these topics have been written by Robertson and Reid (2006), Sminchisescu, Kanaujia, and Metaxas (2006), Weinland, Ronfard, and Boyer (2006), Yilmaz and Shah (2006), and Gorelick, Blank, Shechtman *et al.* (2007).

12.7 Recovering texture maps and albedos

After a 3D model of an object or person has been acquired, the final step in modeling is usually to recover a *texture map* to describe the object’s surface appearance. This first requires establishing a parameterization for the (u, v) texture coordinates as a function of 3D surface position. One simple way to do this is to associate a separate texture map with each triangle (or pair of triangles). More space-efficient techniques involve unwrapping the surface onto one or more maps, e.g., using a subdivision mesh (Section 12.3.2) (Eck, DeRose, Duchamp *et al.* 1995) or a geometry image (Section 12.3.3) (Gu, Gortler, and Hoppe 2002).



Figure 12.22 Estimating the diffuse albedo and reflectance parameters for a scanned 3D model (Sato, Wheeler, and Ikeuchi 1997) © 1997 ACM: (a) set of input images projected onto the model; (b) the complete diffuse reflection (albedo) model; (c) rendering from the reflectance model including the specular component.

Once the (u, v) coordinates for each triangle have been fixed, the perspective projection equations mapping from texture (u, v) to an image j 's pixel (u_j, v_j) coordinates can be obtained by concatenating the affine $(u, v) \rightarrow (X, Y, Z)$ mapping with the perspective homography $(X, Y, Z) \rightarrow (u_j, v_j)$ (Szeliski and Shum 1997). The color values for the (u, v) texture map can then be re-sampled and stored, or the original image can itself be used as the texture source using projective texture mapping (OpenGL-ARB 1997).

The situation becomes more involved when more than one source image is available for appearance recovery, which is the usual case. One possibility is to use a *view-dependent texture map* (Section 13.1.1), in which a different source image (or combination of source images) is used for each polygonal face based on the angles between the virtual camera, the surface normals, and the source images (Debevec, Taylor, and Malik 1996; Pighin, Hecker, Lischinski *et al.* 1998). An alternative approach is to estimate a complete Surface Light Field for each surface point (Wood, Azuma, Aldinger *et al.* 2000), as described in Section 13.3.2.

In some situations, e.g., when using models in traditional 3D games, it is preferable to merge all of the source images into a single coherent texture map during pre-processing. Ideally, each surface triangle should select the source image where it is seen most directly (perpendicular to its normal) and at the resolution best matching the texture map resolution.¹⁴ This can be posed as a graph cut optimization problem, where the smoothness term encourages adjacent triangles to use similar source images, followed by blending to compensate for exposure differences (Lempitsky and Ivanov 2007; Sinha, Steedly, Szeliski *et al.* 2008). Even better results can be obtained by explicitly modeling geometric and photometric misalignments between the source images (Shum and Szeliski 2000; Gal, Wexler, Ofek *et al.* 2010).

These kinds of approaches produce good results when the lighting stays fixed with respect to the object, i.e., when the camera moves around the object or space. When the lighting is strongly directional, however, and the object is being moved relative to this lighting, strong shading effects or specularities may be present, which will interfere with the reliable recovery of a texture (albedo) map. In this case, it is preferable to explicitly undo the shading effects (Section 12.1) by modeling the light source directions and estimating the surface reflectance properties while recovering the texture map (Sato and Ikeuchi 1996; Sato, Wheeler, and Ikeuchi 1997; Yu and Malik 1998; Yu, Debevec, Malik *et al.* 1999). Figure 12.22 shows

¹⁴ When surfaces are seen at oblique viewing angles, it may be necessary to blend different images together to obtain the best resolution (Wang, Kang, Szeliski *et al.* 2001).

the results of one such approach, where the specularities are first removed while estimating the matte reflectance component (albedo) and then later re-introduced by estimating the specular component k_s in a Torrance–Sparrow reflection model (2.91).

12.7.1 Estimating BRDFs

A more ambitious approach to the problem of view-dependent appearance modeling is to estimate a general bidirectional reflectance distribution function (BRDF) for each point on an object’s surface. Dana, van Ginneken, Nayar *et al.* (1999), Jensen, Marschner, Levoy *et al.* (2001), and Lensch, Kautz, Goesele *et al.* (2003) present different techniques for estimating such functions, while Dorsey, Rushmeier, and Sillion (2007) and Weyrich, Lawrence, Lensch *et al.* (2008) present more recent surveys of the topics of BRDF modeling, recovery, and rendering.

As we saw in Section 2.2.2 (2.81), the BRDF can be written as

$$f_r(\theta_i, \phi_i, \theta_r, \phi_r; \lambda), \quad (12.9)$$

where (θ_i, ϕ_i) and (θ_r, ϕ_r) are the angles the incident \hat{v}_i and reflected \hat{v}_r light ray directions make with the local surface coordinate frame $(\hat{d}_x, \hat{d}_y, \hat{n})$ shown in Figure 2.15. When modeling the appearance of an object, as opposed to the appearance of a patch of material, we need to estimate this function at every point (x, y) on the object’s surface, which gives us the *spatially varying* BRDF, or SVBRDF (Weyrich, Lawrence, Lensch *et al.* 2008),

$$f_v(x, y, \theta_i, \phi_i, \theta_r, \phi_r; \lambda). \quad (12.10)$$

If sub-surface scattering effects are being modeled, such as the long-range transmission of light through materials such as alabaster, the eight-dimensional bidirectional scattering-surface reflectance-distribution function (BSSRDF) is used instead,

$$f_e(x_i, y_i, \theta_i, \phi_i, x_e, y_e, \theta_e, \phi_e; \lambda), \quad (12.11)$$

where the e subscript now represents the *emitted* rather than the *reflected* light directions.

Weyrich, Lawrence, Lensch *et al.* (2008) provide a nice survey of these and related topics, including basic photometry, BRDF models, traditional BRDF acquisition using *gonio reflectometry* (the precise measurement of visual angles and reflectances), multiplexed illumination (Schechner, Nayar, and Belhumeur 2009), skin modeling (Debevec, Hawkins, Tchou *et al.* 2000; Weyrich, Matusik, Pfister *et al.* 2006), and image-based acquisition techniques, which simultaneously recover an object’s 3D shape and reflectometry from multiple photographs.

A nice example of this latter approach is the system developed by Lensch, Kautz, Goesele *et al.* (2003), who estimate locally varying BRDFs and refine their shape models using local estimates of surface normals. To build up their models, they first associate a *lumitexels*, which contains a 3D position, a surface normal, and a set of sparse radiance samples, with each surface point. Next, they cluster such lumitexels into materials that share common properties, using a Lafourche reflectance model (Lafourche, Foo, Torrance *et al.* 1997) and a divisive clustering approach (Figure 12.23a). Finally, in order to model detailed spatially varying appearance, each lumitexel (surface point) is projected onto the basis of clustered appearance models (Figure 12.23b).

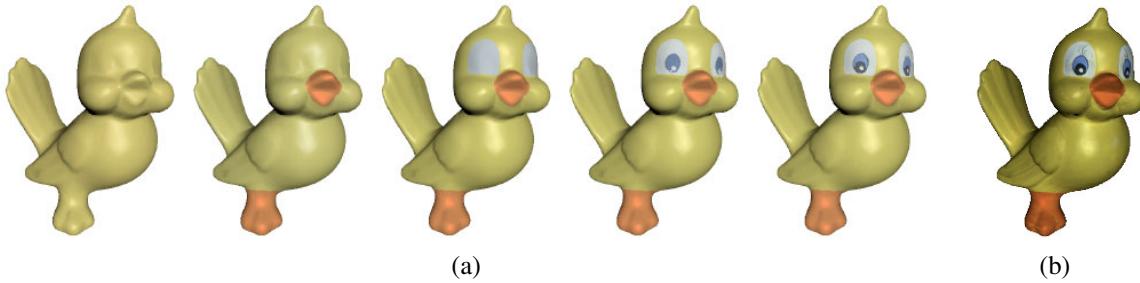


Figure 12.23 Image-based reconstruction of appearance and detailed geometry (Lensch, Kautz, Goesele *et al.* 2003) © 2003 ACM. (a) Appearance models (BRDFs) are re-estimated using divisive clustering. (b) In order to model detailed spatially varying appearance, each lumitexel is projected onto the basis formed by the clustered materials.

While most of the techniques discussed in this section require large numbers of views to estimate surface properties, a challenging future direction will be to take these techniques out of the lab and into the real world, and to combine them with regular and Internet photo image-based modeling approaches.

12.7.2 Application: 3D photography

The techniques described in this chapter for building complete 3D models from multiple images and then recovering their surface appearance have opened up a whole new range of applications that often go under the name *3D photography*. Pollefeys and Van Gool (2002) provide a nice introduction to this field, including the processing steps of feature matching, structure from motion recovery,¹⁵ dense depth map estimation, 3D model building, and texture map recovery. A complete Web-based system for automatically performing all of these tasks, called ARC3D, is described by Vergauwen and Van Gool (2006) and Moons, Van Gool, and Vergauwen (2010). The latter paper provides not only an in-depth survey of this whole field but also a detailed description of their complete end-to-end system.

An alternative to such fully automated systems is to put the user in the loop in what is sometimes called *interactive computer vision*. van den Hengel, Dick, Thormhlen *et al.* (2007) describe their VideoTrace system, which performs automated point tracking and 3D structure recovery from video and then lets the user draw triangles and surfaces on top of the resulting point cloud, as well as interactively adjusting the locations of model vertices. Sinha, Steedly, Szeliski *et al.* (2008) describe a related system that uses matched vanishing points in multiple images (Figure 4.45) to infer 3D line orientations and plane normals. These are then used to guide the user drawing axis-aligned planes, which are automatically fitted to the recovered 3D point cloud. Fully automated variants on these ideas are described by Zebedin, Bauer, Karner *et al.* (2008), Furukawa, Curless, Seitz *et al.* (2009a), Furukawa, Curless, Seitz *et al.* (2009b), Mičušík and Košecká (2009), and Sinha, Steedly, and Szeliski (2009).

As the sophistication and reliability of these techniques continues to improve, we can expect to see even more user-friendly applications for photorealistic 3D modeling from images (Exercise 12.8).

¹⁵ These earlier steps are also discussed in Section 7.4.4.

12.8 Additional reading

Shape from shading is one of the classic problems in computer vision (Horn 1975). Some representative papers in this area include those by Horn (1977), Ikeuchi and Horn (1981), Pentland (1984), Horn and Brooks (1986), Horn (1990), Szeliski (1991a), Mancini and Wolff (1992), Dupuis and Oliensis (1994), and Fua and Leclerc (1995). The collection of papers edited by Horn and Brooks (1989) is a great source of information on this topic, especially the chapter on variational approaches. The survey by Zhang, Tsai, Cryer *et al.* (1999) not only reviews more recent techniques but also provides some comparative results.

Woodham (1981) wrote the seminal paper of photometric stereo. Shape from texture techniques include those by Witkin (1981), Ikeuchi (1981), Blostein and Ahuja (1987), Garding (1992), Malik and Rosenholtz (1997), Liu, Collins, and Tsin (2004), Liu, Lin, and Hays (2004), Hays, Leordeanu, Efros *et al.* (2006), Lin, Hays, Wu *et al.* (2006), Lobay and Forsyth (2006), White and Forsyth (2006), White, Crane, and Forsyth (2007), and Park, Brocklehurst, Collins *et al.* (2009). Good papers and books on depth from defocus have been written by Pentland (1987), Nayar and Nakagawa (1994), Nayar, Watanabe, and Noguchi (1996), Watanabe and Nayar (1998), Chaudhuri and Rajagopalan (1999), and Favaro and Soatto (2006). Additional techniques for recovering shape from various kinds of illumination effects, including inter-reflections (Nayar, Ikeuchi, and Kanade 1991), are discussed in the book on shape recovery edited by Wolff, Shafer, and Healey (1992b).

Active rangefinding systems, which use laser or natural light illumination projected into the scene, have been described by Besl (1989), Rioux and Bird (1993), Kang, Webb, Zitnick *et al.* (1995), Curless and Levoy (1995), Curless and Levoy (1996), Proesmans, Van Gool, and Defoort (1998), Bouguet and Perona (1999), Curless (1999), Hebert (2000), Idan and Yahav (2001), Goesele, Fuchs, and Seidel (2003), Scharstein and Szeliski (2003), Davis, Ramamoorthi, and Rusinkiewicz (2003), Zhang, Curless, and Seitz (2003), Zhang, Snavely, Curless *et al.* (2004), and Moons, Van Gool, and Vergauwen (2010). Individual range scans can be aligned using 3D correspondence and distance optimization techniques such as *iterated closest points* and its variants (Besl and McKay 1992; Zhang 1994; Szeliski and Lavallée 1996; Johnson and Kang 1997; Gold, Rangarajan, Lu *et al.* 1998; Johnson and Hebert 1999; Pulli 1999; David, DeMenthon, Duraiswami *et al.* 2004; Li and Hartley 2007; Enqvist, Josephson, and Kahl 2009). Once they have been aligned, range scans can be merged using techniques that model the signed distance of surfaces to volumetric sample points (Hoppe, DeRose, Duchamp *et al.* 1992; Curless and Levoy 1996; Hilton, Stoddart, Illingworth *et al.* 1996; Wheeler, Sato, and Ikeuchi 1998; Kazhdan, Bolitho, and Hoppe 2006; Lempitsky and Boykov 2007; Zach, Pock, and Bischof 2007b; Zach 2008).

Once constructed, 3D surfaces can be modeled and manipulated using a variety of three-dimensional representations, which include triangle meshes (Eck, DeRose, Duchamp *et al.* 1995; Hoppe 1996), splines (Farin 1992, 1996; Lee, Wolberg, and Shin 1996), subdivision surfaces (Stollnitz, DeRose, and Salesin 1996; Zorin, Schröder, and Sweldens 1996; Warren and Weimer 2001; Peters and Reif 2008), and geometry images (Gu, Gortler, and Hoppe 2002). Alternatively, they can be represented as collections of point samples with local orientation estimates (Hoppe, DeRose, Duchamp *et al.* 1992; Szeliski and Tonnesen 1992; Turk and O'Brien 2002; Pfister, Zwicker, van Baar *et al.* 2000; Alexa, Behr, Cohen-Or *et al.* 2003; Pauly, Keiser, Kobbelt *et al.* 2003; Diebel, Thrun, and Brünig 2006; Guennebaud and Gross

2007; Guennebaud, Germann, and Gross 2008; Oztireli, Guennebaud, and Gross 2008). They can also be modeled using implicit inside–outside characteristic or signed distance functions sampled on regular or irregular (octree) volumetric grids (Lavallée and Szeliski 1995; Szeliski and Lavallée 1996; Frisken, Perry, Rockwood *et al.* 2000; Dinh, Turk, and Slabaugh 2002; Kazhdan, Bolitho, and Hoppe 2006; Lempitsky and Boykov 2007; Zach, Pock, and Bischof 2007b; Zach 2008).

The literature on model-based 3D reconstruction is extensive. For modeling architecture and urban scenes, both interactive and fully automated systems have been developed. A special journal issue devoted to the reconstruction of large-scale 3D scenes (Zhu and Kanade 2008) is a good source of references and Robertson and Cipolla (2009) give a nice description of a complete system. Lots of additional references can be found in Section 12.6.1.

Face and whole body modeling and tracking is a very active sub-field of computer vision, with its own conferences and workshops, e.g., the International Conference on Automatic Face and Gesture Recognition (FG), the IEEE Workshop on Analysis and Modeling of Faces and Gestures, and the International Workshop on Tracking Humans for the Evaluation of their Motion in Image Sequences (THEMIS). Recent survey articles on the topic of whole body modeling and tracking include those by Forsyth, Arikian, Ikemoto *et al.* (2006), Moeslund, Hilton, and Krüger (2006), and Sigal, Balan, and Black (2010).

12.9 Exercises

Ex 12.1: Shape from focus Grab a series of focused images with a digital SLR set to manual focus (or get one that allows for programmatic focus control) and recover the depth of an object.

1. Take some calibration images, e.g., of a checkerboard, so you can compute a mapping between the amount of defocus and the focus setting.
2. Try both a fronto-parallel planar target and one which is slanted so that it covers the working range of the sensor. Which one works better?
3. Now put a real object in the scene and perform a similar focus sweep.
4. For each pixel, compute the local sharpness and fit a parabolic curve over focus settings to find the most in-focus setting.
5. Map these focus settings to depth and compare your result to ground truth. If you are using a known simple object, such as sphere or cylinder (a ball or a soda can), it's easy to measure its true shape.
6. (Optional) See if you can recover the depth map from just two or three focus settings.
7. (Optional) Use an LCD projector to project artificial texture onto the scene. Use a pair of cameras to compare the accuracy of your shape from focus and shape from stereo techniques.
8. (Optional) Create an all-in-focus image using the technique of Agarwala, Dontcheva, Agrawala *et al.* (2004).

Ex 12.2: Shadow striping Implement the handheld shadow striping system of Bouguet and Perona (1999). The basic steps include the following.

1. Set up two background planes behind the object of interest and calculate their orientation relative to the viewer, e.g., with fiducial marks.
2. Cast a moving shadow with a stick across the scene; record the video or capture the data with a webcam.
3. Estimate each light plane equation from the projections of the cast shadow against the two backgrounds.
4. Triangulate to the remaining points on each curve to get a 3D stripe and display the stripes using a 3D graphics engine.
5. (Optional) remove the requirement for a known second (vertical) plane and infer its location (or that of the light source) using the techniques described by Bouguet and Perona (1999). The techniques from Exercise 10.9 may also be helpful here.

Ex 12.3: Range data registration Register two or more 3D datasets using either iterated closest points (ICP) (Besl and McKay 1992; Zhang 1994; Gold, Rangarajan, Lu *et al.* 1998) or octree signed distance fields (Szeliski and Lavallée 1996) (Section 12.2.1).

Apply your technique to narrow-baseline stereo pairs, e.g., obtained by moving a camera around an object, using structure from motion to recover the camera poses, and using a standard stereo matching algorithm.

Ex 12.4: Range data merging Merge the datasets that you registered in the previous exercise using signed distance fields (Curless and Levoy 1996; Hilton, Stoddart, Illingworth *et al.* 1996). You can optionally use an octree to represent and compress this field if you already implemented it in the previous registration step.

Extract a meshed surface model from the signed distance field using marching cubes and display the resulting model.

Ex 12.5: Surface simplification Use progressive meshes (Hoppe 1996) or some other technique from Section 12.3.2 to create a hierarchical simplification of your surface model.

Ex 12.6: Architectural modeler Build a 3D interior or exterior model of some architectural structure, such as your house, from a series of handheld wide-angle photographs.

1. Extract lines and vanishing points (Exercises 4.11–4.15) to estimate the dominant directions in each image.
2. Use structure from motion to recover all of the camera poses and match up the vanishing points.
3. Let the user sketch the locations of the walls by drawing lines corresponding to wall bottoms, tops, and horizontal extents onto the images (Sinha, Steedly, Szeliski *et al.* 2008)—see also Exercise 6.9. Do something similar for openings (doors and windows) and simple furniture (tables and countertops).

4. Convert the resulting polygonal meshes into a 3D model (e.g., VRML) and optionally texture-map these surfaces from the images.

Ex 12.7: Body tracker Download the video sequences from the HumanEva Web site.¹⁶ Either implement a human motion tracker from scratch or extend the code on that Web site (Sigal, Balan, and Black 2010) in some interesting way.

Ex 12.8: 3D photography Combine all of your previously developed techniques to produce a system that takes a series of photographs or a video and constructs a photorealistic texture-mapped 3D model.

¹⁶ <http://vision.cs.brown.edu/humaneva/>.

Chapter 13

Image-based rendering

| | | |
|--------|--|-----|
| 13.1 | View interpolation | 545 |
| 13.1.1 | View-dependent texture maps | 547 |
| 13.1.2 | <i>Application:</i> Photo Tourism | 548 |
| 13.2 | Layered depth images | 549 |
| 13.2.1 | Impostors, sprites, and layers | 549 |
| 13.3 | Light fields and Lumigraphs | 551 |
| 13.3.1 | Unstructured Lumigraph | 554 |
| 13.3.2 | Surface light fields | 555 |
| 13.3.3 | <i>Application:</i> Concentric mosaics | 556 |
| 13.4 | Environment mattes | 556 |
| 13.4.1 | Higher-dimensional light fields | 558 |
| 13.4.2 | The modeling to rendering continuum | 559 |
| 13.5 | Video-based rendering | 560 |
| 13.5.1 | Video-based animation | 560 |
| 13.5.2 | Video textures | 561 |
| 13.5.3 | <i>Application:</i> Animating pictures | 564 |
| 13.5.4 | 3D Video | 564 |
| 13.5.5 | <i>Application:</i> Video-based walkthroughs | 566 |
| 13.6 | Additional reading | 569 |
| 13.7 | Exercises | 570 |

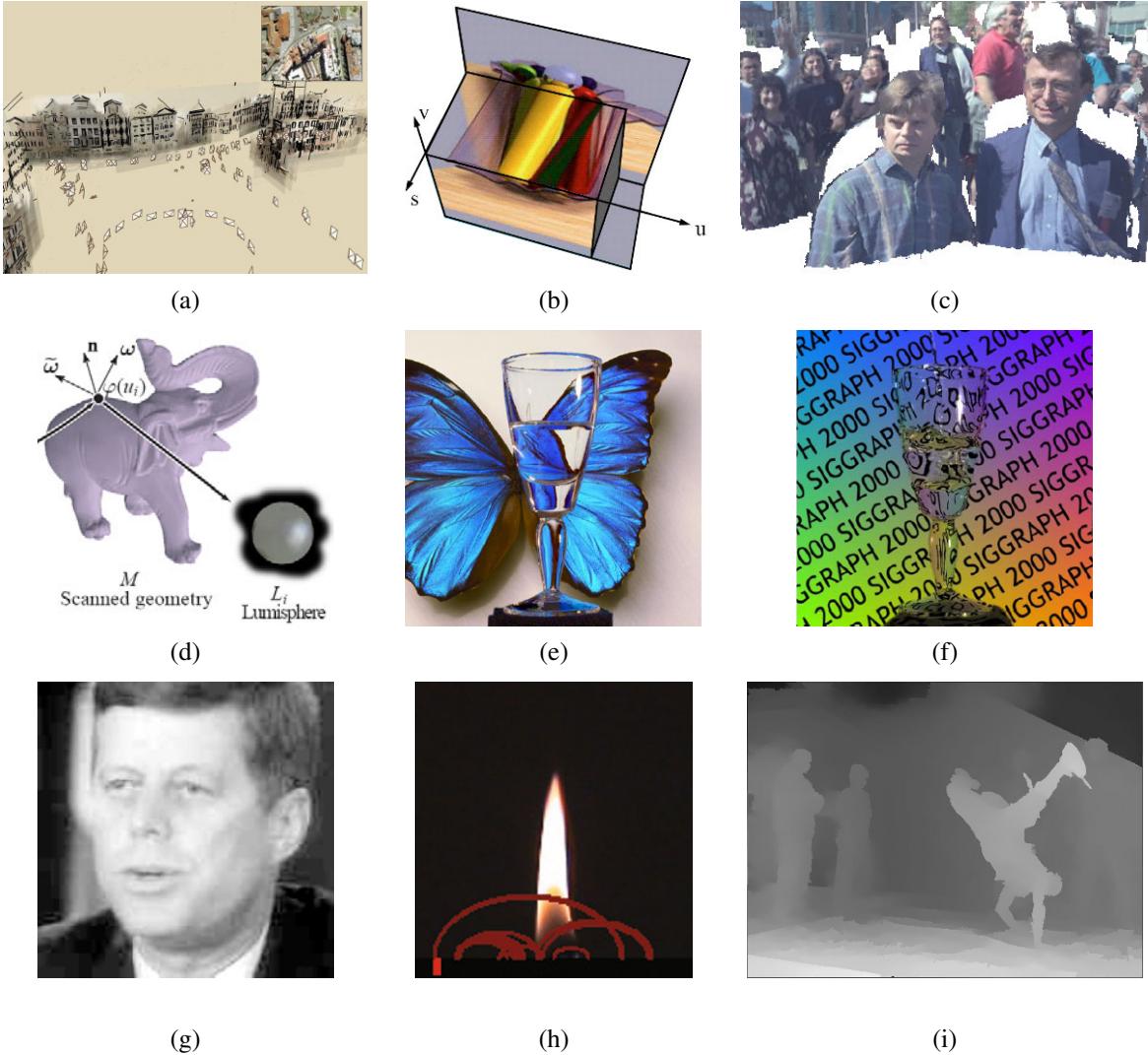


Figure 13.1 Image-based and video-based rendering: (a) a 3D view of a Photo Tourism reconstruction (Snavely, Seitz, and Szeliski 2006) © 2006 ACM; (b) a slice through a 4D light field (Gortler, Grzeszczuk, Szeliski *et al.* 1996) © 1996 ACM; (c) sprites with depth (Shade, Gortler, He *et al.* 1998) © 1998 ACM; (d) surface light field (Wood, Azuma, Aldinger *et al.* 2000) © 2000 ACM; (e) environment matte in front of a novel background (Zongker, Werner, Curless *et al.* 1999) © 1999 ACM; (f) real-time video environment matte (Chuang, Zongker, Hindorff *et al.* 2000) © 2000 ACM; (g) Video Rewrite used to re-animate old video (Bregler, Covell, and Slaney 1997) © 1997 ACM; (h) video texture of a candle flame (Schödl, Szeliski, Salesin *et al.* 2000) © 2000 ACM; (i) video view interpolation (Zitnick, Kang, Uyttendaele *et al.* 2004) © 2004 ACM.

Over the last two decades, image-based rendering has emerged as one of the most exciting applications of computer vision (Kang, Li, Tong *et al.* 2006; Shum, Chan, and Kang 2007). In image-based rendering, 3D reconstruction techniques from computer vision are combined with computer graphics rendering techniques that use multiple views of a scene to create interactive photo-realistic experiences, such as the Photo Tourism system shown in Figure 13.1a. Commercial versions of such systems include immersive street-level navigation in on-line mapping systems¹ and the creation of 3D Photosynths² from large collections of casually acquired photographs.

In this chapter, we explore a variety of image-based rendering techniques, such as those illustrated in Figure 13.1. We begin with *view interpolation* (Section 13.1), which creates a seamless transition between a pair of reference images using one or more pre-computed depth maps. Closely related to this idea are *view-dependent texture maps* (Section 13.1.1), which blend multiple texture maps on a 3D model’s surface. The representations used for both the color imagery and the 3D geometry in view interpolation include a number of clever variants such as *layered depth images* (Section 13.2) and *sprites with depth* (Section 13.2.1).

We continue our exploration of image-based rendering with the *light field* and *Lumigraph* four-dimensional representations of a scene’s appearance (Section 13.3), which can be used to render the scene from any arbitrary viewpoint. Variants on these representations include the *unstructured Lumigraph* (Section 13.3.1), *surface light fields* (Section 13.3.2), *concentric mosaics* (Section 13.3.3), and *environment mattes* (Section 13.4).

The last part of this chapter explores the topic of *video-based rendering*, which uses one or more videos in order to create novel video-based experiences (Section 13.5). The topics we cover include video-based facial animation (Section 13.5.1), as well as *video textures* (Section 13.5.2), in which short video clips can be seamlessly looped to create dynamic real-time video-based renderings of a scene. We close with a discussion of *3D videos* created from multiple video streams (Section 13.5.4), as well as *video-based walkthroughs* of environments (Section 13.5.5), which have found widespread application in immersive outdoor mapping and driving direction systems.

13.1 View interpolation

While the term *image-based rendering* first appeared in the papers by Chen (1995) and McMillan and Bishop (1995), the work on *view interpolation* by Chen and Williams (1993) is considered as the seminal paper in the field. In view interpolation, pairs of rendered color images are combined with their pre-computed depth maps to generate interpolated views that mimic what a virtual camera would see in between the two reference views.

View interpolation combines two ideas that were previously used in computer vision and computer graphics. The first is the idea of pairing a recovered depth map with the reference image used in its computation and then using the resulting texture-mapped 3D model to generate novel views (Figure 11.1). The second is the idea of *morphing* (Section 3.6.3) (Figure 3.53), where correspondences between pairs of images are used to warp each reference image to an in-between location while simultaneously cross-dissolving between the two warped images.

¹ <http://maps.bing.com> and <http://maps.google.com>.

² <http://photosynth.net>.

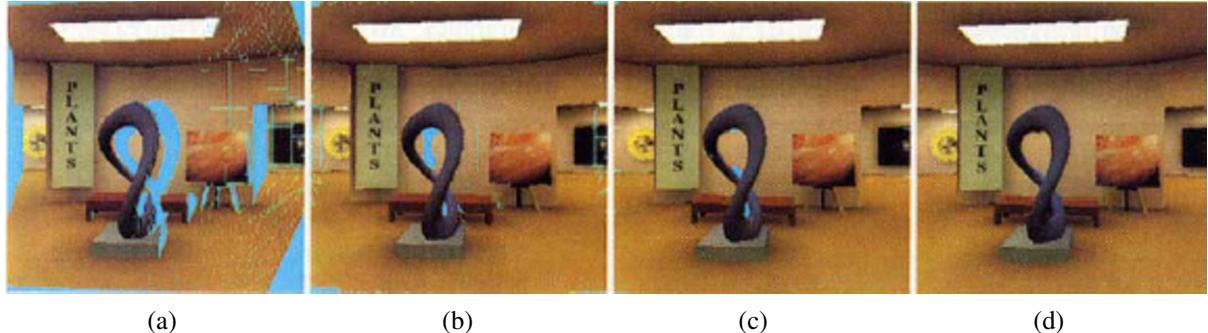


Figure 13.2 View interpolation (Chen and Williams 1993) © 1993 ACM: (a) holes from one source image (shown in blue); (b) holes after combining two widely spaced images; (c) holes after combining two closely spaced images; (d) after interpolation (hole filling).

Figure 13.2 illustrates this process in more detail. First, both source images are warped to the novel view, using both the knowledge of the reference and virtual 3D camera pose along with each image’s depth map (2.68–2.70). In the paper by Chen and Williams (1993), a *forward warping* algorithm (Algorithm 3.1 and Figure 3.46) is used. The depth maps are represented as quadtrees for both space and rendering time efficiency (Samet 1989).

During the forward warping process, multiple pixels (which occlude one another) may land on the same destination pixel. To resolve this conflict, either a *z-buffer* depth value can be associated with each destination pixel or the images can be warped in back-to-front order, which can be computed based on the knowledge of epipolar geometry (Chen and Williams 1993; Laveau and Faugeras 1994; McMillan and Bishop 1995).

Once the two reference images have been warped to the novel view (Figure 13.2a–b), they can be merged to create a coherent composite (Figure 13.2c). Whenever one of the images has a *hole* (illustrated as a cyan pixel), the other image is used as the final value. When both images have pixels to contribute, these can be blended as in usual morphing, i.e., according to the relative distances between the virtual and source cameras. Note that if the two images have very different exposures, which can happen when performing view interpolation on real images, the hole-filled regions and the blended regions will have different exposures, leading to subtle artifacts.

The final step in view interpolation (Figure 13.2d) is to fill any remaining holes or cracks due to the forward warping process or lack of source data (scene visibility). This can be done by copying pixels from the *further* pixels adjacent to the hole. (Otherwise, foreground objects are subject to a “fattening effect”.)

The above process works well for rigid scenes, although its visual quality (lack of aliasing) can be improved using a two-pass, forward–backward algorithm (Section 13.2.1) (Shade, Gortler, He *et al.* 1998) or full 3D rendering (Zitnick, Kang, Uyttendaele *et al.* 2004). In the case where the two reference images are views of a non-rigid scene, e.g., a person smiling in one image and frowning in the other, *view morphing*, which combines ideas from view interpolation with regular morphing, can be used (Seitz and Dyer 1996).

While the original view interpolation paper describes how to generate novel views based on similar pre-computed (linear perspective) images, the *plenoptic modeling* paper of McMillan and Bishop (1995) argues that cylindrical images should be used to store the pre-computed

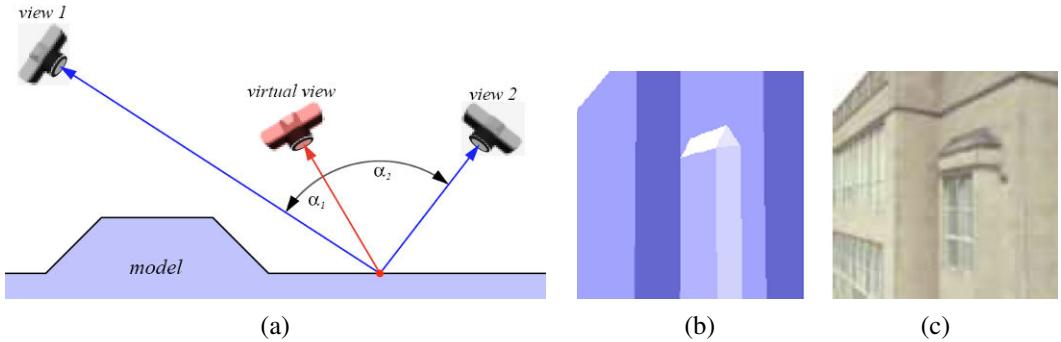


Figure 13.3 View-dependent texture mapping (Debevec, Taylor, and Malik 1996) © 1996 ACM. (a) The weighting given to each input view depends on the relative angles between the novel (virtual) view and the original views; (b) simplified 3D model geometry; (c) with view-dependent texture mapping, the geometry appears to have more detail (recessed windows).

rendering or real-world images. (Chen 1995) also propose using environment maps (cylindrical, cubic, or spherical) as source images for view interpolation.

13.1.1 View-dependent texture maps

View-dependent texture maps (Debevec, Taylor, and Malik 1996) are closely related to view interpolation. Instead of associating a separate depth map with each input image, a single 3D model is created for the scene, but different images are used as texture map sources depending on the virtual camera's current position (Figure 13.3a).³

In more detail, given a new virtual camera position, the similarity of this camera's view of each polygon (or pixel) is compared to that of potential source images. The images are then blended using a weighting that is inversely proportional to the angles α_i between the virtual view and the source views (Figure 13.3a). Even though the geometric model can be fairly coarse (Figure 13.3b), blending between different views gives a strong sense of more detailed geometry because of the parallax (visual motion) between corresponding pixels. While the original paper performs the weighted blend computation separately at each pixel or coarsened polygon face, follow-on work by Debevec, Yu, and Borshukov (1998) presents a more efficient implementation based on precomputing contributions for various portions of viewing space and then using projective texture mapping (OpenGL-ARB 1997).

The idea of view-dependent texture mapping has been used in a large number of subsequent image-based rendering systems, including facial modeling and animation (Pighin, Hecker, Lischinski *et al.* 1998) and 3D scanning and visualization (Pulli, Abi-Rached, Duchamp *et al.* 1998). Closely related to view-dependent texture mapping is the idea of blending between light rays in 4D space, which forms the basis of the Lumigraph and unstructured Lumigraph systems (Section 13.3) (Gortler, Grzeszczuk, Szeliski *et al.* 1996; Buehler, Bosse, McMillan *et al.* 2001).

³ The term *image-based modeling*, which is now commonly used to describe the creation of texture-mapped 3D models from multiple images, appears to have first been used by Debevec, Taylor, and Malik (1996), who also used the term *photogrammetric modeling* to describe the same process.

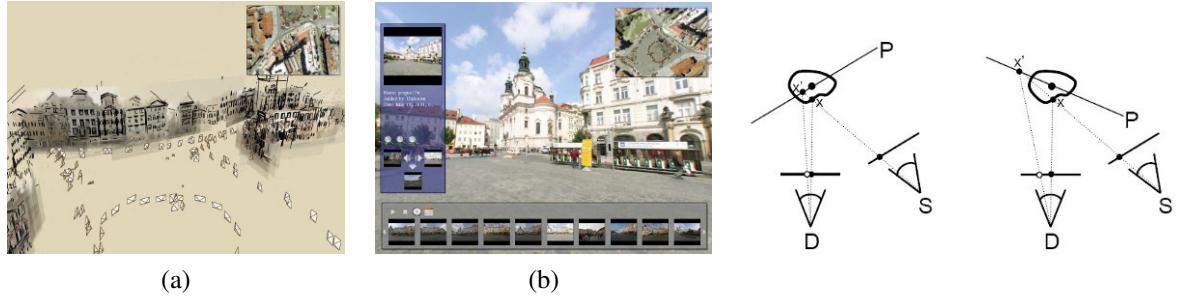


Figure 13.4 Photo Tourism (Snavely, Seitz, and Szeliski 2006): © 2006 ACM: (a) a 3D overview of the scene, with translucent washes and lines painted onto the planar impostors; (b) once the user has selected a region of interest, a set of related thumbnails is displayed along the bottom; (c) planar proxy selection for optimal stabilization (Snavely, Garg, Seitz *et al.* 2008) © 2008 ACM.

In order to provide even more realism in their Façade system, Debevec, Taylor, and Malik (1996) also include a *model-based stereo* component, which optionally computes an offset (parallax) map for each coarse planar facet of their 3D model. They call the resulting analysis and rendering system a *hybrid geometry- and image-based* approach, since it uses traditional 3D geometric modeling to create the global 3D model, but then uses local depth offsets, along with view interpolation, to add visual realism.

13.1.2 Application: Photo Tourism

While view interpolation was originally developed to accelerate the rendering of 3D scenes on low-powered processors and systems without graphics acceleration, it turns out that it can be applied directly to large collections of casually acquired photographs. The *Photo Tourism* system developed by Snavely, Seitz, and Szeliski (2006) uses structure from motion to compute the 3D locations and poses of all the cameras taking the images, along with a sparse 3D point-cloud model of the scene (Section 7.4.4, Figure 7.11).

To perform an image-based exploration of the resulting *sea of images* (Aliaga, Funkhouser, Yanovsky *et al.* 2003), Photo Tourism first associates a 3D proxy with each image. While a triangulated mesh obtained from the point cloud can sometimes form a suitable proxy, e.g., for outdoor terrain models, a simple dominant plane fit to the 3D points visible in each image often performs better, because it does not contain any erroneous segments or connections that pop out as artifacts. As automated 3D modeling techniques continue to improve, however, the pendulum may swing back to more detailed 3D geometry (Goesele, Snavely, Curless *et al.* 2007; Sinha, Steedly, and Szeliski 2009).

The resulting image-based navigation system lets users move from photo to photo, either by selecting cameras from a top-down view of the scene (Figure 13.4a) or by selecting regions of interest in an image, navigating to nearby views, or selecting related thumbnails (Figure 13.4b). To create a background for the 3D scene, e.g., when being viewed from above, non-photorealistic techniques (Section 10.5.2), such as translucent color washes or highlighted 3D line segments, can be used (Figure 13.4a). The system can also be used to annotate regions of images and to automatically propagate such annotations to other photographs.

The 3D planar proxies used in Photo Tourism and the related Photosynth system from Microsoft result in non-photorealistic transitions reminiscent of visual effects such as “page flips”. Selecting a stable 3D axis for all the planes can reduce the amount of swimming and enhance the perception of 3D (Figure 13.4c) (Snavely, Garg, Seitz *et al.* 2008). It is also possible to automatically detect objects in the scene that are seen from multiple views and create “orbits” of viewpoints around such objects. Furthermore, nearby images in both 3D position and viewing direction can be linked to create “virtual paths”, which can then be used to navigate between arbitrary pairs of images, such as those you might take yourself while walking around a popular tourist site (Snavely, Garg, Seitz *et al.* 2008).

The spatial matching of image features and regions performed by Photo Tourism can also be used to infer more information from large image collections. For example, Simon, Snavely, and Seitz (2007) show how the match graph between images of popular tourist sites can be used to find the most *iconic* (commonly photographed) objects in the collection, along with their related tags. In follow-on work, Simon and Seitz (2008) show how such tags can be propagated to sub-regions of each image, using an analysis of which 3D points appear in the central portions of photographs. Extensions of these techniques to *all* of the world’s images, including the use of GPS tags where available, have been investigated as well (Li, Wu, Zach *et al.* 2008; Quack, Leibe, and Van Gool 2008; Crandall, Backstrom, Huttenlocher *et al.* 2009; Li, Crandall, and Huttenlocher 2009; Zheng, Zhao, Song *et al.* 2009).

13.2 Layered depth images

Traditional view interpolation techniques associate a single depth map with each source or reference image. Unfortunately, when such a depth map is warped to a novel view, holes and cracks inevitably appear behind the foreground objects. One way to alleviate this problem is to keep several depth and color values (*depth pixels*) at every pixel in a reference image (or, at least for pixels near foreground–background transitions) (Figure 13.5). The resulting data structure, which is called a *layered depth image* (LDI), can be used to render new views using a back-to-front forward warping (splatting) algorithm (Shade, Gortler, He *et al.* 1998).

13.2.1 Impostors, sprites, and layers

An alternative to keeping lists of color-depth values at each pixel, as is done in the LDI, is to organize objects into different *layers* or *sprites*. The term sprite originates in the computer game industry, where it is used to designate flat animated characters in games such as Pac-Man or Mario Bros. When put into a 3D setting, such objects are often called *impostors*, because they use a piece of flat, alpha-matted geometry to represent simplified versions of 3D objects that are far away from the camera (Shade, Lischinski, Salesin *et al.* 1996; Lengyel and Snyder 1997; Torborg and Kajiya 1996). In computer vision, such representations are usually called *layers* (Wang and Adelson 1994; Baker, Szeliski, and Anandan 1998; Torr, Szeliski, and Anandan 1999; Birchfield, Natarajan, and Tomasi 2007). Section 8.5.2 discusses the topics of transparent layers and reflections, which occur on specular and transparent surfaces such as glass.

While flat layers can often serve as an adequate representation of geometry and appearance for far-away objects, better geometric fidelity can be achieved by also modeling the

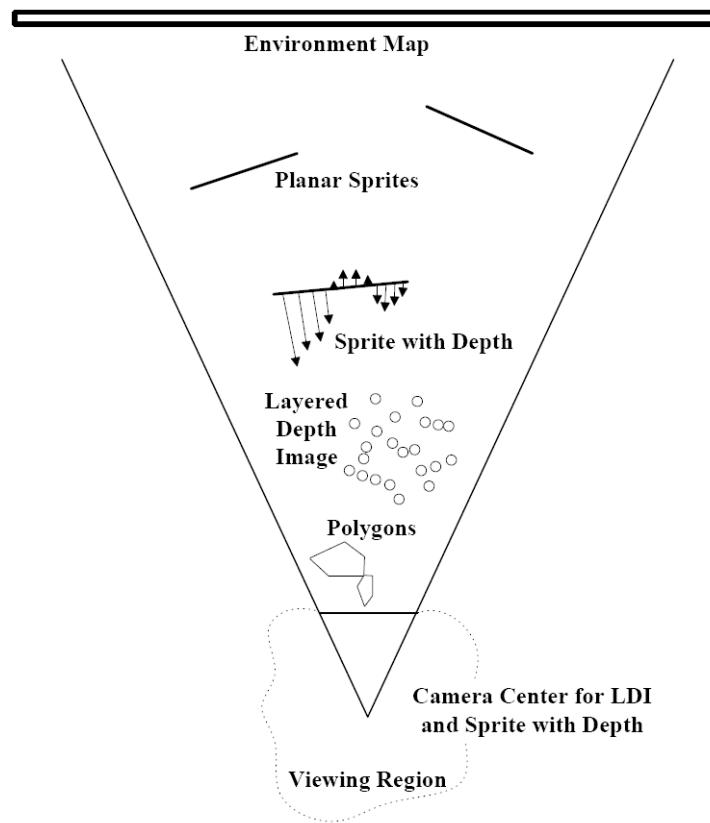


Figure 13.5 A variety of image-based rendering primitives, which can be used depending on the distance between the camera and the object of interest (Shade, Gortler, He *et al.* 1998) © 1998 ACM. Closer objects may require more detailed polygonal representations, while mid-level objects can use a layered depth image (LDI), and far-away objects can use sprites (potentially with depth) and environment maps.



Figure 13.6 Sprites with depth (Shade, Gortler, He *et al.* 1998) © 1998 ACM: (a) alpha-matted color sprite; (b) corresponding relative depth or parallax; (c) rendering without relative depth; (d) rendering with depth (note the curved object boundaries).

per-pixel offsets relative to a base plane, as shown in Figures 13.5 and 13.6a–b. Such representations are called *plane plus parallax* in the computer vision literature (Kumar, Anandan, and Hanna 1994; Sawhney 1994; Szeliski and Coughlan 1997; Baker, Szeliski, and Anandan 1998), as discussed in Section 8.5 (Figure 8.16). In addition to fully automated stereo techniques, it is also possible to paint in depth layers (Kang 1998; Oh, Chen, Dorsey *et al.* 2001; Shum, Sun, Yamazaki *et al.* 2004) or to infer their 3D structure from monocular image cues (Section 14.4.4) (Hoiem, Efros, and Hebert 2005b; Saxena, Sun, and Ng 2009).

How can we render a sprite with depth from a novel viewpoint? One possibility, as with a regular depth map, is to just forward warp each pixel to its new location, which can cause aliasing and cracks. A better way, which we already mentioned in Section 3.6.2, is to first warp the depth (or (u, v) displacement) map to the novel view, fill in the cracks, and then use higher-quality inverse warping to resample the color image (Shade, Gortler, He *et al.* 1998). Figure 13.6d shows the results of applying such a two-pass rendering algorithm. From this still image, you can appreciate that the foreground sprites look more rounded; however, to fully appreciate the improvement in realism, you would have to look at the actual animated sequence.

Sprites with depth can also be rendered using conventional graphics hardware, as described in (Zitnick, Kang, Uyttendaele *et al.* 2004). Rogmans, Lu, Bekaert *et al.* (2009) describe GPU implementations of both real-time stereo matching and real-time forward and inverse rendering algorithms.

13.3 Light fields and Lumigraphs

While image-based rendering approaches can synthesize scene renderings from novel viewpoints, they raise the following more general question:

Is it possible to capture and render the appearance of a scene from all possible viewpoints and, if so, what is the complexity of the resulting structure?

Let us assume that we are looking at a static scene, i.e., one where the objects and illuminants are fixed, and only the observer is moving around. Under these conditions, we can describe each image by the location and orientation of the virtual camera (6 dof) as well as its intrinsics (e.g., its focal length). However, if we capture a two-dimensional *spherical* image around each possible camera location, we can re-render any view from this information.⁴ Thus, taking the cross-product of the three-dimensional space of camera positions with the 2D space of spherical images, we obtain the 5D *plenoptic function* of Adelson and Bergen (1991), which forms the basis of the image-based rendering system of McMillan and Bishop (1995).

Notice, however, that when there is no light dispersion in the scene, i.e., no smoke or fog, all the coincident rays along a portion of free space (between solid or refractive objects) have the same color value. Under these conditions, we can reduce the 5D plenoptic function to the 4D *light field* of all possible rays (Gortler, Grzeszczuk, Szeliski *et al.* 1996; Levoy and Hanrahan 1996; Levoy 2006).⁵

⁴ Since we are counting dimensions, we ignore for now any sampling or resolution issues.

⁵ Levoy and Hanrahan (1996) borrowed the term *light field* from a paper by Gershun (1939). Another name for this representation is the *photic field* (Moon and Spencer 1981).

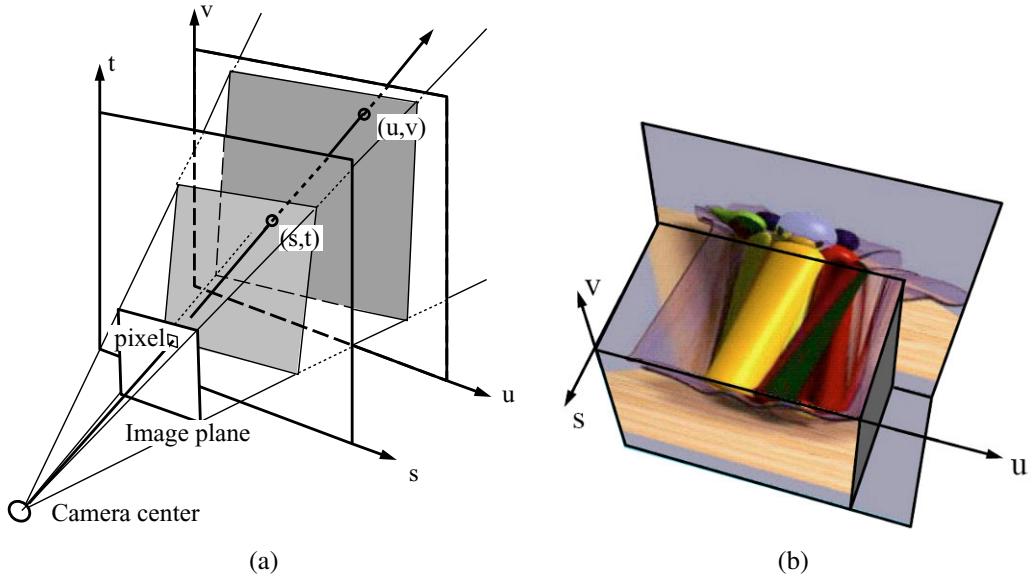


Figure 13.7 The Lumigraph (Gortler, Grzeszczuk, Szeliski *et al.* 1996) © 1996 ACM: (a) a ray is represented by its 4D two-plane parameters (s, t) and (u, v) ; (b) a slice through the 3D light field subset (u, v, s) .

To make the parameterization of this 4D function simpler, let us put two planes in the 3D scene roughly bounding the area of interest, as shown in Figure 13.7a. Any light ray terminating at a camera that lives in front of the st plane (assuming that this space is empty) passes through the two planes at (s, t) and (u, v) and can be described by its 4D coordinate (s, t, u, v) . This diagram (and parameterization) can be interpreted as describing a family of cameras living on the st plane with their image planes being the uv plane. The uv plane can be placed at infinity, which corresponds to all the virtual cameras looking in the same direction.

In practice, if the planes are of finite extent, the finite $light\ slab\ L(s, t, u, v)$ can be used to generate any synthetic view that a camera would see through a (finite) *viewport* in the st plane with a view frustum that wholly intersects the far uv plane. To enable the camera to move all the way around an object, the 3D space surrounding the object can be split into multiple domains, each with its own light slab parameterization. Conversely, if the camera is moving inside a bounded volume of free space looking outward, multiple cube faces surrounding the camera can be used as (s, t) planes.

Thinking about 4D spaces is difficult, so let us drop our visualization by one dimension. If we fix the row value t and constrain our camera to move along the s axis while looking at the uv plane, we can stack all of the stabilized images the camera sees to get the (u, v, s) *epipolar volume*, which we discussed in Section 11.6. A “horizontal” cross-section through this volume is the well-known *epipolar plane image* (Bolles, Baker, and Marimont 1987), which is the us slice shown in Figure 13.7b.

As you can see in this slice, each color pixel moves along a linear track whose slope is related to its depth (parallax) from the uv plane. (Pixels exactly on the uv plane appear “vertical”, i.e., they do not move as the camera moves along s .) Furthermore, pixel tracks

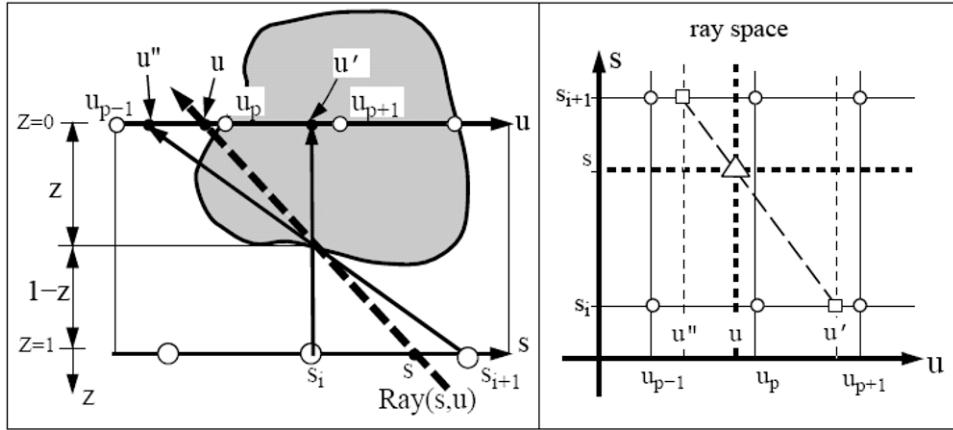


Figure 13.8 Depth compensation in the Lumigraph (Gortler, Grzeszczuk, Szeliski *et al.* 1996) © 1996 ACM. To resample the (s, u) dashed light ray, the u parameter corresponding to each discrete s_i camera location is modified according to the out-of-plane depth z to yield new coordinates u and u' ; in (u, s) ray space, the original sample (Δ) is resampled from the (s_i, u') and (s_{i+1}, u'') samples, which are themselves linear blends of their adjacent (\circ) samples.

occlude one another as their corresponding 3D surface elements occlude. Translucent pixels, however, composite *over* background pixels (Section 3.1.3, (3.8)) rather than occluding them. Thus, we can think of adjacent pixels sharing a similar planar geometry as *EPI strips* or *EPI tubes* (Criminisi, Kang, Swaminathan *et al.* 2005).

The equations mapping from pixels (x, y) in a virtual camera and the corresponding (s, t, u, v) coordinates are relatively straightforward to derive and are sketched out in Exercise 13.7. It is also possible to show that the set of pixels corresponding to a regular orthographic or perspective camera, i.e., one that has a linear projective relationship between 3D points and (x, y) pixels (2.63), lie along a two-dimensional hyperplane in the (s, t, u, v) light field (Exercise 13.7).

While a light field can be used to render a complex 3D scene from novel viewpoints, a much better rendering (with less ghosting) can be obtained if something is known about its 3D geometry. The Lumigraph system of Gortler, Grzeszczuk, Szeliski *et al.* (1996) extends the basic light field rendering approach by taking into account the 3D location of surface points corresponding to each 3D ray.

Consider the ray (s, u) corresponding to the dashed line in Figure 13.8, which intersects the object's surface at a distance z from the uv plane. When we look up the pixel's color in camera s_i (assuming that the light field is discretely sampled on a regular 4D (s, t, u, v) grid), the actual pixel coordinate is u' , instead of the original u value specified by the (s, u) ray. Similarly, for camera s_{i+1} (where $s_i \leq s \leq s_{i+1}$), pixel address u'' is used. Thus, instead of using quadri-linear interpolation of the nearest sampled (s, t, u, v) values around a given ray to determine its color, the (u, v) values are modified for each discrete (s_i, t_i) camera.

Figure 13.8 also shows the same reasoning in *ray space*. Here, the original continuous-valued (s, u) ray is represented by a triangle and the nearby sampled discrete values are shown as circles. Instead of just blending the four nearest samples, as would be indicated

by the vertical and horizontal dashed lines, the modified (s_i, u') and (s_{i+1}, u'') values are sampled instead and their values are then blended.

The resulting rendering system produces images of much better quality than a proxy-free light field and is the method of choice whenever 3D geometry can be inferred. In subsequent work, Isaksen, McMillan, and Gortler (2000) show how a planar proxy for the scene, which is a simpler 3D model, can be used to simplify the resampling equations. They also describe how to create synthetic aperture photos, which mimic what might be seen by a wide-aperture lens, by blending more nearby samples (Levoy and Hanrahan 1996). A similar approach can be used to re-focus images taken with a plenoptic (microlens array) camera (Ng, Levoy, Bréedif *et al.* 2005; Ng 2005) or a light field microscope (Levoy, Ng, Adams *et al.* 2006). It can also be used to see through obstacles, using extremely large synthetic apertures focused on a background that can blur out foreground objects and make them appear translucent (Wilburn, Joshi, Vaish *et al.* 2005; Vaish, Szeliski, Zitnick *et al.* 2006).

Now that we understand how to render new images from a light field, how do we go about capturing such data sets? One answer is to move a calibrated camera with a motion control rig or *gantry*.⁶ Another approach is to take handheld photographs and to determine the pose and intrinsic calibration of each image using either a calibrated stage or structure from motion. In this case, the images need to be *rebinned* into a regular 4D (s, t, u, v) space before they can be used for rendering (Gortler, Grzeszczuk, Szeliski *et al.* 1996). Alternatively, the original images can be used directly using a process called the *unstructured Lumigraph*, which we describe below.

Because of the large number of images involved, light fields and Lumigraphs can be quite voluminous to store and transmit. Fortunately, as you can tell from Figure 13.7b, there is a tremendous amount of redundancy (coherence) in a light field, which can be made even more explicit by first computing a 3D model, as in the Lumigraph. A number of techniques have been developed to compress and progressively transmit such representations (Gortler, Grzeszczuk, Szeliski *et al.* 1996; Levoy and Hanrahan 1996; Rademacher and Bishop 1998; Magnor and Girod 2000; Wood, Azuma, Aldinger *et al.* 2000; Shum, Kang, and Chan 2003; Magnor, Ramanathan, and Girod 2003; Shum, Chan, and Kang 2007).

13.3.1 Unstructured Lumigraph

When the images in a Lumigraph are acquired in an unstructured (irregular) manner, it can be counterproductive to resample the resulting light rays into a regularly binned (s, t, u, v) data structure. This is both because resampling always introduces a certain amount of aliasing and because the resulting gridded light field can be populated very sparsely or irregularly.

The alternative is to render directly from the acquired images, by finding for each light ray in a virtual camera the closest pixels in the original images. The *unstructured Lumigraph* rendering (ULR) system of Buehler, Bosse, McMillan *et al.* (2001) describes how to select such pixels by combining a number of fidelity criteria, including *epipole consistency* (distance of rays to a source camera's center), *angular deviation* (similar incidence direction on the surface), *resolution* (similar sampling density along the surface), *continuity* (to nearby pixels), and *consistency* (along the ray). These criteria can all be combined to determine a weighting

⁶ See <http://lightfield.stanford.edu/acq.html> for a description of some of the gantries and camera arrays built at the Stanford Computer Graphics Laboratory. This Web site also provides a number of light field data sets that are a great source of research and project material.

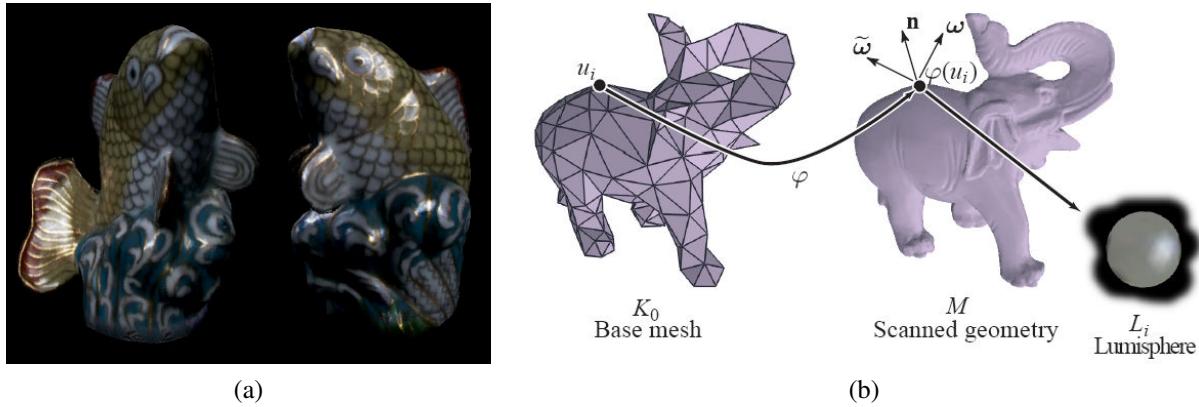


Figure 13.9 Surface light fields (Wood, Azuma, Aldinger *et al.* 2000) © 2000 ACM: (a) example of a highly specular object with strong inter-reflections; (b) the surface light field stores the light emanating from each surface point in all visible directions as a “Lumisphere”.

function between each virtual camera’s pixel and a number of candidate input cameras from which it can draw colors. To make the algorithm more efficient, the computations are performed by discretizing the virtual camera’s image plane using a regular grid overlaid with the polyhedral object mesh model and the input camera centers of projection and interpolating the weighting functions between vertices.

The unstructured Lumigraph generalizes previous work in both image-based rendering and light field rendering. When the input cameras are gridded, the ULR behaves the same way as regular Lumigraph rendering. When fewer cameras are available but the geometry is accurate, the algorithm behaves similarly to view-dependent texture mapping (Section 13.1.1).

13.3.2 Surface light fields

Of course, using a two-plane parameterization for a light field is not the only possible choice. (It is the one usually presented first since the projection equations and visualizations are the easiest to draw and understand.) As we mentioned on the topic of light field compression, if we know the 3D shape of the object or scene whose light field is being modeled, we can effectively compress the field because nearby rays emanating from nearby surface elements have similar color values.

In fact, if the object is totally diffuse, ignoring occlusions, which can be handled using 3D graphics algorithms or z-buffering, all rays passing through a given surface point will have the same color value. Hence, the light field “collapses” to the usual 2D texture-map defined over an object’s surface. Conversely, if the surface is totally specular (e.g., mirrored), each surface point reflects a miniature copy of the environment surrounding that point. In the absence of inter-reflections (e.g., a convex object in a large open space), each surface point simply reflects the far-field *environment map* (Section 2.2.1), which again is two-dimensional. Therefore, it seems that re-parameterizing the 4D light field to lie on the object’s surface can be extremely beneficial.

These observations underlie the *surface light field* representation introduced by Wood, Azuma, Aldinger *et al.* (2000). In their system, an accurate 3D model is built of the object

being represented. Then the *Lumisphere* of all rays emanating from each surface point is estimated or captured (Figure 13.9). Nearby Lumispheres will be highly correlated and hence amenable to both compression and manipulation.

To estimate the diffuse component of each Lumisphere, a median filtering over all visible exiting directions is first performed for each channel. Once this has been subtracted from the Lumisphere, the remaining values, which should consist mostly of the specular components, are *reflected* around the local surface normal (2.89), which turns each Lumisphere into a copy of the local environment around that point. Nearby Lumispheres can then be compressed using predictive coding, vector quantization, or principal component analysis.

The decomposition into a diffuse and specular component can also be used to perform editing or manipulation operations, such as re-painting the surface, changing the specular component of the reflection (e.g., by blurring or sharpening the specular Lumispheres), or even geometrically deforming the object while preserving detailed surface appearance.

13.3.3 Application: Concentric mosaics

A useful and simple version of light field rendering is a panoramic image with parallax, i.e., a video or series of photographs taken from a camera swinging in front of some rotation point. Such panoramas can be captured by placing a camera on a boom on a tripod, or even more simply, by holding a camera at arm's length while rotating your body around a fixed axis.

The resulting set of images can be thought of as a *concentric mosaic* (Shum and He 1999; Shum, Wang, Chai *et al.* 2002) or a *layered depth panorama* (Zheng, Kang, Cohen *et al.* 2007). The term “concentric mosaic” comes from a particular structure that can be used to re-bin all of the sampled rays, essentially associating each column of pixels with the “radius” of the concentric circle to which it is tangent (Shum and He 1999; Peleg, Ben-Ezra, and Pritch 2001).

Rendering from such data structures is fast and straightforward. If we assume that the scene is far enough away, for any virtual camera location, we can associate each column of pixels in the virtual camera with the nearest column of pixels in the input image set. (For a regularly captured set of images, this computation can be performed analytically.) If we have some rough knowledge of the depth of such pixels, columns can be stretched vertically to compensate for the change in depth between the two cameras. If we have an even more detailed depth map (Peleg, Ben-Ezra, and Pritch 2001; Li, Shum, Tang *et al.* 2004; Zheng, Kang, Cohen *et al.* 2007), we can perform pixel-by-pixel depth corrections.

While the virtual camera's motion is constrained to lie in the plane of the original cameras and within the radius of the original capture ring, the resulting experience can exhibit complex rendering phenomena, such as reflections and translucencies, which cannot be captured using a texture-mapped 3D model of the world. Exercise 13.10 has you construct a concentric mosaic rendering system from a series of hand-held photos or video.

13.4 Environment mattes

So far in this chapter, we have dealt with view interpolation and light fields, which are techniques for modeling and rendering complex static scenes seen from different viewpoints.

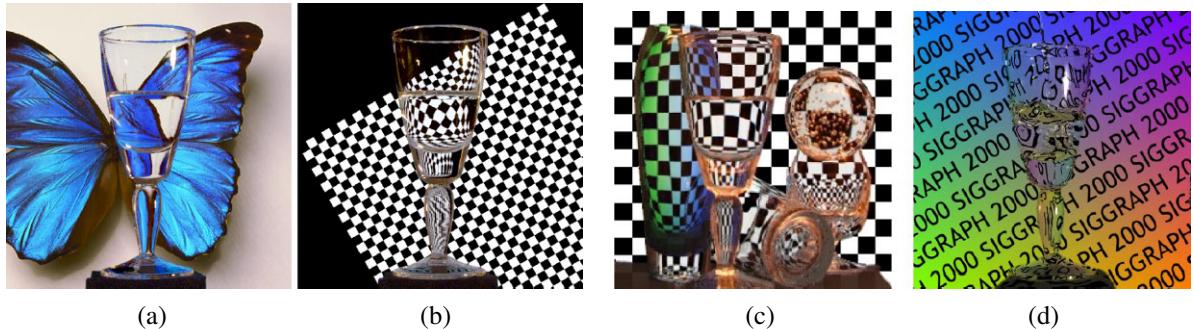


Figure 13.10 Environment mattes: (a–b) a refractive object can be placed in front of a series of backgrounds and their light patterns will be correctly refracted (Zongker, Werner, Curless *et al.* 1999) (c) multiple refractions can be handled using a mixture of Gaussians model and (d) real-time mattes can be pulled using a single graded colored background (Chuang, Zongker, Hindorff *et al.* 2000) © 2000 ACM.

What if instead of moving around a virtual camera, we take a complex, refractive object, such as the water goblet shown in Figure 13.10, and place it in front of a new background? Instead of modeling the 4D space of rays emanating from a scene, we now need to model how each pixel in our view of this object refracts incident light coming from its environment.

What is the intrinsic dimensionality of such a representation and how do we go about capturing it? Let us assume that if we trace a light ray from the camera at pixel (x, y) toward the object, it is reflected or refracted back out toward its environment at an angle (ϕ, θ) . If we assume that other objects and illuminants are sufficiently distant (the same assumption we made for surface light fields in Section 13.3.2), this 4D mapping $(x, y) \rightarrow (\phi, \theta)$ captures all the information between a refractive object and its environment. Zongker, Werner, Curless *et al.* (1999) call such a representation an *environment matte*, since it generalizes the process of object matting (Section 10.4) to not only cut and paste an object from one image into another but also take into account the subtle refractive or reflective interplay between the object and its environment.

Recall from Equations (3.8) and (10.30) that a foreground object can be represented by its premultiplied colors and opacities $(\alpha F, \alpha)$. Such a matte can then be composited onto a new background B using

$$C_i = \alpha_i F_i + (1 - \alpha_i) B_i, \quad (13.1)$$

where i is the pixel under consideration. In environment matting, we augment this equation with a reflective or refractive term to model indirect light paths between the environment and the camera. In the original work of Zongker, Werner, Curless *et al.* (1999), this indirect component I_i is modeled as

$$I_i = R_i \int A_i(\mathbf{x}) B(\mathbf{x}) d\mathbf{x}, \quad (13.2)$$

where A_i is the rectangular *area of support* for that pixel, R_i is the colored reflectance or transmittance (for colored glossy surfaces or glass), and $B(\mathbf{x})$ is the background (environment) image, which is integrated over the area $A_i(\mathbf{x})$. In follow-on work, Chuang, Zongker,

Hindorff *et al.* (2000) use a superposition of oriented Gaussians,

$$I_i = \sum_j R_{ij} \int G_{ij}(\mathbf{x}) B(\mathbf{x}) d\mathbf{x}, \quad (13.3)$$

where each 2D Gaussian

$$G_{ij}(\mathbf{x}) = G_{2D}(\mathbf{x}; \mathbf{c}_{ij}, \boldsymbol{\sigma}_{ij}, \theta_{ij}) \quad (13.4)$$

is modeled by its center \mathbf{c}_{ij} , unrotated widths $\boldsymbol{\sigma}_{ij} = (\sigma_{ij}^x, \sigma_{ij}^y)$, and orientation θ_{ij} .

Given a representation for an environment matte, how can we go about estimating it for a particular object? The trick is to place the object in front of a monitor (or surrounded by a set of monitors), where we can change the illumination patterns $B(\mathbf{x})$ and observe the value of each composite pixel C_i .⁷

As with traditional two-screen matting (Section 10.4.1), we can use a variety of solid colored backgrounds to estimate each pixel’s foreground color $\alpha_i F_i$ and partial coverage (opacity) α_i . To estimate the area of support A_i in (13.2), Zongker, Werner, Curless *et al.* (1999) use a series of periodic horizontal and vertical solid stripes at different frequencies and phases, which is reminiscent of the structured light patterns used in active rangefinding (Section 12.2). For the more sophisticated mixture of Gaussian model (13.3), Chuang, Zongker, Hindorff *et al.* (2000) sweep a series of narrow Gaussian stripes at four different orientations (horizontal, vertical, and two diagonals), which enables them to estimate multiple oriented Gaussian responses at each pixel.

Once an environment matte has been “pulled”, it is then a simple matter to replace the background with a new image $B(\mathbf{x})$ to obtain a novel composite of the object placed in a different environment (Figure 13.10a–c). The use of multiple backgrounds during the matting process, however, precludes the use of this technique with dynamic scenes, e.g., water pouring into a glass (Figure 13.10d). In this case, a single graded color background can be used to estimate a single 2D monochromatic displacement for each pixel (Chuang, Zongker, Hindorff *et al.* 2000).

13.4.1 Higher-dimensional light fields

As you can tell from the preceding discussion, an environment matte in principle maps every pixel (x, y) into a 4D distribution over light rays and is, hence, a six-dimensional representation. (In practice, each 2D pixel’s response is parameterized using a dozen or so parameters, e.g., $\{F, \alpha, B, R, A\}$, instead of a full mapping.) What if we want to model an object’s refractive properties from every potential point of view? In this case, we need a mapping from every incoming 4D light ray to every potential exiting 4D light ray, which is an 8D representation. If we use the same trick as with surface light fields, we can parameterize each surface point by its 4D BRDF to reduce this mapping back down to 6D but this loses the ability to handle multiple refractive paths.

If we want to handle dynamic light fields, we need to add another temporal dimension. (Wenger, Gardner, Tchou *et al.* (2005) gives a nice example of a dynamic appearance and illumination acquisition system.) Similarly, if we want a continuous distribution over wavelengths, this becomes another dimension.

⁷ If we relax the assumption that the environment is distant, the monitor can be placed at several depths to estimate a depth-dependent mapping function (Zongker, Werner, Curless *et al.* 1999).

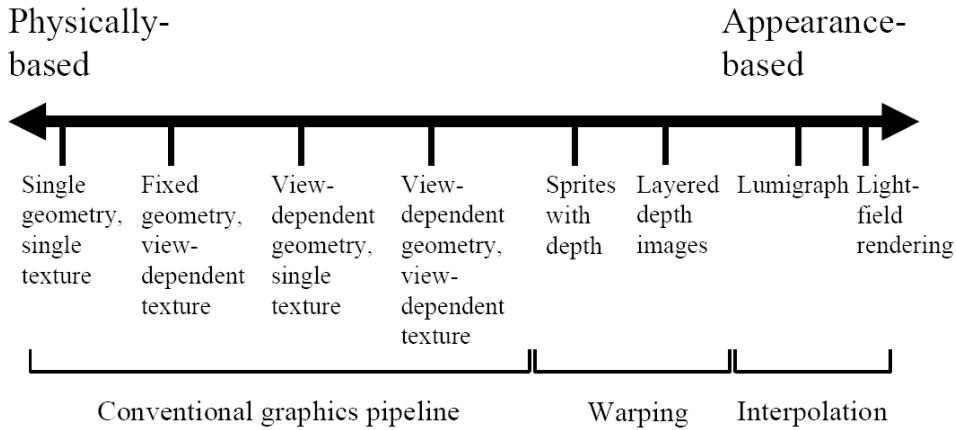


Figure 13.11 The geometry-image continuum in image-based rendering (Kang, Szeliski, and Anandan 2000) © 2000 IEEE. Representations at the left of the spectrum use more detailed geometry and simpler image representations, while representations and algorithms on the right use more images and less geometry.

These examples illustrate how modeling the full complexity of a visual scene through sampling can be extremely expensive. Fortunately, constructing specialized models, which exploit knowledge about the physics of light transport along with the natural coherence of real-world objects, can make these problems more tractable.

13.4.2 The modeling to rendering continuum

The image-based rendering representations and algorithms we have studied in this chapter span a continuum ranging from classic 3D texture-mapped models all the way to pure sampled ray-based representations such as light fields (Figure 13.11). Representations such as view-dependent texture maps and Lumigraphs still use a single global geometric model, but select the colors to map onto these surfaces from nearby images. View-dependent geometry, e.g., multiple depth maps, sidestep the need for coherent 3D geometry, and can sometimes better model local non-rigid effects such as specular motion (Swaminathan, Kang, Szeliski *et al.* 2002; Criminisi, Kang, Swaminathan *et al.* 2005). Sprites with depth and layered depth images use image-based representations of both color and geometry and can be efficiently rendered using warping operations rather than 3D geometric rasterization.

The best choice of representation and rendering algorithm depends on both the quantity and quality of the input imagery as well as the intended application. When nearby views are being rendered, image-based representations capture more of the visual fidelity of the real world because they directly sample its appearance. On the other hand, if only a few input images are available or the image-based models need to be manipulated, e.g., to change their shape or appearance, more abstract 3D representations such as geometric and local reflection models are a better fit. As we continue to capture and manipulate increasingly larger quantities of visual data, research into these aspects of image-based modeling and rendering will continue to evolve.



Figure 13.12 Video Rewrite (Bregler, Covell, and Slaney 1997) © 1997 ACM: the video frames are composed from bits and pieces of old video footage matched to a new audio track.

13.5 Video-based rendering

Since multiple images can be used to render new images or interactive experiences, can something similar be done with video? In fact, a fair amount of work has been done in the area of *video-based rendering* and *video-based animation*, two terms first introduced by Schödl, Szeliski, Salesin *et al.* (2000) to denote the process of generating new video sequences from captured video footage. An early example of such work is Video Rewrite (Bregler, Covell, and Slaney 1997), in which archival video footage is “re-animated” by having actors say new utterances (Figure 13.12). More recently, the term video-based rendering has been used by some researchers to denote the creation of virtual camera moves from a set of synchronized video cameras placed in a studio (Magnor 2005). (The terms *free-viewpoint video* and *3D video* are also sometimes used, see Section 13.5.4.)

In this section, we present a number of video-based rendering systems and applications. We start with *video-based animation* (Section 13.5.1), in which video footage is re-arranged or modified, e.g., in the capture and re-rendering of facial expressions. A special case of this are *video textures* (Section 13.5.2), in which source video is automatically cut into segments and re-looped to create infinitely long video animations. It is also possible to create such animations from still pictures or paintings, by segmenting the image into separately moving regions and animating them using stochastic motion fields (Section 13.5.3).

Next, we turn our attention to *3D video* (Section 13.5.4), in which multiple synchronized video cameras are used to film a scene from different directions. The source video frames can then be re-combined using image-based rendering techniques, such as view interpolation, to create virtual camera paths between the source cameras as part of a real-time viewing experience. Finally, we discuss capturing environments by driving or walking through them with panoramic video cameras in order to create interactive video-based walkthrough experiences (Section 13.5.5).

13.5.1 Video-based animation

As we mentioned above, an early example of video-based animation is Video Rewrite, in which frames from original video footage are rearranged in order to match them to novel spoken utterances, e.g., for movie dubbing (Figure 13.12). This is similar in spirit to the way that *concatenative speech synthesis* systems work (Taylor 2009).

In their system, Bregler, Covell, and Slaney (1997) first use speech recognition to extract phonemes from both the source video material and the novel audio stream. Phonemes are grouped into *triphones* (triplets of phonemes), since these better model the *coarticulation* effect present when people speak. Matching triphones are then found in the source footage and audio track. The mouth images corresponding to the selected video frames are then cut and pasted into the desired video footage being re-animated or dubbed, with appropriate geometric transformations to account for head motion. During the analysis phase, features corresponding to the lips, chin, and head are tracked using computer vision techniques. During synthesis, image morphing techniques are used to blend and stitch adjacent mouth shapes into a more coherent whole. In more recent work, Ezzat, Geiger, and Poggio (2002) describe how to use a *multidimensional morphable model* (Section 12.6.2) combined with regularized trajectory synthesis to improve these results.

A more sophisticated version of this system, called *face transfer*, uses a novel source video, instead of just an audio track, to drive the animation of a previously captured video, i.e., to re-render a video of a talking head with the appropriate visual speech, expression, and head pose elements (Vlasic, Brand, Pfister *et al.* 2005). This work is one of many *performance-driven animation* systems (Section 4.1.5), which are often used to animate 3D facial models (Figures 12.18–12.19). While traditional performance-driven animation systems use marker-based motion capture (Williams 1990; Litwinowicz and Williams 1994; Ma, Jones, Chiang *et al.* 2008), video footage can now often be used directly to control the animation (Buck, Finkelstein, Jacobs *et al.* 2000; Pighin, Szeliski, and Salesin 2002; Zhang, Snavely, Curless *et al.* 2004; Vlasic, Brand, Pfister *et al.* 2005; Roble and Zafar 2009).

In addition to its most common application to facial animation, video-based animation can also be applied to whole body motion (Section 12.6.4), e.g., by matching the flow fields between two different source videos and using one to drive the other (Efros, Berg, Mori *et al.* 2003). Another approach to video-based rendering is to use flow or 3D modeling to *unwrap* surface textures into stabilized images, which can then be manipulated and re-rendered onto the original video (Pighin, Szeliski, and Salesin 2002; Rav-Acha, Kohli, Fitzgibbon *et al.* 2008).

13.5.2 Video textures

Video-based animation is a powerful means of creating photo-realistic videos by re-purposing existing video footage to match some other desired activity or script. What if instead of constructing a special animation or narrative, we simply want the video to continue playing in a plausible manner? For example, many Web sites use images or videos to highlight their destinations, e.g., to portray attractive beaches with surf and palm trees waving in the wind. Instead of using a static image or a video clip that has a discontinuity when it loops, can we transform the video clip into an infinite-length animation that plays forever?

This idea is the basis of *video textures*, in which a short video clip can be arbitrarily extended by re-arranging video frames while preserving visual continuity (Schödl, Szeliski, Salesin *et al.* 2000). The basic problem in creating video textures is how to perform this re-arrangement without introducing visual artifacts. Can you think of how you might do this?

The simplest approach is to match frames by visual similarity (e.g., L_2 distance) and to jump between frames that appear similar. Unfortunately, if the motions in the two frames are different, a dramatic visual artifact will occur (the video will appear to “stutter”). For



Figure 13.13 Video textures (Schödl, Szeliski, Salesin *et al.* 2000) © 2000 ACM: (a) a clock pendulum, with correctly matched direction of motion; (b) a candle flame, showing temporal transition arcs; (c) the flag is generated using morphing at jumps; (d) a bonfire uses longer cross-dissolves; (e) a waterfall cross-dissolves several sequences at once; (f) a smiling animated face; (g) two swinging children are animated separately; (h) the balloons are automatically segmented into separate moving regions; (i) a synthetic fish tank consisting of bubbles, plants, and fish. Videos corresponding to these images can be found at <http://www.cc.gatech.edu/gvu/perception/projects/videotexture/>.

example, if we fail to match the motions of the clock pendulum in Figure 13.13a, it can suddenly change direction in mid-swing.

How can we extend our basic frame matching to also match motion? In principle, we could compute optic flow at each frame and match this. However, flow estimates are often unreliable (especially in textureless regions) and it is not clear how to weight the visual and motion similarities relative to each other. As an alternative, Schödl, Szeliski, Salesin *et al.* (2000) suggest matching *triplets* or larger neighborhoods of adjacent video frames, much in the same way as Video Rewrite matches triphones. Once we have constructed an $n \times n$ similarity matrix between all video frames (where n is the number of frames), a simple finite impulse response (FIR) filtering of each match sequence can be used to emphasize subsequences that match well.

The results of this match computation gives us a *jump table* or, equivalently, a transition probability between any two frames in the original video. This is shown schematically as red arcs in Figure 13.13b, where the red bar indicates which video frame is currently being displayed, and arcs light up as a forward or backward transition is taken. We can view these transition probabilities as encoding the *hidden Markov model* (HMM) that underlies a stochastic video generation process.

Sometimes, it is not possible to find exactly matching subsequences in the original video. In this case, morphing, i.e., warping and blending frames during transitions (Section 3.6.3) can be used to hide the visual differences (Figure 13.13c). If the motion is chaotic enough, as in a bonfire or a waterfall (Figures 13.13d–e), simple blending (extended cross-dissolves) may be sufficient. Improved transitions can also be obtained by performing 3D graph cuts on the spatio-temporal volume around a transition (Kwatra, Schödl, Essa *et al.* 2003).

Video textures need not be restricted to chaotic random phenomena such as fire, wind, and water. Pleasing video textures can be created of people, e.g., a smiling face (as in Figure 13.13f) or someone running on a treadmill (Schödl, Szeliski, Salesin *et al.* 2000). When multiple people or objects are moving independently, as in Figures 13.13g–h, we must first segment the video into independently moving regions and animate each region separately. It is also possible to create large panoramic video textures from a slowly panning camera (Agarwala, Zheng, Pal *et al.* 2005).

Instead of just playing back the original frames in a stochastic (random) manner, video textures can also be used to create scripted or interactive animations. If we extract individual elements, such as fish in a fishtank (Figure 13.13i) into separate *video sprites*, we can animate them along pre-specified paths (by matching the path direction with the original sprite motion) to make our video elements move in a desired fashion (Schödl and Essa 2002). In fact, work on video textures inspired research on systems that re-synthesize new motion sequences from motion capture data, which some people refer to as “mocap soup” (Arikant and Forsyth 2002; Kovar, Gleicher, and Pighin 2002; Lee, Chai, Reitsma *et al.* 2002; Li, Wang, and Shum 2002; Pullen and Bregler 2002).

While video textures primarily analyze the video as a sequence of frames (or regions) that can be re-arranged in time, *temporal textures* (Szummer and Picard 1996; Bar-Joseph, El-Yaniv, Lischinski *et al.* 2001) and *dynamic textures* (Doretto, Chiuso, Wu *et al.* 2003; Yuan, Wen, Liu *et al.* 2004; Doretto and Soatto 2006) treat the video as a 3D spatio-temporal volume with textural properties, which can be described using auto-regressive temporal models.

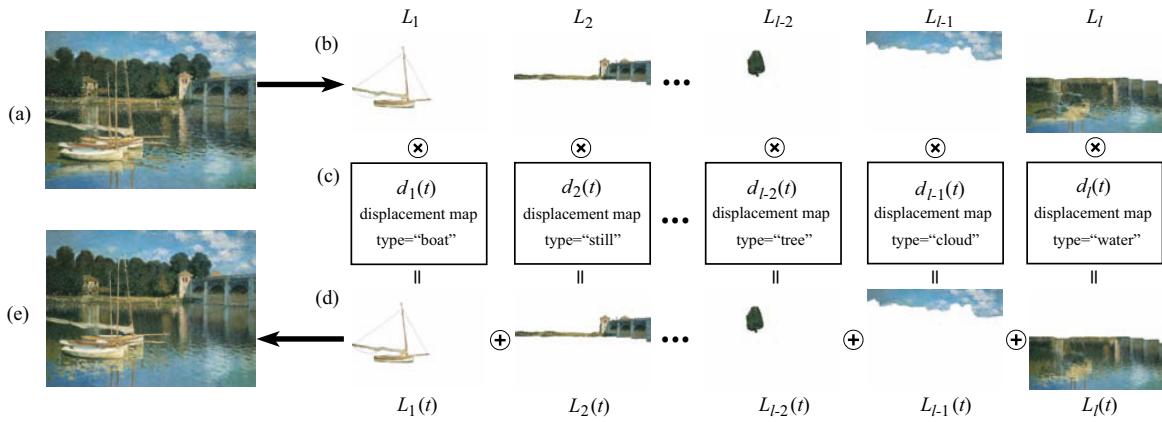


Figure 13.14 Animating still pictures (Chuang, Goldman, Zheng *et al.* 2005) © 2005 ACM. (a) The input still image is manually segmented into (b) several layers. (c) Each layer is then animated with a different stochastic motion texture (d) The animated layers are then composited to produce (e) the final animation

13.5.3 Application: Animating pictures

While video textures can turn a short video clip into an infinitely long video, can the same thing be done with a single still image? The answer is yes, if you are willing to first segment the image into different layers and then animate each layer separately.

Chuang, Goldman, Zheng *et al.* (2005) describe how an image can be decomposed into separate layers using interactive matting techniques. Each layer is then animated using a class-specific synthetic motion. As shown in Figure 13.14, boats rock back and forth, trees sway in the wind, clouds move horizontally, and water ripples, using a shaped noise displacement map. All of these effects can be tied to some global control parameters, such as the velocity and direction of a virtual wind. After being individually animated, the layers can be composited to create a final dynamic rendering.

13.5.4 3D Video

In recent years, the popularity of 3D movies has grown dramatically, with recent releases ranging from *Hannah Montana*, through U2's 3D concert movie, to James Cameron's *Avatar*. Currently, such releases are filmed using stereoscopic camera rigs and displayed in theaters (or at home) to viewers wearing polarized glasses.⁸ In the future, however, home audiences may wish to view such movies with multi-zone auto-stereoscopic displays, where each person gets his or her own customized stereo stream and can move around a scene to see it from different perspectives.⁹

The stereo matching techniques developed in the computer vision community along with image-based rendering (view interpolation) techniques from graphics are both essential components in such scenarios, which are sometimes called *free-viewpoint video* (Carranza, Theobalt, Magnor *et al.* 2003) or *virtual viewpoint video* (Zitnick, Kang, Uyttendaele *et al.* 2004). In

⁸ <http://www.3d-summit.com/>.

⁹ <http://www.siggraph.org/s2008/attendees/caf/3d/>.

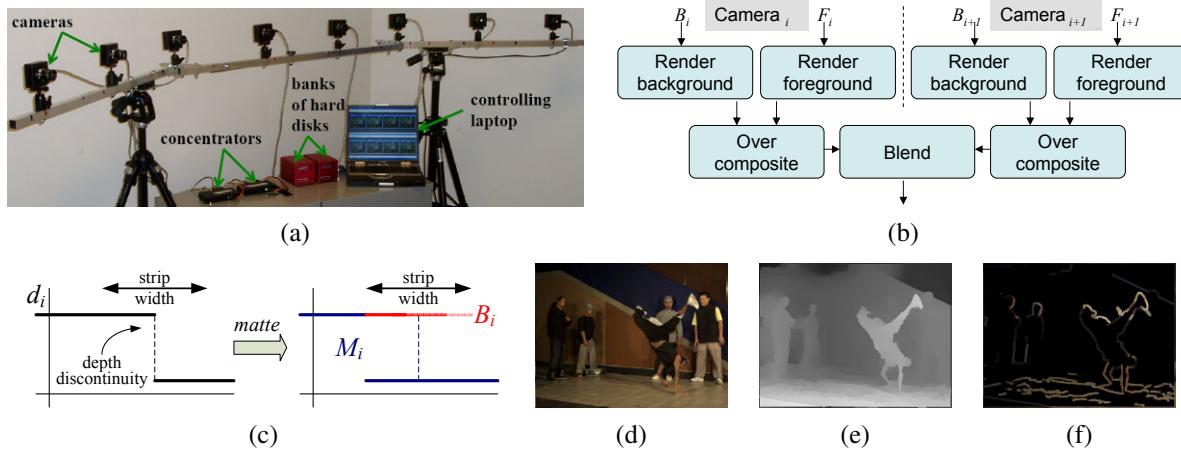


Figure 13.15 Video view interpolation (Zitnick, Kang, Uyttendaele *et al.* 2004) © 2004 ACM: (a) the capture hardware consists of eight synchronized cameras; (b) the background and foreground images from each camera are rendered and composited before blending; (c) the two-layer representation, before and after boundary matting; (d) background color estimates; (e) background depth estimates; (f) foreground color estimates.

addition to solving a series of per-frame reconstruction and view interpolation problems, the depth maps or proxies produced by the analysis phase must be temporally consistent in order to avoid flickering artifacts.

Shum, Chan, and Kang (2007) and Magnor (2005) present nice overviews of various video view interpolation techniques and systems. These include the Virtualized Reality system of Kanade, Rander, and Narayanan (1997) and Vedula, Baker, and Kanade (2005), Immersive Video (Moezzi, Katkere, Kuramura *et al.* 1996), Image-Based Visual Hulls (Matusik, Buehler, Raskar *et al.* 2000; Matusik, Buehler, and McMillan 2001), and Free-Viewpoint Video (Carranza, Theobalt, Magnor *et al.* 2003), which all use global 3D geometric models (surface-based (Section 12.3) or volumetric (Section 12.5)) as their proxies for rendering. The work of Vedula, Baker, and Kanade (2005) also computes *scene flow*, i.e., the 3D motion between corresponding surface elements, which can then be used to perform spatio-temporal interpolation of the multi-view video stream.

The Virtual Viewpoint Video system of Zitnick, Kang, Uyttendaele *et al.* (2004), on the other hand, associates a two-layer depth map with each input image, which allows them to accurately model occlusion effects such as the mixed pixels that occur at object boundaries. Their system, which consists of eight synchronized video cameras connected to a disk array (Figure 13.15a), first uses segmentation-based stereo to extract a depth map for each input image (Figure 13.15e). Near object boundaries (depth discontinuities), the background layer is extended along a strip behind the foreground object (Figure 13.15c) and its color is estimated from the neighboring images where it is not occluded (Figure 13.15d). Automated matting techniques (Section 10.4) are then used to estimate the fractional opacity and color of boundary pixels in the foreground layer (Figure 13.15f).

At render time, given a new virtual camera that lies between two of the original cameras, the layers in the neighboring cameras are rendered as texture-mapped triangles and the foreground layer (which may have fractional opacities) is then composited over the background

layer (Figure 13.15b). The resulting two images are merged and blended by comparing their respective z-buffer values. (Whenever the two z-values are sufficiently close, a linear blend of the two colors is computed.) The interactive rendering system runs in real time using regular graphics hardware. It can therefore be used to change the observer’s viewpoint while playing the video or to freeze the scene and explore it in 3D. More recently, Rogmans, Lu, Bekert *et al.* (2009) have developed GPU implementations of both real-time stereo matching and real-time rendering algorithms, which enable them to explore algorithmic alternatives in a real-time setting.

At present, the depth maps computed from the eight stereo cameras using off-line stereo matching have produced the highest quality depth maps associated with live video.¹⁰ They are therefore often used in studies of 3D video compression, which is an active area of research (Smolic and Kauff 2005; Gotchev and Rosenhahn 2009). Active video-rate depth sensing cameras, such as the 3DV Zcam (Iddan and Yahav 2001), which we discussed in Section 12.2.1, are another potential source of such data.

When large numbers of closely spaced cameras are available, as in the Stanford Light Field Camera (Wilburn, Joshi, Vaish *et al.* 2005), it may not always be necessary to compute explicit depth maps to create video-based rendering effects, although the results are usually of higher quality if you do (Vaish, Szeliski, Zitnick *et al.* 2006).

13.5.5 Application: Video-based walkthroughs

Video camera arrays enable the simultaneous capture of 3D dynamic scenes from multiple viewpoints, which can then enable the viewer to explore the scene from viewpoints near the original capture locations. What if instead we wish to capture an extended area, such as a home, a movie set, or even an entire city?

In this case, it makes more sense to move the camera through the environment and play back the video as an interactive video-based walkthrough. In order to allow the viewer to look around in all directions, it is preferable to use a panoramic video camera (Uyttendaele, Criminisi, Kang *et al.* 2004).¹¹

One way to structure the acquisition process is to capture these images in a 2D horizontal plane, e.g., over a grid superimposed inside a room. The resulting *sea of images* (Aliaga, Funkhouser, Yanovsky *et al.* 2003) can be used to enable continuous motion between the captured locations.¹² However, extending this idea to larger settings, e.g., beyond a single room, can become tedious and data-intensive.

Instead, a natural way to explore a space is often to just walk through it along some pre-specified paths, just as museums or home tours guide users along a particular path, say down the middle of each room.¹³ Similarly, city-level exploration can be achieved by driving down the middle of each street and allowing the user to branch at each intersection. This idea dates back to the Aspen MovieMap project (Lippman 1980), which recorded analog video taken from moving cars onto videodiscs for later interactive playback.

¹⁰ <http://research.microsoft.com/en-us/um/redmond/groups/ivm/vvv/>.

¹¹ See <http://www.cis.upenn.edu/~kostas/omni.html> for descriptions of panoramic (omnidirectional) vision systems and associated workshops.

¹² (The Photo Tourism system of Snavely, Seitz, and Szeliski (2006) applies this idea to less structured collections.

¹³ In computer games, restricting a player to forward and backward motion along predetermined paths is called *rail-based gaming*.

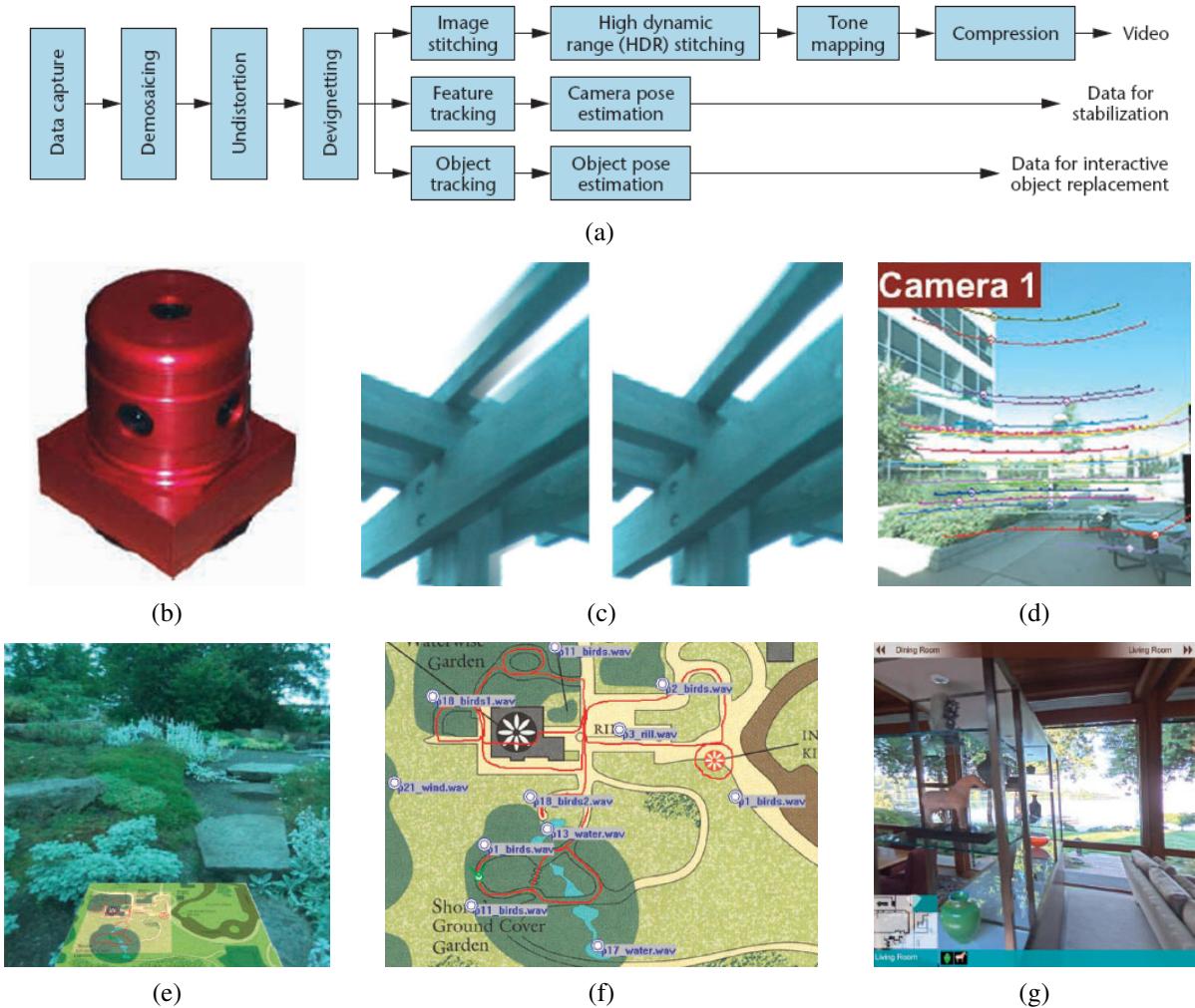


Figure 13.16 Video-based walkthroughs (Uyttendaele, Criminisi, Kang *et al.* 2004) © 2004 IEEE: (a) system diagram of video pre-processing; (b) the Point Grey Ladybug camera; (c) ghost removal using multi-perspective plane sweep; (d) point tracking, used both for calibration and stabilization; (e) interactive garden walkthrough with map below; (f) overhead map authoring and sound placement; (g) interactive home walkthrough with navigation bar (top) and icons of interest (bottom).

Recent improvements in video technology now enable the capture of panoramic (spherical) video using a small co-located array of cameras, such as the Point Grey Ladybug camera¹⁴ (Figure 13.16b) developed by Uyttendaele, Criminisi, Kang *et al.* (2004) for their interactive video-based walkthrough project. In their system, the synchronized video streams from the six cameras (Figure 13.16a) are stitched together into 360° panoramas using a variety of techniques developed specifically for this project.

Because the cameras do not share the same center of projection, parallax between the cameras can lead to ghosting in the overlapping fields of view (Figure 13.16c). To remove this, a multi-perspective plane sweep stereo algorithm is used to estimate per-pixel depths at each column in the overlap area. To calibrate the cameras relative to each other, the camera is spun in place and a constrained structure from motion algorithm (Figure 7.8) is used to estimate the relative camera poses and intrinsics. Feature tracking is then run on the walkthrough video in order to stabilize the video sequence—Liu, Gleicher, Jin *et al.* (2009) have carried out more recent work along these lines.

Indoor environments with windows, as well as sunny outdoor environments with strong shadows, often have a dynamic range that exceeds the capabilities of video sensors. For this reason, the Ladybug camera has a programmable exposure capability that enables the bracketing of exposures at subsequent video frames. In order to merge the resulting video frames into high dynamic range (HDR) video, pixels from adjacent frames need to be motion-compensated before being merged (Kang, Uyttendaele, Winder *et al.* 2003).

The interactive walk-through experience becomes much richer and more navigable if an overview map is available as part of the experience. In Figure 13.16f, the map has annotations, which can show up during the tour, and localized sound sources, which play (with different volumes) when the viewer is nearby. The process of aligning the video sequence with the map can be automated using a process called *map correlation* (Levin and Szeliski 2004).

All of these elements combine to provide the user with a rich, interactive, and immersive experience. Figure 13.16e shows a walk through the Bellevue Botanical Gardens, with an overview map in perspective below the live video window. Arrows on the ground are used to indicate potential directions of travel. The viewer simply orients his view towards one of the arrows (the experience can be driven using a game controller) and “walks” forward along the desired path.

Figure 13.16g shows an indoor home tour experience. In addition to a schematic map in the lower left corner and adjacent room names along the top navigation bar, icons appear along the bottom whenever items of interest, such as a homeowner’s art pieces, are visible in the main window. These icons can then be clicked to provide more information and 3D views.

The development of interactive video tours spurred a renewed interest in 360° video-based virtual travel and mapping experiences, as evidenced by commercial sites such as Google’s Street View and Bing Maps. The same videos can also be used to generate turn-by-turn driving directions, taking advantage of both expanded fields of view and image-based rendering to enhance the experience (Chen, Neubert, Ofek *et al.* 2009).

As we continue to capture more and more of our real world with large amounts of high-quality imagery and video, the interactive modeling, exploration, and rendering techniques described in this chapter will play an even bigger role in bringing virtual experiences based

¹⁴ <http://www.ptgrey.com/>.

on remote areas of the world closer to everyone.

13.6 Additional reading

Two good recent surveys of image-based rendering are by Kang, Li, Tong *et al.* (2006) and Shum, Chan, and Kang (2007), with earlier surveys available from Kang (1999), McMillan and Gortler (1999), and Debevec (1999). The term *image-based rendering* was introduced by McMillan and Bishop (1995), although the seminal paper in the field is the view interpolation paper by Chen and Williams (1993). Debevec, Taylor, and Malik (1996) describe their Façade system, which not only created a variety of image-based modeling tools but also introduced the widely used technique of *view-dependent texture mapping*.

Early work on planar impostors and layers was carried out by Shade, Lischinski, Salesin *et al.* (1996), Lengyel and Snyder (1997), and Torborg and Kajiya (1996), while newer work based on *sprites with depth* is described by Shade, Gortler, He *et al.* (1998).

The two foundational papers in image-based rendering are *Light field rendering* by Levoy and Hanrahan (1996) and *The Lumigraph* by Gortler, Grzeszczuk, Szeliski *et al.* (1996). Buehler, Bosse, McMillan *et al.* (2001) generalize the Lumigraph approach to irregularly spaced collections of images, while Levoy (2006) provides a survey and more gentle introduction to the topic of light field and image-based rendering.

Surface light fields (Wood, Azuma, Aldinger *et al.* 2000) provide an alternative parameterization for light fields with accurately known surface geometry and support both better compression and the possibility of editing surface properties. Concentric mosaics (Shum and He 1999; Shum, Wang, Chai *et al.* 2002) and panoramas with depth (Peleg, Ben-Ezra, and Pritch 2001; Li, Shum, Tang *et al.* 2004; Zheng, Kang, Cohen *et al.* 2007), provide useful parameterizations for light fields captured with panning cameras. Multi-perspective images (Rademacher and Bishop 1998) and manifold projections (Peleg and Herman 1997), although not true light fields, are also closely related to these ideas.

Among the possible extensions of light fields to higher-dimensional structures, environment mattes (Zongker, Werner, Curless *et al.* 1999; Chuang, Zongker, Hindorff *et al.* 2000) are the most useful, especially for placing captured objects into new scenes.

Video-based rendering, i.e., the re-use of video to create new animations or virtual experiences, started with the seminal work of Szummer and Picard (1996), Bregler, Covell, and Slaney (1997), and Schödl, Szeliski, Salesin *et al.* (2000). Important follow-on work to these basic re-targeting approaches was carried out by Schödl and Essa (2002), Kwatra, Schödl, Essa *et al.* (2003), Doretto, Chiuso, Wu *et al.* (2003), Wang and Zhu (2003), Zhong and Sclaroff (2003), Yuan, Wen, Liu *et al.* (2004), Doretto and Soatto (2006), Zhao and Pietikäinen (2007), and Chan and Vasconcelos (2009).

Systems that allow users to change their 3D viewpoint based on multiple synchronized video streams include those by Moezzi, Katkere, Kuramura *et al.* (1996), Kanade, Rander, and Narayanan (1997), Matusik, Buehler, Raskar *et al.* (2000), Matusik, Buehler, and McMillan (2001), Carranza, Theobalt, Magnor *et al.* (2003), Zitnick, Kang, Uyttendaele *et al.* (2004), Magnor (2005), and Vedula, Baker, and Kanade (2005). 3D (multiview) video coding and compression is also an active area of research (Smolic and Kauff 2005; Gotchev and Rosenhahn 2009), with 3D Blu-Ray discs, encoded using the multiview video coding (MVC) extension to H.264/MPEG-4 AVC, expected by the end of 2010.

13.7 Exercises

Ex 13.1: Depth image rendering Develop a “view extrapolation” algorithm to re-render a previously computed stereo depth map coupled with its corresponding reference color image.

1. Use a 3D graphics mesh rendering system such as OpenGL or Direct3D, with two triangles per pixel quad and perspective (projective) texture mapping (Debevec, Yu, and Borshukov 1998).
2. Alternatively, use the one- or two-pass forward warper you constructed in Exercise 3.24, extended using (2.68–2.70) to convert from disparities or depths into displacements.
3. (Optional) Kinks in straight lines introduced during view interpolation or extrapolation are visually noticeable, which is one reason why image morphing systems let you specify line correspondences (Beier and Neely 1992). Modify your depth estimation algorithm to match and estimate the geometry of straight lines and incorporate it into your image-based rendering algorithm.

Ex 13.2: View interpolation Extend the system you created in the previous exercise to render two reference views and then blend the images using a combination of z-buffering, hole filling, and blending (morphing) to create the final image (Section 13.1).

1. (Optional) If the two source images have very different exposures, the hole-filled regions and the blended regions will have different exposures. Can you extend your algorithm to mitigate this?
2. (Optional) Extend your algorithm to perform three-way (trilinear) interpolation between neighboring views. You can triangulate the reference camera poses and use barycentric coordinates for the virtual camera in order to determine the blending weights.

Ex 13.3: View morphing Modify your view interpolation algorithm to perform morphs between views of a non-rigid object, such as a person changing expressions.

1. Instead of using a pure stereo algorithm, use a general flow algorithm to compute displacements, but separate them into a rigid displacement due to camera motion and a non-rigid deformation.
2. At render time, use the rigid geometry to determine the new pixel location but then add a fraction of the non-rigid displacement as well.
3. Alternatively, compute a stereo depth map but let the user specify additional correspondences or use a feature-based matching algorithm to provide them automatically.
4. (Optional) Take a single image, such as the Mona Lisa or a friend’s picture, and create an animated 3D view morph (Seitz and Dyer 1996).
 - (a) Find the vertical axis of symmetry in the image and reflect your reference image to provide a virtual pair (assuming the person’s hairstyle is somewhat symmetric).
 - (b) Use structure from motion to determine the relative camera pose of the pair.

- (c) Use dense stereo matching to estimate the 3D shape.
- (d) Use view morphing to create a 3D animation.

Ex 13.4: View dependent texture mapping Use a 3D model you created along with the original images to implement a view-dependent texture mapping system.

1. Use one of the 3D reconstruction techniques you developed in Exercises 7.3, 11.9, 11.10, or 12.8 to build a triangulated 3D image-based model from multiple photographs.
2. Extract textures for each model face from your photographs, either by performing the appropriate resampling or by figuring out how to use the texture mapping software to directly access the source images.
3. At run time, for each new camera view, select the best source image for each visible model face.
4. Extend this to blend between the top two or three textures. This is trickier, since it involves the use of texture blending or pixel shading (Debevec, Taylor, and Malik 1996; Debevec, Yu, and Borshukov 1998; Pighin, Hecker, Lischinski *et al.* 1998).

Ex 13.5: Layered depth images Extend your view interpolation algorithm (Exercise 13.2) to store more than one depth or color value per pixel (Shade, Gortler, He *et al.* 1998), i.e., a layered depth image (LDI). Modify your rendering algorithm accordingly. For your data, you can use synthetic ray tracing, a layered reconstructed model, or a volumetric reconstruction.

Ex 13.6: Rendering from sprites or layers Extend your view interpolation algorithm to handle multiple planes or sprites (Section 13.2.1) (Shade, Gortler, He *et al.* 1998).

1. Extract your layers using the technique you developed in Exercise 8.9.
2. Alternatively, use an interactive painting and 3D placement system to extract your layers (Kang 1998; Oh, Chen, Dorsey *et al.* 2001; Shum, Sun, Yamazaki *et al.* 2004).
3. Determine a back-to-front order based on expected visibility or add a z-buffer to your rendering algorithm to handle occlusions.
4. Render and composite all of the resulting layers, with optional alpha matting to handle the edges of layers and sprites.

Ex 13.7: Light field transformations Derive the equations relating regular images to 4D light field coordinates.

1. Determine the mapping between the far plane (u, v) coordinates and a virtual camera's (x, y) coordinates.
 - (a) Start by parameterizing a 3D point on the uv plane in terms of its (u, v) coordinates.
 - (b) Project the resulting 3D point to the camera pixels $(x, y, 1)$ using the usual 3×4 camera matrix \mathbf{P} (2.63).
 - (c) Derive the 2D homography relating (u, v) and (x, y) coordinates.

2. Write down a similar transformation for (s, t) to (x, y) coordinates.
3. Prove that if the virtual camera is actually on the (s, t) plane, the (s, t) value depends only on the camera's optical center and is independent of (x, y) .
4. Prove that an image taken by a regular orthographic or perspective camera, i.e., one that has a linear projective relationship between 3D points and (x, y) pixels (2.63), samples the (s, t, u, v) light field along a two-dimensional hyperplane.

Ex 13.8: Light field and Lumigraph rendering Implement a light field or Lumigraph rendering system:

1. Download one of the light field data sets from <http://lightfield.stanford.edu/>.
2. Write an algorithm to synthesize a new view from this light field, using quadri-linear interpolation of (s, t, u, v) ray samples.
3. Try varying the focal plane corresponding to your desired view (Isaksen, McMillan, and Gortler 2000) and see if the resulting image looks sharper.
4. Determine a 3D proxy for the objects in your scene. You can do this by running multi-view stereo over one of your light fields to obtain a depth map per image.
5. Implement the Lumigraph rendering algorithm, which modifies the sampling of rays according to the 3D location of each surface element.
6. Collect a set of images yourself and determine their pose using structure from motion.
7. Implement the unstructured Lumigraph rendering algorithm from Buehler, Bosse, McMillan *et al.* (2001).

Ex 13.9: Surface light fields Construct a surface light field (Wood, Azuma, Aldinger *et al.* 2000) and see how well you can compress it.

1. Acquire an interesting light field of a specular scene or object, or download one from <http://lightfield.stanford.edu/>.
2. Build a 3D model of the object using a multi-view stereo algorithm that is robust to outliers due to specularities.
3. Estimate the Lumisphere for each surface point on the object.
4. Estimate its diffuse components. Is the median the best way to do this? Why not use the minimum color value? What happens if there is Lambertian shading on the diffuse component?
5. Model and compress the remaining portion of the Lumisphere using one of the techniques suggested by Wood, Azuma, Aldinger *et al.* (2000) or invent one of your own.
6. Study how well your compression algorithm works and what artifacts it produces.
7. (Optional) Develop a system to edit and manipulate your surface light field.

Ex 13.10: Handheld concentric mosaics Develop a system to navigate a handheld concentric mosaic.

1. Stand in the middle of a room with a camcorder held at arm's length in front of you and spin in a circle.
2. Use a structure from motion system to determine the camera pose and sparse 3D structure for each input frame.
3. (Optional) Re-bin your image pixels into a more regular concentric mosaic structure.
4. At view time, determine from the new camera's view (which should be near the plane of your original capture) which source pixels to display. You can simplify your computations to determine a source column (and scaling) for each output column.
5. (Optional) Use your sparse 3D structure, interpolated to a dense depth map, to improve your rendering (Zheng, Kang, Cohen *et al.* 2007).

Ex 13.11: Video textures Capture some videos of natural phenomena, such as a water fountain, fire, or smiling face, and loop the video seamlessly into an infinite length video (Schödl, Szeliski, Salesin *et al.* 2000).

1. Compare all the frames in the original clip using an L_2 (sum of square difference) metric. (This assumes the videos were shot on a tripod or have already been stabilized.)
2. Filter the comparison table temporally to accentuate temporal sub-sequences that match well together.
3. Convert your similarity table into a jump probability table through some exponential distribution. Be sure to modify transitions near the end so you do not get “stuck” in the last frame.
4. Starting with the first frame, use your transition table to decide whether to jump forward, backward, or continue to the next frame.
5. (Optional) Add any of the other extensions to the original video textures idea, such as multiple moving regions, interactive control, or graph cut spatio-temporal texture seaming.

Chapter 14

Recognition

| | |
|--|-----|
| 14.1 Object detection | 578 |
| 14.1.1 Face detection | 578 |
| 14.1.2 Pedestrian detection | 585 |
| 14.2 Face recognition | 588 |
| 14.2.1 Eigenfaces | 589 |
| 14.2.2 Active appearance and 3D shape models | 596 |
| 14.2.3 Application: Personal photo collections | 601 |
| 14.3 Instance recognition | 602 |
| 14.3.1 Geometric alignment | 603 |
| 14.3.2 Large databases | 604 |
| 14.3.3 Application: Location recognition | 609 |
| 14.4 Category recognition | 611 |
| 14.4.1 Bag of words | 612 |
| 14.4.2 Part-based models | 615 |
| 14.4.3 Recognition with segmentation | 620 |
| 14.4.4 Application: Intelligent photo editing | 621 |
| 14.5 Context and scene understanding | 625 |
| 14.5.1 Learning and large image collections | 627 |
| 14.5.2 Application: Image search | 630 |
| 14.6 Recognition databases and test sets | 631 |
| 14.7 Additional reading | 631 |
| 14.8 Exercises | 637 |

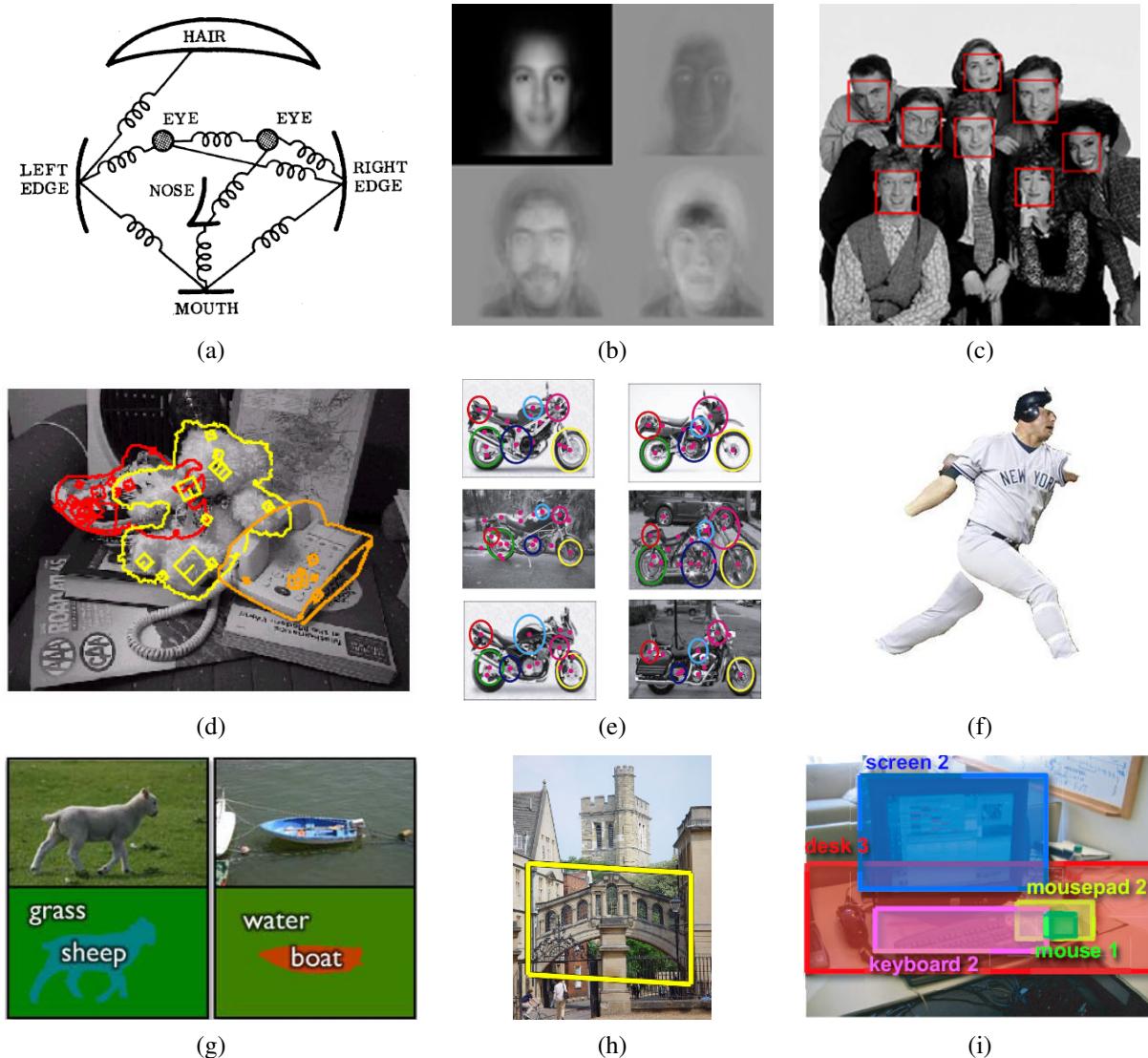


Figure 14.1 Recognition: face recognition with (a) pictorial structures (Fischler and Elschlager 1973) © 1973 IEEE and (b) eigenfaces (Turk and Pentland 1991b); (c) real-time face detection (Viola and Jones 2004) © 2004 Springer; (d) instance (known object) recognition (Lowe 1999) © 1999 IEEE; (e) feature-based recognition (Fergus, Perona, and Zisserman 2007); (f) region-based recognition (Mori, Ren, Efros *et al.* 2004) © 2004 IEEE; (g) simultaneous recognition and segmentation (Shotton, Winn, Rother *et al.* 2009) © 2009 Springer; (h) location recognition (Philbin, Chum, Isard *et al.* 2007) © 2007 IEEE; (i) using context (Russell, Torralba, Liu *et al.* 2007).

Of all the visual tasks we might ask a computer to perform, analyzing a scene and recognizing all of the constituent objects remains the most challenging. While computers excel at accurately reconstructing the 3D shape of a scene from images taken from different views, they cannot name all the objects and animals present in a picture, even at the level of a two-year-old child. There is not even any consensus among researchers on when this level of performance might be achieved.

Why is recognition so hard? The real world is made of a jumble of objects, which all occlude one another and appear in different poses. Furthermore, the variability intrinsic within a class (e.g., dogs), due to complex non-rigid articulation and extreme variations in shape and appearance (e.g., between different breeds), makes it unlikely that we can simply perform exhaustive matching against a database of exemplars.¹

The recognition problem can be broken down along several axes. For example, if we know what we are looking for, the problem is one of *object detection* (Section 14.1), which involves quickly scanning an image to determine where a match may occur (Figure 14.1c). If we have a specific rigid object we are trying to recognize (*instance recognition*, Section 14.3), we can search for characteristic feature points (Section 4.1) and verify that they align in a geometrically plausible way (Section 14.3.1) (Figure 14.1d).

The most challenging version of recognition is general *category* (or *class*) recognition (Section 14.4), which may involve recognizing instances of extremely varied classes such as animals or furniture. Some techniques rely purely on the presence of features (known as a “bag of words” model—see Section 14.4.1), their relative positions (*part-based models* (Section 14.4.2)), Figure 14.1e, while others involve segmenting the image into semantically meaningful regions (Section 14.4.3) (Figure 14.1f). In many instances, recognition depends heavily on the *context* of surrounding objects and scene elements (Section 14.5). Woven into all of these techniques is the topic of *learning* (Section 14.5.1), since hand-crafting specific object recognizers seems like a futile approach given the complexity of the problem.

Given the extremely rich and complex nature of this topic, this chapter is structured to build from simpler concepts to more complex ones. We begin with a discussion of face and object detection (Section 14.1), where we introduce a number of machine-learning techniques such as boosting, neural networks, and support vector machines. Next, we study face recognition (Section 14.2), which is one of the more widely known applications of recognition. This topic serves as an introduction to subspace (PCA) models and Bayesian approaches to recognition and classification. We then present techniques for instance recognition (Section 14.3), building upon earlier topics in this book, such as feature detection, matching, and geometric alignment (Section 14.3.1). We introduce topics from the information and document retrieval communities, such as frequency vectors, feature quantization, and inverted indices (Section 14.3.2). We also present applications of location recognition (Section 14.3.3).

In the second half of the chapter, we address the most challenging variant of recognition, namely the problem of category recognition (Section 14.4). This includes approaches that use bags of features (Section 14.4.1), parts (Section 14.4.2), and segmentation (Section 14.4.3). We show how such techniques can be used to automate photo editing tasks, such as 3D modeling, scene completion, and creating collages (Section 14.4.4). Next, we discuss the role that context can play in both individual object recognition and more holistic scene under-

¹ However, some recent research suggests that direct image matching may be feasible for large enough databases (Russell, Torralba, Liu *et al.* 2007; Malisiewicz and Efros 2008; Torralba, Freeman, and Fergus 2008).

standing (Section 14.5). We close this chapter with a discussion of databases and test sets for constructing and evaluating recognition systems (Section 14.6).

While there is no comprehensive reference on object recognition, an excellent set of notes can be found in the ICCV 2009 short course (Fei-Fei, Fergus, and Torralba 2009), Antonio Torralba's more comprehensive MIT course (Torralba 2008), and two recent collections of papers (Ponce, Hebert, Schmid *et al.* 2006; Dickinson, Leonardis, Schiele *et al.* 2007) and a survey on object categorization (Pinz 2005). An evaluation of some of the best performing recognition algorithms can be found on the PASCAL Visual Object Classes (VOC) Challenge Web site at <http://pascallin.ecs.soton.ac.uk/challenges/VOC/>.

14.1 Object detection

If we are given an image to analyze, such as the group portrait in Figure 14.2, we could try to apply a recognition algorithm to every possible sub-window in this image. Such algorithms are likely to be both slow and error-prone. Instead, it is more effective to construct special-purpose *detectors*, whose job it is to rapidly find likely regions where particular objects might occur.

We begin this section with face detectors, which are some of the more successful examples of recognition. For example, such algorithms are built into most of today's digital cameras to enhance auto-focus and into video conferencing systems to control pan-tilt heads. We then look at pedestrian detectors, as an example of more general methods for object detection. Such detectors can be used in automotive safety applications, e.g., detecting pedestrians and other cars from moving vehicles (Leibe, Cornelis, Cornelis *et al.* 2007).

14.1.1 Face detection

Before face recognition can be applied to a general image, the locations and sizes of any faces must first be found (Figures 14.1c and 14.2). In principle, we could apply a face recognition algorithm at every pixel and scale (Moghaddam and Pentland 1997) but such a process would be too slow in practice.

Over the years, a wide variety of fast face detection algorithms have been developed. Yang, Kriegman, and Ahuja (2002) provide a comprehensive survey of earlier work in this field; Yang's ICPR 2004 tutorial² and the Torralba (2007) short course provide more recent reviews.³

According to the taxonomy of Yang, Kriegman, and Ahuja (2002), face detection techniques can be classified as feature-based, template-based, or appearance-based. Feature-based techniques attempt to find the locations of distinctive image features such as the eyes, nose, and mouth, and then verify whether these features are in a plausible geometrical arrangement. These techniques include some of the early approaches to face recognition (Fischler and Elschlager 1973; Kanade 1977; Yuille 1991), as well as more recent approaches based on modular eigenspaces (Moghaddam and Pentland 1997), local filter jets (Leung, Burl, and Perona 1995; Penev and Atick 1996; Wiskott, Fellous, Krüger *et al.* 1997), support

² <http://vision.ai.uiuc.edu/mhyang/face-detection-survey.html>.

³ An alternative approach to detecting faces is to look for regions of skin color in the image (Forsyth and Fleck 1999; Jones and Rehg 2001). See Exercise 2.8 for some additional discussion and references.

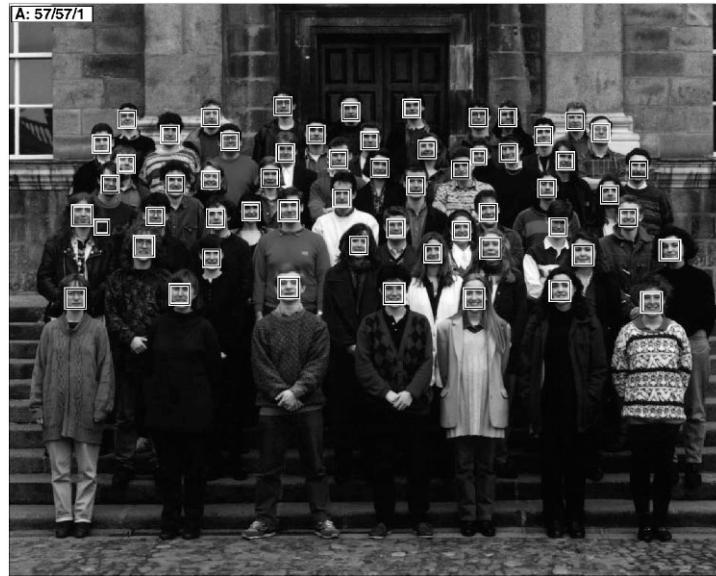


Figure 14.2 Face detection results produced by Rowley, Baluja, and Kanade (1998a) © 1998 IEEE. Can you find the one false positive (a box around a non-face) among the 57 true positive results?

vector machines (Heisele, Ho, Wu *et al.* 2003; Heisele, Serre, and Poggio 2007), and boosting (Schneiderman and Kanade 2004).

Template-based approaches, such as active appearance models (AAMs) (Section 14.2.2), can deal with a wide range of pose and expression variability. Typically, they require good initialization near a real face and are therefore not suitable as fast face detectors.

Appearance-based approaches scan over small overlapping rectangular patches of the image searching for likely face candidates, which can then be refined using a *cascade* of more expensive but selective detection algorithms (Sung and Poggio 1998; Rowley, Baluja, and Kanade 1998a; Romdhani, Torr, Schölkopf *et al.* 2001; Fleuret and Geman 2001; Viola and Jones 2004). In order to deal with scale variation, the image is usually converted into a sub-octave pyramid and a separate scan is performed on each level. Most appearance-based approaches today rely heavily on training classifiers using sets of labeled face and non-face patches.

Sung and Poggio (1998) and Rowley, Baluja, and Kanade (1998a) present two of the earliest appearance-based face detectors and introduce a number of innovations that are widely used in later work by others.

To start with, both systems collect a set of labeled face patches (Figure 14.2) as well as a set of patches taken from images that are known not to contain faces, such as aerial images or vegetation (Figure 14.3b). The collected face images are augmented by artificially mirroring, rotating, scaling, and translating the images by small amounts to make the face detectors less sensitive to such effects (Figure 14.3a).

After an initial set of training images has been collected, some optional pre-processing can be performed, such as subtracting an average gradient (linear function) from the image to compensate for global shading effects and using histogram equalization to compensate for

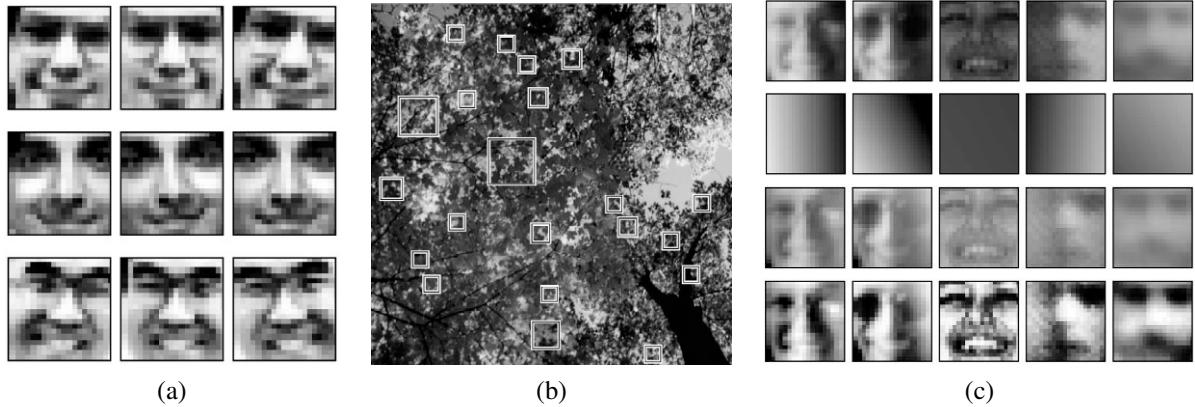


Figure 14.3 Pre-processing stages for face detector training (Rowley, Baluja, and Kanade 1998a) © 1998 IEEE: (a) artificially mirroring, rotating, scaling, and translating training images for greater variability; (b) using images without faces (looking up at a tree) to generate non-face examples; (c) pre-processing the patches by subtracting a best fit linear function (constant gradient) and histogram equalizing.

varying camera contrast (Figure 14.3c).

Clustering and PCA. Once the face and non-face patterns have been pre-processed, Sung and Poggio (1998) cluster each of these datasets into six separate clusters using k-means and then fit PCA subspaces to each of the resulting 12 clusters (Figure 14.4). At detection time, the DIFS and DFFS metrics first developed by Moghaddam and Pentland (1997) (see Figure 14.14 and (14.14)) are used to produce 24 Mahalanobis distance measurements (two per cluster). The resulting 24 measurements are input to a multi-layer perceptron (MLP), which is a neural network with alternating layers of weighted summations and sigmoidal non-linearities trained using the “backpropagation” algorithm (Rumelhart, Hinton, and Williams 1986).

Neural networks. Instead of first clustering the data and computing Mahalanobis distances to the cluster centers, Rowley, Baluja, and Kanade (1998a) apply a neural network (MLP) directly to the 20×20 pixel patches of gray-level intensities, using a variety of differently sized hand-crafted “receptive fields” to capture both large-scale and smaller scale structure (Figure 14.5). The resulting neural network directly outputs the likelihood of a face at the center of every overlapping patch in a multi-resolution pyramid. Since several overlapping patches (in both space and resolution) may fire near a face, an additional merging network is used to merge overlapping detections. The authors also experiment with training several networks and merging their outputs. Figure 14.2 shows a sample result from their face detector.

To make the detector run faster, a separate network operating on 30×30 patches is trained to detect both faces and faces shifted by ± 5 pixels. This network is evaluated at every 10th pixel in the image (horizontally and vertically) and the results of this “coarse” or “sloppy” detector are used to select regions on which to run the slower single-pixel overlap technique. To deal with in-plane rotations of faces, Rowley, Baluja, and Kanade (1998b) train a *router*

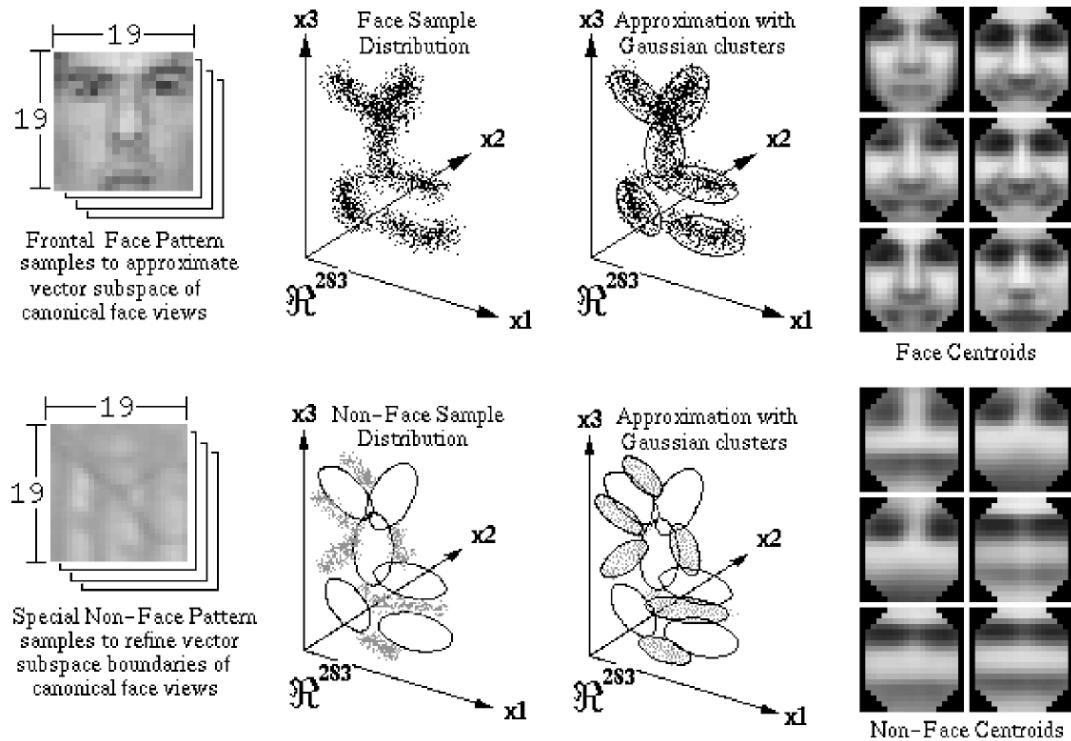


Figure 14.4 Learning a mixture of Gaussians model for face detection (Sung and Poggio 1998) © 1998 IEEE. The face and non-face images (19^2 -long vectors) are first clustered into six separate clusters (each) using k-means and then analyzed using PCA. The cluster centers are shown in the right-hand columns.

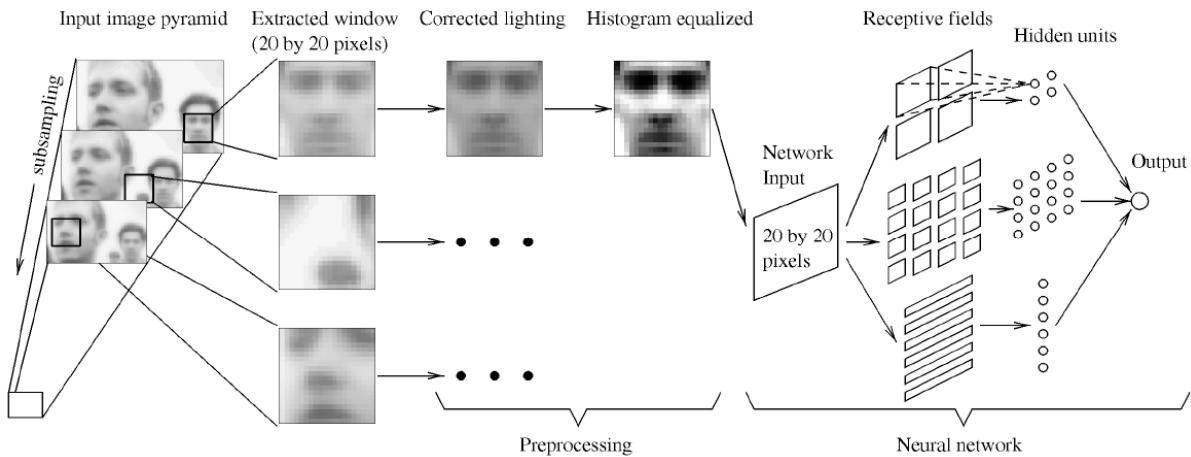


Figure 14.5 A neural network for face detection (Rowley, Baluja, and Kanade 1998a) © 1998 IEEE. Overlapping patches are extracted from different levels of a pyramid and then pre-processed as shown in Figure 14.3b. A three-layer neural network is then used to detect likely face locations.

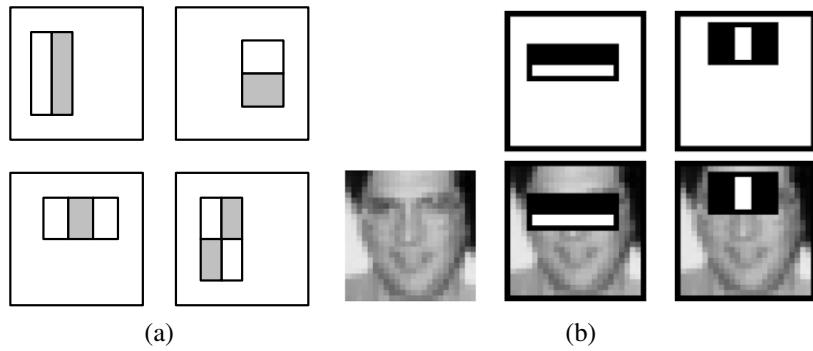


Figure 14.6 Simple features used in boosting-based face detector (Viola and Jones 2004) © 2004 Springer: (a) difference of rectangle feature composed of 2–4 different rectangles (pixels inside the white rectangles are subtracted from the gray ones); (b) the first and second features selected by AdaBoost. The first feature measures the differences in intensity between the eyes and the cheeks, the second one between the eyes and the bridge of the nose.

network to estimate likely rotation angles from input patches and then apply the estimated rotation to each patch before running the result through their upright face detector.

Support vector machines. Instead of using a neural network to classify patches, Osuna, Freund, and Girosi (1997) use a *support vector machine* (SVM) (Hastie, Tibshirani, and Friedman 2001; Schölkopf and Smola 2002; Bishop 2006; Lampert 2008) to classify the same preprocessed patches as Sung and Poggio (1998). An SVM searches for a series of *maximum margin* separating planes in feature space between different classes (in this case, face and non-face patches). In those cases where linear classification boundaries are insufficient, the feature space can be lifted into higher-dimensional features using *kernels* (Hastie, Tibshirani, and Friedman 2001; Schölkopf and Smola 2002; Bishop 2006). SVMs have been used by other researchers for both face detection and face recognition (Heisele, Ho, Wu *et al.* 2003; Heisele, Serre, and Poggio 2007) and are a widely used tool in object recognition in general.

Boosting. Of all the face detectors currently in use, the one introduced by Viola and Jones (2004) is probably the best known and most widely used. Their technique was the first to introduce the concept of *boosting* to the computer vision community, which involves training a series of increasingly discriminating simple classifiers and then blending their outputs (Hastie, Tibshirani, and Friedman 2001; Bishop 2006).

In more detail, boosting involves constructing a *classifier* $h(\mathbf{x})$ as a sum of simple *weak learners*,

$$h(\mathbf{x}) = \text{sign} \left[\sum_{j=0}^{m-1} \alpha_j h_j(\mathbf{x}) \right], \quad (14.1)$$

where each of the weak learners $h_j(\mathbf{x})$ is an extremely simple function of the input, and hence is not expected to contribute much (in isolation) to the classification performance.

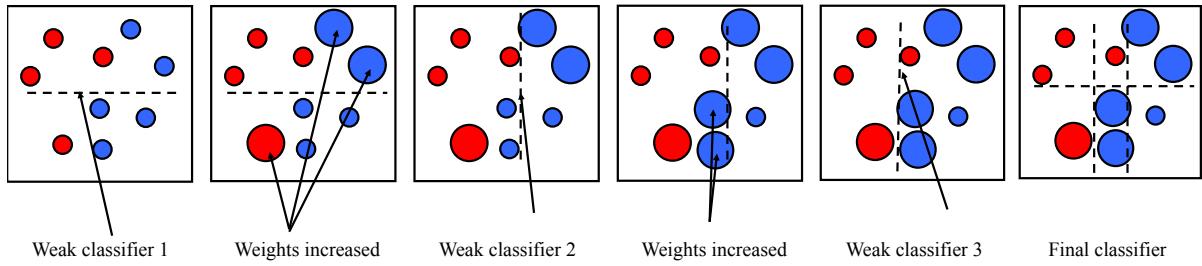


Figure 14.7 Schematic illustration of boosting, courtesy of Svetlana Lazebnik, after original illustrations from Paul Viola and David Lowe. After each weak classifier (decision stump or hyperplane) is selected, data points that are erroneously classified have their weights increased. The final classifier is a linear combination of the simple weak classifiers.

In most variants of boosting, the weak learners are threshold functions,

$$h_j(\mathbf{x}) = a_j[f_j < \theta_j] + b_j[f_j \geq \theta_j] = \begin{cases} a_j & \text{if } f_j < \theta_j \\ b_j & \text{otherwise,} \end{cases} \quad (14.2)$$

which are also known as *decision stumps* (basically, the simplest possible version of *decision trees*). In most cases, it is also traditional (and simpler) to set a_j and b_j to ± 1 , i.e., $a_j = -s_j$, $b_j = +s_j$, so that only the feature f_j , the threshold value θ_j , and the polarity of the threshold $s_j \in \{\pm 1\}$ need to be selected.⁴

In many applications of boosting, the features are simply coordinate axes x_k , i.e., the boosting algorithm selects one of the input vector components as the best one to threshold. In Viola and Jones' face detector, the features are differences of rectangular regions in the input patch, as shown in Figure 14.6. The advantage of using these features is that, while they are more discriminating than single pixels, they are extremely fast to compute once a summed area table has been pre-computed, as described in Section 3.2.3 (3.31–3.32). Essentially, for the cost of an $O(N)$ pre-computation phase (where N is the number of pixels in the image), subsequent differences of rectangles can be computed in $4r$ additions or subtractions, where $r \in \{2, 3, 4\}$ is the number of rectangles in the feature.

The key to the success of boosting is the method for incrementally selecting the weak learners and for re-weighting the training examples after each stage (Figure 14.7). The AdaBoost (Adaptive Boosting) algorithm (Hastie, Tibshirani, and Friedman 2001; Bishop 2006) does this by re-weighting each sample as a function of whether it is correctly classified at each stage, and using the stage-wise average classification error to determine the final weightings α_j among the weak classifiers, as described in Algorithm 14.1. While the resulting classifier is extremely fast in practice, the training time can be quite slow (in the order of weeks), because of the large number of feature (difference of rectangle) hypotheses that need to be examined at each stage.

To further increase the speed of the detector, it is possible to create a *cascade* of classifiers, where each classifier uses a small number of tests (say, a two-term AdaBoost classifier) to reject a large fraction of non-faces while trying to pass through all potential face candidates

⁴Some variants, such as that of Viola and Jones (2004), use $(a_j, b_j) \in [0, 1]$ and adjust the learning algorithm accordingly.

1. Input the positive and negative training examples along with their labels $\{(\mathbf{x}_i, y_i)\}$, where $y_i = 1$ for positive (face) examples and $y_i = -1$ for negative examples.
2. Initialize all the weights to $w_{i,1} \leftarrow \frac{1}{N}$, where N is the number of training examples. (Viola and Jones (2004) use a separate N_1 and N_2 for positive and negative examples.)
3. For each training stage $j = 1 \dots M$:
 - (a) Renormalize the weights so that they sum up to 1 (divide them by their sum).
 - (b) Select the best classifier $h_j(\mathbf{x}; f_j, \theta_j, s_j)$ by finding the one that minimizes the weighted classification error

$$e_j = \sum_{i=0}^{N-1} w_{i,j} e_{i,j}, \quad (14.3)$$

$$e_{i,j} = 1 - \delta(y_i, h_j(\mathbf{x}_i; f_j, \theta_j, s_j)). \quad (14.4)$$

For any given f_j function, the optimal values of (θ_j, s_j) can be found in linear time using a variant of weighted median computation (Exercise 14.2).

- (c) Compute the modified error rate β_j and classifier weight α_j ,

$$\beta_j = \frac{e_j}{1 - e_j} \quad \text{and} \quad \alpha_j = -\log \beta_j. \quad (14.5)$$

- (d) Update the weights according to the classification errors $e_{i,j}$

$$w_{i,j+1} \leftarrow w_{i,j} \beta_j^{1-e_{i,j}}, \quad (14.6)$$

i.e., downweight the training samples that were correctly classified in proportion to the overall classification error.

4. Set the final classifier to

$$h(\mathbf{x}) = \text{sign} \left[\sum_{j=0}^{m-1} \alpha_j h_j(\mathbf{x}) \right]. \quad (14.7)$$

Algorithm 14.1 The AdaBoost training algorithm, adapted from Hastie, Tibshirani, and Friedman (2001), Viola and Jones (2004), and Bishop (2006).

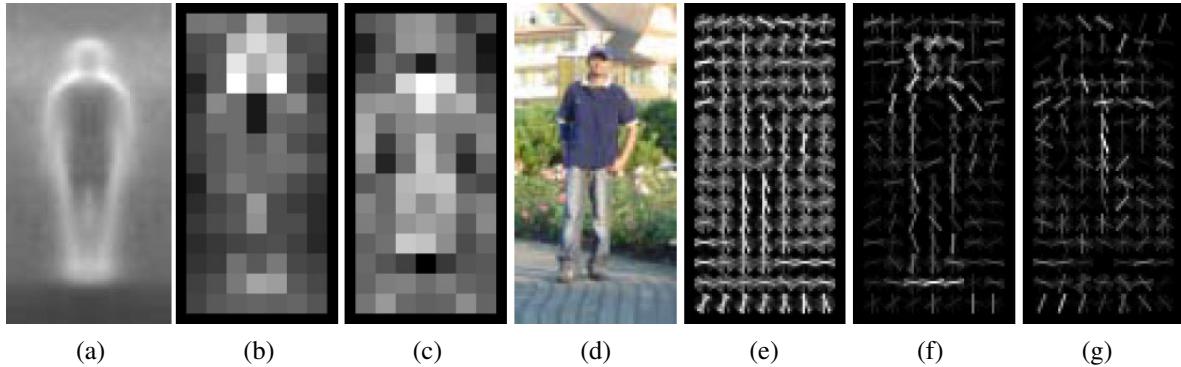


Figure 14.8 Pedestrian detection using histograms of oriented gradients (Dalal and Triggs 2005) © 2005 IEEE: (a) the average gradient image over the training examples; (b) each “pixel” shows the maximum positive SVM weight in the block centered on the pixel; (c) likewise, for the negative SVM weights; (d) a test image; (e) the computed R-HOG (rectangular histogram of gradients) descriptor; (f) the R-HOG descriptor weighted by the positive SVM weights; (g) the R-HOG descriptor weighted by the negative SVM weights.

(Fleuret and Geman 2001; Viola and Jones 2004). An even faster algorithm for performing cascade learning has recently been developed by Brubaker, Wu, Sun *et al.* (2008).

14.1.2 Pedestrian detection

While a lot of the research on object detection has focused on faces, the detection of other objects, such as pedestrians and cars, has also received widespread attention (Gavrila and Philomin 1999; Gavrila 1999; Papageorgiou and Poggio 2000; Mohan, Papageorgiou, and Poggio 2001; Schneiderman and Kanade 2004). Some of these techniques maintain the same focus as face detection on speed and efficiency. Others, however, focus instead on accuracy, viewing detection as a more challenging variant of generic class recognition (Section 14.4) in which the locations and extents of objects are to be determined as accurately as possible. (See, for example, the PASCAL VOC detection challenge, <http://pascallin.ecs.soton.ac.uk/challenges/VOC/>.)

An example of a well-known pedestrian detector is the algorithm developed by Dalal and Triggs (2005), who use a set of overlapping *histogram of oriented gradients* (HOG) descriptors fed into a support vector machine (Figure 14.8). Each HOG has cells to accumulate magnitude-weighted votes for gradients at particular orientations, just as in the scale invariant feature transform (SIFT) developed by Lowe (2004), which we discussed in Section 4.1.2 and Figure 4.18. Unlike SIFT, however, which is only evaluated at interest point locations, HOGs are evaluated on a regular overlapping grid and their descriptor magnitudes are normalized using an even coarser grid; they are only computed at a single scale and a fixed orientation. In order to capture the subtle variations in orientation around a person’s outline, a large number of orientation bins is used and no smoothing is performed in the central difference gradient computation—see the work of Dalal and Triggs (2005) for more implementation details. Figure 14.8d shows a sample input image, while Figure 14.8e shows the associated HOG descriptors.

Once the descriptors have been computed, a support vector machine (SVM) is trained

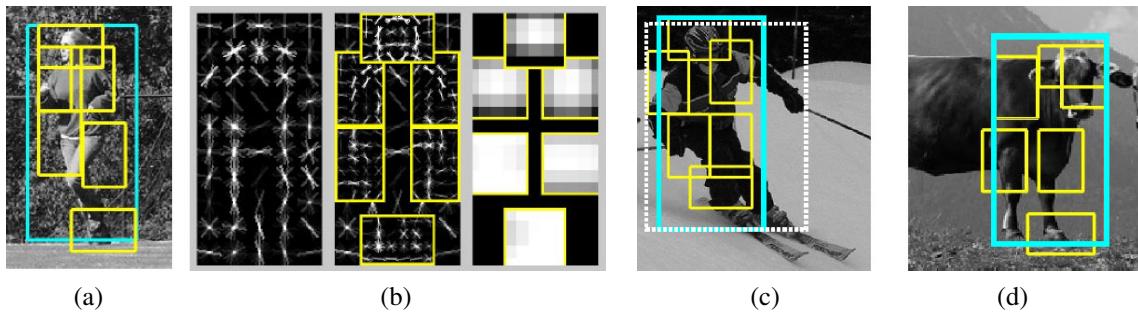


Figure 14.9 Part-based object detection (Felzenszwalb, McAllester, and Ramanan 2008) © 2008 IEEE: (a) An input photograph and its associated person (blue) and part (yellow) detection results. (b) The detection model is defined by a coarse template, several higher resolution part templates, and a spatial model for the location of each part. (c) True positive detection of a skier and (d) false positive detection of a cow (labeled as a person).

on the resulting high-dimensional continuous descriptor vectors. Figures 14.8b–c show a diagram of the (most) positive and negative SVM weights in each block, while Figures 14.8f–g show the corresponding weighted HOG responses for the central input image. As you can see, there are a fair number of positive responses around the head, torso, and feet of the person, and relatively few negative responses (mainly around the middle and the neck of the sweater).

The fields of pedestrian and general object detection have continued to evolve rapidly over the last decade (Belongie, Malik, and Puzicha 2002; Mikolajczyk, Schmid, and Zisserman 2004; Leibe, Seemann, and Schiele 2005; Opel, Pinz, and Zisserman 2006; Torralba 2007; Andriluka, Roth, and Schiele 2009, 2010; Dollàr, Belongie, and Perona 2010). Munder and Gavrila (2006) compare a number of pedestrian detectors and conclude that those based on local receptive fields and SVMs perform the best, with a boosting-based approach coming close. Maji, Berg, and Malik (2008) improve on the best of these results using non-overlapping multi-resolution HOG descriptors and a histogram intersection kernel SVM based on a spatial pyramid match kernel from Lazebnik, Schmid, and Ponce (2006).

When detectors for several different classes are being constructed simultaneously, Torralba, Murphy, and Freeman (2007) show that sharing features and weak learners between detectors yields better performance, both in terms of faster computation times and fewer training examples. To find the features and decision stumps that work best in a shared manner, they introduce a novel *joint boosting* algorithm that optimizes, at each stage, a summed expected exponential loss function using the “gentleboost” algorithm of Friedman, Hastie, and Tibshirani (2000).

In more recent work, Felzenszwalb, McAllester, and Ramanan (2008) extend the histogram of oriented gradients person detector to incorporate flexible parts models (Section 14.4.2). Each part is trained and detected on HOGs evaluated at two pyramid levels below the overall object model and the locations of the parts relative to the parent node (the overall bounding box) are also learned and used during recognition (Figure 14.9b). To compensate for inaccuracies or inconsistencies in the training example bounding boxes (dashed white lines in Figure 14.9c), the “true” location of the parent (blue) bounding box is considered a latent (hidden) variable and is inferred during both training and recognition. Since the locations

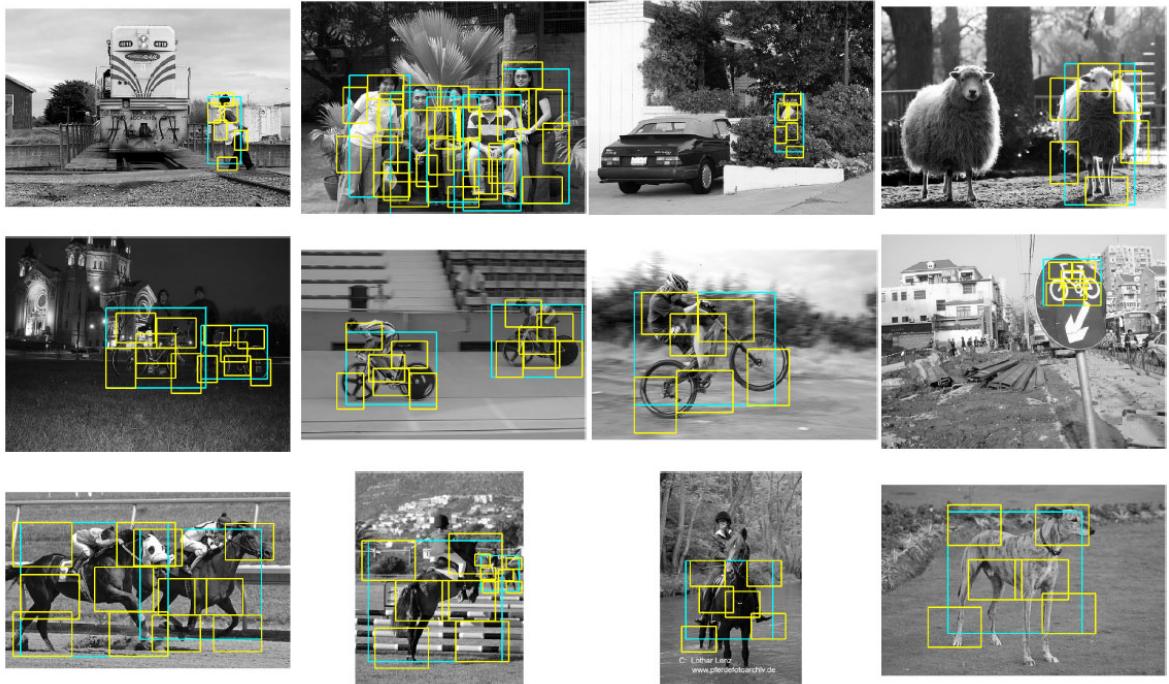


Figure 14.10 Part-based object detection results for people, bicycles, and horses (Felzenszwalb, McAllester, and Ramanan 2008) © 2008 IEEE. The first three columns show correct detections, while the rightmost column shows false positives.

of the parts are also latent, the system can be trained in a semi-supervised fashion, without needing part labels in the training data. An extension to this system (Felzenszwalb, Girshick, McAllester *et al.* 2010), which includes among its improvements a simple contextual model, was among the two best object detection systems in the 2008 Visual Object Classes detection challenge. Other recent improvements to part-based person detection and pose estimation include the work by Andriluka, Roth, and Schiele (2009) and Kumar, Zisserman, and H.S.Torr (2009).

An even more accurate estimate of a person’s pose and location is presented by Rogez, Rihan, Ramalingam *et al.* (2008), who compute both the phase of a person in a walk cycle and the locations of individual joints, using random forests built on top of HOGs (Figure 14.11). Since their system produces full 3D pose information, it is closer in its application domain to 3D person trackers (Sidenbladh, Black, and Fleet 2000; Andriluka, Roth, and Schiele 2010), which we discussed in Section 12.6.4.

One final note on person and object detection. When video sequences are available, the additional information present in the optic flow and motion discontinuities can greatly aid in the detection task, as discussed by Efros, Berg, Mori *et al.* (2003), Viola, Jones, and Snow (2003), and Dalal, Triggs, and Schmid (2006).

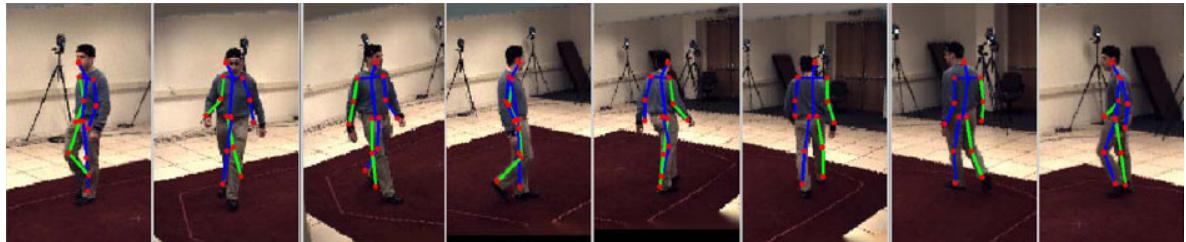


Figure 14.11 Pose detection using random forests (Rogez, Rihan, Ramalingam *et al.* 2008) © 2008 IEEE. The estimated pose (state of the kinematic model) is drawn over each input frame.



Figure 14.12 Humans can recognize low-resolution faces of familiar people (Sinha, Balas, Ostrovsky *et al.* 2006) © 2006 IEEE.

14.2 Face recognition

Among the various recognition tasks that computers might be asked to perform, face recognition is the one where they have arguably had the most success.⁵ While computers cannot pick out suspects from thousands of people streaming in front of video cameras (even people cannot readily distinguish between similar people with whom they are not familiar (O'Toole, Jiang, Roark *et al.* 2006; O'Toole, Phillips, Jiang *et al.* 2009)), their ability to distinguish among a small number of family members and friends has found its way into consumer-level photo applications, such as Picasa and iPhoto. Face recognition can also be used in a variety of additional applications, including human–computer interaction (HCI), identity verification (Kirovski, Jovic, and Jancke 2004), desktop login, parental controls, and patient monitoring (Zhao, Chellappa, Phillips *et al.* 2003).

Today's face recognizers work best when they are given full frontal images of faces under relatively uniform illumination conditions, although databases that include large amounts of pose and lighting variation have been collected (Phillips, Moon, Rizvi *et al.* 2000; Sim,

⁵Instance recognition, i.e., the re-recognition of known objects such as locations or planar objects, is the other most successful application of general image recognition. In the general domain of *biometrics*, i.e., identity recognition, specialized images such as irises and fingerprints perform even better (Jain, Bolle, and Pankanti 1999; Pankanti, Bolle, and Jain 2000; Daugman 2004).

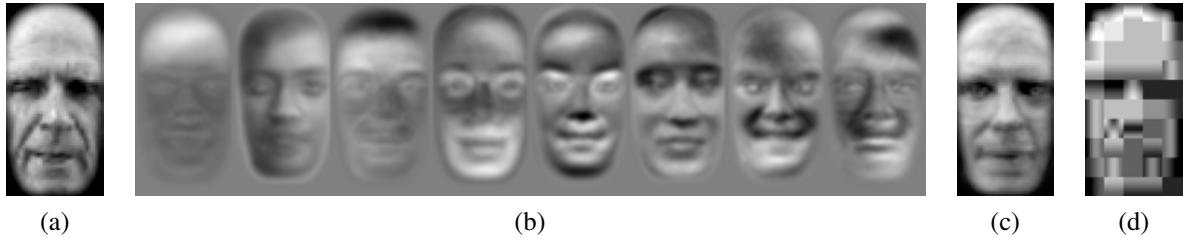


Figure 14.13 Face modeling and compression using eigenfaces (Moghaddam and Pentland 1997) © 1997 IEEE: (a) input image; (b) the first eight eigenfaces; (c) image reconstructed by projecting onto this basis and compressing the image to 85 bytes; (d) image reconstructed using JPEG (530 bytes).

Baker, and Bsat 2003; Gross, Shi, and Cohn 2005; Huang, Ramesh, Berg *et al.* 2007; Phillips, Scruggs, O'Toole *et al.* 2010). (See Table 14.1 in Section 14.6 for more details.)

Some of the earliest approaches to face recognition involved finding the locations of distinctive image features, such as the eyes, nose, and mouth, and measuring the distances between these feature locations (Fischler and Elschlager 1973; Kanade 1977; Yuille 1991). More recent approaches rely on comparing gray-level images projected onto lower dimensional subspaces called *eigenfaces* (Section 14.2.1) and jointly modeling shape and appearance variations (while discounting pose variations) using *active appearance models* (Section 14.2.2).

Descriptions of additional face recognition techniques can be found in a number of surveys and books on this topic (Chellappa, Wilson, and Sirohey 1995; Zhao, Chellappa, Phillips *et al.* 2003; Li and Jain 2005) as well as the Face Recognition Web site.⁶ The survey on face recognition by humans by Sinha, Balas, Ostrovsky *et al.* (2006) is also well worth reading; it includes a number of surprising results, such as humans' ability to recognize low-resolution images of familiar faces (Figure 14.12) and the importance of eyebrows in recognition.

14.2.1 Eigenfaces

Eigenfaces rely on the observation first made by Kirby and Sirovich (1990) that an arbitrary face image \mathbf{x} can be compressed and reconstructed by starting with a mean image \mathbf{m} (Figure 14.1b) and adding a small number of scaled signed images \mathbf{u}_i ,⁷

$$\tilde{\mathbf{x}} = \mathbf{m} + \sum_{i=0}^{M-1} a_i \mathbf{u}_i, \quad (14.8)$$

where the signed basis images (Figure 14.13b) can be derived from an ensemble of training images using *principal component analysis* (also known as *eigenvalue analysis* or the *Karhunen–Loëve transform*). Turk and Pentland (1991a) recognized that the coefficients a_i in the eigenface expansion could themselves be used to construct a fast image matching algorithm.

⁶ <http://www.face-rec.org/>.

⁷ In previous chapters, we used I to indicate images; in this chapter, we use the more abstract quantities \mathbf{x} and \mathbf{u} to indicate collections of pixels in an image turned into a vector.

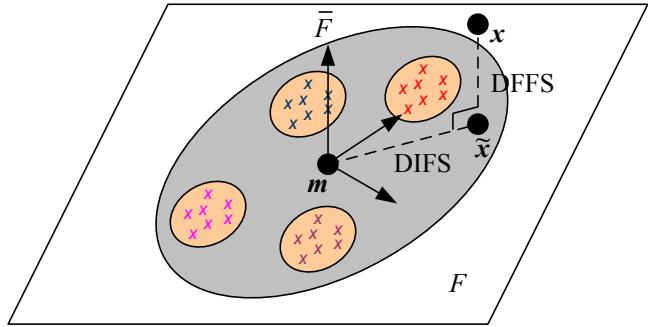


Figure 14.14 Projection onto the linear subspace spanned by the eigenface images (Moghaddam and Pentland 1997) © 1997 IEEE. The distance from face space (DFFS) is the orthogonal distance to the plane, while the distance in face space (DIFS) is the distance along the plane from the mean image. Both distances can be turned into Mahalanobis distances and given probabilistic interpretations.

In more detail, let us start with a collection of *training images* $\{x_j\}$, from which we can compute the mean image \mathbf{m} and a *scatter* or *covariance* matrix

$$\mathbf{C} = \frac{1}{N} \sum_{j=0}^{N-1} (\mathbf{x}_j - \mathbf{m})(\mathbf{x}_j - \mathbf{m})^T. \quad (14.9)$$

We can apply the eigenvalue decomposition (A.6) to represent this matrix as

$$\mathbf{C} = \mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^T = \sum_{i=0}^{N-1} \lambda_i \mathbf{u}_i \mathbf{u}_i^T, \quad (14.10)$$

where the λ_i are the eigenvalues of \mathbf{C} and the \mathbf{u}_i are the *eigenvectors*. For general images, Kirby and Sirovich (1990) call these vectors *eigenpictures*; for faces, Turk and Pentland (1991a) call them *eigenfaces* (Figure 14.13b).⁸

Two important properties of the eigenvalue decomposition are that the optimal (best approximation) coefficients a_i for any new image \mathbf{x} can be computed as

$$a_i = (\mathbf{x} - \mathbf{m}) \cdot \mathbf{u}_i, \quad (14.11)$$

and that, assuming the eigenvalues $\{\lambda_i\}$ are sorted in decreasing order, truncating the approximation given in (14.8) at any point M gives the best possible approximation (least error) between $\tilde{\mathbf{x}}$ and \mathbf{x} . Figure 14.13c shows the resulting approximation corresponding to Figure 14.13a and shows how much better it is at compressing a face image than JPEG.

Truncating the eigenface decomposition of a face image (14.8) after M components is equivalent to projecting the image onto a linear subspace F , which we can call the *face space* (Figure 14.14). Because the eigenvectors (eigenfaces) are orthogonal and of unit norm, the

⁸ In actual practice, the full $P \times P$ scatter matrix (14.9) is never computed. Instead, a smaller $N \times N$ matrix consisting of the inner products between all the signed deviations $(\mathbf{x}_i - \mathbf{m})$ is accumulated instead. See Appendix A.1.2 (A.13–A.14) for details.

distance of a projected face $\tilde{\mathbf{x}}$ to the mean face \mathbf{m} can be written as

$$\text{DIFS} = \|\tilde{\mathbf{x}} - \mathbf{m}\| = \sqrt{\sum_{i=0}^{M-1} a_i^2}, \quad (14.12)$$

where DIFS stands for *distance in face space* (Moghaddam and Pentland 1997). The remaining distance between the original image \mathbf{x} and its projection onto face space $\tilde{\mathbf{x}}$, i.e., the *distance from face space* (DFFS), can be computed directly in pixel space and represents the “faceness” of a particular image.⁹ It is also possible to measure the distance between two different faces in face space as

$$\text{DIFS}(\mathbf{x}, \mathbf{y}) = \|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\| = \sqrt{\sum_{i=0}^{M-1} (a_i - b_i)^2}, \quad (14.13)$$

where the $b_i = (\mathbf{y} - \mathbf{m}) \cdot \mathbf{u}_i$ are the eigenface coefficients corresponding to \mathbf{y} .

Computing such distances in Euclidean vector space, however, does not exploit the additional information that the eigenvalue decomposition of our covariance matrix (14.10) provides. If we interpret the covariance matrix \mathbf{C} as the covariance of a multi-variate Gaussian (Appendix B.1.1),¹⁰ we can turn the DIFS into a log likelihood by computing the *Mahalanobis distance*

$$\text{DIFS}' = \|\tilde{\mathbf{x}} - \mathbf{m}\|_{\mathbf{C}^{-1}} = \sqrt{\sum_{i=0}^{M-1} a_i^2 / \lambda_i^2}. \quad (14.14)$$

Instead of measuring the squared distance along each principal component in face space F , the Mahalanobis distance measures the *ratio* between the squared distance and the corresponding *variance* $\sigma_i^2 = \lambda_i$ and then sums these squared ratios (per-component log-likelihoods). An alternative way to implement this is to pre-scale each eigenvector by the inverse square root of its corresponding eigenvalue,

$$\hat{\mathbf{U}} = \mathbf{U} \mathbf{\Lambda}^{-1/2}. \quad (14.15)$$

This *whitening* transformation then means that Euclidean distances in feature (face) space now correspond directly to log likelihoods (Moghaddam, Jebara, and Pentland 2000). (This same whitening approach can also be used in feature-based matching algorithms, as discussed in Section 4.1.3.)

If the distribution in eigenface space is very elongated, the Mahalanobis distance properly scales the components to come up with a sensible (probabilistic) distance from the mean. A similar analysis can be performed for computing a sensible difference from face space (DFFS) (Moghaddam and Pentland 1997) and the two terms can be combined to produce an estimate of the likelihood of being a true face, which can be useful in doing face detection (Section 14.1.1). More detailed explanations of probabilistic and Bayesian PCA can be found in textbooks on statistical learning (Hastie, Tibshirani, and Friedman 2001; Bishop 2006), which also discuss techniques for selecting the optimum number of components M to use in modeling a distribution.

⁹ This can be used to form a simple face detector, as mentioned in Section 14.1.1.

¹⁰ The ellipse shown in Figure 14.14 denotes an equi-probability contour of this multi-variate Gaussian.

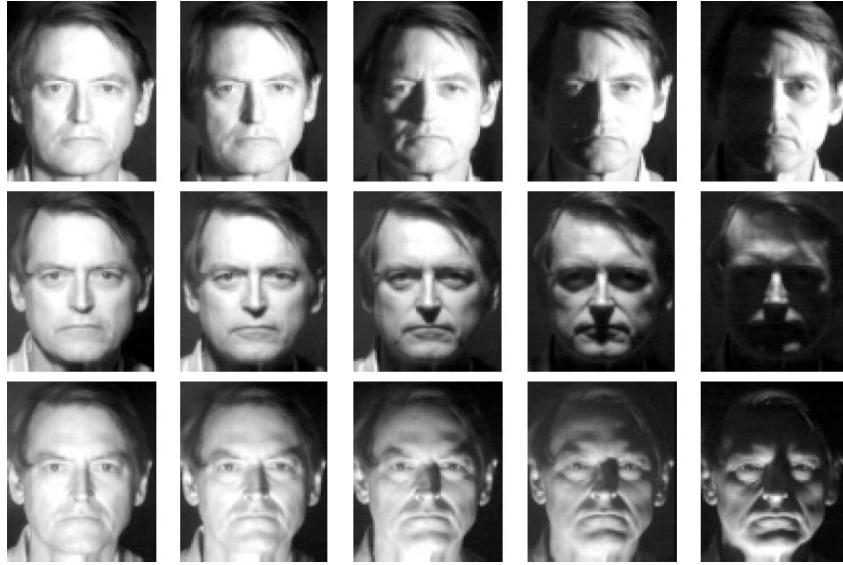


Figure 14.15 Images from the Harvard database used by Belhumeur, Hespanha, and Kriegman (1997) © 1997 IEEE. Note the wide range of illumination variation, which can be more dramatic than inter-personal variations.

One of the biggest advantages of using eigenfaces is that they reduce the comparison of a new face image \mathbf{x} to a prototype (training) face image \mathbf{x}_k (one of the colored \mathbf{x} s in Figure 14.14) from a P -dimensional difference in pixel space to an M -dimensional difference in face space,

$$\|\mathbf{x} - \mathbf{x}_k\| = \|\mathbf{a} - \mathbf{a}_k\|, \quad (14.16)$$

where $\mathbf{a} = \mathbf{U}^T(\mathbf{x} - \mathbf{m})$ (14.11) involves computing a dot product between the signed difference-from-mean image $(\mathbf{x} - \mathbf{m})$ and each of the eigenfaces \mathbf{u}_i . Once again, however, this Euclidean distance ignores the fact that we have more information about face likelihoods available in the distribution of training images.

Consider the set of images of one person taken under a wide range of illuminations shown in Figure 14.15. As you can see, the *intrapersonal* variability within these images is much greater than the typical *extrapersonal* variability between any two people taken under the same illumination. Regular PCA analysis fails to distinguish between these two sources of variability and may, in fact, devote most of its principal components to modeling the intrapersonal variability.

If we are going to approximate faces by a linear subspace, it is more useful to have a space that *discriminates* between different classes (people) and is less sensitive to within-class variations (Belhumeur, Hespanha, and Kriegman 1997). Consider the three classes shown as different colors in Figure 14.16. As you can see, the distributions within a class (indicated by the tilted colored axes) are elongated and tilted with respect to the main face space PCA, which is aligned with the black x and y axes. We can compute the total *within-class* scatter matrix as

$$\mathbf{S}_W = \sum_{k=0}^{K-1} \mathbf{S}_k = \sum_{k=0}^{K-1} \sum_{i \in C_k} (\mathbf{x}_i - \mathbf{m}_k)(\mathbf{x}_i - \mathbf{m}_k)^T, \quad (14.17)$$

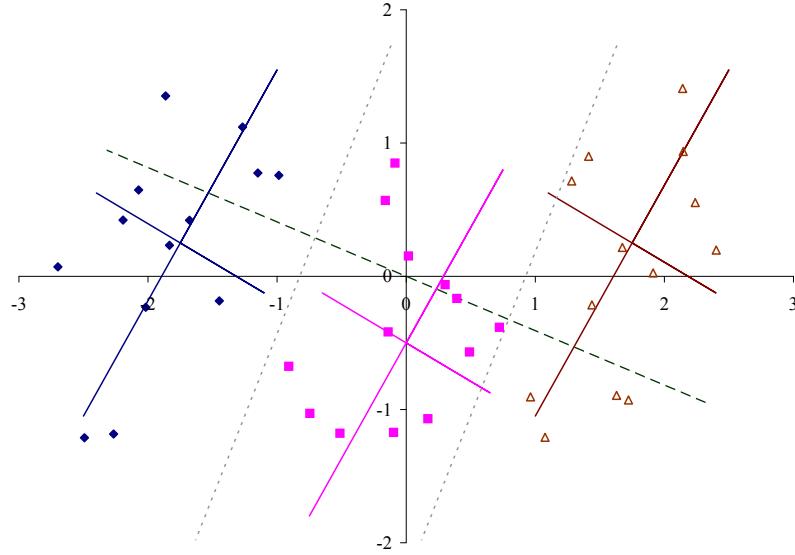


Figure 14.16 Simple example of Fisher linear discriminant analysis. The samples come from three different classes, shown in different colors along with their principal axes, which are scaled to $2\sigma_i$. (The intersections of the tilted axes are the class means \mathbf{m}_k .) The dashed line is the (dominant) Fisher linear discriminant direction and the dotted lines are the linear discriminants between the classes. Note how the discriminant direction is a blend between the principal directions of the between-class and within-class scatter matrices.

where \mathbf{m}_k is the mean of class k and \mathbf{S}_k is its within-class scatter matrix.¹¹ Similarly, we can compute the *between-class* scatter as

$$\mathbf{S}_B = \sum_{k=0}^{K-1} N_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T, \quad (14.18)$$

where N_k are the number of exemplars in each class and \mathbf{m} is the overall mean. For the three distributions shown in Figure 14.16, we have

$$\mathbf{S}_W = 3N \begin{bmatrix} 0.246 & 0.183 \\ 0.183 & 0.457 \end{bmatrix} \quad \text{and} \quad \mathbf{S}_B = N \begin{bmatrix} 6.125 & 0 \\ 0 & 0.375 \end{bmatrix}, \quad (14.19)$$

where $N = N_k = 13$ is the number of samples in each class.

To compute the most discriminating direction, *Fisher's linear discriminant* (FLD) (Belhumeur, Hespanha, and Kriegman 1997; Hastie, Tibshirani, and Friedman 2001; Bishop 2006), which is also known as *linear discriminant analysis* (LDA), selects the direction \mathbf{u} that results in the largest ratio between the projected between-class and within-class variations

$$\mathbf{u}^* = \arg \max_{\mathbf{u}} \frac{\mathbf{u}^T \mathbf{S}_B \mathbf{u}}{\mathbf{u}^T \mathbf{S}_W \mathbf{u}}, \quad (14.20)$$

¹¹ To be consistent with Belhumeur, Hespanha, and Kriegman (1997), we use \mathbf{S}_W and \mathbf{S}_B to denote the scatter matrices, even though we use \mathbf{C} elsewhere (14.9).

which is equivalent to finding the eigenvector corresponding to the largest eigenvalue of the generalized eigenvalue problem

$$\mathbf{S}_B \mathbf{u} = \lambda \mathbf{S}_W \mathbf{u} \quad \text{or} \quad \lambda \mathbf{u} = \mathbf{S}_W^{-1} \mathbf{S}_B \mathbf{u}. \quad (14.21)$$

For the problem shown in Figure 14.16,

$$\mathbf{S}_W^{-1} \mathbf{S}_B = \begin{bmatrix} 11.796 & -0.289 \\ -4.715 & 0.3889 \end{bmatrix} \quad \text{and} \quad \mathbf{u} = \begin{bmatrix} 0.926 \\ -0.379 \end{bmatrix} \quad (14.22)$$

As you can see, using this direction results in a better separation between the classes than using the dominant PCA direction, which is the horizontal axis. In their paper, Belhumeur, Hespanha, and Kriegman (1997) show that Fisherfaces significantly outperform the original eigenfaces algorithm, especially when faces have large amounts of illumination variation, as in Figure 14.15.

An alternative for modeling within-class (intrapersonal) and between-class (extrapersonal) variations is to model each distribution separately and then use Bayesian techniques to find the closest exemplar (Moghaddam, Jebara, and Pentland 2000). Instead of computing the mean for each class and then the within-class and between-class distributions, consider evaluating the difference images

$$\Delta_{ij} = \mathbf{x}_i - \mathbf{x}_j \quad (14.23)$$

between all pairs of training images ($\mathbf{x}_i, \mathbf{x}_j$). The differences between pairs that are in the same class (the same person) are used to estimate the intrapersonal covariance matrix Σ_I , while differences between different people are used to estimate the extrapersonal covariance Σ_E .¹² The principal components (eigenfaces) corresponding to these two classes are shown in Figure 14.17.

At recognition time, we can compute the distance Δ_i between a new face \mathbf{x} and a stored training image \mathbf{x}_i and evaluate its intrapersonal likelihood as

$$p_I(\Delta_i) = p_{\mathcal{N}}(\Delta_i; \Sigma_I) = \frac{1}{|2\pi\Sigma_I|^{1/2}} \exp -\|\Delta_i\|_{\Sigma_I^{-1}}^2, \quad (14.24)$$

where $p_{\mathcal{N}}$ is a normal (Gaussian) distribution with covariance Σ_I and

$$|2\pi\Sigma_I|^{1/2} = (2\pi)^{M/2} \prod_{j=1}^M \lambda_j^{1/2} \quad (14.25)$$

is its volume. The Mahalanobis distance

$$\|\Delta_i\|_{\Sigma_I^{-1}}^2 = \Delta_i^T \Sigma_I^{-1} \Delta_i = \|\mathbf{a}^I - \mathbf{a}_i^I\|^2 \quad (14.26)$$

can be computed more efficiently by first projecting the new image \mathbf{x} into the whitened intrapersonal face space (14.15)

$$\mathbf{a}^I = \hat{\mathbf{U}}^I \mathbf{x} \quad (14.27)$$

and then computing a Euclidean distance to the training image vector \mathbf{a}_i^I , which can be pre-computed offline. The extrapersonal likelihood $p_E(\Delta_i)$ can be computed in a similar fashion.

¹² Note that the difference distributions are zero mean because for every Δ_{ij} there corresponds a negative Δ_{ji} .

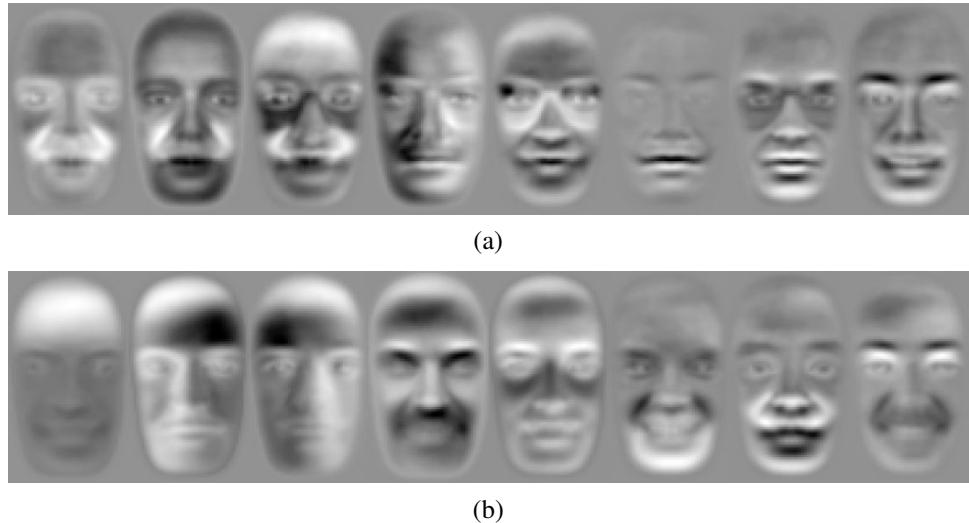


Figure 14.17 “Dual” eigenfaces (Moghaddam, Jebara, and Pentland 2000) © 2000 Elsevier: (a) intrapersonal and (b) extrapersonal.

Once the intrapersonal and extrapersonal likelihoods have been computed, we can compute the Bayesian likelihood of a new image x matching a training image x_i as

$$p(\Delta_i) = \frac{p_I(\Delta_i)l_I}{p_I(\Delta_i)l_I + p_E(\Delta_i)l_E}, \quad (14.28)$$

where l_I and l_E are the prior probabilities of two images being in the same or in different classes (Moghaddam, Jebara, and Pentland 2000). A simpler approach, which does not require the evaluation of extrapersonal probabilities, is to simply choose the training image with the highest likelihood $p_I(\Delta_i)$. In this case, nearest neighbor search techniques in the space spanned by the precomputed $\{\mathbf{a}_i^I\}$ vectors could be used to speed up finding the best match.¹³

Another way to improve the performance of eigenface-based approaches is to break up the image into separate regions such as the eyes, nose, and mouth (Figure 14.18) and to match each of these *modular eigenspaces* independently (Moghaddam and Pentland 1997; Heisele, Ho, Wu *et al.* 2003; Heisele, Serre, and Poggio 2007). The advantage of such a modular approach is that it can tolerate a wider range of viewpoints, because each part can move relative to the others. It also supports a larger variety of combinations, e.g., we can model one person as having a narrow nose and bushy eyebrows, without requiring the eigenfaces to span all possible combinations of nose, mouth, and eyebrows. (If you remember the cardboard children’s books where you can select different top and bottom faces, or Mr. Potato Head, you get the idea.)

Another approach to dealing with large variability in appearance is to create *view-based* (view-specific) eigenspaces, as shown in Figure 14.19 (Moghaddam and Pentland 1997). We can think of these view-based eigenspaces as local descriptors that select different axes depending on which part of the face space you are in. Note that such approaches, however,

¹³ Note that while the covariance matrices Σ_I and Σ_E are computed by looking at differences between *all* pairs of images, the run-time evaluation selects the *nearest* image to determine the facial identity. Whether this is statistically correct is explored in Exercise 14.4.

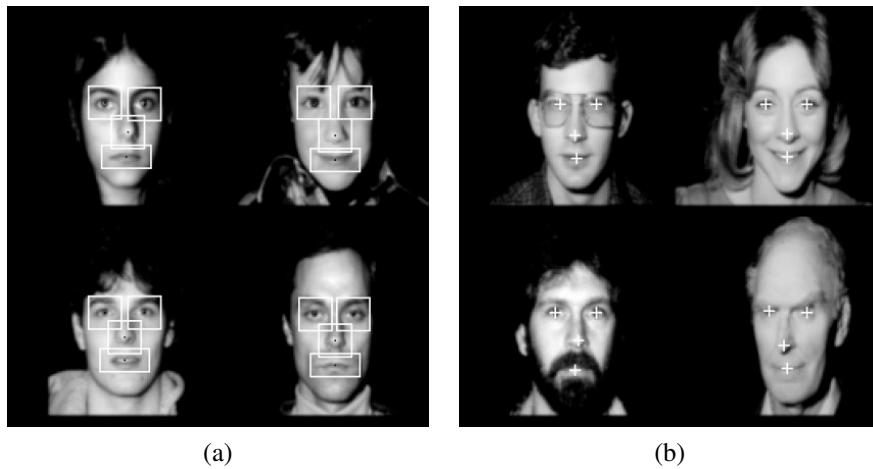


Figure 14.18 Modular eigenspace for face recognition (Moghaddam and Pentland 1997) © 1997 IEEE. (a) By detecting separate features in the faces (eyes, nose, mouth), separate eigenspaces can be estimated for each one. (b) The relative positions of each feature can be detected at recognition time, thus allowing for more flexibility in viewpoint and expression.

potentially require large amounts of training data, i.e., pictures of every person in every possible pose or expression. This is in contrast to the shape and appearance models we study in Section 14.2.2, which can learn deformations across all individuals.

It is also possible to generalize the bilinear factorization implicit in PCA and SVD approaches to multilinear (tensor) formulations that can model several interacting factors simultaneously (Vasilescu and Terzopoulos 2007). These ideas are related to currently active topics in machine learning such as *subspace learning* (Cai, He, Hu *et al.* 2007), *local distance functions* (Frome, Singer, Sha *et al.* 2007), and *metric learning* (Ramanan and Baker 2009). Learning approaches play an increasingly important role in face recognition, e.g., in the work of Sivic, Everingham, and Zisserman (2009) and Guillaumin, Verbeek, and Schmid (2009).

14.2.2 Active appearance and 3D shape models

The need to use modular or view-based eigenspaces for face recognition is symptomatic of a more general observation, i.e., that facial appearance and identifiability depend as much on *shape* as they do on color or texture (which is what eigenfaces capture). Furthermore, when dealing with 3D head rotations, the *pose* of a person's head should be discounted when performing recognition.

In fact, the earliest face recognition systems, such as those by Fischler and Elschlager (1973), Kanade (1977), and Yuille (1991), found distinctive feature points on facial images and performed recognition on the basis of their relative positions or distances. Newer techniques such as *local feature analysis* (Penev and Atick 1996) and *elastic bunch graph matching* (Wiskott, Fellous, Krüger *et al.* 1997) combine local filter responses (jets) at distinctive feature locations together with shape models to perform recognition.

A visually compelling example of why both shape and texture are important is the work of Rowland and Perrett (1995), who manually traced the contours of facial features and then

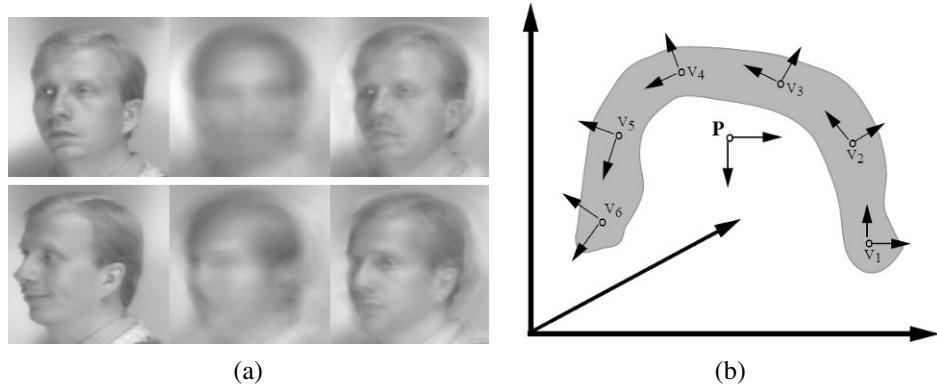


Figure 14.19 View-based eigenspace (Moghaddam and Pentland 1997) © 1997 IEEE. (a) Comparison between a regular (parametric) eigenspace reconstruction (middle column) and a view-based eigenspace reconstruction (right column) corresponding to the input image (left column). The top row is from a training image, the bottom row is from the test set. (b) A schematic representation of the two approaches, showing how each view computes its own local basis representation.

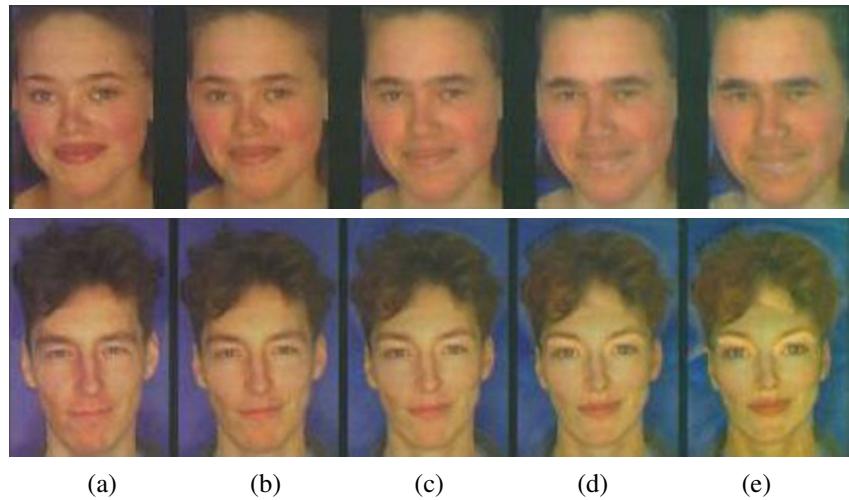


Figure 14.20 Manipulating facial appearance through shape and color (Rowland and Perrett 1995) © 1995 IEEE. By adding or subtracting gender-specific shape and color characteristics to (b) an input image, different amounts of gender variation can be induced. The amounts added (from the mean) are: (a) +50% (gender enhancement), (c) -50% (near "androgyny"), (d) -100% (gender switched), and (e) -150% (opposite gender attributes enhanced).

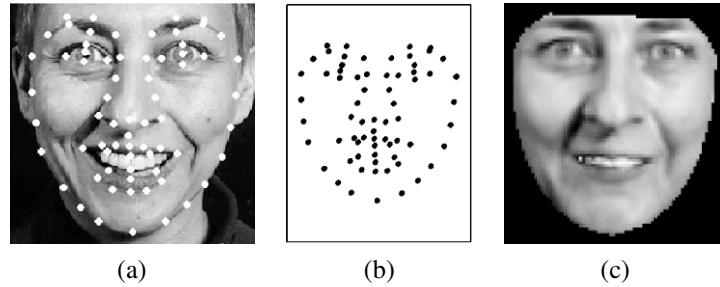


Figure 14.21 Active Appearance Models (Cootes, Edwards, and Taylor 2001) © 2001 IEEE: (a) input image with registered feature points; (b) the feature points (shape vector s); (c) the shape-free appearance image (texture vector t).

used these contours to normalize (warp) each image to a canonical shape. After analyzing both the shape and color images for deviations from the mean, they were able to associate certain shape and color deformations with personal characteristics such as age and gender (Figure 14.20). Their work demonstrates that both shape and color have an important influence on the perception of such characteristics.

Around the same time, researchers in computer vision were beginning to use simultaneous shape deformations and texture interpolation to model the variability in facial appearance caused by identity or expression (Beymer 1996; Vetter and Poggio 1997), developing techniques such as Active Shape Models (Lanitis, Taylor, and Cootes 1997), 3D Morphable Models (Blanz and Vetter 1999), and Elastic Bunch Graph Matching (Wiskott, Fellous, Krüger *et al.* 1997).¹⁴

Of all these techniques, the *active appearance models* (AAMs) of Cootes, Edwards, and Taylor (2001) are among the most widely used for face recognition and tracking. Like other shape and texture models, an AAM models both the variation in the shape of an image s , which is normally encoded by the location of key feature points on the image (Figure 14.21b), as well as the variation in texture t , which is normalized to a canonical shape before being analyzed (Figure 14.21c).¹⁵

Both shape and texture are represented as deviations from a mean shape \bar{s} and texture \bar{t} ,

$$s = \bar{s} + U_s a \quad (14.29)$$

$$t = \bar{t} + U_t a, \quad (14.30)$$

where the eigenvectors in U_s and U_t have been pre-scaled (whitened) so that unit vectors in a represent one standard deviation of variation observed in the training data. In addition to these principal deformations, the shape parameters are transformed by a global similarity to match the location, size, and orientation of a given face. Similarly, the texture image contains a scale and offset to best match novel illumination conditions.

As you can see, the same appearance parameters a in (14.29–14.30) simultaneously control both the shape and texture deformations from the mean, which makes sense if we believe

¹⁴ We have already seen the application of PCA to 3D head and face modeling and animation in Section 12.6.3.

¹⁵ When only the shape variation is being captured, such models are called *active shape models* (ASMs) (Cootes, Cooper, Taylor *et al.* 1995; Davies, Twining, and Taylor 2008). These were already discussed in Section 5.1.1 (5.13–5.17).

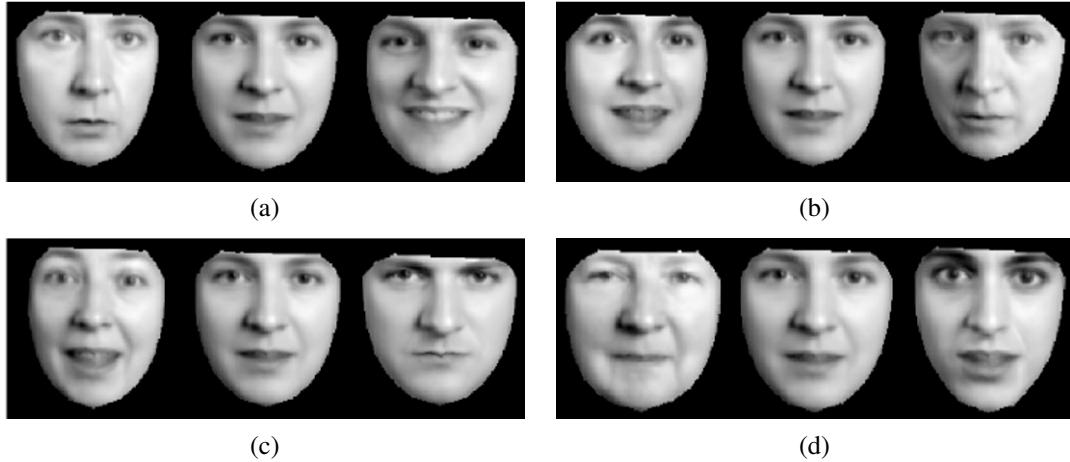


Figure 14.22 Principal modes of variation in active appearance models (Cootes, Edwards, and Taylor 2001) © 2001 IEEE. The four images show the effects of simultaneously changing the first four modes of variation in both shape and texture by $\pm\sigma$ from the mean. You can clearly see how the shape of the face and the shading are simultaneously affected.

them to be correlated. Figure 14.22 shows how moving three standard deviations along each of the first four principal directions ends up changing several correlated factors in a person’s appearance, including expression, gender, age, and identity.

In order to fit an active appearance model to a novel image, Cootes, Edwards, and Taylor (2001) pre-compute a set of “difference decomposition” images, using an approach related to other fast techniques for incremental tracking, such as those we discussed in Sections 4.1.4, 8.1.3, and 8.2 (Gleicher 1997; Hager and Belhumeur 1998), which often *learn* a discriminative mapping between matching errors and incremental displacements (Avidan 2001; Jurie and Dhome 2002; Liu, Chen, and Kumar 2003; Sclaroff and Isidoro 2003; Romdhani and Vetter 2003; Williams, Blake, and Cipolla 2003).

In more detail, Cootes, Edwards, and Taylor (2001) compute the derivatives of a set of training images with respect to each of the parameters in \mathbf{a} using finite differences and then compute a set of *displacement weight* images

$$\mathbf{W} = \left[\frac{\partial \mathbf{x}^T}{\partial \mathbf{a}} \quad \frac{\partial \mathbf{x}}{\partial \mathbf{a}} \right]^{-1} \frac{\partial \mathbf{x}^T}{\partial \mathbf{a}}, \quad (14.31)$$

which can be multiplied by the current error residual to produce an update step in the parameters, $\delta \mathbf{a} = -\mathbf{W} \mathbf{r}$. Matthews and Baker (2004) use their *inverse compositional method*, which they first developed for parametric optical flow (8.64–8.65), to further speed up active appearance model fitting and tracking. Examples of AAMs being fitted to two input images are shown in Figure 14.23.

Although active appearance models are primarily designed to accurately capture the variability in appearance and deformation that are characteristic of faces, they can be adapted to face recognition by computing an identity subspace that separates variation in identity from other sources of variability such as lighting, pose, and expression (Costen, Cootes, Edwards *et al.* 1999). The basic idea, which is modeled after similar work in eigenfaces (Belhumeur,



Figure 14.23 Multiresolution model fitting (search) in active appearance models (Cootes, Edwards, and Taylor 2001) © 2001 IEEE. The columns show the initial model, the results after 3, 8, and 11 iterations, and the final convergence. The rightmost column shows the input image.

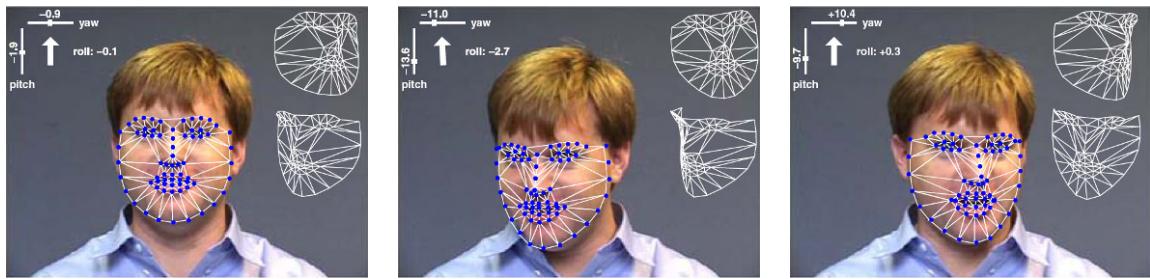


Figure 14.24 Head tracking with 3D AAMs (Matthews, Xiao, and Baker 2007) © 2007 Springer. Each image shows a video frame along with the estimate yaw, pitch, and roll parameters and the fitted 3D deformable mesh.

Hespanha, and Kriegman 1997; Moghaddam, Jebara, and Pentland 2000), is to compute separate statistics for intrapersonal and extrapersonal variation and then find discriminating directions in these subspaces. While AAMs have sometimes been used directly for recognition (Blanz and Vetter 2003), their main use in the context of recognition is to align faces into a canonical pose (Liang, Xiao, Wen *et al.* 2008) so that more traditional methods of face recognition (Penev and Atick 1996; Wiskott, Fellous, Krüger *et al.* 1997; Ahonen, Hadid, and Pietikäinen 2006; Zhao and Pietikäinen 2007; Cao, Yin, Tang *et al.* 2010) can be used. AAMs (or, actually, their simpler version, Active Shape Models (ASMs)) can also be used to align face images to perform automated morphing (Zanella and Fuentes 2004).

Active appearance models continue to be an active research area, with enhancements to deal with illumination and viewpoint variation (Gross, Baker, Matthews *et al.* 2005) as well as occlusions (Gross, Matthews, and Baker 2006). One of the most significant extensions is to construct 3D models of shape (Matthews, Xiao, and Baker 2007), which are much better at capturing and explaining the full variability of facial appearance across wide changes in pose.

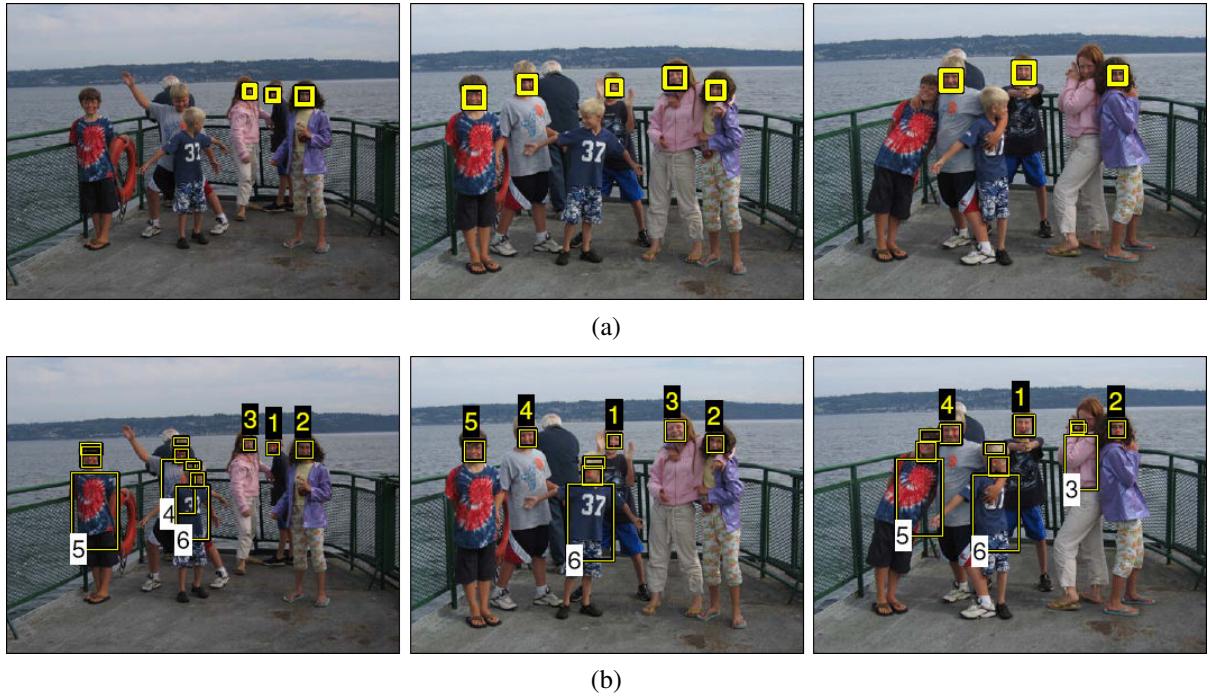


Figure 14.25 Person detection and re-recognition using a combined face, hair, and torso model (Sivic, Zitnick, and Szeliski 2006) © 2006 Springer. (a) Using face detection alone, several of the heads are missed. (b) The combined face and clothing model successfully re-finds all the people.

Such models can be constructed either from monocular video sequences (Matthews, Xiao, and Baker 2007), as shown in Figure 14.24, or from multi-view video sequences (Ramnath, Koterba, Xiao *et al.* 2008), which provide even greater reliability and accuracy in reconstruction and tracking. (For a recent review of progress in head pose estimation, please see the survey paper by Murphy-Chutorian and Trivedi (2009).)

14.2.3 Application: Personal photo collections

In addition to digital cameras automatically finding faces to aid in auto-focusing and video cameras finding faces in video conferencing to center on the speaker (either mechanically or digitally), face detection has found its way into most consumer-level photo organization packages, such as iPhoto, Picasa, and Windows Live Photo Gallery. Finding faces and allowing users to tag them makes it easier to find photos of selected people at a later date or to automatically share them with friends. In fact, the ability to tag friends in photos is one of the more popular features on Facebook.

Sometimes, however, faces can be hard to find and recognize, especially if they are small, turned away from the camera, or otherwise occluded. In such cases, combining face recognition with person detection and clothes recognition can be very effective, as illustrated in Figure 14.25 (Sivic, Zitnick, and Szeliski 2006). Combining person recognition with other kinds of context, such as location recognition (Section 14.3.3) or activity or event recognition, can also help boost performance (Lin, Kapoor, Hua *et al.* 2010).



Figure 14.26 Recognizing objects in a cluttered scene (Lowe 2004) © 2004 Springer. Two of the training images in the database are shown on the left. They are matched to the cluttered scene in the middle using SIFT features, shown as small squares in the right image. The affine warp of each recognized database image onto the scene is shown as a larger parallelogram in the right image.

14.3 Instance recognition

General object recognition falls into two broad categories, namely *instance recognition* and *class recognition*. The former involves re-recognizing a known 2D or 3D rigid object, potentially being viewed from a novel viewpoint, against a cluttered background, and with partial occlusions. The latter, which is also known as *category-level* or *generic* object recognition (Ponce, Hebert, Schmid *et al.* 2006), is the much more challenging problem of recognizing any instance of a particular general class such as “cat”, “car”, or “bicycle”.

Over the years, many different algorithms have been developed for instance recognition. Mundy (2006) surveys earlier approaches, which focused on extracting lines, contours, or 3D surfaces from images and matching them to known 3D object models. Another popular approach was to acquire images from a large set of viewpoints and illuminations and to represent them using an eigenspace decomposition (Murase and Nayar 1995). More recent approaches (Lowe 2004; Rothganger, Lazebnik, Schmid *et al.* 2006; Ferrari, Tuytelaars, and Van Gool 2006b; Gordon and Lowe 2006; Obdržálek and Matas 2006; Sivic and Zisserman 2009) tend to use viewpoint-invariant 2D features, such as those we saw in Section 4.1.2. After extracting informative sparse 2D features from both the new image and the images in the database, image features are matched against the object database, using one of the sparse feature matching strategies described in Section 4.1.3. Whenever a sufficient number of matches have been found, they are verified by finding a geometric transformation that aligns the two sets of features (Figure 14.26).

Below, we describe some of the techniques that have been proposed for representing the geometric relationships between such features (Section 14.3.1). We also discuss how to make the feature matching process more efficient using ideas from text and information retrieval (Section 14.3.2).

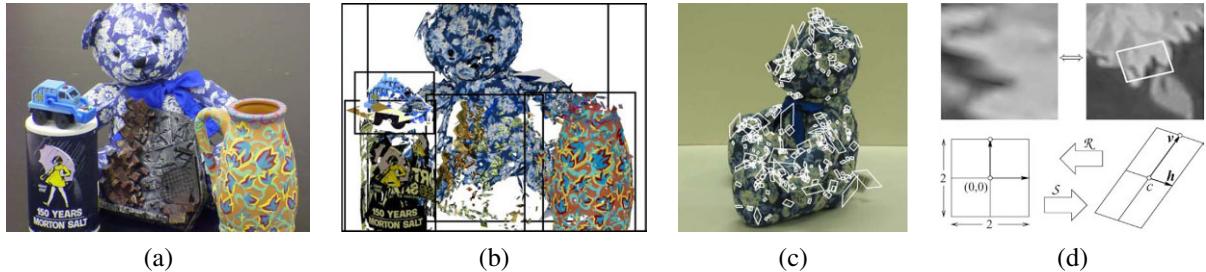


Figure 14.27 3D object recognition with affine regions (Rothganger, Lazebnik, Schmid *et al.* 2006) © 2006 Springer: (a) sample input image; (b) five of the recognized (reprojected) objects along with their bounding boxes; (c) a few of the local affine regions; (d) local affine region (patch) reprojected into a canonical (square) frame, along with its geometric affine transformations.

14.3.1 Geometric alignment

To recognize one or more instances of some known objects, such as those shown in the left column of Figure 14.26, the recognition system first extracts a set of interest points in each database image and stores the associated descriptors (and original positions) in an indexing structure such as a search tree (Section 4.1.3). At recognition time, features are extracted from the new image and compared against the stored object features. Whenever a sufficient number of matching features (say, three or more) are found for a given object, the system then invokes a *match verification* stage, whose job is to determine whether the spatial arrangement of matching features is consistent with those in the database image.

Because images can be highly cluttered and similar features may belong to several objects, the original set of feature matches can have a large number of outliers. For this reason, Lowe (2004) suggests using a Hough transform (Section 4.3.2) to accumulate votes for likely geometric transformations. In his system, he uses an affine transformation between the database object and the collection of scene features, which works well for objects that are mostly planar, or where at least several corresponding features share a quasi-planar geometry.¹⁶

Since SIFT features carry with them their own location, scale, and orientation, Lowe uses a four-dimensional similarity transformation as the original Hough binning structure, i.e., each bin denotes a particular location for the object center, scale, and in-plane rotation. Each matching feature votes for the nearest 2^4 bins and peaks in the transform are then selected for a more careful affine motion fit. Figure 14.26 (right image) shows three instances of the two objects on the left that were recognized by the system. Obdržálek and Matas (2006) generalize Lowe’s approach to use feature descriptors with full local affine frames and evaluate their approach on a number of object recognition databases.

Another system that uses local affine frames is the one developed by Rothganger, Lazebnik, Schmid *et al.* (2006). In their system, the affine region detector of Mikolajczyk and Schmid (2004) is used to rectify local image patches (Figure 14.27d), from which both a SIFT descriptor and a 10×10 UV color histogram are computed and used for matching and recognition. Corresponding patches in different views of the same object, along with

¹⁶ When a larger number of features is available, a full fundamental matrix can be used (Brown and Lowe 2002; Gordon and Lowe 2006). When image stitching is being performed (Brown and Lowe 2007), the motion models discussed in Section 9.1 can be used instead.

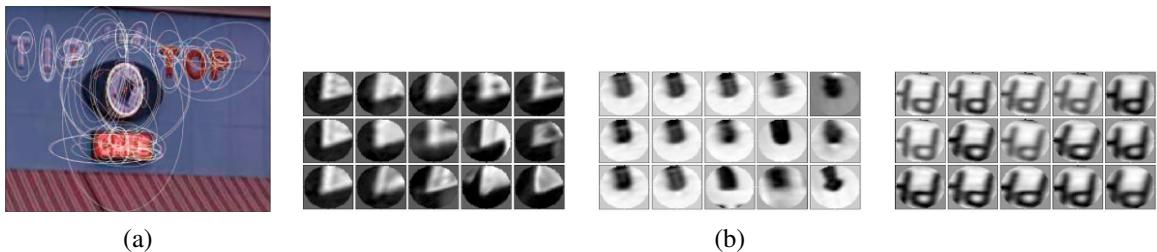


Figure 14.28 Visual words obtained from elliptical normalized affine regions (Sivic and Zisserman 2009) © 2009 IEEE. (a) Affine covariant regions are extracted from each frame and clustered into visual words using k-means clustering on SIFT descriptors with a learned Mahalanobis distance. (b) The central patch in each grid shows the query and the surrounding patches show the nearest neighbors.

their local affine deformations, are used to compute a 3D affine model for the object using an extension of the factorization algorithm of Section 7.3, which can then be upgraded to a Euclidean reconstruction (Tomasi and Kanade 1992).

At recognition time, local Euclidean neighborhood constraints are used to filter potential matches, in a manner analogous to the affine geometric constraints used by Lowe (2004) and Obdržálek and Matas (2006). Figure 14.27 shows the results of recognizing five objects in a cluttered scene using this approach.

While feature-based approaches are normally used to detect and localize known objects in scenes, it is also possible to get pixel-level segmentations of the scene based on such matches. Ferrari, Tuytelaars, and Van Gool (2006b) describe such a system for simultaneously recognizing objects and segmenting scenes, while Kannala, Rahtu, Brandt *et al.* (2008) extend this approach to non-rigid deformations. Section 14.4.3 re-visits this topic of joint recognition and segmentation in the context of generic class (category) recognition.

14.3.2 Large databases

As the number of objects in the database starts to grow large (say, millions of objects or video frames being searched), the time it takes to match a new image against each database image can become prohibitive. Instead of comparing the images one at a time, techniques are needed to quickly narrow down the search to a few likely images, which can then be compared using a more detailed and conservative verification stage.

The problem of quickly finding partial matches between documents is one of the central problems in *information retrieval* (IR) (Baeza-Yates and Ribeiro-Neto 1999; Manning, Raghavan, and Schütze 2008). The basic approach in fast document retrieval algorithms is to pre-compute an *inverted index* between individual words and the documents (or Web pages or news stories) where they occur. More precisely, the *frequency* of occurrence of particular words in a document is used to quickly find documents that match a particular query.

Sivic and Zisserman (2009) were the first to adapt IR techniques to visual search. In their Video Google system, affine invariant features are first detected in all the video frames they are indexing using both *shape adapted* regions around Harris feature points (Schaffalitzky and Zisserman 2002; Mikolajczyk and Schmid 2004) and maximally stable extremal regions (Matas, Chum, Urban *et al.* 2004), (Section 4.1.1), as shown in Figure 14.28a. Next, 128-



Figure 14.29 Matching based on visual words (Sivic and Zisserman 2009) © 2009 IEEE. (a) Features in the query region on the left are matched to corresponding features in a highly ranked video frame. (b) Results after removing the stop words and filtering the results using spatial consistency.

dimensional SIFT descriptors are computed from each normalized region (i.e., the patches shown in Figure 14.28b). Then, an average covariance matrix for these descriptors is estimated by accumulating statistics for features tracked from frame to frame. The feature descriptor covariance Σ is then used to define a Mahalanobis distance between feature descriptors,

$$d(\mathbf{x}_0, \mathbf{x}_1) = \|\mathbf{x}_0 - \mathbf{x}_1\|_{\Sigma^{-1}} = \sqrt{(\mathbf{x}_0 - \mathbf{x}_1)^T \Sigma^{-1} (\mathbf{x}_0 - \mathbf{x}_1)}. \quad (14.32)$$

In practice, feature descriptors are *whitened* by pre-multiplying them by $\Sigma^{-1/2}$ so that Euclidean distances can be used.¹⁷

In order to apply fast information retrieval techniques to images, the high-dimensional feature descriptors that occur in each image must first be mapped into discrete *visual words*. Sivic and Zisserman (2003) perform this mapping using k-means clustering, while some of newer methods discussed below (Nistér and Stewénius 2006; Philbin, Chum, Isard *et al.* 2007) use alternative techniques, such as vocabulary trees or randomized forests. To keep the clustering time manageable, only a few hundred video frames are used to learn the cluster centers, which still involves estimating several thousand clusters from about 300,000 descriptors. At visual query time, each feature in a new query region (e.g., Figure 14.28a, which is a cropped region from a larger video frame) is mapped to its corresponding visual word. To keep very common patterns from contaminating the results, a *stop list* of the most common visual words is created and such words are dropped from further consideration.

Once a query image or region has been mapped into its constituent visual words, likely matching images or video frames must then be retrieved from the database. Information retrieval systems do this by matching word distributions (*term frequencies*) n_{id}/n_d between the query and target documents, where n_{id} is how many times word i occurs in document d , and n_d is the total number of words in document d . In order to downweight words that occur frequently and to focus the search on rarer (and hence, more informative) terms, an *inverse document frequency* weighting $\log N/N_i$ is applied, where N_i is the number of documents containing word i , and N is the total number of documents in the database. The combination of these two factors results in the *term frequency-inverse document frequency* (*tf-idf*) measure,

$$t_i = \frac{n_{id}}{n_d} \log \frac{N}{N_i}. \quad (14.33)$$

¹⁷ Note that the computation of feature covariances from matched feature points is much more sensible than simply performing a PCA on the descriptor space (Winder and Brown 2007). This corresponds roughly to the *within-class* scatter matrix (14.17) we studied in Section 14.2.1.

1. **Vocabulary construction (off-line)**
 - (a) Extract affine covariant regions from each database image.
 - (b) Compute descriptors and optionally whiten them to make Euclidean distances meaningful (Sivic and Zisserman 2009).
 - (c) Cluster the descriptors into visual words, either using k-means (Sivic and Zisserman 2009), hierarchical clustering (Nistér and Stewénius 2006), or randomized k-d trees (Philbin, Chum, Isard *et al.* 2007).
 - (d) Decide which words are too common and put them in the stop list.
2. **Database construction (off-line)**
 - (a) Compute term frequencies for the visual word in each image, document frequencies for each word, and normalized *tf-idf* vectors for each document.
 - (b) Compute inverted indices from visual words to images (with word counts).
3. **Image retrieval (on-line)**
 - (a) Extract regions, descriptors, and visual words, and compute a *tf-idf* vector for the query image or region.
 - (b) Retrieve the top image candidates, either by exhaustively comparing sparse *tf-idf* vectors (Sivic and Zisserman 2009) or by using inverted indices to examine only a subset of the images (Nistér and Stewénius 2006).
 - (c) Optionally re-rank or verify all the candidate matches, using either spatial consistency (Sivic and Zisserman 2009) or an affine (or simpler) transformation model (Philbin, Chum, Isard *et al.* 2007).
 - (d) Optionally expand the answer set by re-submitting highly ranked matches as new queries (Chum, Philbin, Sivic *et al.* 2007).

Algorithm 14.2 Image retrieval using visual words (Sivic and Zisserman 2009; Nistér and Stewénius 2006; Philbin, Chum, Isard *et al.* 2007; Chum, Philbin, Sivic *et al.* 2007; Philbin, Chum, Sivic *et al.* 2008).

At match time, each document (or query region) is represented by its *tf-idf* vector,

$$\mathbf{t} = (t_1, \dots, t_i, \dots, t_m). \quad (14.34)$$

The similarity between two documents is measured by the dot product between their corresponding normalized vectors $\hat{\mathbf{t}} = \mathbf{t}/\|\mathbf{t}\|$, which means that their dissimilarity is proportional to their Euclidean distance. In their journal paper, [Sivic and Zisserman \(2009\)](#) compare this simple metric to a dozen other metrics and conclude that it performs just about as well as more complicated metrics. Because the number of non-zero t_i terms in a typical query or document is small ($M \approx 200$) compared to the number of visual words ($V \approx 20,000$), the distance between pairs of (sparse) *tf-idf* vectors can be computed quite quickly.

After retrieving the top $N_s = 500$ documents based on word frequencies, [Sivic and Zisserman \(2009\)](#) re-rank these results using spatial consistency. This step involves taking every matching feature and counting the number of $k = 15$ nearest adjacent features that also match between the two documents. (This latter process is accelerated using inverted files, which we discuss in more detail below.) As shown in Figure 14.29, this step helps remove spurious false positive matches and produces a better estimate of which frames and regions in the video are actually true matches. Algorithm 14.2 summarizes the processing steps involved in image retrieval using visual words.

While this approach works well for tens of thousand of visual words and thousands of keyframes, as the size of the database continues to increase, both the time to quantize each feature and to find potential matching frames or images can become prohibitive. [Nistér and Stewénius \(2006\)](#) address this problem by constructing a hierarchical *vocabulary tree*, where feature vectors are hierarchically clustered into a k -way tree of prototypes. (This technique is also known as *tree-structured vector quantization* ([Gersho and Gray 1991](#)).) At both database construction time and query time, each descriptor vector is compared to several prototypes at a given level in the vocabulary tree and the branch with the closest prototype is selected for further refinement (Figure 14.30). In this way, vocabularies with millions (10^6) of words can be supported, which enables individual words to be far more discriminative, while only requiring $10 \cdot 6$ comparisons for quantizing each descriptor.

At query time, each node in the vocabulary tree keeps its own inverted file index, so that features that match a particular node in the tree can be rapidly mapped to potential matching images. (Interior leaf nodes just use the inverted indices of their corresponding leaf-node descendants.) To score a particular query *tf-idf* vector \mathbf{t}_q against all document vectors $\{\mathbf{t}_j\}$ using an L_p metric,¹⁸ the non-zero t_{iq} entries in \mathbf{t}_q are used to fetch corresponding non-zero t_{ij} entries, and the L_p norm is efficiently computed as

$$\|\mathbf{t}_q - \mathbf{t}_j\|_p^p = 2 + \sum_{i|t_{iq}>0 \wedge t_{ij}>0} (|t_{iq} - t_{ij}|^p - |t_{iq}|^p - |t_{ij}|^p). \quad (14.35)$$

In order to mitigate quantization errors due to noise in the descriptor vectors, [Nistér and Stewénius \(2006\)](#) not only score leaf nodes in the vocabulary tree (corresponding to visual words), but also score interior nodes in the tree, which correspond to clusters of similar visual words.

Because of the high efficiency in both quantizing and scoring features, their vocabulary-tree-based recognition system is able to process incoming images in real time against a

¹⁸ In their actual implementation, [Nistér and Stewénius \(2006\)](#) use an L_1 metric.

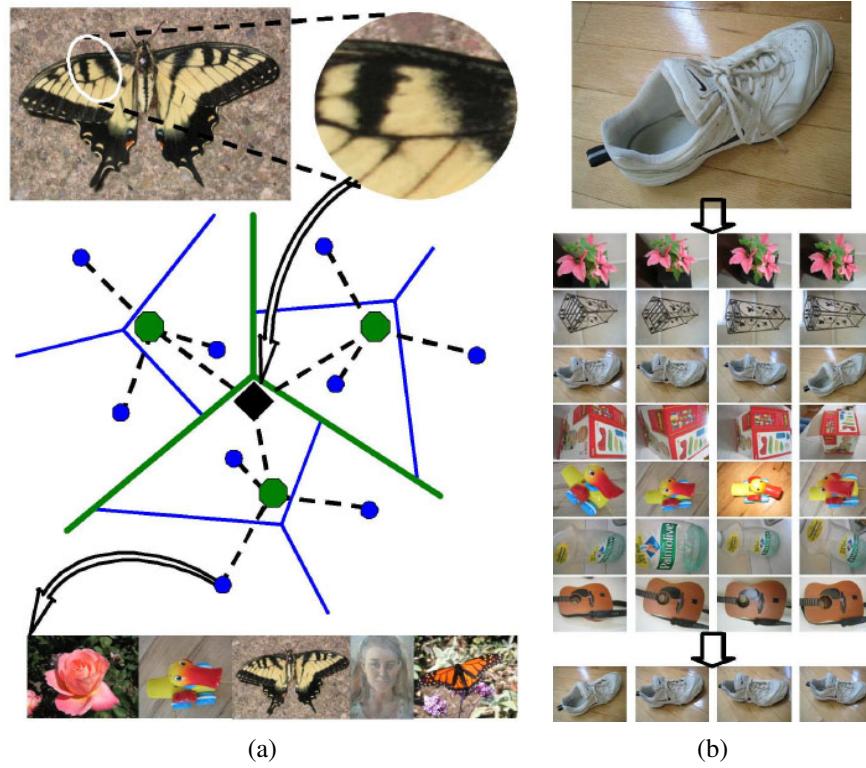


Figure 14.30 Scalable recognition using a vocabulary tree (Nistér and Stewénius 2006) © 2006 IEEE. (a) Each MSER elliptical region is converted into a SIFT descriptor, which is then quantized by comparing it hierarchically to some prototype descriptors in a vocabulary tree. Each leaf node stores its own inverted index (sparse list of non-zero $tf\text{-}idf$ counts) into images that contain that feature. (b) A recognition result, showing a query image (top row) being indexed into a database of 6000 test images and correctly finding the corresponding four images.

database of 40,000 CD covers and at 1Hz when matching a database of one million frames taken from six feature-length movies. Figure 14.30b shows some typical images from the database of objects taken under varying viewpoints and illumination that was used to train and test the vocabulary tree recognition system.

The state of the art in instance recognition continues to improve rapidly. Philbin, Chum, Isard *et al.* (2007) have shown that randomized forest of k-d trees perform better than vocabulary trees on a large location recognition task (Figure 14.31). They also compare the effects of using different 2D motion models (Section 2.1.2) in the verification stage. In follow-on work, Chum, Philbin, Sivic *et al.* (2007) apply another idea from information retrieval, namely *query expansion*, which involves re-submitting top-ranked images from the initial query as additional queries to generate additional candidate results, to further improve recognition rates for difficult (occluded or oblique) examples. Philbin, Chum, Sivic *et al.* (2008) show how to mitigate quantization problems in visual words selection using *soft assignment*, where each feature descriptor is mapped to a number of visual words based on its distance from the cluster prototypes. The soft weights derived from these distances are used, in turn, to weight the counts used in the $tf\text{-}idf$ vectors and to retrieve additional images for later verification.

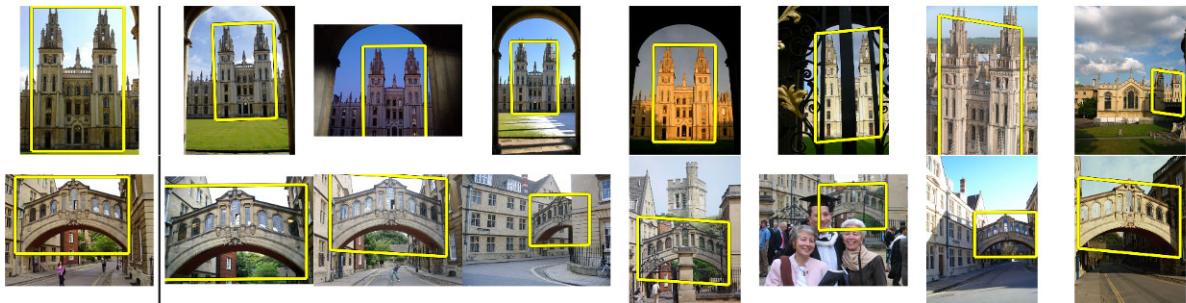


Figure 14.31 Location or building recognition using randomized trees (Philbin, Chum, Isard *et al.* 2007) © 2007 IEEE. The left image is the query, the other images are the highest-ranked results.

Taken together, these recent advances hold the promise of extending current instance recognition algorithms to performing Web-scale retrieval and matching tasks (Agarwal, Snavely, Simon *et al.* 2009; Agarwal, Furukawa, Snavely *et al.* 2010; Snavely, Simon, Goesele *et al.* 2010).

14.3.3 Application: Location recognition

One of the most exciting applications of instance recognition today is in the area of location recognition, which can be used both in desktop applications (where did I take this holiday snap?) and in mobile (cell-phone) applications. The latter case includes not only finding out your current location based on a cell-phone image but also providing you with navigation directions or annotating your images with useful information, such as building names and restaurant reviews (i.e., a portable form of *augmented reality*).

Some approaches to location recognition assume that the photos consist of architectural scenes for which vanishing directions can be used to pre-rectify the images for easier matching (Robertson and Cipolla 2004). Other approaches use general affine covariant interest points to perform *wide baseline matching* (Schaffalitzky and Zisserman 2002). The Photo Tourism system of Snavely, Seitz, and Szeliski (2006) (Section 13.1.2) was the first to apply these kinds of ideas to large-scale image matching and (implicit) location recognition from Internet photo collections taken under a wide variety of viewing conditions.

The main difficulty in location recognition is in dealing with the extremely large community (user-generated) photo collections on Web sites such as Flickr (Philbin, Chum, Isard *et al.* 2007; Chum, Philbin, Sivic *et al.* 2007; Philbin, Chum, Sivic *et al.* 2008; Turcot and Lowe 2009) or commercially captured databases (Schindler, Brown, and Szeliski 2007). The prevalence of commonly appearing elements such as foliage, signs, and common architectural elements further complicates the task. Figure 14.31 shows some results on location recognition from community photo collections, while Figure 14.32 shows sample results from denser commercially acquired datasets. In the latter case, the overlap between adjacent database images can be used to verify and prune potential matches using “temporal” filtering, i.e., requiring the query image to match nearby overlapping database images before accepting the match.

Another variant on location recognition is the automatic discovery of *landmarks*, i.e.,



Figure 14.32 Feature-based location recognition (Schindler, Brown, and Szeliski 2007) © 2007 IEEE: (a) three typical series of overlapping street photos; (b) handheld camera shots and (c) their corresponding database photos.

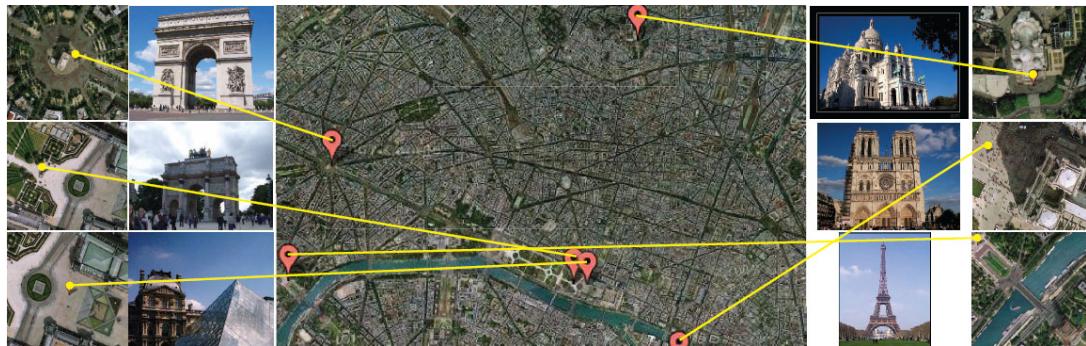


Figure 14.33 Automatic mining, annotation, and localization of community photo collections (Quack, Leibe, and Van Gool 2008) © 2008 ACM. This figure does not show the textual annotations or corresponding Wikipedia entries, which are also discovered.

frequently photographed objects and locations. Simon, Snavely, and Seitz (2007) show how these kinds of objects can be discovered simply by analyzing the matching graph constructed as part of the 3D modeling process in Photo Tourism. More recent work has extended this approach to larger data sets using efficient clustering techniques (Philbin and Zisserman 2008; Li, Wu, Zach *et al.* 2008; Chum, Philbin, and Zisserman 2008; Chum and Matas 2010) as well as combining meta-data such as GPS and textual tags with visual search (Quack, Leibe, and Van Gool 2008; Crandall, Backstrom, Huttenlocher *et al.* 2009), as shown in Figure 14.33. It is now even possible to automatically associate object tags with images based on their co-occurrence in multiple loosely tagged images (Simon and Seitz 2008; Gammeter, Bossard, Quack *et al.* 2009).

The concept of organizing the world’s photo collections by location has even been recently extended to organizing all of the universe’s (astronomical) photos in an application called *astrometry*, <http://astrometry.net/>. The technique used to match any two star fields is

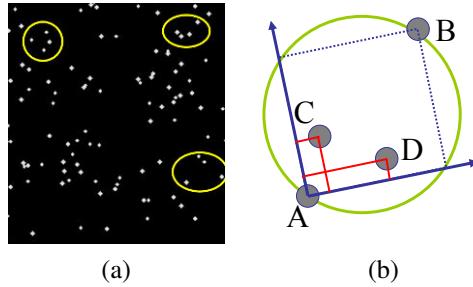


Figure 14.34 Locating star fields using astrometry, <http://astrometry.net/>. (a) Input star field and some selected star quads. (b) The 2D coordinates of stars C and D are encoded relative to the unit square defined by A and B.

to take quadruplets of nearby stars (a pair of stars and another pair inside their diameter) to form a 30-bit *geometric hash* by encoding the relative positions of the second pair of points using the inscribed square as the reference frame, as shown in Figure 14.34. Traditional information retrieval techniques (k-d trees built for different parts of a sky atlas) are then used to find matching quads as potential star field location hypotheses, which can then be verified using a similarity transform.

14.4 Category recognition

While instance recognition techniques are relatively mature and are used in commercial applications, such as Photosynth (Section 13.1.2), generic category (class) recognition is still a largely unsolved problem. Consider for example the set of photographs in Figure 14.35, which shows objects taken from 10 different visual categories. (I’ll leave it up to you to name each of the categories.) How would you go about writing a program to categorize each of these images into the appropriate class, especially if you were also given the choice “none of the above”?

As you can tell from this example, visual category recognition is an *extremely* challenging problem; no one has yet constructed a system that approaches the performance level of a two-year-old child. However, the progress in the field has been quite dramatic, if judged by how much better today’s algorithms are compared to those of a decade ago.

Figure 14.54 shows a sample image from each of the 20 categories used in the 2008 PASCAL Visual Object Classes Challenge. The yellow boxes represent the extent of each of the objects found in a given image. On such *closed world* collections where the task is to decide among 20 categories, today’s classification algorithms can do remarkably well.

In this section, we look at a number of approaches to solving category recognition. While historically, *part-based* representations and recognition algorithms (Section 14.4.2) were the preferred approach (Fischler and Elschlager 1973; Felzenszwalb and Huttenlocher 2005; Fergus, Perona, and Zisserman 2007), we begin by describing simpler *bag-of-features* approaches (Section 14.4.1) that represent objects and images as unordered collections of feature descriptors. We then look at the problem of simultaneously segmenting images while recognizing objects (Section 14.4.3) and also present some applications of such techniques to photo manipulation (Section 14.4.4). In Section 14.5, we look at how context and scene un-

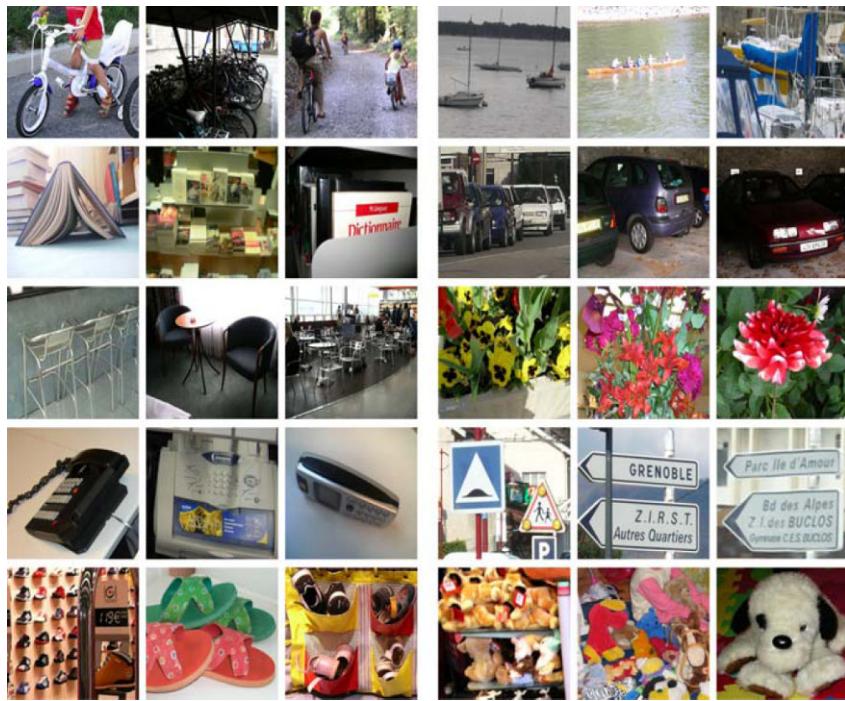


Figure 14.35 Sample images from the Xerox 10 class dataset (Csurka, Dance, Perronnin *et al.* 2006) © 2007 Springer. Imagine trying to write a program to distinguish such images from other photographs.

derstanding, as well as machine learning, can improve overall recognition results. Additional details on the techniques presented in this section can be found in (Pinz 2005; Ponce, Hebert, Schmid *et al.* 2006; Dickinson, Leonardis, Schiele *et al.* 2007; Fei-Fei, Fergus, and Torralba 2009).

14.4.1 Bag of words

One of the simplest algorithms for category recognition is the *bag of words* (also known as *bag of features* or *bag of keypoints*) approach (Csurka, Dance, Fan *et al.* 2004; Lazebnik, Schmid, and Ponce 2006; Csurka, Dance, Perronnin *et al.* 2006; Zhang, Marszalek, Lazebnik *et al.* 2007). As shown in Figure 14.36, this algorithm simply computes the distribution (histogram) of visual words found in the query image and compares this distribution to those found in the training images. We have already seen elements of this approach in Section 14.3.2, Equations (14.33–14.35) and Algorithm 14.2. The biggest difference from instance recognition is the absence of a geometric verification stage (Section 14.3.1), since individual instances of generic visual categories, such as those shown in Figure 14.35, have relatively little spatial coherence to their features (but see the work by Lazebnik, Schmid, and Ponce (2006)).

Csurka, Dance, Fan *et al.* (2004) were the first to use the term *bag of keypoints* to describe such approaches and among the first to demonstrate the utility of frequency-based techniques for category recognition. Their original system used affine covariant regions and SIFT de-

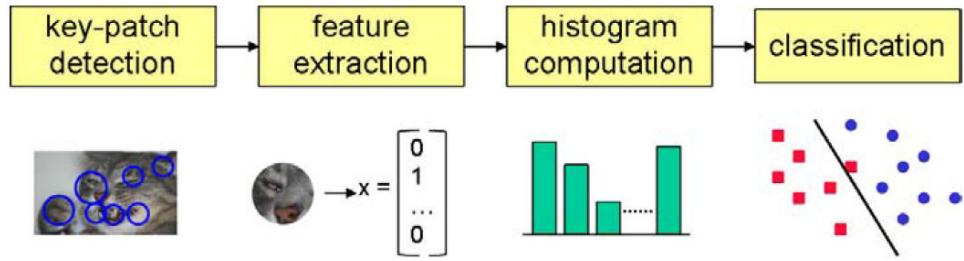


Figure 14.36 A typical processing pipeline for a bag-of-words category recognition system (Csurka, Dance, Perronnin *et al.* 2006) © 2007 Springer. Features are first extracted at keypoints and then quantized to get a distribution (histogram) over the learned *visual words* (feature cluster centers). The feature distribution histogram is used to learn a decision surface using a classification algorithm, such as a support vector machine.

scriptors, k-means visual vocabulary construction, and both a naïve Bayesian classifier and support vector machines for classification. (The latter was found to perform better.) Their newer system (Csurka, Dance, Perronnin *et al.* 2006) uses regular (non-affine) SIFT patches, boosting instead of SVMs, and incorporates a small amount of geometric consistency information.

Zhang, Marszalek, Lazebnik *et al.* (2007) perform a more detailed study of such bag of features systems. They compare a number of feature detectors (Harris–Laplace (Mikolajczyk and Schmid 2004) and Laplacian (Lindeberg 1998b)), descriptors (SIFT, RIFT, and SPIN (Lazebnik, Schmid, and Ponce 2005)), and SVM kernel functions. To estimate distances for the kernel function, they form an *image signature*

$$S = ((t_1, \mathbf{m}_1), \dots, (t_m, \mathbf{m}_m)), \quad (14.36)$$

analogous to the *tf-idf* vector \mathbf{t} in (14.34), where the cluster centers \mathbf{m}_i are made explicit. They then investigate two different kernels for comparing such image signatures. The first is the *earth mover's distance* (EMD) (Rubner, Tomasi, and Guibas 2000),

$$EMD(S, S') = \frac{\sum_i \sum_j f_{ij} d(\mathbf{m}_i, \mathbf{m}'_j)}{\sum_i \sum_j f_{ij}}, \quad (14.37)$$

where f_{ij} is a *flow* value that can be computed using a linear program and $d(\mathbf{m}_i, \mathbf{m}'_j)$ is the *ground distance* (Euclidean distance) between \mathbf{m}_i and \mathbf{m}'_j . Note that the EMD can be used to compare two signatures of different lengths, where the entries do not need to correspond. The second is a χ^2 distance

$$\chi^2(S, S') = \frac{1}{2} \sum_i \frac{(t_i - t'_i)^2}{t_i + t'_i}, \quad (14.38)$$

which measures the likelihood that the two signatures were generated from consistent random processes. These distance metrics are then converted into SVM kernels using a generalized Gaussian kernel

$$K(S, S') = \exp\left(-\frac{1}{A} D(S, S')\right), \quad (14.39)$$

where A is a scaling parameter set to the mean distance between training images. In their experiments, they find that the EMD works best for visual category recognition and the χ^2 measure is best for texture recognition.

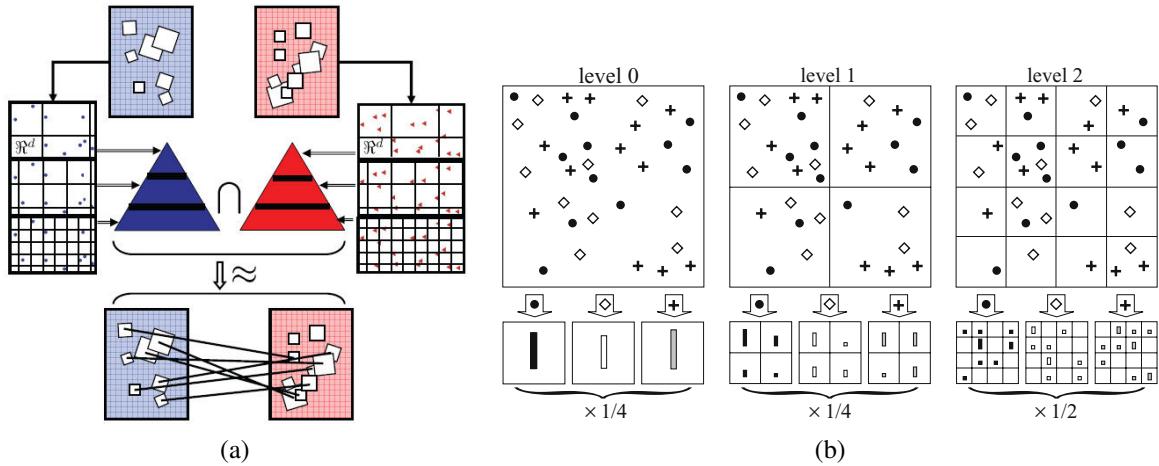


Figure 14.37 Comparing collections of feature vectors using pyramid matching. (a) The feature-space pyramid match kernel (Grauman and Darrell 2007b) constructs a pyramid in high-dimensional feature space and uses it to compute distances (and implicit correspondences) between sets of feature vectors. (b) Spatial pyramid matching (Lazebnik, Schmid, and Ponce 2006) © 2006 IEEE divides the image into a pyramid of pooling regions and computes separate visual word histograms (distributions) inside each spatial bin.

Instead of quantizing feature vectors to visual words, Grauman and Darrell (2007b) develop a technique for directly computing an approximate distance between two variably sized collections of feature vectors. Their approach is to bin the feature vectors into a multi-resolution pyramid defined in feature space (Figure 14.37a) and count the number of features that land in corresponding bins B_{il} and B'_{il} (Figure 14.38a–c). The distance between the two sets of feature vectors (which can be thought of as points in a high-dimensional space) is computed using histogram intersection between corresponding bins

$$C_l = \sum_i \min(B_{il}, B'_{il}) \quad (14.40)$$

(Figure 14.38d). These per-level counts are then summed up in a weighted fashion

$$D_\Delta = \sum_l w_l N_l \quad \text{with} \quad N_l = C_l - C_{l-1} \quad \text{and} \quad w_l = \frac{1}{d2^l} \quad (14.41)$$

(Figure 14.38e), which discounts matches already found at finer levels while weighting finer matches more heavily. (d is the dimension of the embedding space, i.e., the length of the feature vectors.) In follow-on work, Grauman and Darrell (2007a) show how an explicit construction of the pyramid can be avoided using hashing techniques.

Inspired by this work, Lazebnik, Schmid, and Ponce (2006) show how a similar idea can be employed to augment bags of keypoints with loose notions of 2D spatial location analogous to the pooling performed by SIFT (Lowe 2004) and “gist” (Torralba, Murphy, Freeman *et al.* 2003). In their work, they extract affine region descriptors (Lazebnik, Schmid, and Ponce 2005) and quantize them into visual words. (Based on previous results by Fei-Fei and Perona (2005), the feature descriptors are extracted densely (on a regular grid) over the image, which can be helpful in describing textureless regions such as the sky.) They then form

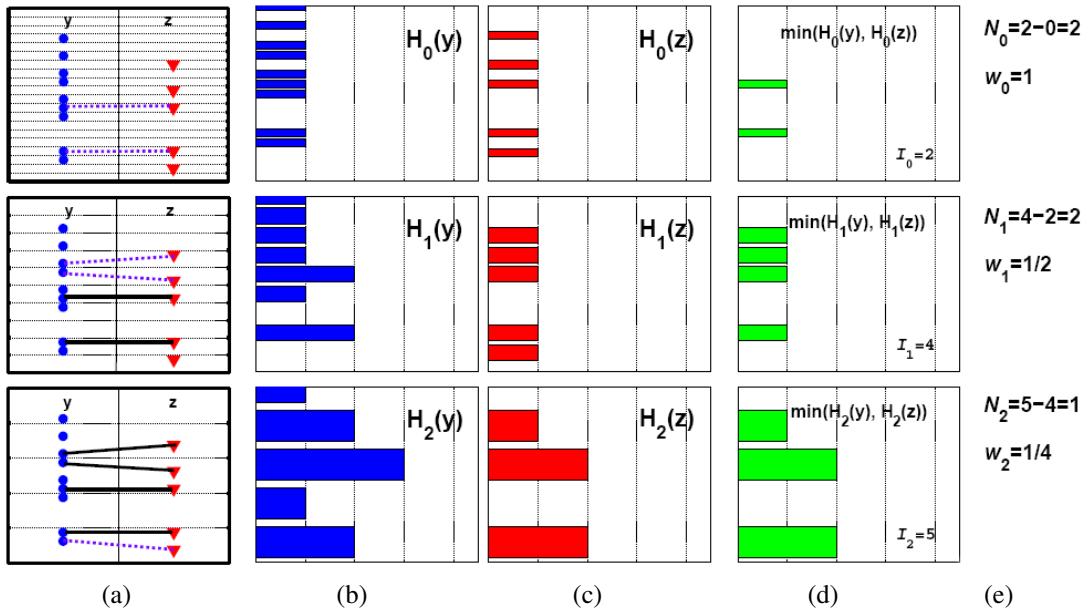


Figure 14.38 A one-dimensional illustration of comparing collections of feature vectors using the pyramid match kernel (Grauman and Darrell 2007b): (a) distribution of feature vectors (point sets) into the pyramidal bins; (b–c) histogram of point counts in bins B_{il} and B'_{il} for the two images; (d) histogram intersections (minimum values); (e) per-level similarity scores, which are weighted and summed to form the final distance/similarity metric.

a spatial pyramid of bins containing word counts (histograms), as shown in Figure 14.37b, and use a similar pyramid match kernel to combine histogram intersection counts in a hierarchical fashion.

The debate about whether to use quantized feature descriptors or continuous descriptors and also whether to use sparse or dense features continues to this day. Boiman, Shechtman, and Irani (2008) show that if query images are compared to *all* the features representing a given class, rather than just each class image individually, nearest-neighbor matching followed by a naïve Bayes classifier outperforms quantized visual words (Figure 14.39). Instead of using generic feature detectors and descriptors, some authors have been investigating *learning* class-specific features (Ferencz, Learned-Miller, and Malik 2008), often using randomized forests (Philbin, Chum, Isard *et al.* 2007; Moosmann, Nowak, and Jurie 2008; Shotton, Johnson, and Cipolla 2008) or combining the feature generation and image classification stages (Yang, Jin, Sukthankar *et al.* 2008). Others, such as Serre, Wolf, and Poggio (2005) and Mutch and Lowe (2008) use hierarchies of dense feature transforms inspired by biological (visual cortical) processing combined with SVMs for final classification.

14.4.2 Part-based models

Recognizing an object by finding its constituent parts and measuring their geometric relationships is one of the oldest approaches to object recognition (Fischler and Elschlager 1973; Kanade 1977; Yuille 1991). We have already seen examples of part-based approaches being used for face recognition (Figure 14.18) (Moghaddam and Pentland 1997; Heisele, Ho, Wu

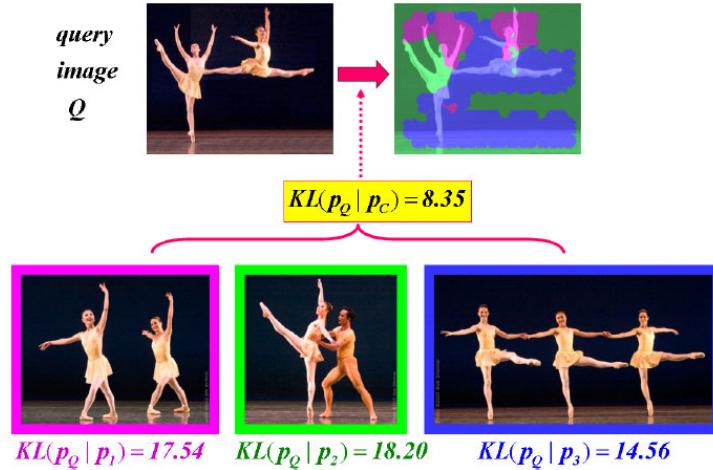


Figure 14.39 “Image-to-Image” vs. “Image-to-Class” distance comparison (Boiman, Shechtman, and Irani 2008) © 2008 IEEE. The query image on the upper left may not match the feature distribution of any of the database images in the bottom row. However, if each feature in the query is matched to its closest analog in *all* the class images, a good match can be found.

et al. 2003; Heisele, Serre, and Poggio 2007) and pedestrian detection (Figure 14.9) (Felzenszwalb, McAllester, and Ramanan 2008).

In this section, we look more closely at some of the central issues in part-based recognition, namely, the representation of geometric relationships, the representation of individual parts, and algorithms for learning such descriptions and recognizing them at run time. More details on part-based models for recognition can be found in the course notes of Fergus (2007b, 2009).

The earliest approaches to representing geometric relationships were dubbed *pictorial structures* by Fischler and Elschlager (1973) and consisted of spring-like connections between different feature locations (Figure 14.1a). To fit a pictorial structure to an image, an energy function of the form

$$E = \sum_i V_i(\mathbf{l}_i) + \sum_{ij \in E} V_{ij}(\mathbf{l}_i, \mathbf{l}_j) \quad (14.42)$$

is minimized over all potential part locations or poses $\{\mathbf{l}_i\}$ and pairs of parts (i, j) for which an edge (geometric relationship) exists in E . Note how this energy is closely related to that used with Markov random fields (3.108–3.109), which can be used to embed pictorial structures in a probabilistic framework that makes parameter learning easier (Felzenszwalb and Huttenlocher 2005).

Part-based models can have different topologies for the geometric connections between the parts (Figure 14.41). For example, Felzenszwalb and Huttenlocher (2005) restrict the connections to a tree (Figure 14.41d), which makes learning and inference more tractable. A tree topology enables the use of a recursive Viterbi (dynamic programming) algorithm (Pearl 1988; Bishop 2006), in which leaf nodes are first optimized as a function of their parents, and the resulting values are then plugged in and eliminated from the energy function—see Appendix B.5.2. The Viterbi algorithm computes an optimal match in $O(N^2|E| + NP)$ time,

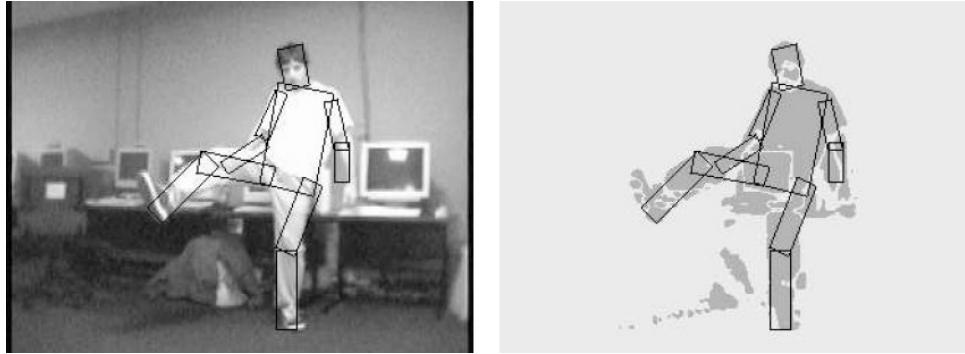


Figure 14.40 Using pictorial structures to locate and track a person (Felzenszwalb and Huttenlocher 2005) © 2005 Springer. The structure consists of articulated rectangular body parts (torso, head, and limbs) connected in a tree topology that encodes relative part positions and orientations. To fit a pictorial structure model, a binary silhouette image is first computed using background subtraction.

where N is the number of potential locations or poses for each part, $|E|$ is the number of edges (pairwise constraints), and $P = |V|$ is the number of parts (vertices in the graphical model, which is equal to $|E| + 1$ in a tree). To further increase the efficiency of the inference algorithm, Felzenszwalb and Huttenlocher (2005) restrict the pairwise energy functions $V_{ij}(\mathbf{l}_i, \mathbf{l}_j)$ to be Mahalanobis distances on functions of location variables and then use fast distance transform algorithms to minimize each pairwise interaction in time that is closer to linear in N .

Figure 14.40 shows the results of using their pictorial structures algorithm to fit an articulated body model to a binary image obtained by background segmentation. In this application of pictorial structures, parts are parameterized by the locations, sizes, and orientations of their approximating rectangles. Unary matching potentials $V_i(\mathbf{l}_i)$ are determined by counting the percentage of foreground and background pixels inside and just outside the tilted rectangle representing each part.

Over the last decade, a large number of different graphical models have been proposed for part-based recognition, as shown in Figure 14.41. Carneiro and Lowe (2006) discuss a number of these models and propose one of their own, which they call a *sparse flexible model*; it involves ordering the parts and having each part's location depend on at most k of its ancestor locations.

The simplest models, which we saw in Section 14.4.1, are bags of words, where there are no geometric relationships between different parts or features. While such models can be very efficient, they have a very limited capacity to express the spatial arrangement of parts. Trees and stars (a special case of trees where all leaf nodes are directly connected to a common root) are the most efficient in terms of inference and hence also learning (Felzenszwalb and Huttenlocher 2005; Fergus, Perona, and Zisserman 2005; Felzenszwalb, McAllester, and Ramanan 2008). Directed acyclic graphs (Figure 14.41f–g) come next in terms of complexity and can still support efficient inference, although at the cost of imposing a causal structure on the part model (Bouchard and Triggs 2005; Carneiro and Lowe 2006). k -fans, in which a clique of size k forms the root of a star-shaped model (Figure 14.41c) have inference complexity $O(N^{k+1})$, although with distance transforms and Gaussian priors, this can be lowered to

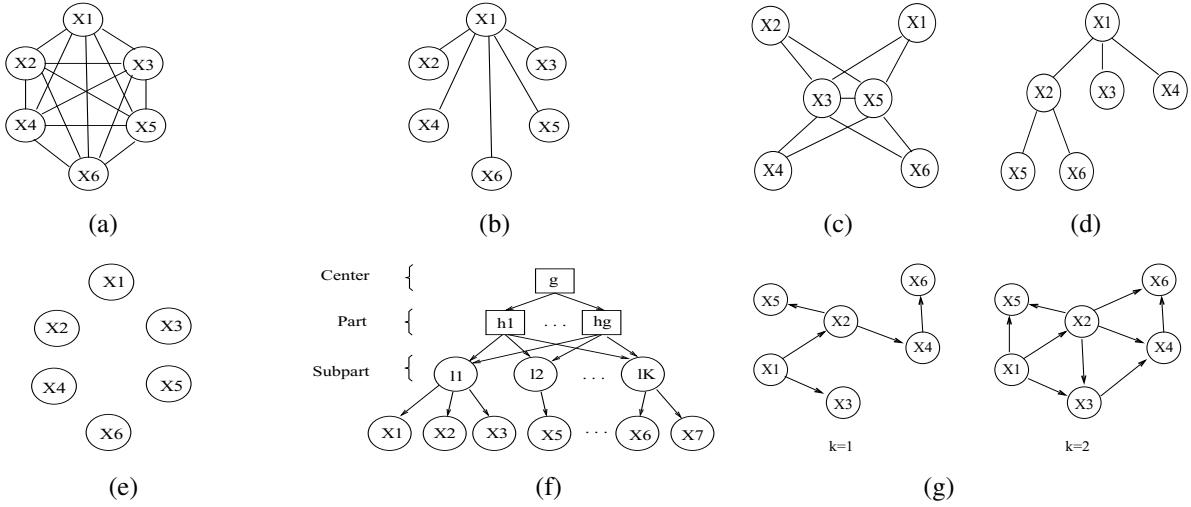


Figure 14.41 Graphical models for geometric spatial priors (Carneiro and Lowe 2006) © 2006 Springer: (a) constellation (Fergus, Perona, and Zisserman 2007); (b) star (Crandall, Felzenszwalb, and Huttenlocher 2005; Fergus, Perona, and Zisserman 2005); (c) k -fan ($k = 2$) (Crandall, Felzenszwalb, and Huttenlocher 2005); (d) tree (Felzenszwalb and Huttenlocher 2005); (e) bag of features (Csurka, Dance, Fan *et al.* 2004); (f) hierarchy (Bouchard and Triggs 2005); (g) sparse flexible model (Carneiro and Lowe 2006).

$O(N^k)$ (Crandall, Felzenszwalb, and Huttenlocher 2005; Crandall and Huttenlocher 2006). Finally, fully connected *constellation* models (Figure 14.41a) are the most general, but the assignment of features to parts becomes intractable for moderate numbers of parts P , since the complexity of such an assignment is $O(N^P)$ (Fergus, Perona, and Zisserman 2007).

The original constellation model was developed by Burl, Weber, and Perona (1998) and consists of a number of parts whose relative positions are encoded by their mean locations and a full covariance matrix, which is used to denote not only positional uncertainty but also potential correlations (covariance) between different parts (Figure 14.42a). Weber, Welling, and Perona (2000) extended this technique to a weakly supervised setting, where both the appearance of each part and its locations are automatically learned given only whole image labels. Fergus, Perona, and Zisserman (2007) further extend this approach to simultaneous learning of appearance and shape models from scale-invariant keypoint detections.

Figure 14.42a shows the shape model learned for the motorcycle class. The top figure shows the mean relative locations for each part along with their position covariances (inter-part covariances are not shown) and likelihood of occurrence. The bottom curve shows the Gaussian PDFs for the relative log-scale of each part with respect to the “landmark” feature. Figure 14.42b shows the appearance model learned for each part, visualized as the patches around detected features in the training database that best match the appearance model. Figure 14.42c shows the features detected in the test database (pink dots) along with the corresponding parts that they were assigned to (colored circles). As you can see, the system has successfully learned and then used a fairly complex model of motorcycle appearance.

The part-based approach to recognition has also been extended to learning new categories from small numbers of examples, building on recognition components developed for other classes (Fei-Fei, Fergus, and Perona 2006). More complex hierarchical part-based models can

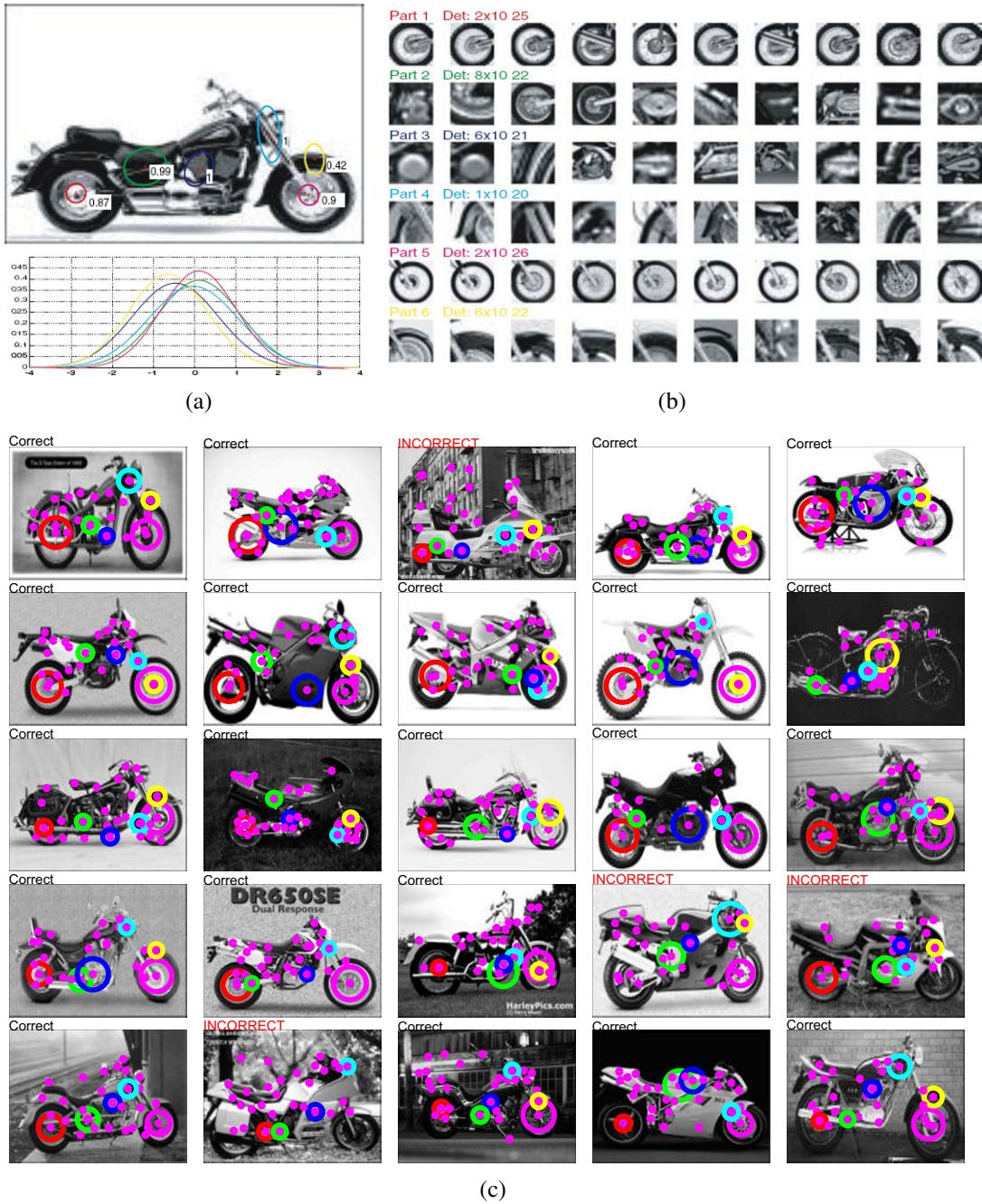


Figure 14.42 Part-based recognition (Fergus, Perona, and Zisserman 2007) © 2007 Springer: (a) locations and covariance ellipses for each part, along with their occurrence probabilities (top) and relative log-scale densities (bottom); (b) part examples drawn from the training images that best match the average appearance; (c) recognition results for the motorcycle class, showing detected features (pink dots) and parts (colored circles).

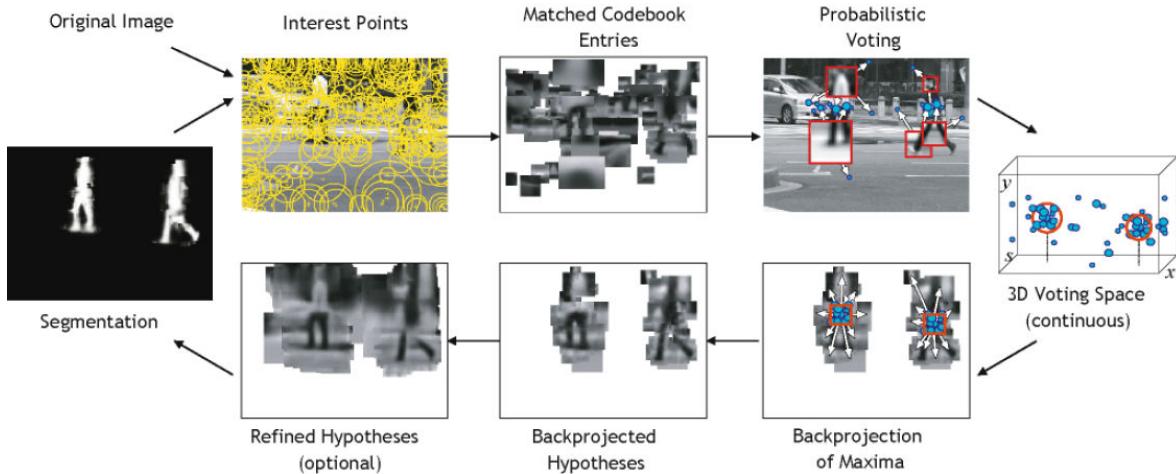


Figure 14.43 Interleaved recognition and segmentation (Leibe, Leonardis, and Schiele 2008) © 2008 Springer. The process starts by re-recognition visual words (codebook entries) in a new image (scene) and having each part vote for likely locations and size in a 3D (x, y, s) voting space (top row). Once a maximum has been found, the parts (features) corresponding to this instance are determined by *backprojecting* the contributing votes. The foreground–background segmentation for each object can be found by backprojecting probabilistic masks associated with each codebook entry. The whole recognition and segmentation process can then be repeated.

be developed using the concept of grammars (Bouchard and Triggs 2005; Zhu and Mumford 2006). A simpler way to use parts is to have keypoints that are recognized as being part of a class vote for the estimated part locations, as shown in the top row of Figure 14.43 (Leibe, Leonardis, and Schiele 2008). (Implicitly, this corresponds to having a star-shaped geometric model.)

14.4.3 Recognition with segmentation

The most challenging version of generic object recognition is to simultaneously perform recognition with accurate boundary segmentation (Fergus 2007a). For instance recognition (Section 14.3.1), this can sometimes be achieved by backprojecting the object model into the scene (Lowe 2004), as shown in Figure 14.1d, or matching portions of the new scene to pre-learned (segmented) object models (Ferrari, Tuytelaars, and Van Gool 2006b; Kannala, Rahtu, Brandt *et al.* 2008).

For more complex (flexible) object models, such as those for humans Figure 14.1f, a different approach is to pre-segment the image into larger or smaller pieces (Chapter 5) and then match such pieces to portions of the model (Mori, Ren, Efros *et al.* 2004; Mori 2005; He, Zemel, and Ray 2006; Gu, Lim, Arbelaez *et al.* 2009).

An alternative approach by Leibe, Leonardis, and Schiele (2008), which we introduced in the previous section, votes for potential object locations and scales based on the detection of features corresponding to pre-clustered visual codebook entries (Figure 14.43). To support segmentation, each codebook entry has an associated foreground–background mask, which is learned as part of the codebook clustering process from pre-labeled object segmentation masks. During recognition, once a maximum in the voting space is found, the masks

associated with the entries that voted for this instance are combined to obtain an object segmentation, as shown on the left side of Figure 14.43.

A more holistic approach to recognition and segmentation is to formulate the problem as one of labeling every pixel in an image with its class membership, and to solve this problem using energy minimization or Bayesian inference techniques, i.e., conditional random fields (Section 3.7.2, (3.118)) (Kumar and Hebert 2006; He, Zemel, and Carreira-Perpiñán 2004). The TextonBoost system of Shotton, Winn, Rother *et al.* (2009) uses unary (pixel-wise) potentials based on image-specific color distributions (Section 5.5) (Boykov and Jolly 2001; Rother, Kolmogorov, and Blake 2004), location information (e.g., foreground objects are more likely to be in the middle of the image, sky is likely to be higher, and road is likely to be lower), and novel texture-layout classifiers trained using shared boosting. It also uses traditional pairwise potentials that look at image color gradients (Veksler 2001; Boykov and Jolly 2001; Rother, Kolmogorov, and Blake 2004). The texton-layout features first filter the image with a series of 17 oriented filter banks and then cluster the responses to classify each pixel into 30 different texton classes (Malik, Belongie, Leung *et al.* 2001). The responses are then filtered using offset rectangular regions trained with joint boosting (Viola and Jones 2004) to produce the texton-layout features used as unary potentials.

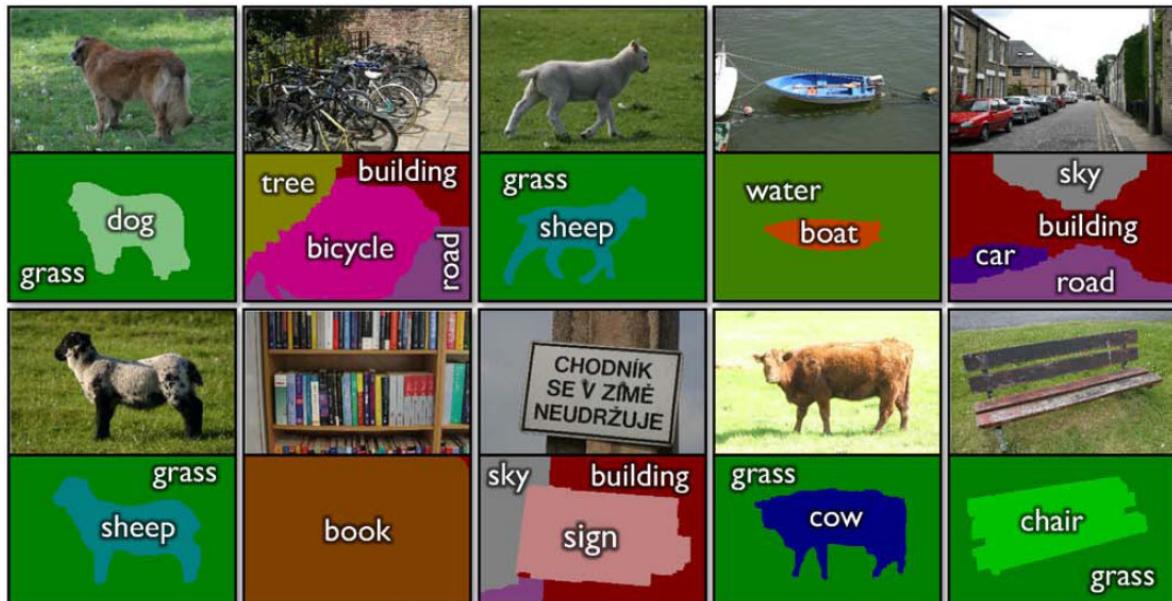
Figure 14.44a shows some examples of images successfully labeled and segmented using TextonBoost, while Figure 14.44b shows examples where it does not do as well. As you can see, this kind of semantic labeling can be extremely challenging.

The TextonBoost conditional random field framework has been extended to LayoutCRFs by Winn and Shotton (2006), who incorporate additional constraints to recognize multiple object instances and deal with occlusions (Figure 14.45), and even more recently by Hoiem, Rother, and Winn (2007) to incorporate full 3D models.

Conditional random fields continue to be widely used and extended for simultaneous recognition and segmentation applications (Kumar and Hebert 2006; He, Zemel, and Ray 2006; Levin and Weiss 2006; Verbeek and Triggs 2007; Yang, Meer, and Foran 2007; Rabenovich, Vedaldi, Galleguillos *et al.* 2007; Batra, Sukthankar, and Chen 2008; Larlus and Jurie 2008; He and Zemel 2008; Kumar, Torr, and Zisserman 2010), producing some of the best results on the difficult PASCAL VOC segmentation challenge (Shotton, Johnson, and Cipolla 2008; Kohli, Ladický, and Torr 2009). Approaches that first segment the image into unique or multiple segmentations (Borenstein and Ullman 2008; He, Zemel, and Ray 2006; Russell, Efros, Sivic *et al.* 2006) (potentially combined with CRF models) also do quite well: Csurka and Perronnin (2008) have one of the top algorithms in the VOC segmentation challenge. Hierarchical (multi-scale) and grammar (parsing) models are also sometimes used (Tu, Chen, Yuille *et al.* 2005; Zhu, Chen, Lin *et al.* 2008).

14.4.4 Application: Intelligent photo editing

Recent advances in object recognition and scene understanding have greatly increased the power of intelligent (semi-automated) photo editing applications. One example is the Photo Clip Art system of Lalonde, Hoiem, Efros *et al.* (2007), which recognizes and segments objects of interest, such as pedestrians, in Internet photo collections and then allows users to paste them into their own photos. Another is the scene completion system of Hays and Efros (2007), which tackles the same *inpainting* problem we studied in Section 10.5. Given an image in which we wish to erase and fill in a large section (Figure 14.46a–b), where do you



(a)



(b)

Figure 14.44 Simultaneous recognition and segmentation using TextronBoost (Shotton, Winn, Rother *et al.* 2009)
© 2009 Springer: (a) successful recognition results; (b) less successful results.

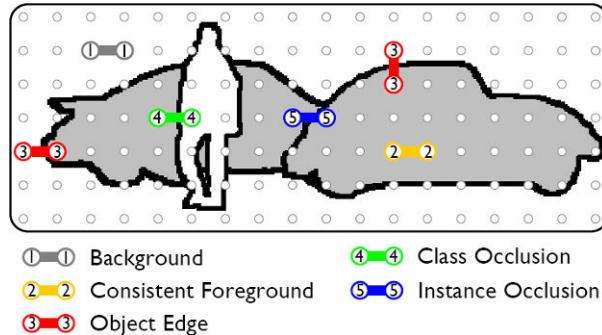


Figure 14.45 Layout consistent random field (Winn and Shotton 2006) © 2006 IEEE. The numbers indicate the kind of neighborhood relations that can exist between pixels assigned to the same or different classes. Each pairwise relationship carries its own likelihood (energy penalty).

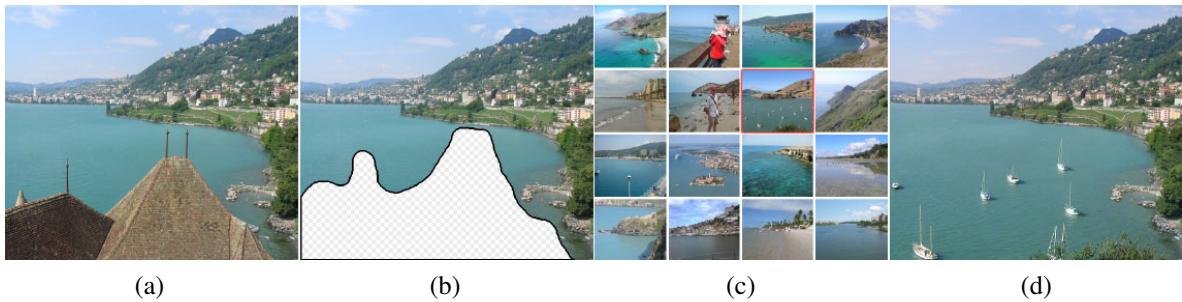


Figure 14.46 Scene completion using millions of photographs (Hays and Efros 2007) © 2007 ACM: (a) original image; (b) after unwanted foreground removal; (c) plausible scene matches, with the one the user selected highlighted in red; (d) output image after replacement and blending.

get the pixels to fill in the gaps in the edited image? Traditional approaches either use smooth continuation (Bertalmio, Sapiro, Caselles *et al.* 2000) or borrowing pixels from other parts of the image (Efros and Leung 1999; Criminisi, Pérez, and Toyama 2004; Efros and Freeman 2001). With the advent of huge repositories of images on the Web (a topic we return to in Section 14.5.1), it often makes more sense to find a *different* image to serve as the source of the missing pixels.

In their system, Hays and Efros (2007) compute the *gist* of each image (Oliva and Torralba 2001; Torralba, Murphy, Freeman *et al.* 2003) to find images with similar colors and composition. They then run a graph cut algorithm that minimizes image gradient differences and composite the new replacement piece into the original image using Poisson image blending (Section 9.3.4) (Pérez, Gangnet, and Blake 2003). Figure 14.46d shows the resulting image with the erased foreground rooftops region replaced with sailboats.

A different application of image recognition and segmentation is to infer 3D structure from a single photo by recognizing certain scene structures. For example, Criminisi, Reid, and Zisserman (2000) detect vanishing points and have the user draw basic structures, such as walls, in order to infer the 3D geometry (Section 6.3.3). Hoiem, Efros, and Hebert (2005a) on the other hand, work with more “organic” scenes such as the one shown in Figure 14.47.

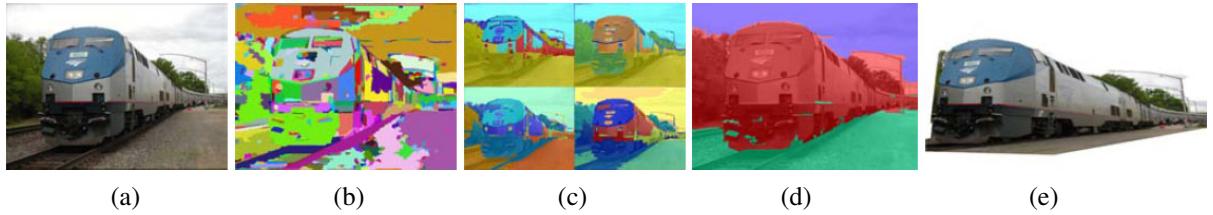


Figure 14.47 Automatic photo pop-up (Hoiem, Efros, and Hebert 2005a) © 2005 ACM: (a) input image; (b) superpixels are grouped into (c) multiple regions; (d) labelings indicating ground (green), vertical (red), and sky (blue); (e) novel view of resulting piecewise-planar 3D model.

Their system uses a variety of classifiers and statistics learned from labeled images to classify each pixel as either ground, vertical, or sky (Figure 14.47d). To do this, they begin by computing superpixels (Figure 14.47b) and then group them into plausible regions that are likely to share similar geometric labels (Figure 14.47c). After all the pixels have been labeled, the boundaries between the vertical and ground pixels can be used to infer 3D lines along which the image can be folded into a “pop-up” (after removing the sky pixels), as shown in Figure 14.47e. In related work, Saxena, Sun, and Ng (2009) develop a system that directly infers the depth and orientation of each pixel instead of using just three geometric class labels.

Face detection and localization can also be used in a variety of photo editing applications (in addition to being used in-camera to provide better focus, exposure, and flash settings). Zanella and Fuentes (2004) use active shape models (Section 14.2.2) to register facial features for creating automated morphs. Rother, Bordeaux, Hamadi *et al.* (2006) use face and sky detection to determine regions of interest in order to decide which pieces from a collection of images to stitch into a collage. Bitouk, Kumar, Dhillon *et al.* (2008) describe a system that matches a given face image to a large collection of Internet face images, which can then be used (with careful relighting algorithms) to replace the face in the original image. Applications they describe include de-identification and getting the best possible smile from everyone in a “burst mode” group shot. Leyvand, Cohen-Or, Dror *et al.* (2008) show how accurately locating facial features using an active shape model (Cootes, Edwards, and Taylor 2001; Zhou, Gu, and Zhang 2003) can be used to warp such features (and hence the image) towards configurations resembling those found in images whose facial attractiveness was highly rated, thereby “beautifying” the image without completely losing a person’s identity.

Most of these techniques rely either on a set of labeled training images, which is an essential component of all learning techniques, or the even more recent explosion in images available on the Internet. The assumption in some of this work (and in recognition systems based on such very large databases (Section 14.5.1)) is that as the collection of accessible (and potentially partially labeled) images gets larger, finding a close match gets easier. As Hays and Efros (2007) state in their abstract “Our chief insight is that while the space of images is effectively infinite, the space of semantically differentiable scenes is actually not that large.” In an interesting commentary on their paper, Levoy (2008) disputes this assertion, claiming that “features in natural scenes form a heavy-tailed distribution, meaning that while some features in photographs are more common than others, the relative occurrence of less common features drops slowly. In other words, there are many unusual photographs in the world.” He does, however agree that in computational photography, as in many other applications such

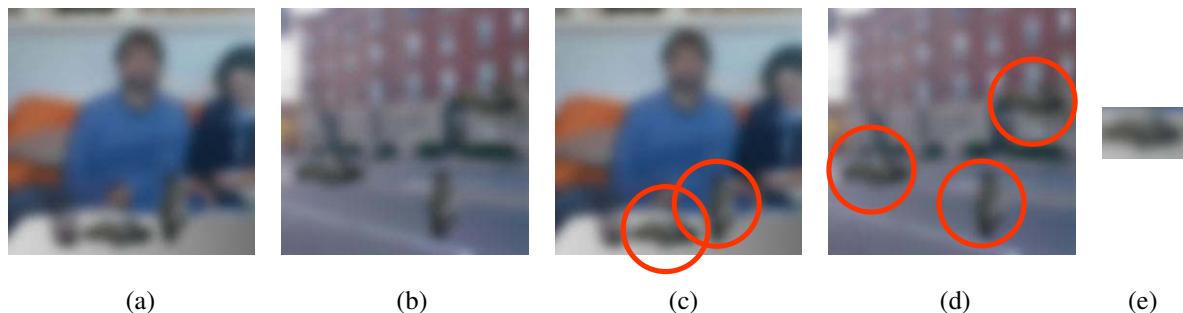


Figure 14.48 The importance of context (images courtesy of Antonio Torralba). Can you name all of the objects in images (a–b), especially those that are circled in (c–d). Look carefully at the circled objects. Did you notice that they all have the same shape (after being rotated), as shown in column (e)?

as speech recognition, synthesis, and translation, “simple machine learning algorithms often outperform more sophisticated ones if trained on large enough databases.” He also goes on to point out both the potential advantages of such systems, such as better automatic color balancing, and potential issues and pitfalls with the kind of image fakery that these new approaches enable.

For additional examples of photo editing and computational photography applications enabled by Internet computer vision, please see recent workshops on this topic,¹⁹ as well as the special journal issue (Avidan, Baker, and Shan 2010), and the course on Internet Vision by Tamara Berg (2008).

14.5 Context and scene understanding

Thus far, we have mostly considered the task of recognizing and localizing objects in isolation from that of understanding the scene (context) in which the object occur. This is a severe limitation, as context plays a very important role in human object recognition (Oliva and Torralba 2007). As we will see in this section, it can greatly improve the performance of object recognition algorithms (Divvala, Hoiem, Hays *et al.* 2009), as well as providing useful semantic clues for general scene understanding (Torralba 2008).

Consider the two photographs in Figure 14.48a–b. Can you name all of the objects, especially those circled in images (c–d)? Now have a closer look at the circled objects. Do see any similarity in their shapes? In fact, if you rotate them by 90°, they are all the same as the “blob” shown in Figure 14.48e. So much for our ability to recognize object by their shape! Another (perhaps more artificial) example of recognition in context is shown in Figure 14.49. Try to name all of the letters and numbers, and then see if you guessed right.

Even though we have not addressed context explicitly earlier in this chapter, we have already seen several instances of this general idea being used. A simple way to incorporate spatial information into a recognition algorithm is to compute feature statistics over different regions, as in the spatial pyramid system of Lazebnik, Schmid, and Ponce (2006). Part-based models (Section 14.4.2, Figures 14.40–14.43), use a kind of local context, where various parts need to be arranged in a proper geometric relationship to constitute an object.

¹⁹ <http://www.internetvisioner.org/>.



Figure 14.49 More examples of context: read the letters in the first group, the numbers in the second, and the letters and numbers in the third. (Images courtesy of Antonio Torralba.)

The biggest difference between part-based and context models is that the latter combine objects into scenes and the number of constituent objects from each class is not known in advance. In fact, it is possible to combine part-based and context models into the same recognition architecture (Murphy, Torralba, and Freeman 2003; Suderth, Torralba, Freeman *et al.* 2008; Crandall and Huttenlocher 2007).

Consider the street and office scenes shown in Figure 14.50a–b. If we have enough training images with labeled regions, such as buildings, cars, and roads or monitors, keyboards, and mice, we can develop a geometric model for describing their relative positions. Suderth, Torralba, Freeman *et al.* (2008) develop such a model, which can be thought of as a two-level constellation model. At the top level, the distributions of objects relative to each other (say, buildings with respect to cars) is modeled as a Gaussian (Figure 14.50c, upper right corners). At the bottom level, the distribution of parts (affine covariant features) with respect to the object center is modeled using a mixture of Gaussians (Figure 14.50c, lower two rows). However, since the number of objects in the scene and parts in each object is unknown, a *latent Dirichlet process* (LDP) is used to model object and part creation in a generative framework. The distributions for all of the objects and parts are learned from a large labeled database and then later used during inference (recognition) to label the elements of a scene.

Another example of context is in simultaneous segmentation and recognition (Section 14.4.3) (Figures 14.44–14.45), where the arrangements of various objects in a scene are used as part of the labeling process. Torralba, Murphy, and Freeman (2004) describe a conditional random field where the estimated locations of building and roads influence the detection of cars, and where boosting is used to learn the structure of the CRF. Rabinovich, Vedaldi, Galleguillos *et al.* (2007) use context to improve the results of CRF segmentation by noting that certain adjacencies (relationships) are more likely than others, e.g., a person is more likely to be on a horse than on a dog.

Context also plays an important role in 3D inference from single images (Figure 14.47), using computer vision techniques for labeling pixels as belonging to the ground, vertical surfaces, or sky (Hoiem, Efros, and Hebert 2005a,b). This line of work has been extended to a more holistic approach that simultaneously reasons about object identity, location, surface orientations, occlusions, and camera viewing parameters (Hoiem, Efros, and Hebert 2008a,b).

A number of approaches use the *gist* of a scene (Torralba 2003; Torralba, Murphy, Freeman *et al.* 2003) to determine where instances of particular objects are likely to occur. For example, Murphy, Torralba, and Freeman (2003) train a regressor to predict the vertical loca-

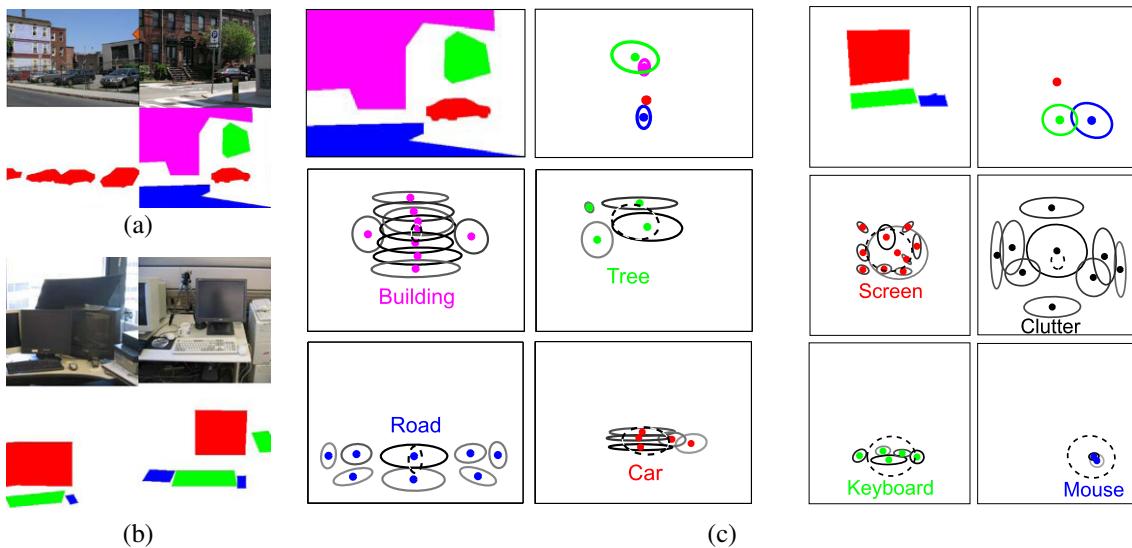


Figure 14.50 Contextual scene models for object recognition (Sudderth, Torralba, Freeman *et al.* 2008) © 2008 Springer: (a) some street scenes and their corresponding labels (magenta = buildings, red = cars, green = trees, blue = road); (b) some office scenes (red = computer screen, green = keyboard, blue = mouse); (c) learned contextual models built from these labeled scenes. The top row shows a sample label image and the distribution of the objects relative to the center red (car or screen) object. The bottom rows show the distributions of parts that make up each object.

tions of objects such as pedestrians, cars, and buildings (or screens and keyboard for indoor office scenes) based on the gist of an image. These location distributions are then used with classic object detectors to improve the performance of the detectors. Gists can also be used to directly match complete images, as we saw in the scene completion work of Hays and Efros (2007).

Finally, some of the most recent work in scene understanding exploits the existence of large numbers of labeled (or even unlabeled) images to perform matching directly against whole images, where the images themselves implicitly encode the expected relationships between objects (Figure 14.51) (Russell, Torralba, Liu *et al.* 2007; Malisiewicz and Efros 2008). We discuss such techniques in the next section, where we look at the influence that large image databases have had on object recognition and scene understanding.

14.5.1 Learning and large image collections

Given how learning techniques are widely used in recognition algorithms, you may wonder whether the topic of learning deserves its own section (or even chapter), or whether it is just part of the basic fabric of all recognition tasks. In fact, trying to build a recognition system without lots of training data for anything other than a basic pattern such as a UPC code has proven to be a dismal failure.

In this chapter, we have already seen lots of techniques borrowed from the machine learning, statistics, and pattern recognition communities. These include principal component, subspace, and discriminant analysis (Section 14.2.1) and more sophisticated discriminative clas-

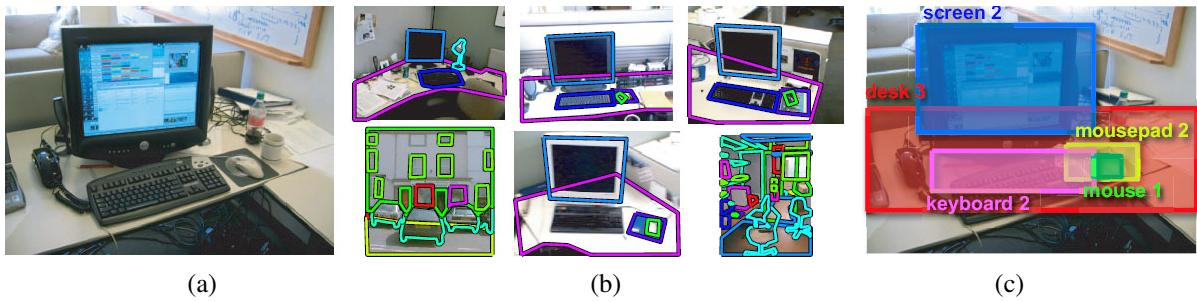


Figure 14.51 Recognition by scene alignment (Russell, Torralba, Liu *et al.* 2007): (a) input image; (b) matched images with similar scene configurations; (c) final labeling of the input image.

sification algorithms such as neural networks, support vector machines, and boosting (Section 14.1.1). Some of the best-performing techniques on challenging recognition benchmarks (Varma and Ray 2007; Felzenszwalb, McAllester, and Ramanan 2008; Fritz and Schiele 2008; Vedaldi, Gulshan, Varma *et al.* 2009) rely heavily on the latest machine learning techniques, whose development is often being driven by challenging vision problems (Freeman, Perona, and Schölkopf 2008).

A distinction sometimes made in the recognition community is between problems where most of the variables of interest (say, parts) are already (partially) labeled and systems that learn more of the problem structure with less supervision (Fergus, Perona, and Zisserman 2007; Fei-Fei, Fergus, and Perona 2006). In fact, recent work by Sivic, Russell, Zisserman *et al.* (2008) has demonstrated the ability to learn visual hierarchies (hierarchies of object parts with related visual appearance) and scene segmentations in a totally unsupervised framework.

Perhaps the most dramatic change in the recognition community has been the appearance of very large databases of training images.²⁰ Early learning-based algorithms, such as those for face and pedestrian detection (Section 14.1), used relatively few (in the hundreds) labeled examples to train recognition algorithm parameters (say, the thresholds used in boosting). Today, some recognition algorithms use databases such as LabelMe (Russell, Torralba, Murphy *et al.* 2008), which contain tens of thousands of labeled examples.

The existence of such large databases opens up the possibility of matching directly against the training images rather than using them to learn the parameters of recognition algorithms. Russell, Torralba, Liu *et al.* (2007) describe a system where a new image is matched against each of the training images, from which a consensus labeling for the unknown objects in the scene can be inferred, as shown in Figure 14.51. Malisiewicz and Efros (2008) start by over-segmenting each image and then use the LabelMe database to search for similar images and configurations in order to obtain per-pixel category labelings. It is also possible to combine feature-based correspondence algorithms with large labeled databases to perform simultaneous recognition and segmentation (Liu, Yuen, and Torralba 2009).

When the database of images becomes large enough, it is even possible to directly match complete images with the expectation of finding a good match. Torralba, Freeman, and Fergus (2008) start with a database of 80 million tiny (32×32) images and compensate for the poor accuracy in their image labels, which are collected automatically from the Internet, by using

²⁰ We have already seen some computational photography applications of such databases in Section 14.4.4.

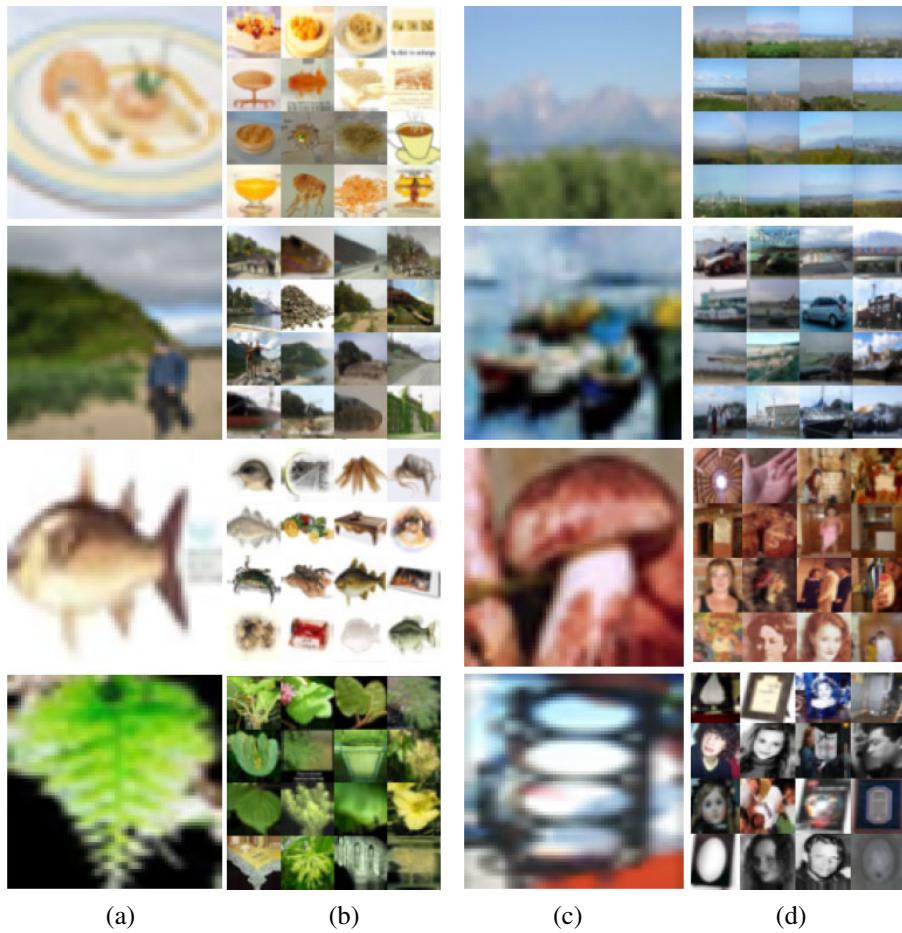


Figure 14.52 Recognition using tiny images (Torralba, Freeman, and Fergus 2008) © 2008 IEEE: columns (a) and (c) show sample input images and columns (b) and (d) show the corresponding 16 nearest neighbors in the database of 80 million tiny images.

a semantic taxonomy (Wordnet) to infer the most likely labels for a new image. Somewhere in the 80 million images, there are enough examples to associate some set of images with each of the 75,000 non-abstract nouns in Wordnet that they use in their system. Some sample recognition results are shown in Figure 14.52.

Another example of a large labeled database of images is ImageNet (Deng, Dong, Socher *et al.* 2009), which is collecting images for the 80,000 nouns (synonym sets) in WordNet (Fellbaum 1998). As of April 2010, about 500–1000 carefully vetted examples for 14841 synsets have been collected (Figure 14.53). The paper by Deng, Dong, Socher *et al.* (2009) also has a nice review of related databases.

As we mentioned in Section 14.4.3, the existence of large databases of partially labeled Internet imagery has given rise to a new sub-field of Internet computer vision, with its own workshops²¹ and a special journal issue (Avidan, Baker, and Shan 2010).

²¹ <http://www.internetvisioner.org/>.

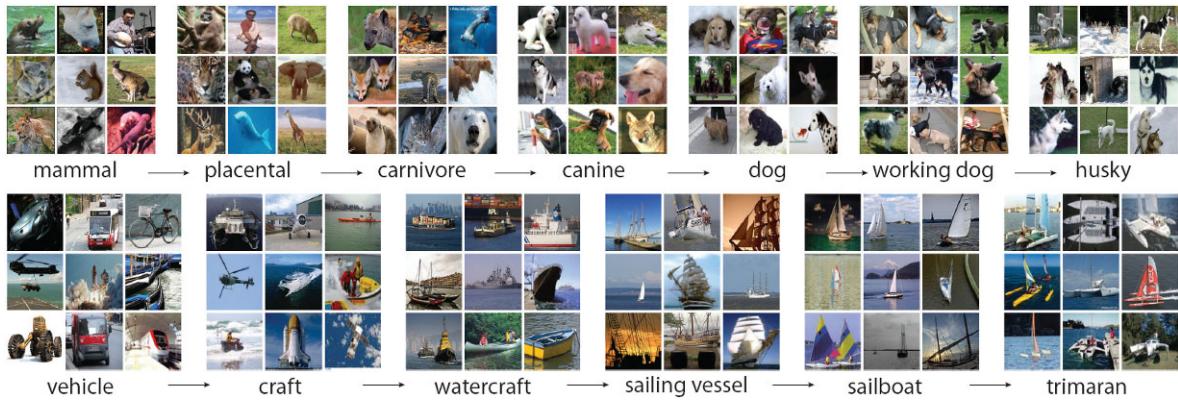


Figure 14.53 ImageNet (Deng, Dong, Socher *et al.* 2009) © 2009 IEEE. This database contains over 500 carefully vetted images for each of 14,841 (as of April, 2010) nouns from the WordNet hierarchy.

14.5.2 Application: Image search

Even though visual recognition algorithms are by some measures still in their infancy, they are already starting to have some impact on image search, i.e., the retrieval of images from the Web using combinations of keywords and visual similarity. Today, most image search engines rely mostly on textual keywords found in captions, nearby text, and filenames, augmented by user click-through data (Craswell and Szummer 2007). As recognition algorithms continue to improve, however, visual features and visual similarity will start being used to recognize images with missing or erroneous keywords.

The topic of searching by visual similarity has a long history and goes by a variety of names, including content-based image retrieval (CBIR) (Smeulders, Worring, Santini *et al.* 2000; Lew, Sebe, Djeraba *et al.* 2006; Vasconcelos 2007; Datta, Joshi, Li *et al.* 2008) and query by image content (QBIC) (Flickner, Sawhney, Niblack *et al.* 1995). Original publications in these fields were based primarily on simple whole-image similarity metrics, such as color and texture (Swain and Ballard 1991; Jacobs, Finkelstein, and Salesin 1995; Manjunath and Ma 1996).

In more recent work, Fergus, Perona, and Zisserman (2004) use a feature-based learning and recognition algorithm to re-rank the outputs from a traditional keyword-based image search engine. In follow-on work, Fergus, Fei-Fei, Perona *et al.* (2005) cluster the results returned by image search using an extension of probabilistic latent semantic analysis (PLSA) (Hofmann 1999) and then select the clusters associated with the highest ranked results as the representative images for that category.

Even more recent work relies on carefully annotated image databases such as LabelMe (Russell, Torralba, Murphy *et al.* 2008). For example, Malisiewicz and Efros (2008) describe a system that, given a query image, can find similar LabelMe images, whereas Liu, Yuen, and Torralba (2009) combine feature-based correspondence algorithms with the labeled database to perform simultaneous recognition and segmentation.

14.6 Recognition databases and test sets

In addition to rapid advances in machine learning and statistical modeling techniques, one of the key ingredients in the continued improvement of recognition algorithms has been the increased availability and quality of image recognition databases.

Tables 14.1 and 14.2, which are based on similar tables in Fei-Fei, Fergus, and Torralba (2009), updated with more recent entries and URLs, show some of the mostly widely used recognition databases. Some of these databases, such as the ones for face recognition and localization, date back over a decade. The most recent ones, such as the PASCAL database, are refreshed annually with ever more challenging problems. Table 14.1 shows examples of databases used primarily for (whole image) recognition while Table 14.2 shows databases where more accurate localization or segmentation information is available and expected.

Ponce, Berg, Everingham *et al.* (2006) discuss some of the problems with earlier datasets and describe how the latest PASCAL Visual Object Classes Challenge aims to overcome these. Some examples of the 20 visual classes in the 2008 challenge are shown in Figure 14.54. The slides from the VOC workshops,²² are a great source for pointers to the best recognition techniques currently available.

Two of the most recent trends in recognition databases are the emergence of Web-based annotation and data collection tools, and the use of search and recognition algorithms to build up databases (Ponce, Berg, Everingham *et al.* 2006). Some of the most interesting work in human annotation of images comes from a series of interactive multi-person games such as ESP (von Ahn and Dabbish 2004) and Peekaboom (von Ahn, Liu, and Blum 2006). In these games, people help each other guess the identity of a hidden image by giving textual clues as to its contents, which implicitly labels either the whole image or just regions. A more “serious” volunteer effort is the LabelMe database, in which vision researchers contribute manual polygonal region annotations in return for gaining access to the database (Russell, Torralba, Murphy *et al.* 2008).

The use of computer vision algorithms for collecting recognition databases dates back to the work of Fergus, Fei-Fei, Perona *et al.* (2005), who cluster the results returned by Google image search using an extension of PLSA and then select the clusters associated with the highest ranked results. More recent examples of related techniques include the work of Berg and Forsyth (2006) and Li and Fei-Fei (2010).

Whatever methods are used to collect and validate recognition databases, they will continue to grow in size, utility, and difficulty from year to year. They will also continue to be an essential component of research into the recognition and scene understanding problems, which remain, as always, the grand challenges of computer vision.

14.7 Additional reading

Although there are currently no specialized textbooks on image recognition and scene understanding, some surveys (Pinz 2005) and collections of papers (Ponce, Hebert, Schmid *et al.* 2006; Dickinson, Leonardis, Schiele *et al.* 2007) can be found that describe the latest approaches. Other good sources of recent research are courses on this topic, such as the ICCV

²² <http://pascallin.ecs.soton.ac.uk/challenges/VOC/>.

| Name / URL | Extents | Contents / Reference |
|---|--------------------------|---|
| <i>Face and person recognition</i> | | |
| Yale face database http://www1.cs.columbia.edu/~belhumeur/ | Centered face images | Frontal faces Belhumeur, Hespanha, and Kriegman (1997) |
| Resources for face detection http://vision.ai.uiuc.edu/mhyang/face-detection-survey.html | Various databases | Faces in various poses Yang, Kriegman, and Ahuja (2002) |
| FERET http://www.frvt.org/FERET | Centered face images | Frontal faces Phillips, Moon, Rizvi <i>et al.</i> (2000) |
| FRVT http://www.frvt.org/ | Centered face images | Faces in various poses Phillips, Scruggs, O'Toole <i>et al.</i> (2010) |
| CMU PIE database http://www.ri.cmu.edu/projects/project_418.html | Centered face image | Faces in various poses Sim, Baker, and Bsat (2003) |
| CMU Multi-PIE database http://multipie.org | Centered face image | Faces in various poses Gross, Matthews, Cohn <i>et al.</i> (2010) |
| Faces in the Wild http://vis-www.cs.umass.edu/lfw/ | Internet images | Faces in various poses Huang, Ramesh, Berg <i>et al.</i> (2007) |
| Consumer image person DB http://chenlab.ece.cornell.edu/people/Andy/GallagherDataset.html | Complete images | People Gallagher and Chen (2008) |
| <i>Object recognition</i> | | |
| Caltech 101 http://www.vision.caltech.edu/Image_Datasets/Caltech101/ | Segmentation masks | 101 categories Fei-Fei, Fergus, and Perona (2006) |
| Caltech 256 http://www.vision.caltech.edu/Image_Datasets/Caltech256/ | Centered objects | 256 categories and clutter Griffin, Holub, and Perona (2007) |
| COIL-100 http://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php | Centered objects | 100 instances Nene, Nayar, and Murase (1996) |
| ETH-80 http://www.mis.tu-darmstadt.de/datasets | Centered objects | 8 instances, 10 views Leibe and Schiele (2003) |
| Instance recognition benchmark http://vis.uky.edu/~stewe/ukbench/ | Objects in various poses | 2550 objects Nistér and Stewénius (2006) |
| Oxford buildings dataset http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/ | Pictures of buildings | 5062 images Philbin, Chum, Isard <i>et al.</i> (2007) |
| NORB http://www.cs.nyu.edu/~ylclab/data/norb-v1.0/ | Bounding box | 50 toys LeCun, Huang, and Bottou (2004) |
| Tiny images http://people.csail.mit.edu/torralba/tinyimages/ | Complete images | 75,000 (Wordnet) things Torralba, Freeman, and Fergus (2008) |
| ImageNet http://www.image-net.org/ | Complete images | 14,000 (Wordnet) things Deng, Dong, Socher <i>et al.</i> (2009) |

Table 14.1 Image databases for recognition, adapted and expanded from Fei-Fei, Fergus, and Torralba (2009).

| Name / URL | Extents | Contents / Reference |
|---|---------------------|--|
| <i>Object detection / localization</i> | | |
| CMU frontal faces http://vasc.ri.cmu.edu/idb/html/face/frontal_images | Patches | Frontal faces Rowley, Baluja, and Kanade (1998a) |
| MIT frontal faces http://cbcl.mit.edu/software-datasets/FaceData2.html | Patches | Frontal faces Sung and Poggio (1998) |
| CMU face detection databases http://www.ri.cmu.edu/research_project_detail.html?project_id=419 | Multiple faces | Faces in various poses Schneiderman and Kanade (2004) |
| UIUC Image DB http://l2r.cs.uiuc.edu/~cogcomp/Data/Car/ | Bounding boxes | Cars Agarwal and Roth (2002) |
| Caltech Pedestrian Dataset http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians/ | Bounding boxes | Pedestrians Dollàr, Wojek, Schiele <i>et al.</i> (2009) |
| Graz-02 Database http://www.emt.tugraz.at/~pinz/data/GRAZ_02/ | Segmentation masks | Bikes, cars, people Opelt, Pinz, Fussenegger <i>et al.</i> (2006) |
| ETHZ Toys http://www.vision.ee.ethz.ch/~calvin/datasets.html | Cluttered images | Toys, boxes, magazines Ferrari, Tuytelaars, and Van Gool (2006b) |
| TU Darmstadt DB http://www.vision.ee.ethz.ch/~bleibe/data/datasets.html | Segmentation masks | Motorbikes, cars, cows Leibe, Leonardis, and Schiele (2008) |
| MSR Cambridge http://research.microsoft.com/en-us/projects/objectclassrecognition/ | Segmentation masks | 23 classes Shotton, Winn, Rother <i>et al.</i> (2009) |
| LabelMe dataset http://labelme.csail.mit.edu/ | Polygonal boundary | >500 categories Russell, Torralba, Murphy <i>et al.</i> (2008) |
| Lotus Hill http://www.imageparsing.com/ | Segmentation masks | Scenes and hierarchies Yao, Yang, Lin <i>et al.</i> (2010) |
| <i>On-line annotation tools</i> | | |
| ESP game http://www.gwap.com/gwap/ | Image descriptions | Web images von Ahn and Dabbish (2004) |
| Peekaboom http://www.gwap.com/gwap/ | Labeled regions | Web images von Ahn, Liu, and Blum (2006) |
| LabelMe http://labelme.csail.mit.edu/ | Polygonal boundary | High-resolution images Russell, Torralba, Murphy <i>et al.</i> (2008) |
| <i>Collections of challenges</i> | | |
| PASCAL http://pascallin.ecs.soton.ac.uk/challenges/VOC/ | Segmentation, boxes | Various Everingham, Van Gool, Williams <i>et al.</i> (2010) |

Table 14.2 Image databases for detection and localization, adapted and expanded from Fei-Fei, Fergus, and Torralba (2009).

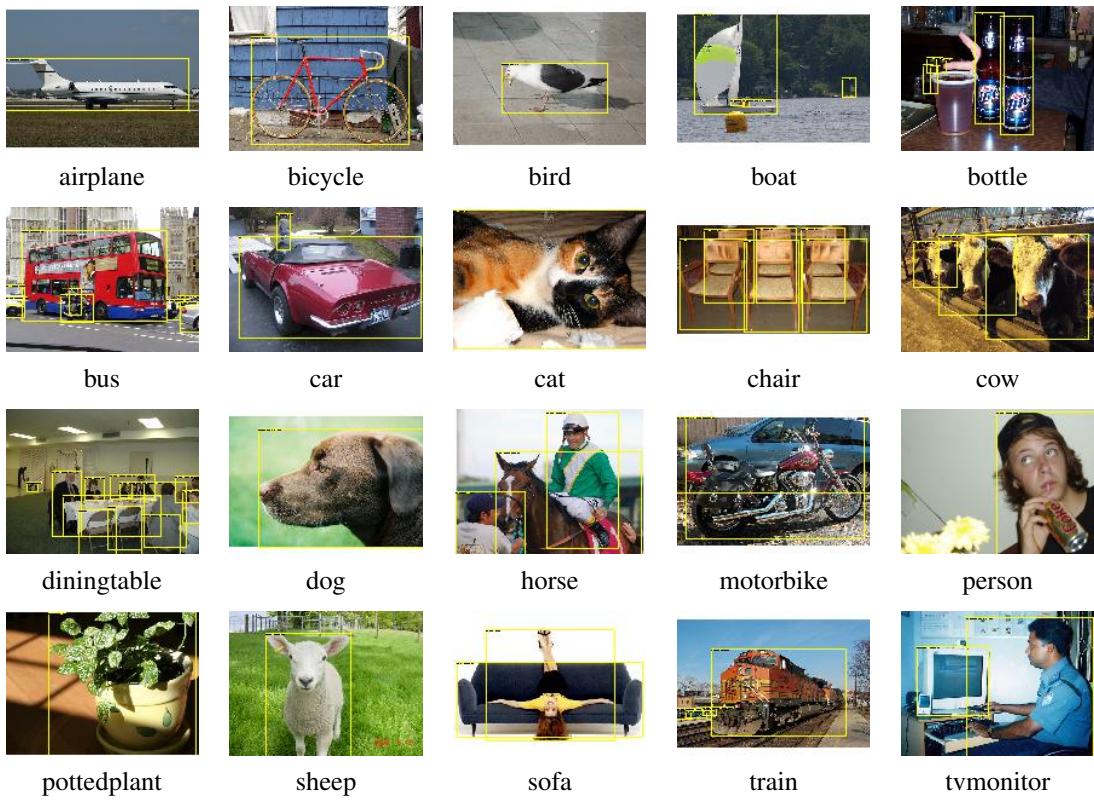


Figure 14.54 Sample images from the PASCAL Visual Object Classes Challenge 2008 (VOC2008) database (Everingham, Van Gool, Williams *et al.* 2008). The original images were obtained from flickr (<http://www.flickr.com/>) and the database rights are explained on <http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2008/>.

2009 short course (Fei-Fei, Fergus, and Torralba 2009) and Antonio Torralba's more comprehensive MIT course (Torralba 2008). The PASCAL VOC Challenge Web site contains workshop slides that summarize today's best performing algorithms.

The literature on face, pedestrian, car, and other object detection is quite extensive. Seminal papers in face detection include those by Osuna, Freund, and Girosi (1997), Sung and Poggio (1998), Rowley, Baluja, and Kanade (1998a), and Viola and Jones (2004), with Yang, Kriegman, and Ahuja (2002) providing a comprehensive survey of early work in this field. More recent examples include (Heisele, Ho, Wu *et al.* 2003; Heisele, Serre, and Poggio 2007).

Early work in pedestrian and car detection was carried out by Gavrila and Philomin (1999), Gavrila (1999), Papageorgiou and Poggio (2000), Mohan, Papageorgiou, and Poggio (2001), and Schneiderman and Kanade (2004). More recent examples include the work of Belongie, Malik, and Puzicha (2002), Mikolajczyk, Schmid, and Zisserman (2004), Dalal and Triggs (2005), Leibe, Seemann, and Schiele (2005), Dalal, Triggs, and Schmid (2006), Opelt, Pinz, and Zisserman (2006), Torralba (2007), Andriluka, Roth, and Schiele (2008), Felzenszwalb, McAllester, and Ramanan (2008), Rogez, Rihan, Ramalingam *et al.* (2008), Andriluka, Roth, and Schiele (2009), Kumar, Zisserman, and H.S.Torr (2009), Dollàr, Belongie, and Perona (2010), and Felzenszwalb, Girshick, McAllester *et al.* (2010).

While some of the earliest approaches to face recognition involved finding the distinc-

tive image features and measuring the distances between them (Fischler and Elschlager 1973; Kanade 1977; Yuille 1991), more recent approaches rely on comparing gray-level images, often projected onto lower dimensional subspaces (Turk and Pentland 1991a; Belhumeur, Hespanha, and Kriegman 1997; Moghaddam and Pentland 1997; Moghaddam, Jebara, and Pentland 2000; Heisele, Ho, Wu *et al.* 2003; Heisele, Serre, and Poggio 2007). Additional details on principal component analysis (PCA) and its Bayesian counterparts can be found in Appendix B.1.1 and books and articles on this topic (Hastie, Tibshirani, and Friedman 2001; Bishop 2006; Roweis 1998; Tipping and Bishop 1999; Leonardis and Bischof 2000; Vidal, Ma, and Sastry 2010). The topics of subspace learning, local distance functions, and metric learning are covered by Cai, He, Hu *et al.* (2007), Frome, Singer, Sha *et al.* (2007), Gillaumin, Verbeek, and Schmid (2009), Ramanan and Baker (2009), and Sivic, Everingham, and Zisserman (2009). An alternative to directly matching gray-level images or patches is to use non-linear local transforms such as local binary patterns (Ahonen, Hadid, and Pietikäinen 2006; Zhao and Pietikäinen 2007; Cao, Yin, Tang *et al.* 2010).

In order to boost the performance of what are essentially 2D appearance-based models, a variety of shape and pose deformation models have been developed (Beymer 1996; Vetter and Poggio 1997), including Active Shape Models (Lanitis, Taylor, and Cootes 1997; Cootes, Cooper, Taylor *et al.* 1995; Davies, Twining, and Taylor 2008), Elastic Bunch Graph Matching (Wiskott, Fellous, Krüger *et al.* 1997), 3D Morphable Models (Blanz and Vetter 1999), and Active Appearance Models (Costen, Cootes, Edwards *et al.* 1999; Cootes, Edwards, and Taylor 2001; Gross, Baker, Matthews *et al.* 2005; Gross, Matthews, and Baker 2006; Matthews, Xiao, and Baker 2007; Liang, Xiao, Wen *et al.* 2008; Ramnath, Koterba, Xiao *et al.* 2008). The topic of head pose estimation, in particular, is covered in a recent survey by Murphy-Chutorian and Trivedi (2009).

Additional information about face recognition can be found in a number of surveys and books on this topic (Chellappa, Wilson, and Sirohey 1995; Zhao, Chellappa, Phillips *et al.* 2003; Li and Jain 2005) as well as on the Face Recognition Web site.²³ Databases for face recognition are discussed by Phillips, Moon, Rizvi *et al.* (2000), Sim, Baker, and Bsat (2003), Gross, Shi, and Cohn (2005), Huang, Ramesh, Berg *et al.* (2007), and Phillips, Scruggs, O'Toole *et al.* (2010).

Algorithms for instance recognition, i.e., the detection of static man-made objects that only vary slightly in appearance but may vary in 3D pose, are mostly based on detecting 2D points of interest and describing them using viewpoint-invariant descriptors (Lowe 2004; Rothganger, Lazebnik, Schmid *et al.* 2006; Ferrari, Tuytelaars, and Van Gool 2006b; Gordon and Lowe 2006; Obdržálek and Matas 2006; Kannala, Rahtu, Brandt *et al.* 2008; Sivic and Zisserman 2009).

As the size of the database being matched increases, it becomes more efficient to quantize the visual descriptors into words (Sivic and Zisserman 2003; Schindler, Brown, and Szeliski 2007; Sivic and Zisserman 2009; Turcot and Lowe 2009), and to then use information-retrieval techniques, such as inverted indices (Nistér and Stewénius 2006; Philbin, Chum, Isard *et al.* 2007; Philbin, Chum, Sivic *et al.* 2008), query expansion (Chum, Philbin, Sivic *et al.* 2007; Agarwal, Snavely, Simon *et al.* 2009), and min hashing (Philbin and Zisserman 2008; Li, Wu, Zach *et al.* 2008; Chum, Philbin, and Zisserman 2008; Chum and Matas 2010) to perform efficient retrieval and clustering.

²³ <http://www.face-rec.org/>.

A number of surveys, collections of papers, and course notes have been written on the topic of category recognition (Pinz 2005; Ponce, Hebert, Schmid *et al.* 2006; Dickinson, Leonardis, Schiele *et al.* 2007; Fei-Fei, Fergus, and Torralba 2009). Some of the seminal papers on the bag of words (bag of keypoints) approach to whole-image category recognition have been written by Csurka, Dance, Fan *et al.* (2004), Lazebnik, Schmid, and Ponce (2006), Csurka, Dance, Perronnin *et al.* (2006), Grauman and Darrell (2007b), and Zhang, Marszałek, Lazebnik *et al.* (2007). Additional and more recent papers in this area include Sivic, Russell, Efros *et al.* (2005), Serre, Wolf, and Poggio (2005), Opelt, Pinz, Fussenegger *et al.* (2006), Grauman and Darrell (2007a), Torralba, Murphy, and Freeman (2007), Boiman, Shechtman, and Irani (2008), Ferencz, Learned-Miller, and Malik (2008), and Mutch and Lowe (2008). It is also possible to recognize objects based on their contours, e.g., using shape contexts (Belongie, Malik, and Puzicha 2002) or other techniques (Jurie and Schmid 2004; Shotton, Blake, and Cipolla 2005; Opelt, Pinz, and Zisserman 2006; Ferrari, Tuytelaars, and Van Gool 2006a).

Many object recognition algorithms use part-based decompositions to provide greater invariance to articulation and pose. Early algorithms focused on the relative positions of the parts (Fischler and Elschlager 1973; Kanade 1977; Yuille 1991) while newer algorithms use more sophisticated models of appearance (Felzenszwalb and Huttenlocher 2005; Fergus, Perona, and Zisserman 2007; Felzenszwalb, McAllester, and Ramanan 2008). Good overviews on part-based models for recognition can be found in the course notes of Fergus 2007b; 2009.

Carneiro and Lowe (2006) discuss a number of graphical models used for part-based recognition, which include trees and stars (Felzenszwalb and Huttenlocher 2005; Fergus, Perona, and Zisserman 2005; Felzenszwalb, McAllester, and Ramanan 2008), k -fans (Crandall, Felzenszwalb, and Huttenlocher 2005; Crandall and Huttenlocher 2006), and constellations (Burl, Weber, and Perona 1998; Weber, Welling, and Perona 2000; Fergus, Perona, and Zisserman 2007). Other techniques that use part-based recognition include those developed by Dorkó and Schmid (2003) and Bar-Hillel, Hertz, and Weinshall (2005).

Combining object recognition with scene segmentation can yield strong benefits. One approach is to pre-segment the image into pieces and then match the pieces to portions of the model (Mori, Ren, Efros *et al.* 2004; Mori 2005; He, Zemel, and Ray 2006; Russell, Efros, Sivic *et al.* 2006; Borenstein and Ullman 2008; Csurka and Perronnin 2008; Gu, Lim, Arbelaez *et al.* 2009). Another is to vote for potential object locations and scales based on object detection (Leibe, Leonardis, and Schiele 2008). One of the currently most popular approaches is to use conditional random fields (Kumar and Hebert 2006; He, Zemel, and Carreira-Perpiñán 2004; He, Zemel, and Ray 2006; Levin and Weiss 2006; Winn and Shotton 2006; Hoiem, Rother, and Winn 2007; Rabinovich, Vedaldi, Galleguillos *et al.* 2007; Verbeek and Triggs 2007; Yang, Meer, and Foran 2007; Batra, Sukthankar, and Chen 2008; Larlus and Jurie 2008; He and Zemel 2008; Shotton, Winn, Rother *et al.* 2009; Kumar, Torr, and Zisserman 2010), which produce some of the best results on the difficult PASCAL VOC segmentation challenge (Shotton, Johnson, and Cipolla 2008; Kohli, Ladický, and Torr 2009).

More and more recognition algorithms are starting to use scene context as part of their recognition strategy. Representative papers in this area include those by Torralba (2003), Torralba, Murphy, Freeman *et al.* (2003), Murphy, Torralba, and Freeman (2003), Torralba, Murphy, and Freeman (2004), Crandall and Huttenlocher (2007), Rabinovich, Vedaldi, Galleguillos *et al.* (2007), Russell, Torralba, Liu *et al.* (2007), Hoiem, Efros, and Hebert (2008a),

Hoiem, Efros, and Hebert (2008b), Sudderth, Torralba, Freeman *et al.* (2008), and Divvala, Hoiem, Hays *et al.* (2009).

Sophisticated machine learning techniques are also becoming a key component of successful object detection and recognition algorithms (Varma and Ray 2007; Felzenszwalb, McAllester, and Ramanan 2008; Fritz and Schiele 2008; Sivic, Russell, Zisserman *et al.* 2008; Vedaldi, Gulshan, Varma *et al.* 2009), as is exploiting large human-labeled databases (Russell, Torralba, Liu *et al.* 2007; Malisiewicz and Efros 2008; Torralba, Freeman, and Fergus 2008; Liu, Yuen, and Torralba 2009). Rough three-dimensional models are also making a comeback for recognition, as evidenced in some recent papers (Savarese and Fei-Fei 2007, 2008; Sun, Su, Savarese *et al.* 2009; Su, Sun, Fei-Fei *et al.* 2009). As always, the latest conferences on computer vision are your best reference for the newest algorithms in this rapidly evolving field.

14.8 Exercises

Ex 14.1: Face detection Build and test one of the face detectors presented in Section 14.1.1.

1. Download one or more of the labeled face detection databases in Table 14.2.
2. Generate your own negative examples by finding photographs that do not contain any people.
3. Implement one of the following face detectors (or devise one of your own):
 - boosting (Algorithm 14.1) based on simple area features, with an optional cascade of detectors (Viola and Jones 2004);
 - PCA face subspace (Moghaddam and Pentland 1997);
 - distances to clustered face and non-face prototypes, followed by a neural network (Sung and Poggio 1998) or SVM (Osuna, Freund, and Girosi 1997) classifier;
 - a multi-resolution neural network trained directly on normalized gray-level patches (Rowley, Baluja, and Kanade 1998a).
4. Test the performance of your detector on the database by evaluating the detector at every location in a sub-octave pyramid. Optionally retrain your detector on false positive examples you get on non-face images.

Ex 14.2: Determining the threshold for AdaBoost Given a set of function evaluations on the training examples \mathbf{x}_i , $f_i = f(\mathbf{x}_i) \in \pm 1$, training labels $y_i \in \pm 1$, and weights $w_i \in (0, 1)$, as explained in Algorithm 14.1, devise an efficient algorithm to find values of θ and $s = \pm 1$ that maximize

$$\sum_i w_i y_i h(s f_i, \theta), \quad (14.43)$$

where $h(x, \theta) = \text{sign}(x - \theta)$.

Ex 14.3: Face recognition using eigenfaces Collect a set of facial photographs and then build a recognition system to re-recognize the same people.

1. Take several photos of each of your classmates and store them.
2. Align the images by automatically or manually detecting the corners of the eyes and using a similarity transform to stretch and rotate each image to a canonical position.
3. Compute the average image and a PCA subspace for the face images
4. Take a new set of photographs a week later and use them as your test set.
5. Compare each new image to each database image and select the nearest one as the recognized identity. Verify that the distance in PCA space is close to the distance computed with a full SSD (sum of squared difference) measure.
6. (Optional) Compute different principal components for identity and expression, and use them to improve your recognition results.

Ex 14.4: Bayesian face recognition Moghaddam, Jebara, and Pentland (2000) compute separate covariance matrices Σ_I and Σ_E by looking at differences between *all* pairs of images. At run time, they select the *nearest* image to determine the facial identity. Does it make sense to estimate statistics for all pairs of images and use them for testing the distance to the nearest exemplar? Discuss whether this is statistically correct.

How is the all-pair intrapersonal covariance matrix Σ_I related to the within-class scatter matrix S_W ? Does a similar relationship hold between Σ_E and S_B ?

Ex 14.5: Modular eigenfaces Extend your face recognition system to separately match the eye, nose, and mouth regions, as shown in Figure 14.18.

1. After normalizing face images to a canonical scale and location, manually segment out some of the eye, nose, and face regions.
2. Build separate detectors for these three (or four) kinds of region, either using a subspace (PCA) approach or one of the techniques presented in Section 14.1.1.
3. For each new image to be recognized, first detect the locations of the facial features.
4. Then, match the individual features against your database and note the locations of these features.
5. Train and test a classifier that uses the individual feature matching IDs as well as (optionally) the feature locations to perform face recognition.

Ex 14.6: Recognition-based color balancing Build a system that recognizes the most important color areas in common photographs (sky, grass, skin) and color balances the image accordingly. Some references and ideas for skin detection are given in Exercise 2.8 and by Forsyth and Fleck (1999), Jones and Rehg (2001), Vezhnevets, Sazonov, and Andreeva (2003), and Kakumanu, Makrigiannis, and Bourbakis (2007). These may give you ideas for how to detect other regions or you can try more sophisticated MRF-based approaches (Shotton, Winn, Rother *et al.* 2009).

Ex 14.7: Pedestrian detection Build and test one of the pedestrian detectors presented in Section 14.1.2.

Ex 14.8: Simple instance recognition Use the feature detection, matching, and alignment algorithms you developed in Exercises 4.1–4.4 and 9.2 to find matching images given a query image or region (Figure 14.26).

Evaluate several feature detectors, descriptors, and robust geometric verification strategies, either on your own or by comparing your results with those of classmates.

Ex 14.9: Large databases and location recognition Extend the previous exercise to larger databases using quantized visual words and information retrieval techniques, as described in Algorithm 14.2.

Test your algorithm on a large database, such as the one used by Nistér and Stewénius (2006) or Philbin, Chum, Sivic *et al.* (2008), which are listed in Table 14.1. Alternatively, use keyword search on the Web or in a photo sharing site (e.g., for a city) to create your own database.

Ex 14.10: Bag of words Adapt the feature extraction and matching pipeline developed in Exercise 14.8 to category (class) recognition, using some of the techniques described in Section 14.4.1.

1. Download the training and test images from one or more of the databases listed in Tables 14.1 and 14.2, e.g., Caltech 101, Caltech 256, or PASCAL VOC.
2. Extract features from each of the training images, quantize them, and compute the $tf-idf$ vectors (bag of words histograms).
3. As an option, consider not quantizing the features and using pyramid matching (14.40–14.41) (Grauman and Darrell 2007b) or using a spatial pyramid for greater selectivity (Lazebnik, Schmid, and Ponce 2006).
4. Choose a classification algorithm (e.g., nearest neighbor classification or support vector machine) and “train” your recognizer, i.e., build up the appropriate data structures (e.g., k-d trees) or set the appropriate classifier parameters.
5. Test your algorithm on the test data set using the same pipeline you developed in steps 2–4 and compare your results to the best reported results.
6. Explain why your results differ from the previously reported ones and give some ideas for how you could improve your system.

You can find a good synopsis of the best-performing classification algorithms and their approaches in the report of the PASCAL Visual Object Classes Challenge found on their Web site (<http://pascallin.ecs.soton.ac.uk/challenges/VOC/>).

Ex 14.11: Object detection and localization Extend the classification algorithm developed in the previous exercise to localize the objects in an image by reporting a bounding box around each detected object. The easiest way to do this is to use a sliding window approach. Some pointers to recent techniques in this area can be found in the workshop associated with the PASCAL VOC 2008 Challenge.

Ex 14.12: Part-based recognition Choose one or more of the techniques described in Section 14.4.2 and implement a part-based recognition system. Since these techniques are fairly involved, you will need to read several of the research papers in this area, select which general approach you want to follow, and then implement your algorithm. A good starting point could be the paper by Felzenszwalb, McAllester, and Ramanan (2008), since it performed well in the PASCAL VOC 2008 detection challenge.

Ex 14.13: Recognition and segmentation Choose one or more of the techniques described in Section 14.4.3 and implement a simultaneous recognition and segmentation system. Since these techniques are fairly involved, you will need to read several of the research papers in this area, select which general approach you want to follow, and then implement your algorithm. Test your algorithm on one or more of the segmentation databases in Table 14.2.

Ex 14.14: Context Implement one or more of the context and scene understanding systems described in Section 14.5 and report on your experience. Does context or whole scene understanding perform better at naming objects than stand-alone systems?

Ex 14.15: Tiny images Download the tiny images database from <http://people.csail.mit.edu/torralba/tinyimages/> and build a classifier based on comparing your test images directly against all of the labeled training images. Does this seem like a promising approach?

Chapter 15

Conclusion

In this book, we have covered a broad range of computer vision topics. Starting with image formation, we have seen how images can be pre-processed to remove noise or blur, segmented into regions, or converted into feature descriptors. Multiple images can be matched and registered, with the results used to estimate motion, track people, reconstruct 3D models, or merge images into more attractive and interesting composites and renderings. Images can also be analyzed to produce semantic descriptions of their content. However, the gap between computer and human performance in this area is still large and is likely to remain so for many years.

Our study has also exposed us to a wide range of mathematical techniques. These include continuous mathematics, such as signal processing, variational approaches, three-dimensional and projective geometry, linear algebra, and least squares. We have also studied topics in discrete mathematics and computer science, such as graph algorithms, combinatorial optimization, and even database techniques for information retrieval. Since many problems in computer vision are inverse problems that involve estimating unknown quantities from noisy input data, we have also looked at Bayesian statistical inference techniques, as well as machine learning techniques to learn probabilistic models from large amounts of training data. As the availability of partially labeled visual imagery on the Internet continues to increase exponentially, this latter approach will continue to have a major impact on our field.

You may ask: why is our field so broad and aren't there any unifying principles that can be used to simplify our study? Part of the answer lies in the expansive definition of computer vision, which is the analysis of images and video, as well as the incredible complexity inherent in the formation of visual imagery. In some ways, our field is as complex as the study of automotive engineering, which requires an understanding of internal combustion, mechanics, aerodynamics, ergonomics, electrical circuitry, and control systems, among other topics. Computer vision similarly draws on a wide variety of sub-disciplines, which makes it challenging to cover in a one-semester course, let alone to achieve mastery during a course of graduate studies. Conversely, the incredible breadth and technical complexity of computer vision problems is what draws many people to this research field.

Because of this richness and the difficulty in making and measuring progress, I have attempted to instill in my students and in readers of this book a discipline founded on principles from engineering, science, and statistics.

The engineering approach to problem solving is to first carefully define the overall problem being tackled and to question the basic assumptions and goals inherent in this process. Once this has been done, a number of alternative solutions or approaches are implemented and carefully tested, paying attention to issues such as reliability and computational cost. Finally, one or more solutions are deployed and evaluated in real-world settings. For this reason, this book contains many different alternatives for solving vision problems, many of which are sketched out in the exercises for students to implement and test on their own.

The scientific approach builds upon a basic understanding of physical principles. In the case of computer vision, this includes the physics of man-made and natural structures, image formation, including lighting and atmospheric effects, optics, and noisy sensors. The task is to then invert this formation using stable and efficient algorithms to obtain reliable descriptions of the scene and other quantities of interest. The scientific approach also encourages us to formulate and test hypotheses, which is similar to the extensive testing and evaluation inherent in engineering disciplines.

Lastly, because so much about the image formation process is inherently uncertain and ambiguous, a statistical approach that models both uncertainty in the world (e.g., the number and types of animals in a picture) and noise in the image formation process, is often essential. Bayesian inference techniques can then be used to combine prior and measurement models to estimate the unknowns and to model their uncertainty. Machine learning techniques can be used to create the probabilistic models in the first place. Efficient learning and inference algorithms, such as dynamic programming, graph cuts, and belief propagation, often play a crucial role in this process.

Given the breadth of material we have covered in this book, what new developments are we likely to see in the future? As I have mentioned before, one of the recent trends in computer vision is using the massive amounts of partially labeled visual data on the Internet as sources for learning visual models of scenes and objects. We have already seen data-driven approaches succeed in related fields such as speech recognition, machine translation, speech and music synthesis, and even computer graphics (both in image-based rendering and animation from motion capture). A similar process has been occurring in computer vision, with some of the most exciting new work occurring at the intersection of the object recognition and machine learning fields.

More traditional quantitative techniques in computer vision such as motion estimation, stereo correspondence, and image enhancement, all benefit from better prior models for images, motions, and disparities, as well as efficient statistical inference techniques such as those for inhomogeneous and higher-order Markov random fields. Some techniques, such as feature matching and structure from motion, have matured to where they can be applied to almost arbitrary collections of images of static scenes. This has resulted in an explosion of work in 3D modeling from Internet datasets, which again is related to visual recognition from massive amounts of data.

While these are all encouraging developments, the gap between human and machine performance in semantic scene understanding remains large. It may be many years before computers can name and outline all of the objects in a photograph with the same skill as a two-year-old child. However, we have to remember that human performance is often the result of many years of training and familiarity and often works best in special ecologically important situations. For example, while humans appear to be experts at face recognition, our actual

performance when shown people we do not know well is not that good. Combining vision algorithms with general inference techniques that reason about the real world will likely lead to more breakthroughs, although some of the problems may turn out to be “AI-complete”, in the sense that a full emulation of human experience and intelligence may be necessary.

Whatever the outcome of these research endeavors, computer vision is already having a tremendous impact in many areas, including digital photography, visual effects, medical imaging, safety and surveillance, and Web-based search. The breadth of the problems and techniques inherent in this field, combined with the richness of the mathematics and the utility of the resulting algorithms, will ensure that this remains an exciting area of study for years to come.

Appendix A

Linear algebra and numerical techniques

| | | |
|-------|---|-----|
| A.1 | Matrix decompositions | 646 |
| A.1.1 | Singular value decomposition | 646 |
| A.1.2 | Eigenvalue decomposition | 647 |
| A.1.3 | QR factorization | 649 |
| A.1.4 | Cholesky factorization | 650 |
| A.2 | Linear least squares | 651 |
| A.2.1 | Total least squares | 653 |
| A.3 | Non-linear least squares | 654 |
| A.4 | Direct sparse matrix techniques | 655 |
| A.4.1 | Variable reordering | 656 |
| A.5 | Iterative techniques | 656 |
| A.5.1 | Conjugate gradient | 657 |
| A.5.2 | Preconditioning | 659 |
| A.5.3 | Multigrid | 660 |

In this appendix, we introduce some elements of linear algebra and numerical techniques that are used elsewhere in the book. We start with some basic decompositions in matrix algebra, including the singular value decomposition (SVD), eigenvalue decompositions, and other matrix decompositions (factorizations). Next, we look at the problem of linear least squares, which can be solved using either the QR decomposition or normal equations. This is followed by non-linear least squares, which arise when the measurement equations are not linear in the unknowns or when robust error functions are used. Such problems require iteration to find a solution. Next, we look at direct solution (factorization) techniques for sparse problems, where the ordering of the variables can have a large influence on the computation and memory requirements. Finally, we discuss iterative techniques for solving large linear (or linearized) least squares problems. Good general references for much of this material include the work by Björck (1996), Golub and Van Loan (1996), Trefethen and Bau (1997), Meyer (2000), Nocedal and Wright (2006), and Björck and Dahlquist (2010).

A note on vector and matrix indexing. To be consistent with the rest of the book and with the general usage in the computer science and computer vision communities, I adopt a 0-based indexing scheme for vector and matrix element indexing. Please note that most mathematical textbooks and papers use 1-based indexing, so you need to be aware of the differences when you read this book.

Software implementations. Highly optimized and tested libraries corresponding to the algorithms described in this appendix are readily available and are listed in Appendix C.2.

A.1 Matrix decompositions

In order to better understand the structure of matrices and more stably perform operations such as inversion and system solving, a number of decompositions (or factorizations) can be used. In this section, we review singular value decomposition (SVD), eigenvalue decomposition, QR factorization, and Cholesky factorization.

A.1.1 Singular value decomposition

One of the most useful decompositions in matrix algebra is the *singular value decomposition* (SVD), which states that any real-valued $M \times N$ matrix \mathbf{A} can be written as

$$\begin{aligned} \mathbf{A}_{M \times N} &= \mathbf{U}_{M \times P} \Sigma_{P \times P} \mathbf{V}_{P \times N}^T & (A.1) \\ &= \left[\mathbf{u}_0 \mid \cdots \mid \mathbf{u}_{p-1} \right] \left[\begin{array}{ccc} \sigma_0 & & \\ & \ddots & \\ & & \sigma_{p-1} \end{array} \right] \left[\begin{array}{c} \mathbf{v}_0^T \\ \vdots \\ \mathbf{v}_{p-1}^T \end{array} \right], \end{aligned}$$

where $P = \min(M, N)$. The matrices \mathbf{U} and \mathbf{V} are orthonormal, i.e., $\mathbf{U}^T \mathbf{U} = \mathbf{I}$ and $\mathbf{V}^T \mathbf{V} = \mathbf{I}$, and so are their column vectors,

$$\mathbf{u}_i \cdot \mathbf{u}_j = \mathbf{v}_i \cdot \mathbf{v}_j = \delta_{ij}. \quad (A.2)$$

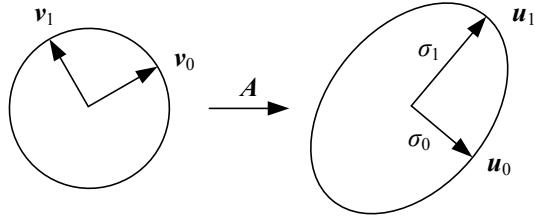


Figure A.1 The action of a matrix \mathbf{A} can be visualized by thinking of the domain as being spanned by a set of orthonormal vectors \mathbf{v}_j , each of which is transformed to a new orthogonal vector \mathbf{u}_j with a length σ_j . When \mathbf{A} is interpreted as a covariance matrix and its eigenvalue decomposition is performed, each of the \mathbf{u}_j axes denote a principal direction (component) and each σ_j denotes one standard deviation along that direction.

The singular values are all non-negative and can be ordered in decreasing order

$$\sigma_0 \geq \sigma_1 \geq \cdots \geq \sigma_{p-1} \geq 0. \quad (\text{A.3})$$

A geometric intuition for the SVD of a matrix \mathbf{A} can be obtained by re-writing $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T$ in (A.2) as

$$\mathbf{AV} = \mathbf{U}\Sigma \quad \text{or} \quad \mathbf{Av}_j = \sigma_j \mathbf{u}_j. \quad (\text{A.4})$$

This formula says that the matrix \mathbf{A} takes any basis vector \mathbf{v}_j and maps it to a direction \mathbf{u}_j with length σ_j , as shown in Figure A.1

If only the first r singular values are positive, the matrix \mathbf{A} is of *rank r* and the index p in the SVD decomposition (A.2) can be replaced by r . (In other words, we can drop the last $p - r$ columns of \mathbf{U} and \mathbf{V} .)

An important property of the singular value decomposition of a matrix (also true for the eigenvalue decomposition of a real symmetric non-negative definite matrix) is that if we truncate the expansion

$$\mathbf{A} = \sum_{j=0}^t \sigma_j \mathbf{u}_j \mathbf{v}_j^T, \quad (\text{A.5})$$

we obtain the best possible least squares approximation to the original matrix \mathbf{A} . This is used both in eigenface-based face recognition systems (Section 14.2.1) and in the separable approximation of convolution kernels (3.21).

A.1.2 Eigenvalue decomposition

If the matrix \mathbf{C} is symmetric ($m = n$),¹ it can be written as an eigenvalue decomposition,

$$\begin{aligned} \mathbf{C} &= \mathbf{U}\Lambda\mathbf{U}^T = \left[\begin{array}{c|c|c} \mathbf{u}_0 & \cdots & \mathbf{u}_{n-1} \end{array} \right] \left[\begin{array}{ccc} \lambda_0 & & \\ & \ddots & \\ & & \lambda_{n-1} \end{array} \right] \left[\begin{array}{c} \mathbf{u}_0^T \\ \vdots \\ \mathbf{u}_{n-1}^T \end{array} \right] \\ &= \sum_{i=0}^{n-1} \lambda_i \mathbf{u}_i \mathbf{u}_i^T. \end{aligned} \quad (\text{A.6})$$

¹ In this appendix, we denote symmetric matrices using \mathbf{C} and general rectangular matrices using \mathbf{A} .

(The eigenvector matrix \mathbf{U} is sometimes written as Φ and the eigenvectors \mathbf{u} as ϕ .) In this case, the eigenvalues

$$\lambda_0 \geq \lambda_1 \geq \cdots \geq \lambda_{n-1} \quad (\text{A.7})$$

can be both positive and negative.²

A special case of the symmetric matrix \mathbf{C} occurs when it is constructed as the sum of a number of outer products

$$\mathbf{C} = \sum_i \mathbf{a}_i \mathbf{a}_i^T = \mathbf{A} \mathbf{A}^T, \quad (\text{A.8})$$

which often occurs when solving least squares problems (Appendix A.2), where the matrix \mathbf{A} consists of all the \mathbf{a}_i column vectors stacked side-by-side. In this case, we are guaranteed that all of the eigenvalues λ_i are non-negative. The associated matrix \mathbf{C} is *positive semi-definite*

$$\mathbf{x}^T \mathbf{C} \mathbf{x} \geq 0, \quad \forall \mathbf{x}. \quad (\text{A.9})$$

If the matrix \mathbf{C} is of full rank, the eigenvalues are all positive and the matrix is called *symmetric positive definite* (SPD).

Symmetric positive semi-definite matrices also arise in the statistical analysis of data, since they represent the *covariance* of a set of $\{\mathbf{x}_i\}$ points around their mean $\bar{\mathbf{x}}$,

$$\mathbf{C} = \frac{1}{n} \sum_i (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T. \quad (\text{A.10})$$

In this case, performing the eigenvalue decomposition is known as *principal component analysis* (PCA), since it models the principal directions (and magnitudes) of variation of the point distribution around their mean, as shown in Section 5.1.1 (5.13–5.15), Section 14.2.1 (14.9), and Appendix B.1.1 (B.10). Figure A.1 shows how the principal components of the covariance matrix \mathbf{C} denote the principal axes \mathbf{u}_j of the uncertainty ellipsoid corresponding to this point distribution and how the $\sigma_j = \sqrt{\lambda_j}$ denote the standard deviations along each axis.

The eigenvalues and eigenvectors of \mathbf{C} and the singular values and singular vectors of \mathbf{A} are closely related. Given

$$\mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T, \quad (\text{A.11})$$

we get

$$\mathbf{C} = \mathbf{A} \mathbf{A}^T = \mathbf{U} \Sigma \mathbf{V}^T \mathbf{V} \Sigma \mathbf{U}^T = \mathbf{U} \Lambda \mathbf{U}^T. \quad (\text{A.12})$$

From this, we see that $\lambda_i = \sigma_i^2$ and that the left singular vectors of \mathbf{A} are the eigenvectors of \mathbf{C} .

This relationship gives us an efficient method for computing the eigenvalue decomposition of large matrices that are rank deficient, such as the scatter matrices observed in computing eigenfaces (Section 14.2.1). Observe that the covariance matrix \mathbf{C} in (14.9) is exactly the same as \mathbf{C} in (A.8). Note also that the individual difference-from-mean images $\mathbf{a}_i = \mathbf{x}_i - \bar{\mathbf{x}}$ are long vectors of length P (the number of pixels in the image), while the total number of exemplars N (the number of faces in the training database) is much smaller. Instead of forming $\mathbf{C} = \mathbf{A} \mathbf{A}^T$, which is $P \times P$, we form the matrix

$$\hat{\mathbf{C}} = \mathbf{A}^T \mathbf{A}, \quad (\text{A.13})$$

² Eigenvalue decompositions can be computed for non-symmetric matrices but the eigenvalues and eigenvectors can have complex entries in that case.

which is $N \times N$. (This involves taking the dot product between every pair of difference images \mathbf{a}_i and \mathbf{a}_j .) The eigenvalues of $\hat{\mathbf{C}}$ are the squared singular values of \mathbf{A} , namely Σ^2 , and are hence also the eigenvalues of \mathbf{C} . The eigenvectors of $\hat{\mathbf{C}}$ are the right singular vectors \mathbf{V} of \mathbf{A} , from which the desired eigenfaces \mathbf{U} , which are the left singular vectors of \mathbf{A} , can be computed as

$$\mathbf{U} = \mathbf{AV}\Sigma^{-1}. \quad (\text{A.14})$$

This final step is essentially computing the eigenfaces as linear combinations of the difference images (Turk and Pentland 1991a). If you have access to a high-quality linear algebra package such as LAPACK, routines for efficiently computing a small number of the left singular vectors and singular values of rectangular matrices such as \mathbf{A} are usually provided (Appendix C.2). However, if storing all of the images in memory is prohibitive, the construction of $\hat{\mathbf{C}}$ in (A.13) can be used instead.

How can eigenvalue and singular value decompositions actually be computed? Notice that an eigenvector is defined by the equation

$$\lambda_i \mathbf{u}_i = \mathbf{Cu}_i \quad \text{or} \quad (\lambda_i \mathbf{I} - \mathbf{C})\mathbf{u}_i = 0. \quad (\text{A.15})$$

(This can be derived from (A.6) by post-multiplying both sides by \mathbf{u}_i .) Since the latter equation is *homogeneous*, i.e., it has a zero right-hand-side, it can only have a non-zero (non-trivial) solution for \mathbf{u}_i if the system is rank deficient, i.e.,

$$|(\lambda \mathbf{I} - \mathbf{C})| = 0. \quad (\text{A.16})$$

Evaluating this determinant yields a *characteristic* polynomial equation in λ , which can be solved for small problems, e.g., 2×2 or 3×3 matrices, in closed form.

For larger matrices, iterative algorithms that first reduce the matrix \mathbf{C} to a real symmetric tridiagonal form using orthogonal transforms and then perform QR iterations are normally used (Golub and Van Loan 1996; Trefethen and Bau 1997; Björck and Dahlquist 2010). Since these techniques are rather involved, it is best to use a linear algebra package such as LAPACK (Anderson, Bai, Bischof *et al.* 1999)—see Appendix C.2.

Factorization with missing data requires different kinds of iterative algorithms, which often involve either hallucinating the missing terms or minimizing some weighted reconstruction metric, which is intrinsically much more challenging than regular factorization. This area has been widely studied in computer vision (Shum, Ikeuchi, and Reddy 1995; De la Torre and Black 2003; Huynh, Hartley, and Heyden 2003; Buchanan and Fitzgibbon 2005; Gross, Matthews, and Baker 2006; Torresani, Hertzmann, and Bregler 2008) and is sometimes called *generalized PCA*. However, this term is also sometimes used to denote algebraic subspace clustering techniques, which is the subject of a forthcoming monograph by Vidal, Ma, and Sastry (2010).

A.1.3 QR factorization

A widely used technique for stably solving poorly conditioned least squares problems (Björck 1996) and as the basis of more complex algorithms, such as computing the SVD and eigenvalue decompositions, is the QR factorization,

$$\mathbf{A} = \mathbf{QR}, \quad (\text{A.17})$$

```

procedure Cholesky( $C, R$ ):
     $R = C$ 
    for  $i = 0 \dots n - 1$ 
        for  $j = i + 1 \dots n - 1$ 
             $R_{j,j:n-1} = R_{j,j:n-1} - r_{ij}r_{ii}^{-1}R_{i,j:n-1}$ 
             $R_{i,i:n-1} = r_{ii}^{-1/2}R_{i,i:n-1}$ 

```

Algorithm A.1 Cholesky decomposition of the matrix C into its upper triangular form R .

where Q is an *orthonormal* (or *unitary*) matrix $QQ^T = I$ and R is upper triangular.³ In computer vision, QR can be used to convert a camera matrix into a rotation matrix and an upper-triangular calibration matrix (6.35) and also in various self-calibration algorithms (Section 7.2.2). The most common algorithms for computing QR decompositions, modified Gram–Schmidt, Householder transformations, and Givens rotations, are described by Golub and Van Loan (1996), Trefethen and Bau (1997), and Björck and Dahlquist (2010) and are also found in LAPACK. Unlike the SVD and eigenvalue decompositions, QR factorization does not require iteration and can be computed exactly in $O(MN^2 + N^3)$ operations, where M is the number of rows and N is the number of columns (for a tall matrix).

A.1.4 Cholesky factorization

Cholesky factorization can be applied to any symmetric positive definite matrix C to convert it into a product of symmetric lower and upper triangular matrices,

$$C = LL^T = R^T R, \quad (\text{A.18})$$

where L is a lower-triangular matrix and R is an upper-triangular matrix. Unlike Gaussian elimination, which may require pivoting (row and column reordering) or may become unstable (sensitive to roundoff errors or reordering), Cholesky factorization remains stable for positive definite matrices, such as those that arise from normal equations in least squares problems (Appendix A.2). Because of the form of (A.18), the matrices L and R are sometimes called *matrix square roots*.⁴

The algorithm to compute an upper triangular Cholesky decomposition of C is a straightforward symmetric generalization of Gaussian elimination and is based on the decomposition (Björck 1996; Golub and Van Loan 1996)

$$C = \begin{bmatrix} \gamma & c^T \\ c & C_{11} \end{bmatrix} \quad (\text{A.19})$$

³ The term “R” comes from the German name for the lower–upper (LU) decomposition, which is LR for “links” and “rechts” (left and right of the diagonal).

⁴ In fact, there exists a whole family of matrix square roots. Any matrix of the form LQ or QR , where Q is a unitary matrix, is a square root of C .

$$= \begin{bmatrix} \gamma^{1/2} & \mathbf{0}^T \\ \mathbf{c}\gamma^{-1/2} & \mathbf{I} \end{bmatrix} \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{C}_{11} - \mathbf{c}\gamma^{-1}\mathbf{c}^T \end{bmatrix} \begin{bmatrix} \gamma^{1/2} & \gamma^{-1/2}\mathbf{c}^T \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \quad (\text{A.20})$$

$$= \mathbf{R}_0^T \mathbf{C}_1 \mathbf{R}_0, \quad (\text{A.21})$$

which, through recursion, can be turned into

$$\mathbf{C} = \mathbf{R}_0^T \dots \mathbf{R}_{n-1}^T \mathbf{R}_{n-1} \dots \mathbf{R}_0 = \mathbf{R}^T \mathbf{R}. \quad (\text{A.22})$$

Algorithm A.1 provides a more procedural definition, which can store the upper-triangular matrix \mathbf{R} in the same space as \mathbf{C} , if desired. The total operation count for Cholesky factorization is $O(N^3)$ for a dense matrix but can be significantly lower for sparse matrices with low fill-in (Appendix A.4).

Note that Cholesky decomposition can also be applied to block-structured matrices, where the term γ in (A.19) is now a square block sub-matrix and \mathbf{c} is a rectangular matrix (Golub and Van Loan 1996). The computation of square roots can be avoided by leaving the γ on the diagonal of the middle factor in (A.20), which results in the $\mathbf{C} = \mathbf{LDL}^T$ factorization, where \mathbf{D} is a diagonal matrix. However, since square roots are relatively fast on modern computers, this is not worth the bother and Cholesky factorization is usually preferred.

A.2 Linear least squares

Least squares fitting problems are pervasive in computer vision. For example, the alignment of images based on matching feature points involves the minimization of a squared distance objective function (6.2),

$$E_{\text{LS}} = \sum_i \|\mathbf{r}_i\|^2 = \sum_i \|\mathbf{f}(\mathbf{x}_i; \mathbf{p}) - \mathbf{x}'_i\|^2, \quad (\text{A.23})$$

where

$$\mathbf{r}_i = \mathbf{f}(\mathbf{x}_i; \mathbf{p}) - \mathbf{x}'_i = \hat{\mathbf{x}}'_i - \tilde{\mathbf{x}}'_i \quad (\text{A.24})$$

is the *residual* between the measured location $\hat{\mathbf{x}}'_i$ and its corresponding current *predicted* location $\tilde{\mathbf{x}}'_i = \mathbf{f}(\mathbf{x}_i; \mathbf{p})$. More complex versions of least squares problems, such as large-scale structure from motion (Section 7.4), may involve the minimization of functions of thousands of variables. Even problems such as image filtering (Section 3.4.3) and regularization (Section 3.7.1) may involve the minimization of sums of squared errors.

Figure A.2a shows an example of a simple least squares line fitting problem, where the quantities being estimated are the line equation parameters (m, b) . When the sampled vertical values y_i are assumed to be noisy versions of points on the line $y = mx + b$, the optimal estimates for (m, b) can be found by minimizing the squared vertical residuals

$$E_{\text{VLS}} = \sum_i |y_i - (mx_i + b)|^2. \quad (\text{A.25})$$

Note that the function being fitted need not itself be linear to use linear least squares. All that is required is that the function be linear in the unknown parameters. For example, polynomial fitting can be written as

$$E_{\text{PLS}} = \sum_i |y_i - (\sum_{j=0}^p a_j x_i^j)|^2, \quad (\text{A.26})$$

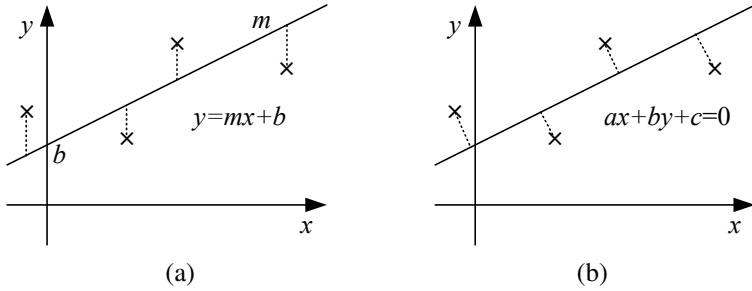


Figure A.2 Least squares regression. (a) The line $y = mx + b$ is fit to the four noisy data points, $\{(x_i, y_i)\}$, denoted by \times by minimizing the squared vertical residuals between the data points and the line, $\sum_i \|y_i - (mx_i + b)\|^2$. (b) When the measurements $\{(x_i, y_i)\}$ are assumed to have noise in all directions, the sum of orthogonal squared distances to the line $\sum_i \|ax_i + by_i + c\|^2$ is minimized using total least squares.

while sinusoid fitting with unknown amplitude A and phase ϕ (but known frequency f) can be written as

$$E_{\text{SLS}} = \sum_i |y_i - A \sin(2\pi f x_i + \phi)|^2 = \sum_i |y_i - (B \sin 2\pi f x_i + C \cos 2\pi f x_i)|^2, \quad (\text{A.27})$$

which is linear in (B, C) .

In general, it is more common to denote the unknown parameters using \boldsymbol{x} and to write the general form of linear least squares as⁵

$$E_{\text{LLS}} = \sum_i |\mathbf{a}_i \boldsymbol{x} - b_i|^2 = \|\mathbf{A}\boldsymbol{x} - \mathbf{b}\|^2. \quad (\text{A.28})$$

Expanding the above equation gives us

$$E_{\text{LLS}} = \boldsymbol{x}^T (\mathbf{A}^T \mathbf{A}) \boldsymbol{x} - 2\boldsymbol{x}^T (\mathbf{A}^T \mathbf{b}) + \|\mathbf{b}\|^2, \quad (\text{A.29})$$

whose minimum value for \boldsymbol{x} can be found by solving the associated *normal equations* (Björck 1996; Golub and Van Loan 1996)

$$(\mathbf{A}^T \mathbf{A}) \boldsymbol{x} = \mathbf{A}^T \mathbf{b}. \quad (\text{A.30})$$

The preferred way to solve the normal equations is to use Cholesky factorization. Let

$$\mathbf{C} = \mathbf{A}^T \mathbf{A} = \mathbf{R}^T \mathbf{R}, \quad (\text{A.31})$$

where \mathbf{R} is the upper-triangular Cholesky factor of the Hessian \mathbf{C} , and

$$\mathbf{d} = \mathbf{A}^T \mathbf{b}. \quad (\text{A.32})$$

After factorization, the solution for \boldsymbol{x} can be obtained as

$$\mathbf{R}^T \mathbf{z} = \mathbf{d}, \quad \mathbf{R}\boldsymbol{x} = \mathbf{z}, \quad (\text{A.33})$$

⁵ Be extra careful in interpreting the variable names here. In the 2D line-fitting example, x is used to denote the horizontal axis, but in the general least squares problem, $\boldsymbol{x} = (m, b)$ denotes the unknown parameter vector.

which involves the solution of two triangular systems, i.e., forward and backward substitution (Björck 1996).

In cases where the least squares problem is numerically poorly conditioned (which should generally be avoided by adding sufficient regularization or prior knowledge about the parameters, (Appendix A.3)), it is possible to use QR factorization or SVD directly on the matrix \mathbf{A} (Björck 1996; Golub and Van Loan 1996; Trefethen and Bau 1997; Nocedal and Wright 2006; Björck and Dahlquist 2010), e.g.,

$$\mathbf{A}\mathbf{x} = \mathbf{Q}\mathbf{R}\mathbf{x} = \mathbf{b} \quad \longrightarrow \quad \mathbf{R}\mathbf{x} = \mathbf{Q}^T\mathbf{b}. \quad (\text{A.34})$$

Note that the upper triangular matrices \mathbf{R} produced by the Cholesky factorization of $\mathbf{C} = \mathbf{A}^T\mathbf{A}$ and the QR factorization of \mathbf{A} are the same, but that solving (A.34) is generally more stable (less sensitive to roundoff error) but slower (by a constant factor).

A.2.1 Total least squares

In some problems, e.g., when performing geometric line fitting in 2D images or 3D plane fitting to point cloud data, instead of having measurement error along one particular axis, the measured points have uncertainty in all directions, which is known as the *errors-in-variables* model (Van Huffel and Lemmerling 2002; Matei and Meer 2006). In this case, it makes more sense to minimize a set of homogeneous squared errors of the form

$$E_{\text{TLS}} = \sum_i (\mathbf{a}_i \cdot \mathbf{x})^2 = \|\mathbf{Ax}\|^2, \quad (\text{A.35})$$

which is known as *total least squares* (TLS) (Van Huffel and Vandewalle 1991; Björck 1996; Golub and Van Loan 1996; Van Huffel and Lemmerling 2002).

The above error metric has a trivial minimum solution at $\mathbf{x} = 0$ and is, in fact, homogeneous in \mathbf{x} . For this reason, we augment this minimization problem with the requirement that $\|\mathbf{x}\|^2 = 1$, which results in the eigenvalue problem

$$\mathbf{x} = \arg \min_{\mathbf{x}} \mathbf{x}^T (\mathbf{A}^T \mathbf{A}) \mathbf{x} \quad \text{such that} \quad \|\mathbf{x}\|^2 = 1. \quad (\text{A.36})$$

The value of \mathbf{x} that minimizes this constrained problem is the eigenvector associated with the smallest eigenvalue of $\mathbf{A}^T \mathbf{A}$. This is the same as the last right singular vector of \mathbf{A} , since

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}, \quad (\text{A.37})$$

$$\mathbf{A}^T \mathbf{A} = \mathbf{V}\Sigma^2\mathbf{V}, \quad (\text{A.38})$$

$$\mathbf{A}^T \mathbf{A} \mathbf{v}_k = \sigma_k^2, \quad (\text{A.39})$$

which is minimized by selecting the smallest σ_k value.

Figure A.2b shows a line fitting problem where, in this case, the measurement errors are assumed to be isotropic in (x, y) . The solution for the best line equation $ax + by + c = 0$ is found by minimizing

$$E_{\text{TLS-2D}} = \sum_i (ax_i + by_i + c)^2, \quad (\text{A.40})$$

i.e., finding the eigenvector associated with the smallest eigenvalue of⁶

$$\mathbf{C} = \mathbf{A}^T \mathbf{A} = \sum_i \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \begin{bmatrix} x_i & y_i & 1 \end{bmatrix}. \quad (\text{A.41})$$

Notice, however, that minimizing $\sum_i (\mathbf{a}_i \mathbf{x})^2$ in (A.35) is only statistically optimal (Appendix B.1.1) if all of the measured terms in the \mathbf{a}_i , e.g., the $(x_i, y_i, 1)$ measurements, have equal noise. This is definitely not the case in the line-fitting example of Figure A.2b (A.40), since the 1 values are noise-free. To mitigate this, we first subtract the mean x and y values from all the measured points

$$\hat{x}_i = x_i - \bar{x} \quad (\text{A.42})$$

$$\hat{y}_i = y_i - \bar{y} \quad (\text{A.43})$$

and then fit the 2D line equation $a(x - \bar{x}) + b(y - \bar{y}) = 0$ by minimizing

$$E_{\text{TLS-2Dm}} = \sum_i (a\hat{x}_i + b\hat{y}_i)^2. \quad (\text{A.44})$$

The more general case where each individual measurement component can have different noise level, as is the case in estimating essential and fundamental matrices (Section 7.2), is called the *heteroscedastic* errors-in-variable (HEIV) model and is discussed by Matei and Meer (2006).

A.3 Non-linear least squares

In many vision problems, such as structure from motion, the least squares problem formulated in (A.23) involves functions $\mathbf{f}(\mathbf{x}_i; \mathbf{p})$ that are *not* linear in the unknown parameters \mathbf{p} . This problem is known as *non-linear least squares* or *non-linear regression* (Björck 1996; Madsen, Nielsen, and Tingleff 2004; Nocedal and Wright 2006). It is usually solved by iteratively re-linearizing (A.23) around the current estimate of \mathbf{p} using the gradient derivative (Jacobian) $\mathbf{J} = \partial \mathbf{f} / \partial \mathbf{p}$ and computing an incremental improvement $\Delta \mathbf{p}$.

As shown in Equations (6.13–6.17), this results in

$$E_{\text{NLS}}(\Delta \mathbf{p}) = \sum_i \|\mathbf{f}(\mathbf{x}_i; \mathbf{p} + \Delta \mathbf{p}) - \mathbf{x}'_i\|^2 \quad (\text{A.45})$$

$$\approx \sum_i \|\mathbf{J}(\mathbf{x}_i; \mathbf{p}) \Delta \mathbf{p} - \mathbf{r}_i\|^2, \quad (\text{A.46})$$

where the Jacobians $\mathbf{J}(\mathbf{x}_i; \mathbf{p})$ and residual vectors \mathbf{r}_i play the same role in forming the normal equations as \mathbf{a}_i and b_i in (A.28).

Because the above approximation only holds near a local minimum or for small values of $\Delta \mathbf{p}$, the update $\mathbf{p} \leftarrow \mathbf{p} + \Delta \mathbf{p}$ may not always decrease the summed square residual error (A.45). One way to mitigate this problem is to take a smaller step,

$$\mathbf{p} \leftarrow \mathbf{p} + \alpha \Delta \mathbf{p}, \quad 0 < \alpha \leq 1. \quad (\text{A.47})$$

⁶ Again, be careful with the variable names here. The measurement equation is $\mathbf{a}_i = (x_i, y_i, 1)$ and the unknown parameters are $\mathbf{x} = (a, b, c)$.

A simple way to determine a reasonable value of α is to start with 1 and successively halve the value, which is a simple form of *line search* (Al-Baali and Fletcher. 1986; Björck 1996; Nocedal and Wright 2006).

Another approach to ensuring a downhill step in error is to add a diagonal damping term to the approximate Hessian

$$\mathbf{C} = \sum_i \mathbf{J}^T(\mathbf{x}_i) \mathbf{J}(\mathbf{x}_i), \quad (\text{A.48})$$

i.e., to solve

$$[\mathbf{C} + \lambda \operatorname{diag}(\mathbf{C})] \Delta \mathbf{p} = \mathbf{d}, \quad (\text{A.49})$$

where

$$\mathbf{d} = \sum_i \mathbf{J}^T(\mathbf{x}_i) \mathbf{r}_i, \quad (\text{A.50})$$

which is called a *damped Gauss–Newton* method. The damping parameter λ is increased if the squared residual is not decreasing as fast as expected, i.e., as predicted by (A.46), and is decreased if the expected decrease is obtained (Madsen, Nielsen, and Tingleff 2004). The combination of the Newton (first-order Taylor series) approximation (A.46) and the adaptive damping parameter λ is commonly known as the Levenberg–Marquardt algorithm (Levenberg 1944; Marquardt 1963) and is an example of more general *trust region methods*, which are discussed in more detail in (Björck 1996; Conn, Gould, and Toint 2000; Madsen, Nielsen, and Tingleff 2004; Nocedal and Wright 2006).

When the initial solution is far away from its quadratic region of convergence around a local minimum, *large residual methods*, e.g., *Newton-type methods*, which add a second-order term to the Taylor series expansion in (A.46), may converge faster. Quasi-Newton methods such as BFGS, which require only gradient evaluations, can also be useful if memory size is an issue. Such techniques are discussed in textbooks and papers on numerical optimization (Toint 1987; Björck 1996; Conn, Gould, and Toint 2000; Nocedal and Wright 2006).

A.4 Direct sparse matrix techniques

Many optimization problems in computer vision, such as bundle adjustment (Szeliski and Kang 1994; Triggs, McLauchlan, Hartley *et al.* 1999; Hartley and Zisserman 2004; Snavely, Seitz, and Szeliski 2008b; Agarwal, Snavely, Simon *et al.* 2009) have Jacobian and (approximate) Hessian matrices that are extremely sparse (Section 7.4.1). For example, Figure 7.9a shows the *bipartite* model typical of structure from motion problems, in which most points are only observed by a subset of the cameras, which results in the sparsity patterns for the Jacobian and Hessian shown in Figure 7.9b–c.

Whenever the Hessian matrix is sparse enough, it is more efficient to use sparse Cholesky factorization instead of regular Cholesky factorization. In such sparse direct techniques, the Hessian matrix \mathbf{C} and its associated Cholesky factor \mathbf{R} are stored in *compressed form*, in which the amount of storage is proportional to the number of (potentially) non-zero entries (Björck 1996; Davis 2006).⁷ Algorithms for computing the non-zero elements in \mathbf{C} and \mathbf{R}

⁷ For example, you can store a list of (i, j, c_{ij}) triples. One example of such a scheme is *compressed sparse row (CSR)* storage. An alternative storage method called *skyline*, which stores adjacent vertical spans of non-zero elements (Bathe 2007), is sometimes used in finite element analysis. Banded systems such as snakes (5.3) can store just the non-zero band elements (Björck 1996, Section 6.2) and can be solved in $O(nb^2)$, where n is the number of

from the sparsity pattern of the Jacobian matrix \mathbf{J} are given by Björck (1996, Section 6.4), and algorithms for computing the numerical Cholesky and QR decompositions (once the sparsity pattern has been computed and storage allocated) are discussed by Björck (1996, Section 6.5).

A.4.1 Variable reordering

The key to efficiently solving sparse problems using direct (non-iterative) techniques is to determine an efficient *ordering* for the variables, which reduces the amount of *fill-in*, i.e., the number of non-zero entries in \mathbf{R} that were zero in the original \mathbf{C} matrix. We already saw in Section 7.4.1 how storing the more numerous 3D point parameters before the camera parameters and using the Schur complement (7.56) results in a more efficient algorithm. Similarly, sorting parameters by time in video-based reconstruction problems usually results in lower fill-in. Furthermore, any problem whose adjacency graph (the graph corresponding to the sparsity pattern) is a tree can be solved in linear time with an appropriate reordering of the variables (putting all the children before their parents). All of these are examples of good reordering techniques.

In the general case of unstructured data, there are many heuristics available to find good reorderings (Björck 1996; Davis 2006).⁸ For general adjacency (sparsity) graphs, *minimum degree orderings* generally produce good results. For planar graphs, which often arise on image or spline grids (Section 8.3), *nested dissection*, which recursively splits the graph into two equal halves along a *frontier* (or boundary) of small size, generally works well. Such *domain decomposition* (or *multi-frontal*) techniques also enable the use of parallel processing, since independent sub-graphs can be processed in parallel on separate processors (Davis 2008).

The overall set of steps used to perform the direct solution of sparse least squares problems are summarized in Algorithm A.2, which is a modified version of Algorithm 6.6.1 by Björck (1996, Section 6.6)). If a series of related least squares problems is being solved, as is the case in iterative non-linear least squares (Appendix A.3), steps 1–3 can be performed ahead of time and reused for each new invocation with different \mathbf{C} and \mathbf{d} values. When the problem is block-structured, as is the case in structure from motion where point (structure) variables have dense 3×3 sub-entries in \mathbf{C} and cameras have 6×6 (or larger) entries, the cost of performing the reordering computation is small compared to the actual numerical factorization, which can benefit from block-structured matrix operations (Golub and Van Loan 1996). It is also possible to apply sparse reordering and multifrontal techniques to QR factorization (Davis 2008), which may be preferable when the least squares problems are poorly conditioned.

A.5 Iterative techniques

When problems become large, the amount of memory required to store the Hessian matrix \mathbf{C} and its factor \mathbf{R} , and the amount of time it takes to compute the factorization, can become prohibitively large, especially when there are large amounts of fill-in. This is often the case with image processing problems defined on pixel grids, since, even with the optimal reordering (nested dissection) the amount of fill can still be large.

variables and b is the bandwidth.

⁸Finding the optimal reordering with minimal fill-in is provably NP-hard.

procedure *SparseCholeskySolve*(\mathbf{C}, \mathbf{d}):

1. Determine symbolically the structure of \mathbf{C} , i.e., the adjacency graph.
2. (Optional) Compute a reordering for the variables, taking into account any block structure inherent in the problem.
3. Determine the fill-in pattern for \mathbf{R} and allocate the compressed storage for \mathbf{R} as well as storage for the permuted right hand side $\hat{\mathbf{d}}$.
4. Copy the elements of \mathbf{C} and \mathbf{d} into \mathbf{R} and $\hat{\mathbf{d}}$, permuting the values according to the computed ordering.
5. Perform the numerical factorization of \mathbf{R} using Algorithm A.1.
6. Solve the factored system (A.33), i.e.,

$$\mathbf{R}^T \mathbf{z} = \hat{\mathbf{d}}, \quad \mathbf{R}\mathbf{x} = \mathbf{z}.$$

7. Return the solution \mathbf{x} , after undoing the permutation.

Algorithm A.2 Sparse least squares using a sparse Cholesky decomposition of the matrix \mathbf{C} .

A preferable approach to solving such linear systems is to use iterative techniques, which compute a series of estimates that converge to the final solution, e.g., by taking a series of downhill steps in an energy function such as (A.29).

A large number of iterative techniques have been developed over the years, including such well-known algorithms as successive overrelaxation and multi-grid. These are described in specialized textbooks on iterative solution techniques (Axelsson 1996; Saad 2003) as well as in more general books on numerical linear algebra and least squares techniques (Björck 1996; Golub and Van Loan 1996; Trefethen and Bau 1997; Nocedal and Wright 2006; Björck and Dahlquist 2010).

A.5.1 Conjugate gradient

The iterative solution technique that often performs best is conjugate gradient descent, which takes a series of downhill steps that are *conjugate* to each other with respect to the \mathbf{C} matrix, i.e., if the \mathbf{u} and \mathbf{v} descent directions satisfy $\mathbf{u}^T \mathbf{C} \mathbf{v} = 0$. In practice, conjugate gradient descent outperforms other kinds of gradient descent algorithm because its convergence rate is proportional to the square root of the *condition number* of \mathbf{C} instead of the condition number itself.⁹ Shewchuk (1994) provides a nice introduction to this topic, with clear intuitive explanations of the reasoning behind the conjugate gradient algorithm and its performance.

Algorithm A.3 describes the conjugate gradient algorithm and its related least squares counterpart, which can be used when the original set of least squares linear equations are

⁹ The condition number $\kappa(\mathbf{C})$ is the ratio of the largest and smallest eigenvalues of \mathbf{C} . The actual convergence rate depends on the clustering of the eigenvalues, as discussed in the references cited in this section.

$$\text{ConjugateGradient}(\mathbf{C}, \mathbf{d}, \mathbf{x}_0)$$

1. $\mathbf{r}_0 = \mathbf{d} - \mathbf{C}\mathbf{x}_0$
2. $\mathbf{p}_0 = \mathbf{r}_0$
3. **for** $k = 0 \dots$
4. $\mathbf{w}_k = \mathbf{C}\mathbf{p}_k$
5. $\alpha_k = \|\mathbf{r}_k\|^2 / (\mathbf{p}_k \cdot \mathbf{w}_k)$
6. $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$
7. $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{w}_k$
- 8.
9. $\beta_{k+1} = \|\mathbf{r}_{k+1}\|^2 / \|\mathbf{r}_k\|^2$
10. $\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$

$$\text{ConjugateGradientLS}(\mathbf{A}, \mathbf{b}, \mathbf{x}_0)$$

1. $\mathbf{q}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0, \quad \mathbf{r}_0 = \mathbf{A}^T \mathbf{q}_0$
2. $\mathbf{p}_0 = \mathbf{r}_0$
3. **for** $k = 0 \dots$
4. $\mathbf{v}_k = \mathbf{A}\mathbf{p}_k$
5. $\alpha_k = \|\mathbf{r}_k\|^2 / \|\mathbf{v}_k\|^2$
6. $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$
7. $\mathbf{q}_{k+1} = \mathbf{q}_k - \alpha_k \mathbf{v}_k$
8. $\mathbf{r}_{k+1} = \mathbf{A}^T \mathbf{q}_{k+1}$
9. $\beta_{k+1} = \|\mathbf{r}_{k+1}\|^2 / \|\mathbf{r}_k\|^2$
10. $\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$

Algorithm A.3 Conjugate gradient and conjugate gradient least squares algorithms. The algorithm is described in more detail in the text, but in brief, they choose descent directions \mathbf{p}_k that are conjugate to each other with respect to \mathbf{C} by computing a factor β by which to discount the previous search direction \mathbf{p}_{k-1} . They then find the optimal step size α and take a downhill step by an amount $\alpha_k \mathbf{p}_k$.

available in the form of $\mathbf{Ax} = \mathbf{b}$ (A.28). While it is easy to convince yourself that the two forms are mathematically equivalent, the least squares form is preferable if rounding errors start to affect the results because of poor conditioning. It may also be preferable if, due to the sparsity structure of \mathbf{A} , multiplies with the original \mathbf{A} matrix are faster or more space efficient than multiplies with \mathbf{C} .

The conjugate gradient algorithm starts by computing the current residual $\mathbf{r}_0 = \mathbf{d} - \mathbf{Cx}_0$, which is the direction of steepest descent of the energy function (A.28). It sets the original descent direction $\mathbf{p}_0 = \mathbf{r}_0$. Next, it multiplies the descent direction by the quadratic form (Hessian) matrix \mathbf{C} and combines this with the residual to estimate the optimal step size α_k . The solution vector \mathbf{x}_k and the residual vector \mathbf{r}_k are then updated using this step size. (Notice how the least squares variant of the conjugate gradient algorithm splits the multiplication by the $\mathbf{C} = \mathbf{A}^T \mathbf{A}$ matrix across steps 4 and 8.) Finally, a new search direction is calculated by first computing a factor β as the ratio of current to previous residual magnitudes. The new search direction \mathbf{p}_{k+1} is then set to the residual plus β times the old search direction \mathbf{p}_k , which keeps the directions conjugate with respect to \mathbf{C} .

It turns out that conjugate gradient descent can also be directly applied to non-quadratic energy functions, e.g., those arising from non-linear least squares (Appendix A.3). Instead of explicitly forming a local quadratic approximation \mathbf{C} and then computing residuals \mathbf{r}_k , non-linear conjugate gradient descent computes the gradient of the energy function E (A.45) directly inside each iteration and uses it to set the search direction (Nocedal and Wright 2006). Since the quadratic approximation to the energy function may not exist or may be inaccurate,

line search is often used to determine the step size α_k . Furthermore, to compensate for errors in finding the true function minimum, alternative formulas for β_{k+1} such as Polak–Ribière,

$$\beta_{k+1} = \frac{\nabla E(\mathbf{x}_{k+1})[\nabla E(\mathbf{x}_{k+1}) - \nabla E(\mathbf{x}_k)]}{\|\nabla E(\mathbf{x}_k)\|^2} \quad (\text{A.51})$$

are often used (Nocedal and Wright 2006).

A.5.2 Preconditioning

As we mentioned previously, the rate of convergence of the conjugate gradient algorithm is governed in large part by the condition number $\kappa(\mathbf{C})$. Its effectiveness can therefore be increased dramatically by reducing this number, e.g., by rescaling elements in \mathbf{x} , which corresponds to rescaling rows and columns in \mathbf{C} .

In general, preconditioning is usually thought of as a change of basis from the vector \mathbf{x} to a new vector

$$\hat{\mathbf{x}} = \mathbf{S}\mathbf{x}. \quad (\text{A.52})$$

The corresponding linear system being solved then becomes

$$\mathbf{A}\mathbf{S}^{-1}\hat{\mathbf{x}} = \mathbf{S}^{-1}\mathbf{b} \quad \text{or} \quad \hat{\mathbf{A}}\hat{\mathbf{x}} = \hat{\mathbf{b}}, \quad (\text{A.53})$$

with a corresponding least squares energy (A.29) of the form

$$E_{\text{PLS}} = \hat{\mathbf{x}}^T(\mathbf{S}^{-T}\mathbf{C}\mathbf{S}^{-1})\hat{\mathbf{x}} - 2\hat{\mathbf{x}}^T(\mathbf{S}^{-T}\mathbf{d}) + \|\hat{\mathbf{b}}\|^2. \quad (\text{A.54})$$

The actual preconditioned matrix $\hat{\mathbf{C}} = \mathbf{S}^{-T}\mathbf{C}\mathbf{S}^{-1}$ is usually not explicitly computed. Instead, Algorithm A.3 is extended to insert \mathbf{S}^{-T} and \mathbf{S}^T operations at the appropriate places (Björck 1996; Golub and Van Loan 1996; Trefethen and Bau 1997; Saad 2003; Nocedal and Wright 2006).

A good preconditioner \mathbf{S} is easy and cheap to compute, but is also a decent approximation to a square root of \mathbf{C} , so that $\kappa(\mathbf{S}^{-T}\mathbf{C}\mathbf{S}^{-1})$ is closer to 1. The simplest such choice is the square root of the diagonal matrix $\mathbf{S} = \mathbf{D}^{1/2}$, with $\mathbf{D} = \text{diag}(\mathbf{C})$. This has the advantage that any scalar change in variables (e.g., using radians instead of degrees for angular measurements) has no effect on the range of convergence of the iterative technique. For problems that are naturally block-structured, e.g., for structure from motion, where 3D point positions or 6D camera poses are being estimated, a block diagonal preconditioner is often a good choice.

A wide variety of more sophisticated preconditioners have been developed over the years (Björck 1996; Golub and Van Loan 1996; Trefethen and Bau 1997; Saad 2003; Nocedal and Wright 2006), many of which can be directly applied to problems in computer vision (Byröd and øAström 2009; Jeong, Nistér, Steedly *et al.* 2010; Agarwal, Snavely, Seitz *et al.* 2010). Some of these are based on an *incomplete Cholesky* factorization of \mathbf{C} , i.e., one in which the amount of fill-in in \mathbf{R} is strictly limited, e.g., to just the original non-zero elements in \mathbf{C} .¹⁰ Other preconditioners are based on a sparsified, e.g., tree-based or clustered, approximation to \mathbf{C} (Koutis 2007; Koutis and Miller 2008; Grady 2008; Koutis, Miller, and Tolliver 2009), since these are known to have efficient inversion properties.

¹⁰ If a complete Cholesky factorization $\mathbf{C} = \mathbf{R}^T\mathbf{R}$ is used, we get $\hat{\mathbf{C}} = \mathbf{R}^{-T}\mathbf{C}\mathbf{R}^{-1} = \mathbf{I}$ and all iterative algorithms converge in a single step, thereby obviating the need to use them, but the complete factorization is often too expensive. Note that incomplete factorization can also benefit from reordering.

For grid-based image-processing applications, *parallel* or *hierarchical* preconditioners often perform extremely well (Yserentant 1986; Szeliski 1990b; Pentland 1994; Saad 2003; Szeliski 2006b). These approaches use a change of basis transformation S that resembles the pyramidal or wavelet representations discussed in Section 3.5, and are hence amenable to parallel and GPU-based implementations. Coarser elements in the new representation quickly converge to the low-frequency components in the solution, while finer-level elements encode the higher-frequency components. Some of the relationships between hierarchical preconditioners, incomplete Cholesky factorization, and multigrid techniques are explored by Saad (2003) and Szeliski (2006b).

A.5.3 Multigrid

One other class of iterative techniques widely used in computer vision is *multigrid* techniques (Briggs, Henson, and McCormick 2000; Trottenberg, Oosterlee, and Schuller 2000), which have been applied to problems such as surface interpolation (Terzopoulos 1986a), optical flow (Terzopoulos 1986a; Bruhn, Weickert, Kohlberger *et al.* 2006), high dynamic range tone mapping (Fattal, Lischinski, and Werman 2002), colorization (Levin, Lischinski, and Weiss 2004), natural image matting (Levin, Lischinski, and Weiss 2008), and segmentation (Grady 2008).

The main idea behind multigrid is to form coarser (lower-resolution) versions of the problems and use them to compute the low-frequency components of the solution. However, unlike simple coarse-to-fine techniques, which use the coarse solutions to initialize the fine solution, multigrid techniques only *correct* the low-frequency component of the current solution and use multiple rounds of coarsening and refinement (in what are often called “V” and “W” patterns of motion across the pyramid) to obtain rapid convergence.

On certain simple homogeneous problems (such as solving Poisson equations), multigrid techniques can achieve optimal performance, i.e., computation times linear in the number of variables. However, for more inhomogeneous problems or problems on irregular grids, variants on these techniques, such as *algebraic multigrid* (AMG) approaches, which look at the structure of C to derive coarse level problems, may be preferable. Saad (2003) has a nice discussion of the relationship between multigrid and parallel preconditioners and on the relative merits of using multigrid or conjugate gradient approaches.

Appendix B

Bayesian modeling and inference

| | | |
|-------|---|-----|
| B.1 | Estimation theory | 662 |
| B.1.1 | Likelihood for multivariate Gaussian noise | 663 |
| B.2 | Maximum likelihood estimation and least squares | 665 |
| B.3 | Robust statistics | 666 |
| B.4 | Prior models and Bayesian inference | 667 |
| B.5 | Markov random fields | 668 |
| B.5.1 | Gradient descent and simulated annealing | 670 |
| B.5.2 | Dynamic programming | 670 |
| B.5.3 | Belief propagation | 672 |
| B.5.4 | Graph cuts | 674 |
| B.5.5 | Linear programming | 676 |
| B.6 | Uncertainty estimation (error analysis) | 678 |

The following problem commonly recurs in this book: Given a number of measurements (images, feature positions, etc.), estimate the values of some unknown structure or parameter (camera positions, object shape, etc.). These kinds of problems are in general called *inverse* problems because they involve estimating unknown model parameters instead of simulating the forward formation equations.¹ Computer graphics is a classic forward modeling problem (given some objects, cameras, and lighting, simulate the images that would result), while computer vision problems are usually of the inverse kind (given one or more images, recover the scene that gave rise to these images).

Given an instance of an inverse problem, there are, in general, several ways to proceed. For instance, through clever (or sometimes straightforward) algebraic manipulation, a closed form solution for the unknowns can sometimes be derived. Consider, for example, the *camera matrix calibration* problem (Section 6.2.1): given an image of a calibration pattern consisting of known 3D point positions, compute the 3×4 camera matrix \mathbf{P} that maps these points onto the image plane.

In more detail, we can write this problem as (6.33–6.34)

$$x_i = \frac{p_{00}X_i + p_{01}Y_i + p_{02}Z_i + p_{03}}{p_{20}X_i + p_{21}Y_i + p_{22}Z_i + p_{23}} \quad (\text{B.1})$$

$$y_i = \frac{p_{10}X_i + p_{11}Y_i + p_{12}Z_i + p_{13}}{p_{20}X_i + p_{21}Y_i + p_{22}Z_i + p_{23}}, \quad (\text{B.2})$$

where (x_i, y_i) is the feature position of the i th point measured in the image plane, (X_i, Y_i, Z_i) is the corresponding 3D point position, and the p_{ij} are the unknown entries of the camera matrix \mathbf{P} . Moving the denominator over to the left hand side, we end up with a set of simultaneous linear equations,

$$x_i(p_{20}X_i + p_{21}Y_i + p_{22}Z_i + p_{23}) = p_{00}X_i + p_{01}Y_i + p_{02}Z_i + p_{03}, \quad (\text{B.3})$$

$$y_i(p_{20}X_i + p_{21}Y_i + p_{22}Z_i + p_{23}) = p_{10}X_i + p_{11}Y_i + p_{12}Z_i + p_{13}, \quad (\text{B.4})$$

which we can solve using linear least squares (Appendix A.2) to obtain an estimate of \mathbf{P} .

The question then arises: is this set of equations the right ones to be solving? If the measurements are totally noise-free or we do not care about getting the best possible answer, then the answer is yes. However, in general, we cannot be sure that we have a reasonable algorithm unless we make a model of the likely sources of error and devise an algorithm that performs as well as possible given these potential errors.

B.1 Estimation theory

The study of such inference problems from noisy data is often called *estimation theory* (Gelb 1974), and its extension to problems where we explicitly choose a loss function is called *statistical decision theory* (Berger 1993; Hastie, Tibshirani, and Friedman 2001; Bishop 2006; Robert 2007). We first start by writing down the forward process that leads from our unknowns (and knowns) to a set of noise-corrupted measurements. We then devise an algorithm that will give us an estimate (or set of estimates) that are both insensitive to the noise (as best they can be) and also quantify the reliability of these estimates.

¹ In machine learning, these problems are called *regression problems*, because we are trying to estimate a *continuous* quantity from noisy inputs, as opposed to a discrete *classification* task (Bishop 2006).

The specific equations above (B.1) are just a particular instance of a more general set of *measurement equations*,

$$\mathbf{y}_i = \mathbf{f}_i(\mathbf{x}) + \mathbf{n}_i. \quad (\text{B.5})$$

Here, the \mathbf{y}_i are the noise-corrupted *measurements*, e.g., (x_i, y_i) in Equation (B.1), and \mathbf{x} is the unknown *state vector*.²

Each measurement comes with its associated *measurement model* $\mathbf{f}_i(\mathbf{x})$, which maps the unknown into that particular measurement. An alternative formulation would be to have one general function $\mathbf{f}(\mathbf{x}, \mathbf{p}_i)$ and to use a per-measurement parameter vector \mathbf{p}_i to distinguish between different measurements, e.g., (X_i, Y_i, Z_i) in Equation (B.1). Note that the use of the $\mathbf{f}_i(\mathbf{x})$ form makes it straightforward to have measurements of different dimensions, which becomes useful when we start adding in prior information (Appendix B.4).

Each measurement is also contaminated with some noise \mathbf{n}_i . In Equation (B.5), we have indicated that \mathbf{n}_i is a zero-mean normal (Gaussian) random variable with a covariance matrix Σ_i . In general, the noise need not be Gaussian and, in fact, it is usually prudent to assume that some measurements may be outliers. However, we defer this discussion to Appendix B.3, after we have explored the simpler Gaussian noise case more fully. We also assume that the noise vectors \mathbf{n}_i are independent. In the case where they are not (e.g., when some constant gain or offset contaminates all of the pixels in a given image), we can add this effect as a *nuisance parameter* to our state vector \mathbf{x} and later estimate its value (and discard it, if so desired).

B.1.1 Likelihood for multivariate Gaussian noise

Given all of the noisy measurements $\mathbf{y} = \{\mathbf{y}_i\}$, we would like to infer a probability distribution on the unknown \mathbf{x} vector. We can write the *likelihood* of having observed the $\{\mathbf{y}_i\}$ given a particular value of \mathbf{x} as

$$L = p(\mathbf{y}|\mathbf{x}) = \prod_i p(\mathbf{y}_i|\mathbf{x}) = \prod_i p(\mathbf{y}_i|\mathbf{f}_i(\mathbf{x})) = \prod_i p(\mathbf{n}_i). \quad (\text{B.6})$$

When each noise vector \mathbf{n}_i is a multivariate Gaussian with covariance Σ_i ,

$$\mathbf{n}_i \sim \mathcal{N}(0, \Sigma_i), \quad (\text{B.7})$$

we can write this likelihood as

$$\begin{aligned} L &= \prod_i |2\pi\Sigma_i|^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{y}_i - \mathbf{f}_i(\mathbf{x}))^T \Sigma_i^{-1} (\mathbf{y}_i - \mathbf{f}_i(\mathbf{x}))\right) \\ &= \prod_i |2\pi\Sigma_i|^{-1/2} \exp\left(-\frac{1}{2}\|\mathbf{y}_i - \mathbf{f}_i(\mathbf{x})\|_{\Sigma_i^{-1}}^2\right), \end{aligned} \quad (\text{B.8})$$

where the matrix norm $\|\mathbf{x}\|_{\mathbf{A}}^2$ is a shorthand notation for $\mathbf{x}^T \mathbf{A} \mathbf{x}$.

The norm $\|\mathbf{y}_i - \bar{\mathbf{y}}_i\|_{\Sigma_i^{-1}}$ is often called the *Mahalanobis distance* (5.26 and 14.14) and is used to measure the distance between a measurement and the mean of a multivariate Gaussian distribution. Contours of equal Mahalanobis distance are equi-probability contours. Note

² In the Kalman filtering literature (Gelb 1974), it is more common to use \mathbf{z} instead of \mathbf{y} to denote measurements.

that when the measurement covariance is isotropic (the same in all directions), i.e., when $\Sigma_i = \sigma_i^2 \mathbf{I}$, the likelihood can be written as

$$L = \prod_i (2\pi\sigma_i^2)^{-N_i/2} \exp\left(-\frac{1}{2\sigma_i^2}\|\mathbf{y}_i - \mathbf{f}_i(\mathbf{x})\|^2\right), \quad (\text{B.9})$$

where N_i is the length of the i th measurement vector \mathbf{y}_i .

We can more easily visualize the structure of the covariance matrix and the corresponding Mahalanobis distance if we first perform an *eigenvalue* or *principal component* analysis (PCA) of the covariance matrix (A.6),

$$\Sigma = \Phi \operatorname{diag}(\lambda_0 \dots \lambda_{N-1}) \Phi^T. \quad (\text{B.10})$$

Equal-probability contours of the corresponding multi-variate Gaussian, which are also equi-distance contours in the Mahalanobis distance (Figure 14.14), are multi-dimensional ellipsoids whose axis directions are given by the columns of Φ (the *eigenvectors*) and whose lengths are given by the $\sigma_j = \sqrt{\lambda_j}$ (Figure A.1).

It is usually more convenient to work with the negative log likelihood, which we can think of as a *cost* or *energy*

$$E = -\log L = \frac{1}{2} \sum_i (\mathbf{y}_i - \mathbf{f}_i(\mathbf{x}))^T \Sigma_i^{-1} (\mathbf{y}_i - \mathbf{f}_i(\mathbf{x})) + k \quad (\text{B.11})$$

$$= \frac{1}{2} \sum_i \|\mathbf{y}_i - \mathbf{f}_i(\mathbf{x})\|_{\Sigma_i^{-1}}^2 + k, \quad (\text{B.12})$$

where $k = \sum_i \log |2\pi\Sigma_i|$ is a constant that depends on the measurement variances, but is independent of \mathbf{x} .

Notice that the inverse covariance $C_i = \Sigma_i^{-1}$ plays the role of a *weight* on each of the measurement error *residuals*, i.e., the difference between the contaminated measurement \mathbf{y}_i and its uncontaminated (predicted) value $\mathbf{f}_i(\mathbf{x})$. In fact, the inverse covariance is often called the (Fisher) *information matrix* (Bishop 2006), since it tells us how much information is contained in a given measurement, i.e., how well it constrains the final estimate. We can also think of this matrix as denoting the amount of *confidence* to associate with each measurement (hence the letter C).

In this formulation, it is quite acceptable for some information matrices to be singular (of degenerate rank) or even zero (if the measurement is missing altogether). Rank-deficient measurements often occur, for example, when using a line feature or edge to measure a 3D edge-like feature, since its exact position along the edge is unknown (of infinite or extremely large variance) §8.1.3.

In order to make the distinction between the noise contaminated measurement and its expected value for a particular setting of \mathbf{x} more explicit, we adopt the notation $\tilde{\mathbf{y}}$ for the former (think of the tilde as the approximate or noisy value) and $\hat{\mathbf{y}} = \mathbf{f}_i(\mathbf{x})$ for the latter (think of the hat as the predicted or expected value). We can then write the negative log likelihood as

$$E = -\log L = \sum_i \|\tilde{\mathbf{y}}_i - \hat{\mathbf{y}}_i\|_{\Sigma_i^{-1}}^2 + k. \quad (\text{B.13})$$

B.2 Maximum likelihood estimation and least squares

Now that we have presented the likelihood and log likelihood functions, how can we find the optimal value for our state estimate \mathbf{x} ? One plausible choice might be to select the value of \mathbf{x} that maximizes $L = p(\mathbf{y}|\mathbf{x})$. In fact, in the absence of any prior model for \mathbf{x} (Appendix B.4), we have

$$L = p(\mathbf{y}|\mathbf{x}) = p(\mathbf{y}, \mathbf{x}) = p(\mathbf{x}|\mathbf{y}).$$

Therefore, choosing the value of \mathbf{x} that maximizes the likelihood is equivalent to choosing the maximum of our probability density estimate for \mathbf{x} .

When might this be a good idea? If the data (measurements) constrain the possible values of \mathbf{x} so that they all cluster tightly around one value (e.g., if the distribution $p(\mathbf{x}|\mathbf{y})$ is a unimodal Gaussian), the maximum likelihood estimate is the optimal one in that it is both unbiased and has the least possible variance. In many other cases, e.g., if a single estimate is all that is required, it is still often the best estimate.³ However, if the probability is multi-modal, i.e., it has several local minima in the log likelihood (Figure 5.7), much more care may be required. In particular, it might be necessary to defer certain decisions (such as the ultimate position of an object being tracked) until more measurements have been taken. The CONDENSATION algorithm presented in Section 5.1.2 is one possible method for modeling and updating such multi-modal distributions but is just one example of more general *particle filtering* and *Markov Chain Monte Carlo* (MCMC) techniques (Andrieu, de Freitas, Doucet *et al.* 2003; Bishop 2006; Koller and Friedman 2009).

Another possible way to choose the best estimate is to maximize the *expected utility* (or, conversely, to minimize the expected risk or loss) associated with obtaining the correct estimate, i.e., by minimizing

$$E_{\text{loss}}(\mathbf{x}, \mathbf{y}) = \int l(\mathbf{x} - \mathbf{z}) p(\mathbf{z}|\mathbf{y}) d\mathbf{z}. \quad (\text{B.14})$$

For example, if a robot wants to avoid hitting a wall at all costs, the loss function will be high whenever the estimate underestimates the true distance to the wall. When $l(\mathbf{x} - \mathbf{y}) = \delta(\mathbf{x} - \mathbf{y})$, we obtain the maximum likelihood estimate, whereas when $l(\mathbf{x} - \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^2$, we obtain the *mean square error* (MSE) or *expected value* estimate. The explicit modeling of a utility or loss function is what characterizes *statistical decision theory* (Berger 1993; Hastie, Tibshirani, and Friedman 2001; Bishop 2006; Robert 2007).

How do we find the maximum likelihood estimate? If the measurement noise is Gaussian, we can minimize the quadratic objective function (B.13). This becomes even simpler if the measurement equations are linear, i.e.,

$$\mathbf{f}_i(\mathbf{x}) = \mathbf{H}_i \mathbf{x}, \quad (\text{B.15})$$

where \mathbf{H} is the *measurement matrix* relating unknown state variables \mathbf{x} to measurements $\tilde{\mathbf{y}}$. In this case, (B.13) becomes

$$E = \sum_i \|\tilde{\mathbf{y}}_i - \mathbf{H}_i \mathbf{x}\|_{\boldsymbol{\Sigma}_i^{-1}} = \sum_i (\tilde{\mathbf{y}}_i - \mathbf{H}_i \mathbf{x})^T \mathbf{C}_i (\tilde{\mathbf{y}}_i - \mathbf{H}_i \mathbf{x}), \quad (\text{B.16})$$

³ According to the Gauss-Markov theorem, least squares produces the best linear unbiased estimator (BLUE) for a linear measurement model regardless of the actual noise distribution, assuming that the noise is zero mean and uncorrelated.

which is a simple quadratic form in \mathbf{x} , which can be solved using linear least squares (Appendix A.2). When the measurements are non-linear, the system must be solved iteratively using non-linear least squares (Appendix A.3).

B.3 Robust statistics

In Appendix B.1.1, we assumed that the noise being added to each measurement (B.5) was multivariate Gaussian (B.7). This is an appropriate model if the noise is the result of lots of tiny errors being added together, e.g., from thermal noise in a silicon imager. In most cases, however, measurements can be contaminated with larger *outliers*, i.e., gross failures in the measurement process. Examples of such outliers include bad feature matches (Section 6.1.4), occlusions in stereo matching (Chapter 11), and discontinuities in an otherwise smooth image, depth map, or label image (Sections 3.7.1 and 3.7.2).

In such cases, it makes more sense to model the measurement noise with a long-tailed *contaminated* noise model such as a Laplacian. The negative log likelihood in this case, rather than being quadratic in the measurement residuals (B.12–B.16), has a slower growth in the penalty function to account for the increased likelihood of large errors.

This formulation of the inference problem is called an *M-estimator* in the robust statistics literature (Huber 1981; Hampel, Ronchetti, Rousseeuw *et al.* 1986; Black and Rangarajan 1996; Stewart 1999) and involves applying a robust penalty function $\rho(r)$ to the residuals

$$E_{\text{RLS}}(\Delta \mathbf{p}) = \sum_i \rho(\|\mathbf{r}_i\|) \quad (\text{B.17})$$

instead of squaring them.

As we mentioned in Section 6.1.4, we can take the derivative of this function with respect to \mathbf{p} and set it to 0,

$$\sum_i \psi(\|\mathbf{r}_i\|) \frac{\partial \|\mathbf{r}_i\|}{\partial \mathbf{p}} = \sum_i \frac{\psi(\|\mathbf{r}_i\|)}{\|\mathbf{r}_i\|} \mathbf{r}_i^T \frac{\partial \mathbf{r}_i}{\partial \mathbf{p}} = 0, \quad (\text{B.18})$$

where $\psi(r) = \rho'(r)$ is the derivative of ρ and is called the *influence function*. If we introduce a *weight function*, $w(r) = \Psi(r)/r$, we observe that finding the stationary point of (B.17) using (B.18) is equivalent to minimizing the *iteratively re-weighted least squares* (IRLS) problem

$$E_{\text{IRLS}} = \sum_i w(\|\mathbf{r}_i\|) \|\mathbf{r}_i\|^2, \quad (\text{B.19})$$

where the $w(\|\mathbf{r}_i\|)$ play the same local weighting role as $\mathbf{C}_i = \boldsymbol{\Sigma}_i^{-1}$ in (B.12). Black and Anandan (1996) describe a variety of robust penalty functions and their corresponding influence and weighting function.

The IRLS algorithm alternates between computing the influence functions $w(\|\mathbf{r}_i\|)$ and solving the resulting weighted least squares problem (with fixed w values). Alternative incremental robust least squares algorithms can be found in the work of Sawhney and Ayer (1996); Black and Anandan (1996); Black and Rangarajan (1996); Baker, Gross, Ishikawa *et al.* (2003) and textbooks and tutorials on robust statistics (Huber 1981; Hampel, Ronchetti, Rousseeuw *et al.* 1986; Rousseeuw and Leroy 1987; Stewart 1999). It is also possible to apply general optimization techniques (Appendix A.3) directly to the non-linear cost function given in Equation (B.19), which may sometimes have better convergence properties.

Most robust penalty functions involve a scale parameter, which should typically be set to the variance (or standard deviation, depending on the formulation) of the non-contaminated (inlier) noise. Estimating such noise levels directly from the measurements or their residuals, however, can be problematic, as such estimates themselves become contaminated by outliers. The robust statistics literature contains a variety of techniques to estimate such parameters. One of the simplest and most effective is the *median absolute deviation* (MAD),

$$MAD = \text{med}_i \|\mathbf{r}_i\|, \quad (\text{B.20})$$

which, when multiplied by 1.4, provides a robust estimate of the standard deviation of the inlier noise process.

As mentioned in Section 6.1.4, it is often better to start iterative non-linear minimization techniques, such as IRLS, in the vicinity of a good solution by first randomly selecting small subsets of measurements until a good set of inliers is found. The best known of these techniques is RANdom SAmple Consensus (RANSAC) (Fischler and Bolles 1981), although even better variants such as Preemptive RANSAC (Nistér 2003) and PROgressive SAmple Consensus (PROSAC) (Chum and Matas 2005) have since been developed.

B.4 Prior models and Bayesian inference

While maximum likelihood estimation can often lead to good solutions, in some cases the range of possible solutions consistent with the measurements is too large to be useful. For example, consider the problem of image denoising (Sections 3.4.4 and 3.7.3). If we estimate each pixel separately based on just its noisy version, we cannot make any progress, as there are a large number of values that could lead to each noisy measurement.⁴ Instead, we need to rely on typical properties of images, e.g., that they tend to be piecewise smooth (Section 3.7.1).

The propensity of images to be piecewise smooth can be encoded in a *prior distribution* $p(\mathbf{x})$, which measures the likelihood of an image being a natural image. For example, to encode piecewise smoothness, we can use a *Markov random field* model (3.109 and B.24) whose negative log likelihood is proportional to a robustified measure of image smoothness (gradient magnitudes).

Prior models need not be restricted to image processing applications. For example, we may have some external knowledge about the rough dimensions of an object being scanned, the focal length of a lens being calibrated, or the likelihood that a particular object might appear in an image. All of these are examples of prior distributions or probabilities and they can be used to produce more reliable estimates.

As we have already seen in (3.68) and (3.106), Bayes' Rule states that a *posterior* distribution $p(\mathbf{x}|\mathbf{y})$ over the unknowns \mathbf{x} given the measurements \mathbf{y} can be obtained by multiplying the measurement likelihood $p(\mathbf{y}|\mathbf{x})$ by the prior distribution $p(\mathbf{x})$,

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})}, \quad (\text{B.21})$$

where $p(\mathbf{y}) = \int_{\mathbf{x}} p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$ is a normalizing constant used to make the $p(\mathbf{x}|\mathbf{y})$ distribution *proper* (integrate to 1). Taking the negative logarithm of both sides of Equation (B.21), we

⁴ In fact, the maximum likelihood estimate is just the noisy image itself.

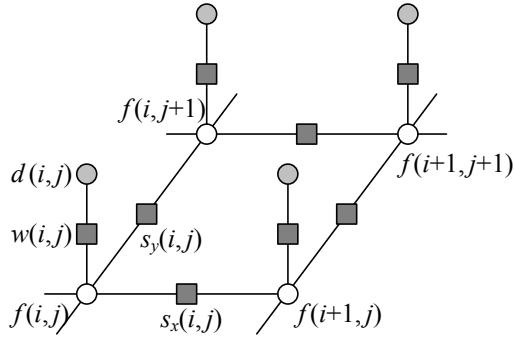


Figure B.1 Graphical model for an \mathcal{N}_4 neighborhood Markov random field. The white circles are the unknowns $f(i, j)$, while the dark circles are the input data $d(i, j)$. The $s_x(i, j)$ and $s_y(i, j)$ black boxes denote arbitrary *interaction potentials* between adjacent nodes in the random field, and the $w(i, j)$ denote the *data penalty* functions. They are all examples of the general potentials $V_{i,j,k,l}(f(i, j), f(k, l))$ used in Equation (B.24).

get

$$-\log p(\mathbf{x}|\mathbf{y}) = -\log p(\mathbf{y}|\mathbf{x}) - \log p(\mathbf{x}) + \log p(\mathbf{y}), \quad (\text{B.22})$$

which is the *negative posterior log likelihood*. It is common to drop the constant $\log p(\mathbf{y})$ because its value does not matter during energy minimization. However, if the prior distribution $p(\mathbf{x})$ depends on some unknown parameters, we may wish to keep $\log p(\mathbf{y})$ in order to compute the most likely value of these parameters using *Occam's razor*, i.e., by maximizing the likelihood of the observations, or to select the correct number of free parameters using *model selection* (Hastie, Tibshirani, and Friedman 2001; Torr 2002; Bishop 2006; Robert 2007).

To find the most likely (*maximum a posteriori* or MAP) solution \mathbf{x} given some measurements \mathbf{y} , we simply minimize this negative log likelihood, which can also be thought of as an *energy*,

$$E(\mathbf{x}, \mathbf{y}) = E_d(\mathbf{x}, \mathbf{y}) + E_p(\mathbf{x}). \quad (\text{B.23})$$

The first term $E_d(\mathbf{x}, \mathbf{y})$ is the *data energy* or *data penalty* and measures the negative log likelihood that the measurements \mathbf{y} were observed given the unknown state \mathbf{x} . The second term $E_p(\mathbf{x})$ is the *prior energy* and it plays a role analogous to the smoothness energy in regularization. Note that the MAP estimate may not always be desirable, since it selects the “peak” in the posterior distribution rather than some more stable statistic such as MSE—see the discussion in Appendix B.2 about loss functions and decision theory.

B.5 Markov random fields

Markov random fields (Blake, Kohli, and Rother 2010) are the most popular types of prior model for gridded image-like data,⁵ which include not only regular natural images (Section 3.7.2) but also two-dimensional fields such as optic flow (Chapter 8) or depth maps (Chapter 11), as well as binary fields, such as segmentations (Section 5.5).

⁵ Alternative formulations include power spectra (Section 3.4.3) and non-local means (Buades, Coll, and Morel 2008).

As we discussed in Section 3.7.2, the prior probability $p(\mathbf{x})$ for a Markov random field is a *Gibbs* or *Boltzmann distribution*, whose negative log likelihood (according to the Hammer-sley–Clifford Theorem) can be written as a sum of pairwise *interaction potentials*,

$$E_p(\mathbf{x}) = \sum_{\{(i,j),(k,l)\} \in \mathcal{N}} V_{i,j,k,l}(f(i,j), f(k,l)), \quad (\text{B.24})$$

where $\mathcal{N}(i,j)$ denotes the *neighbors* of pixel (i,j) . In the more general case, MRFs can also contain unary potentials, as well as *higher-order potentials* defined over larger cardinality *cliques* (Kindermann and Snell 1980; Geman and Geman 1984; Bishop 2006; Potetz and Lee 2008; Kohli, Kumar, and Torr 2009; Kohli, Ladický, and Torr 2009; Rother, Kohli, Feng *et al.* 2009; Alahari, Kohli, and Torr 2011). They can also contain *line processes*, i.e., additional binary variables that mediate discontinuities between adjacent elements (Geman and Geman 1984). Black and Rangarajan (1996) show how independent line process variables can be eliminated and incorporated into regular MRFs using robust pairwise penalty functions.

The most commonly used neighborhood in Markov random field modeling is the \mathcal{N}_4 neighborhood, where each pixel in the field $f(i,j)$ interacts only with its immediate neighbors—Figure B.1 shows such an \mathcal{N}_4 MRF. The $s_x(i,j)$ and $s_y(i,j)$ black boxes denote arbitrary interaction potentials between adjacent nodes in the random field and the $w(i,j)$ denote the elemental data penalty terms in E_d (B.23). These square nodes can also be interpreted as *factors* in a *factor graph* version of the undirected graphical model (Bishop 2006; Wainwright and Jordan 2008; Koller and Friedman 2009), which is another name for interaction potentials. (Strictly speaking, the factors are improper probability functions whose product is the un-normalized posterior distribution.)

More complex and higher-dimensional interaction models and neighborhoods are also possible. For example, 2D grids can be enhanced with the addition of diagonal connections (an \mathcal{N}_8 neighborhood) or even larger numbers of pairwise terms (Boykov and Kolmogorov 2003; Rother, Kolmogorov, Lempitsky *et al.* 2007). 3D grids can be used to compute globally optimal segmentations in 3D volumetric medical images (Boykov and Funka-Lea 2006) (Section 5.5.1). Higher-order cliques can also be used to develop more sophisticated models (Potetz and Lee 2008; Kohli, Ladický, and Torr 2009; Kohli, Kumar, and Torr 2009).

One of the biggest challenges in using MRF models is to develop efficient *inference algorithms* that will find low-energy solutions (Veksler 1999; Boykov, Veksler, and Zabih 2001; Kohli 2007; Kumar 2008). Over the years, a large variety of such algorithms have been developed, including simulated annealing, graph cuts, and loopy belief propagation. The choice of inference technique can greatly affect the overall performance of a vision system. For example, most of the top-performing algorithms on the Middlebury Stereo Evaluation page either use belief propagation or graph cuts.

In the next few subsections, we review some of the more widely used MRF inference techniques. More in-depth descriptions of most of these algorithms can be found in a recently published book on advances in MRF techniques (Blake, Kohli, and Rother 2010). Experimental comparisons, along with test datasets and reference software, are provided by Szeliski, Zabih, Scharstein *et al.* (2008).⁶

⁶ <http://vision.middlebury.edu/MRF/>.

B.5.1 Gradient descent and simulated annealing

The simplest optimization technique is gradient descent, which minimizes the energy by changing independent subsets of nodes to take on lower-energy configurations. Such techniques go under a variety of names, including *contextual classification* (Kittler and Föglein 1984) and *iterated conditional modes* (ICM) (Besag 1986).⁷ Variables can either be updated sequentially, e.g., in raster scan, or in parallel, e.g., using red–black coloring on a checkerboard. Chou and Brown (1990) suggests using highest confidence first (HCF), i.e., choosing variables based on how large a difference they make in reducing the energy.

The problem with gradient descent is that it is prone to getting stuck in local minima, which is almost always the case with MRF problems. One way around this is to use *stochastic gradient descent* or *Markov chain Monte Carlo* (MCMC) (Metropolis, Rosenbluth, Rosenbluth *et al.* 1953), i.e., to randomly take occasional uphill steps in order to get out of such minima. One popular update rule is the *Gibbs sampler* (Geman and Geman 1984); rather than choosing the lowest energy state for a variable being updated, it chooses the state with probability

$$p(\mathbf{x}) \propto e^{-E(\mathbf{x})/T}, \quad (\text{B.25})$$

where T is called the *temperature* and controls how likely the system is to choose a more random update. Stochastic gradient descent is usually combined with *simulated annealing* (Kirkpatrick, Gelatt, and Vecchi 1983), which starts at a relatively high temperature, thereby randomly exploring a large part of the state space, and gradually cools (anneals) the temperature to find a good local minimum. During the late 1980s, simulated annealing was the method of choice for solving MRF inference problems (Szeliski 1986; Marroquin, Mitter, and Poggio 1985; Barnard 1989).

Another variant on simulated annealing is the Swendsen–Wang algorithm (Swendsen and Wang 1987; Barbu and Zhu 2003, 2005). Here, instead of “flipping” (changing) single variables, a connected subset of variables, chosen using a random walk based on MRF connectivity strengths, is selected as the basic update unit. This can sometimes help make larger state changes, and hence find better-quality solutions in less time.

While simulated annealing has largely been superseded by the newer graph cuts and loopy belief propagation techniques, it still occasionally finds use, especially in highly connected and highly non-submodular graphs (Rother, Kolmogorov, Lempitsky *et al.* 2007).

B.5.2 Dynamic programming

Dynamic programming (DP) is an efficient inference procedure that works for any tree-structured graphical model, i.e., one that does not have any cycles. Given such a tree, pick any node as the root r and figuratively pick up the tree by its root. The depth or distance of all the other nodes from this root induces a partial ordering over the vertices, from which a total ordering can be obtained by arbitrarily breaking ties. Let us now lay out this graph as a tree with the root on the right and indices increasing from left to right, as shown in Figure B.2a.

Before describing the DP algorithm, let us re-write the potential function of Equation (B.24)

⁷ The name comes from iteratively setting variables to the mode (most likely, i.e., lowest energy) state conditioned on its currently fixed neighbors.

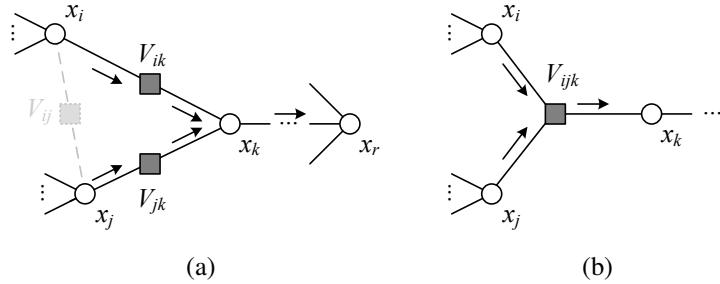


Figure B.2 Dynamic programming over a tree drawn as a factor graph. (a) To compute the lowest energy solution $\hat{E}_k(x_k)$ at node x_k conditioned on the best solutions to the left of this node, we enumerate all possible values of $\hat{E}_i(x_i) + V_{ik}(x_i, x_k)$ and pick the smallest one (and similarly for j). (b) For higher-order cliques, we need to try all combinations of (x_i, x_j) in order to select the best possible configuration. The arrows show the basic flow of the computation. The lightly shaded factor V_{ij} in (a) shows an additional connection that turns the tree into a cyclic graph, for which exact inference cannot be efficiently computed.

in a more general but succinct form,

$$E(\mathbf{x}) = \sum_{(i,j) \in \mathcal{N}} V_{i,j}(x_i, x_j) + \sum_i V_i(x_i), \quad (\text{B.26})$$

where instead of using pixel indices (i, j) and (k, l) , we just use scalar index variables i and j . We also replace the function value $f(i, j)$ with the more succinct notation x_i , with the $\{x_i\}$ variables making up the state vector \mathbf{x} . We can simplify this function even further by adding dummy nodes (vertices) i^- for every node that has a non-zero $V_i(x_i)$ and setting $V_{i,i^-}(x_i, x_{i^-}) = V_i(x_i)$, which lets us drop the V_i terms from (B.26).

Dynamic programming proceeds by computing partial sums in a left-to-right fashion, i.e., in order of increasing variable index. Let \mathcal{C}_k be the children of k , i.e., $i < k, (i, k) \in \mathcal{N}$. Then, define

$$\tilde{E}_k(\mathbf{x}) = \sum_{i < k, j \leq k} V_{i,j}(x_i, x_j) = \sum_{i \in \mathcal{C}_k} \left[V_{i,k}(x_i, x_k) + \tilde{E}_i(\mathbf{x}) \right], \quad (\text{B.27})$$

as a partial sum of (B.26) over all variables up to and including k , i.e., over all parts of the graph shown in Figure B.2a to the left of x_k . This sum depends on the state of all the unknown variables in \mathbf{x} with $i \leq k$.

Now suppose we wish to find the setting for all variables $i < k$ that minimizes this sum. It turns out that we can use a simple recursive formula

$$\hat{E}_k(x_k) = \min_{\{x_i, i < k\}} \tilde{E}_k(\mathbf{x}) = \sum_{i \in \mathcal{C}_k} \min_{x_i} \left[V_{i,k}(x_i, x_k) + \hat{E}_i(x_i) \right] \quad (\text{B.28})$$

to find this minimum. Visually, this is easy to understand. Looking at Figure B.2a, associate an energy $\hat{E}_k(x_k)$ with each node k and each possible setting of its value x_k that is based on the *best* possible setting of variables to the left of that node. It is easy to convince yourself that in this figure, you only need to know $\hat{E}_i(x_i)$ and $\hat{E}_j(x_j)$ in order to compute this value.

Once the flow of information in the tree has been processed from left to right, the minimum value of $\hat{E}_r(x_r)$ at the root gives the MAP (lowest-energy) solution for $E(\mathbf{x})$. The

root node is set to the choice of x_r that minimizes this function, and other nodes are set in a *backward chaining* pass by selecting the values of child nodes $i \in \mathcal{C}_k$ that were minimal in the original recursion (B.28).

Dynamic programming is not restricted to trees with pairwise potentials. Figure B.2b shows an example of a three-way potential $V_{ijk}(x_i, x_j, x_k)$ inside a tree. To compute the optimum value of $\hat{E}_k(x_k)$, the recursion formula in (B.28) now has to evaluate the minimum over all combinations of possible state values leading into a factor node (gray box). For this reason, dynamic programming is normally exponential in complexity in the order of the clique size, i.e., a clique of size n with l labels at each node requires the evaluation of l^{n-1} possible states (Potetz and Lee 2008; Kohli, Kumar, and Torr 2009). However, for certain kinds of potential functions $V_{i,k}(x_i, x_k)$, including the Potts model (delta function), absolute values (total variation), and quadratic (Gaussian MRF), Felzenszwalb and Huttenlocher (2006) show how to reduce the complexity of the min-finding step (B.28) from $O(l^2)$ to $O(l)$. In Appendix B.5.3, we also discuss how Potetz and Lee (2008) reduce the complexity for special kinds of higher-order clique, i.e., linear summations followed by non-linearities.

Figure B.2a also shows what happens if we add an extra factor between nodes i and j . In this case, the graph is no longer a tree, i.e., it contains a cycle. It is no longer possible to use the recursion formula (B.28), since $\hat{E}_i(x_i)$ now appears in two different terms inside the summation, i.e., as a child of both nodes j and k , and the same setting for x_i may not minimize both. In other words, when loops exist, there is no ordering of the variables that allows the recursion (elimination) in (B.28) to be well-founded.

It is, however, possible to convert small loops into higher-order factors and to solve these as shown in Figure B.2b. However, graphs with long loops or meshes result in extremely large clique sizes and hence an amount of computation potentially exponential in the size of the graph.

B.5.3 Belief propagation

Belief propagation is an inference technique originally developed for trees (Pearl 1988) but more recently extended to “loopy” (cyclic) graphs such as MRFs (Frey and MacKay 1997; Freeman, Pasztor, and Carmichael 2000; Yedidia, Freeman, and Weiss 2001; Weiss and Freeman 2001a,b; Yuille 2002; Sun, Zheng, and Shum 2003; Felzenszwalb and Huttenlocher 2006). It is closely related to dynamic programming, in that both techniques pass messages forward and backward over a tree or graph. In fact, one of the two variants of belief propagation, the *max-product rule*, performs the exact same computation (inference) as dynamic programming, albeit using probabilities instead of energies.

Recall that the energy we are minimizing in MAP estimation (B.26) is the negative log likelihood (B.12, B.13, and B.22) of a factored Gibbs posterior distribution,

$$p(\mathbf{x}) = \prod_{(i,j) \in \mathcal{N}} \phi_{i,j}(x_i, x_j), \quad (\text{B.29})$$

where

$$\phi_{i,j}(x_i, x_j) = e^{-V_{i,j}(x_i, x_j)} \quad (\text{B.30})$$

are the pairwise *interaction potentials*. We can rewrite (B.27) as

$$\tilde{p}_k(\mathbf{x}) = \prod_{i < k, j \leq k} \phi_{i,j}(x_i, x_j) = \prod_{i \in \mathcal{C}_k} \tilde{p}_{i,k}(\mathbf{x}), \quad (\text{B.31})$$

where

$$\tilde{p}_{i,k}(\mathbf{x}) = \phi_{i,k}(x_i, x_k) \tilde{p}_i(\mathbf{x}). \quad (\text{B.32})$$

We can therefore rewrite (B.28) as

$$\hat{p}_k(x_k) = \max_{\{x_i, i < k\}} \tilde{p}_k(\mathbf{x}) = \prod_{i \in C_k} \hat{p}_{i,k}(\mathbf{x}), \quad (\text{B.33})$$

with

$$\hat{p}_{i,k}(\mathbf{x}) = \max_{x_i} \phi_{i,k}(x_i, x_k) \hat{p}_i(\mathbf{x}). \quad (\text{B.34})$$

Equation (B.34) is the *max* update rule evaluated at all square box factors in Figure B.2a, while (B.33) is the *product* rule evaluated at the nodes. The probability distribution $\hat{p}_{i,k}(\mathbf{x})$ is often interpreted as a *message* passing information about child i to parent k and is hence written as $m_{i,k}(x_k)$ (Yedidia, Freeman, and Weiss 2001) or $\mu_{i \rightarrow k}(x_k)$ (Bishop 2006).

The max-product rule can be used to compute the MAP estimate in a tree using the same kind of forward and backward sweep as in dynamic programming (which is sometimes called the *max-sum* algorithm (Bishop 2006)). An alternative rule, known as the *sum–product*, sums over all possible values in (B.34) rather than taking the maximum, in essence computing the *expected* distribution rather than the *maximum likelihood* distribution. This produces a set of probability estimates that can be used to compute the *marginal* distributions $b_i(x_i) = \sum_{\mathbf{x} \setminus x_i} p(\mathbf{x})$ (Pearl 1988; Yedidia, Freeman, and Weiss 2001; Bishop 2006).

Belief propagation may not produce optimal estimates for cyclic graphs for the same reason that dynamic programming fails to work, i.e., because a node with multiple parents may take on different optimal values for each of the parents, i.e., there is no unique elimination ordering. Early algorithms for extending belief propagation to graphs with cycles, dubbed *loopy belief propagation*, performed the updates in parallel over the graph, i.e., using *synchronous updates* (Frey and MacKay 1997; Freeman, Pasztor, and Carmichael 2000; Yedidia, Freeman, and Weiss 2001; Weiss and Freeman 2001a,b; Yuille 2002; Sun, Zheng, and Shum 2003; Felzenszwalb and Huttenlocher 2006).

For example, Felzenszwalb and Huttenlocher (2006) split an \mathcal{N}_4 graph into its red and black (checkerboard) components and alternate between sending messages from the red nodes to the black and vice versa. They also use multi-grid (coarser level) updates to speed up the convergence. As discussed previously, to reduce the complexity of the basic max-product update rule (B.28) from $O(l^2)$ to $O(l)$, they develop specialized update algorithms for several cost functions $V_{i,k}(x_i, x_k)$, including the Potts model (delta function), absolute values (total variation), and quadratic (Gaussian MRF). A related algorithm, *mean field diffusion* (Scharstein and Szeliski 1998), also uses synchronous updates between nodes to compute marginal distributions. Yuille (2010) discusses the relationships between mean field theory and loopy belief propagation.

More recent loopy belief propagation algorithms and their variants use sequential scans through the graph (Szeliski, Zabih, Scharstein *et al.* 2008). For example, Tappen and Freeman (2003) pass messages from left to right along each row and then reverse the direction once they reach the end. This is similar to treating each row as an independent tree (chain), except that messages from nodes above and below the row are also incorporated. They then perform similar computations along columns. These sequential updates allow the information to propagate much more quickly across the image than synchronous updates.

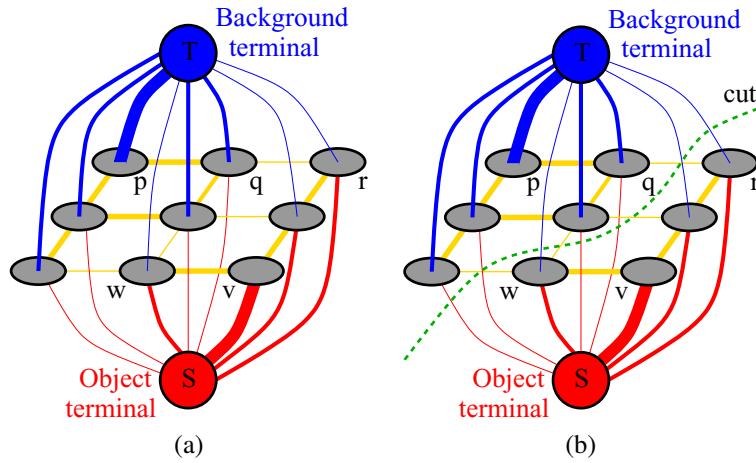


Figure B.3 Graph cuts for minimizing binary sub-modular MRF energies (Boykov and Jolly 2001) © 2001 IEEE: (a) energy function encoded as a max flow problem; (b) the minimum cut determines the region boundary.

The other belief propagation variant tested by Szeliski, Zabih, Scharstein *et al.* (2008), which they call BP-S or TRW-S, is based on Kolmogorov's (2006) sequential extension of the *tree-reweighted message passing* of Wainwright, Jaakkola, and Willsky (2005). TRW first selects a set of trees from the neighborhood graph and computes a set of probability distributions over each tree. These are then used to reweight the messages being passed during loopy belief propagation. The sequential version of TRW, called TRW-S, processes nodes in scan-line order, with a forward and backward pass. In the forward pass, each node sends messages to its right and bottom neighbors. In the backward pass, messages are sent to the left and upper neighbors. TRW-S also computes a lower bound on the energy, which is used by Szeliski, Zabih, Scharstein *et al.* (2008) to estimate how close to the best possible solution all of the MRF inference algorithms being evaluated get.

As with dynamic programming, belief propagation techniques also become less efficient as the order of each factor clique increases. Potetz and Lee (2008) shows how this complexity can be reduced back to linear in the clique order for continuous-valued problems where the factors involve linear summations followed by a non-linearity, which is typical of more sophisticated MRF models such as fields of experts (Roth and Black 2009) and steerable random fields (Roth and Black 2007b). Kohli, Kumar, and Torr (2009) and Alahari, Kohli, and Torr (2011) develop alternative ways for dealing with higher-order cliques in the context of graph cut algorithms.

B.5.4 Graph cuts

The computer vision community has adopted “graph cuts” as an informal name to describe a large family of MRF inference algorithms based on solving one or more min-cut or max-flow problems (Boykov, Veksler, and Zabih 2001; Boykov and Kolmogorov 2010; Boykov, Veksler, and Zabih 2010; Ishikawa and Veksler 2010).

The simplest example of an MRF graph cut is the polynomial-time algorithm for performing exact minimization of a binary MRF originally developed by Greig, Porteous, and Seheult

(1989) and brought to the attention of the computer vision community by Boykov, Veksler, and Zabih (2001) and Boykov and Jolly (2001). The basic construction of the min-cut graph from an MRF energy function is shown in Figure B.3 and described in Sections 3.7.2 and 5.5. In brief, the nodes in an MRF are connected to special source and sink nodes, and the minimum cut between these two nodes, whose cost is exactly that of the MRF energy under a binary assignment of labels, is computed using a polynomial-time max flow algorithm (Goldberg and Tarjan 1988; Boykov and Kolmogorov 2004).

As discussed in Section 5.5, important extensions of this basic algorithm have been made for the case of directed edges (Kolmogorov and Boykov 2005), larger neighborhoods (Boykov and Kolmogorov 2003; Kolmogorov and Boykov 2005), connectivity priors (Vicente, Kolmogorov, and Rother 2008), and shape priors (Lempitsky and Boykov 2007; Lempitsky, Blake, and Rother 2008). Kolmogorov and Zabih (2004) formally characterize the class of binary energy potentials (*regularity conditions*) for which these algorithms find the global minimum. Komodakis, Tziritas, and Paragios (2008) and Rother, Kolmogorov, Lempitsky *et al.* (2007) provide good algorithms for the cases when they do not.

Binary MRF problems can also be approximately solved by turning them into continuous $[0, 1]$ problems, solving them either as linear systems (Grady 2006; Sinop and Grady 2007; Grady and Alvino 2008; Grady 2008; Grady and Ali 2008; Singaraju, Grady, and Vidal 2008; Couprise, Grady, Najman *et al.* 2009) (the *random walker model*) or by computing geodesic distances (Bai and Sapiro 2009; Criminisi, Sharp, and Blake 2008) and then thresholding the results. More details on these techniques are provided in Section 5.5 and a nice review can be found in the work of Singaraju, Grady, Sinop *et al.* (2010). A different connection to continuous segmentation techniques, this time to the literature on level sets (Section 5.1.4), is made by Boykov, Kolmogorov, Cremers *et al.* (2006), who develop an approach to solving surface propagation PDEs based on combinatorial graph cut algorithms—Boykov and Funka-Lea (2006) discuss this and related techniques.

Multi-valued MRF inference problems usually require solving a series of related binary MRF problems (Boykov, Veksler, and Zabih 2001), although for special cases, such as some convex functions, a single graph cut may suffice (Ishikawa 2003; Schlesinger and Flach 2006). The seminal work in this area is that of Boykov, Veksler, and Zabih (2001), who introduced two algorithms, called the *swap move* and the *expansion move*, which are sketched in Figure B.4. The α - β -swap move selects two labels (usually by cycling through all possible pairings) and then formulates a binary MRF problem that allows any pixels currently labeled as either α or β to optionally switch their values to the other label. The α -expansion move allows any pixel in the MRF to take on the α label or to keep its current identity. It is easy to see by inspection that both of these moves result in binary MRFs with well-defined energy functions.

Because these algorithms use a binary MRF optimization inside their inner loop, they are subject to the constraints on the energy functions that occur in the binary labeling case (Kolmogorov and Zabih 2004). However, more recent algorithms such as those developed by Komodakis, Tziritas, and Paragios (2008) and Rother, Kolmogorov, Lempitsky *et al.* (2007) can be used to provide approximate solutions for more general energy functions. Efficient algorithms for re-using previous solutions (*flow- and cut-recycling*) have been developed for on-line applications such as *dynamic MRFs* (Kohli and Torr 2005; Juan and Boykov 2006; Alahari, Kohli, and Torr 2011) and coarse-to-fine banded graph cuts (Agarwala, Zheng, Pal *et*

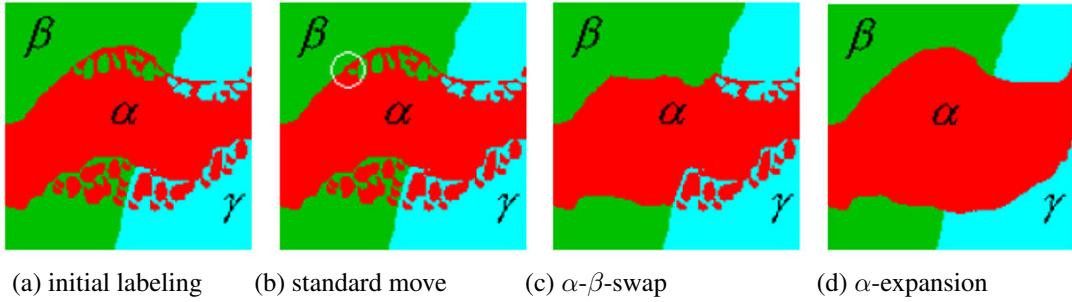


Figure B.4 Multi-level graph optimization from (Boykov, Veksler, and Zabih 2001) © 2001 IEEE: (a) initial problem configuration; (b) the standard move changes only one pixel; (c) the α - β -swap optimally exchanges all α - and β -labeled pixels; (d) the α -expansion move optimally selects among current pixel values and the α label.

al. 2005; Lombaert, Sun, Grady *et al.* 2005; Juan and Boykov 2006). It is also now possible to minimize the number of labels used as part of the alpha-expansion process (Delong, Osokin, Isack *et al.* 2010).

In experimental comparisons, α -expansions usually converge faster to a good solution than α - β -swaps (Szeliski, Zabih, Scharstein *et al.* 2008), especially for problems that involve large regions of identical labels, such as the labeling of source imagery in image stitching (Figure 3.60). For truncated convex energy functions defined over ordinal values, more accurate algorithms that consider complete ranges of labels inside each min-cut and often produce lower energies have been developed (Veksler 2007; Kumar and Torr 2008; Kumar, Veksler, and Torr 2010). The whole field of efficient MRF inference algorithms is rapidly developing, as witnessed by a recent special journal issue (Kohli and Torr 2008; Komodakis, Tziritas, and Paragios 2008; Olsson, Eriksson, and Kahl 2008; Potetz and Lee 2008), articles (Alahari, Kohli, and Torr 2011), and a forthcoming book (Blake, Kohli, and Rother 2010).

B.5.5 Linear programming

⁸ Many successful algorithms for MRF optimization are based on the *linear programming* (LP) relaxation of the energy function (Weiss, Yanover, and Meltzer 2010). For some practical MRF problems, LP-based techniques can produce globally minimal solutions (Meltzer, Yanover, and Weiss 2005), even though MRF inference is in general NP-hard. In order to describe this relaxation, let us first rewrite the energy function (B.26) as

$$E(\mathbf{x}) = \sum_{(i,j) \in \mathcal{N}} V_{i,j}(x_i, x_j) + \sum_i V_i(x_i) \quad (\text{B.35})$$

$$= \sum_{i,j,\alpha,\beta} V_{i,j}(\alpha, \beta) x_{i,j;\alpha,\beta} + \sum_{i,\alpha} V_i(\alpha) x_{i;\alpha} \quad (\text{B.36})$$

$$\text{subject to } x_{i;\alpha} = \sum_{\beta} x_{i,j;\alpha,\beta} \quad \forall (i,j) \in \mathcal{N}, \alpha, \quad (\text{B.37})$$

$$x_{j;\beta} = \sum_{\alpha} x_{i,j;\alpha,\beta} \quad \forall (i,j) \in \mathcal{N}, \beta, \quad \text{and} \quad (\text{B.38})$$

$$x_{i,\alpha}, x_{i,j;\alpha,\beta} \in \{0, 1\}. \quad (\text{B.39})$$

⁸ This section was contributed by Vladimir Kolmogorov. Thanks!

Here, α and β range over label values and $x_{i;\alpha} = \delta(x_i - \alpha)$ and $x_{ij;\alpha\beta} = \delta(x_i - \alpha)\delta(x_j - \beta)$ are indicator variables of assignments $x_i = \alpha$ and $(x_i, x_j) = (\alpha, \beta)$, respectively. The LP relaxation is obtained by replacing the discreteness constraints (B.39) with linear constraints $x_{ij;\alpha\beta} \in [0, 1]$. It is easy to show that the optimal value of (B.36) is a lower bound on (B.26).

This relaxation has been extensively studied in the literature, starting with the work of Schlesinger (1976). An important question is how to solve this LP efficiently. Unfortunately, general-purpose LP solvers cannot handle large problems in vision (Yanover, Meltzer, and Weiss 2006). A large number of customized iterative techniques have been proposed. Most of these solve the dual problem, i.e., they formulate a lower bound on (B.36) and then try to maximize this bound. The bound is often formulated using a convex combination of trees, as proposed in (Wainwright, Jaakkola, and Willsky 2005).

The LP lower bound can be maximized via a number of techniques, such as *max-sum diffusion* (Werner 2007), *tree-reweighted message passing* (TRW) (Wainwright, Jaakkola, and Willsky 2005; Kolmogorov 2006), subgradient methods (Schlesinger and Giginyak 2007a,b; Komodakis, Paragios, and Tziritas 2007), and Bregman projections (Ravikumar, Agarwal, and Wainwright 2008). Note that the max-sum diffusion and TRW algorithms are not guaranteed to converge to a global maximum of LP—they may get stuck at a suboptimal point (Kolmogorov 2006; Werner 2007). However, in practice, this does not appear to be a problem (Kolmogorov 2006).

For some vision applications, algorithms based on relaxation (B.36) produce excellent results. However, this is not guaranteed in all cases—after all, the problem is NP-hard. Recently, researchers have investigated alternative linear programming relaxations (Sontag and Jaakkola 2007; Sontag, Meltzer, Globerson *et al.* 2008; Komodakis and Paragios 2008; Schraudolph 2010). These algorithms are capable of producing tighter bounds compared to (B.36) at the expense of additional computational cost.

LP relaxation and alpha expansion. Solving a linear program produces primal and dual solutions that satisfy *complementary slackness conditions*. In general, the primal solution of (B.36) does not have to be integer-valued so, in practice, we may have to round it to obtain a valid labeling \boldsymbol{x} . An alternative proposed by Komodakis and Tziritas (2007a); Komodakis, Tziritas, and Paragios (2007) is to search for primal and dual solutions such that they satisfy *approximate complementary slackness conditions* and the primal solution is already integer-valued. Several max-flow-based algorithms are proposed by (Komodakis and Tziritas 2007a; Komodakis, Tziritas, and Paragios 2007) for this purpose and the *Fast-PD* method (Komodakis, Tziritas, and Paragios 2007) is shown to perform best. In the case of metric interactions, the default version of Fast-PD produces the same primal solution as the alpha-expansion algorithm (Boykov, Veksler, and Zabih 2001). This provides an interesting interpretation of the alpha expansion algorithm as trying to approximately solve relaxation (B.36).

Unlike the standard alpha expansion algorithm, Fast-PD also maintains a dual solution and thus runs faster in practice. Fast-PD can be extended to the case of semi-metric interactions (Komodakis, Tziritas, and Paragios 2007). The primal version of such extension was also given by Rother, Kumar, Kolmogorov *et al.* (2005).

B.6 Uncertainty estimation (error analysis)

In addition to computing the most likely estimate, many applications require an estimate for the *uncertainty* in this estimate.⁹ The most general way to do this is to compute a complete probability distribution over all of the unknowns but this is generally intractable. The one special case where it is easy to obtain a simple description for this distribution is linear estimation problems with Gaussian noise, where the joint energy function (negative log likelihood of the posterior estimate) is a quadratic. In this case, the posterior distribution is a multi-variate Gaussian and the covariance can be computed directly from the inverse of the problem Hessian. (Another name for the inverse covariance matrix, which is equal to the Hessian in such simple cases, is the *information matrix*.)

Even here, however, the full covariance matrix may be too large to compute and store. For example, in large structure from motion problems, a large sparse Hessian normally results in a full dense covariance matrix. In such cases, it is often considered acceptable to report only the variance in the estimated quantities or simple covariance estimates on individual parameters, such as 3D point positions or camera pose estimates (Szeliski 1990a). More insight into the problem, e.g., the dominant *modes* of uncertainty, can be obtained using eigenvalue analysis (Szeliski and Kang 1997).

For problems where the posterior energy is non-quadratic, e.g., in non-linear or robustified least squares, it is still often possible to obtain an estimate of the Hessian in the vicinity of the optimal solution. In this case, the *Cramer–Rao lower bound* on the uncertainty (covariance) can be computed as the inverse of the Hessian. Another way of saying this is that while the local Hessian can underestimate how “wide” the energy function can be, the covariance can never be smaller than the estimate based on this local quadratic approximation. It is also possible to estimate a different kind of uncertainty (min-marginal energies) in general MRFs where the MAP inference is performed using graph cuts (Kohli and Torr 2008).

While many computer vision applications ignore uncertainty modeling, it is often useful to compute these estimates just to get an intuitive feeling for the reliability of the estimates. Certain applications, such as Kalman filtering, require the computation of this uncertainty (either explicitly as posterior covariances or implicitly as inverse covariances) in order to optimally integrate new measurements with previously computed estimates.

⁹ This is particularly true of classic photogrammetry applications, where the reporting of precision is almost always considered mandatory (Förstner 2005).

Appendix C

Supplementary material

| | | |
|-----|-------------------------------|-----|
| C.1 | Data sets | 680 |
| C.2 | Software | 682 |
| C.3 | Slides and lectures | 689 |
| C.4 | Bibliography | 690 |

In this final appendix, I summarize some of the supplementary materials that may be useful to students, instructors, and researchers. The book's Web site at <http://szeliski.org/Book> contains updated lists of datasets and software, so please check there as well.

C.1 Data sets

One of the keys to developing reliable vision algorithms is to test your procedures on challenging and representative data sets. When ground truth or other people's results are available, such test can be even more informative (and quantitative).

Over the years, a large number of datasets have been developed for testing and evaluating computer vision algorithms. A number of these datasets (and software) are indexed on the Computer Vision Homepage.¹ Some newer Web sites, such as CVonline (<http://homepages.inf.ed.ac.uk/rbf/CVonline/>), VisionBib.Com (<http://datasets.visionbib.com/>), and Computer Vision online (<http://computervisiononline.com/>), have more recent pointers.

Below, I list some of the more popular data sets, grouped by the book chapters to which they most closely correspond:

Chapter 2: Image formation

CURET: Columbia-Utrecht Reflectance and Texture Database, <http://www1.cs.columbia.edu/CAVE/software/curet/> (Dana, van Ginneken, Nayar *et al.* 1999).

Middlebury Color Datasets: registered color images taken by different cameras to study how they transform gamuts and colors, <http://vision.middlebury.edu/color/data/> (Chakrabarti, Scharstein, and Zickler 2009).

Chapter 3: Image processing

Middlebury test datasets for evaluating MRF minimization/inference algorithms, <http://vision.middlebury.edu/MRF/results/> (Szeliski, Zabih, Scharstein *et al.* 2008).

Chapter 4: Feature detection and matching

Affine Covariant Features database for evaluating feature detector and descriptor matching quality and repeatability, <http://www.robots.ox.ac.uk/~vgg/research/affine/> (Mikolajczyk and Schmid 2005; Mikolajczyk, Tuytelaars, Schmid *et al.* 2005).

Database of matched image patches for learning and feature descriptor evaluation, <http://cvlab.epfl.ch/~brown/patchdata/patchdata.html> (Winder and Brown 2007; Hua, Brown, and Winder 2007).

Chapter 5: Segmentation

Berkeley Segmentation Dataset and Benchmark of 1000 images labeled by 30 humans, along with an evaluation, <http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/> (Martin, Fowlkes, Tal *et al.* 2001).

¹ <http://www.cs.cmu.edu/~cil/vision.html>, although it has not been maintained since 2004.

Weizmann segmentation evaluation database of 100 grayscale images with ground truth segmentations, http://www.wisdom.weizmann.ac.il/~vision/Seg_Evaluation_DB/index.html (Alpert, Galun, Basri *et al.* 2007).

Chapter 8: Dense motion estimation

The Middlebury optic flow evaluation Web site, <http://vision.middlebury.edu/flow/data> (Baker, Scharstein, Lewis *et al.* 2009).

The Human-Assisted Motion Annotation database,
<http://people.csail.mit.edu/celiu/motionAnnotation/> (Liu, Freeman, Adelson *et al.* 2008)

Chapter 10: Computational photography

High Dynamic Range radiance maps, <http://www.debevec.org/Research/HDR/> (Debevec and Malik 1997).

Alpha matting evaluation Web site, <http://alphamatting.com/> (Rhemann, Rother, Wang *et al.* 2009).

Chapter 11: Stereo correspondence

Middlebury Stereo Datasets and Evaluation, <http://vision.middlebury.edu/stereo/> (Scharstein and Szeliski 2002).

Stereo Classification and Performance Evaluation of different aggregation costs for stereo matching, <http://www.vision.deis.unibo.it/spe/SPEHome.aspx> (Tombari, Mattoccia, Di Stefano *et al.* 2008).

Middlebury Multi-View Stereo Datasets, <http://vision.middlebury.edu/mview/data/> (Seitz, Curless, Diebel *et al.* 2006).

Multi-view and Oxford Colleges building reconstructions, <http://www.robots.ox.ac.uk/~vgg/data/data-mview.html>.

Multi-View Stereo Datasets, <http://cvlab.epfl.ch/data/strechamvs/> (Strecha, Fransens, and Van Gool 2006).

Multi-View Evaluation, <http://cvlab.epfl.ch/~strecha/multiview/> (Strecha, von Hansen, Van Gool *et al.* 2008).

Chapter 12: 3D reconstruction

HumanEva: synchronized video and motion capture dataset for evaluation of articulated human motion, <http://vision.cs.brown.edu/humaneva/> (Sigal, Balan, and Black 2010).

Chapter 13: Image-based rendering

The (New) Stanford Light Field Archive, <http://lightfield.stanford.edu/> (Wilburn, Joshi, Vaish *et al.* 2005).

Virtual Viewpoint Video: multi-viewpoint video with per-frame depth maps, <http://research.microsoft.com/en-us/um/redmond/groups/ivm/vvv/> (Zitnick, Kang, Uyttendaele *et al.* 2004).

Chapter 14: Recognition

For a list of visual recognition datasets, see Tables 14.1–14.2. In addition to those, there are also:

Buffy pose classes, http://www.robots.ox.ac.uk/~vgg/data/buffy_pose_classes/ and Buffy stickmen V2.1, <http://www.robots.ox.ac.uk/~vgg/data/stickmen/index.html> (Ferrari, Marin-Jimenez, and Zisserman 2009; Eichner and Ferrari 2009).

H3D database of pose/joint annotated photographs of humans, <http://www.eecs.berkeley.edu/~lbourdev/h3d/> (Bourdev and Malik 2009).

Action Recognition Datasets, <http://www.cs.berkeley.edu/projects/vision/action>, has pointers to several datasets for action and activity recognition, as well as some papers. The human action database at <http://www.nada.kth.se/cvap/actions/> contains more action sequences.

C.2 Software

One of the best sources for computer vision algorithms is the Open Source Computer Vision (OpenCV) library (<http://opencv.willowgarage.com/wiki/>), which was developed by Gary Bradski and his colleagues at Intel and is now being maintained and extended at Willow Garage (Bradsky and Kaehler 2008). A partial list of the available functions, taken from <http://opencv.willowgarage.com/documentation/cpp/> includes:

- image processing and transforms (filtering, morphology, pyramids);
- geometric image transformations (rotations, resizing);
- miscellaneous image transformations (Fourier transforms, distance transforms);
- histograms;
- segmentation (watershed, mean shift);
- feature detection (Canny, Harris, Hough, MSER, SURF);
- motion analysis and object tracking (Lucas–Kanade, mean shift);
- camera calibration and 3D reconstruction;
- machine learning (k nearest neighbors, support vector machines, decision trees, boosting, random trees, expectation-maximization, and neural networks).

The Intel Performance Primitives (IPP) library, <http://software.intel.com/en-us/intel-ipp/>, contains highly optimized code for a variety of image processing tasks. Many of the routines in OpenCV take advantage of this library, if it is installed, to run even faster. In terms of

functionality, it has many of the same operators as those found in OpenCV, plus additional libraries for image and video compression, signal and speech processing, and matrix algebra.

The MATLAB Image Processing Toolbox, <http://www.mathworks.com/products/image/>, contains routines for spatial transformations (rotations, resizing), normalized cross-correlation, image analysis and statistics (edges, Hough transform), image enhancement (adaptive histogram equalization, median filtering) and restoration (deblurring), linear filtering (convolution), image transforms (Fourier and DCT), and morphological operations (connected components and distance transforms).

Two older libraries, which no longer appear to be under active development but contain many useful routines, are VXL (C++ Libraries for Computer Vision Research and Implementation, <http://vxl.sourceforge.net/>) and LTI-Lib 2 (<http://www.ie.itcr.ac.cr/palvarado/ltilib-2/homepage/>).

Photo editing and viewing packages, such as Windows Live Photo Gallery, iPhoto, Picasa, GIMP, and IrfanView, can be useful for performing common processing tasks, converting formats, and viewing your results. They can also serve as interesting reference implementations for image processing algorithms (such as tone correction or denoising) that you are trying to develop from scratch.

There are also software packages and infrastructure that can be helpful for building real-time video processing demos. Vision on Tap (<http://www.visionontap.com/>) provides a Web service that will process your webcam video in real time (Chiu and Raskar 2009). Video-Man (VideoManager, <http://videomanlib.sourceforge.net/>) can be useful for getting real-time video-based demos and applications running. You can also use `imread` in MATLAB to read directly from any URL, such as a webcam.

Below, I list some additional software that can be found on the Web, grouped by the book chapters to which they most correspond:

Chapter 3: Image processing

matlabPyrTools—MATLAB source code for Laplacian pyramids, QMF/Wavelets, and steerable pyramids, <http://www.cns.nyu.edu/~lcv/software.php> (Simoncelli and Adelson 1990a; Simoncelli, Freeman, Adelson *et al.* 1992).

BLS-GSM image denoising, <http://decsai.ugr.es/~javier/denoise/> (Portilla, Strela, Wainwright *et al.* 2003).

Fast bilateral filtering code, <http://people.csail.mit.edu/jiawen/#code> (Chen, Paris, and Durand 2007).

C++ implementation of the fast distance transform algorithm, <http://people.cs.uchicago.edu/~pff/dt/> (Felzenszwalb and Huttenlocher 2004a).

GREYC's Magic Image Converter, including image restoration software using regularization and anisotropic diffusion, <http://gmic.sourceforge.net/gimp.shtml> (Tschumperlé and Deriche 2005).

Chapter 4: Feature detection and matching

VLFeat, an open and portable library of computer vision algorithms, <http://vlfeat.org/> (Vedaldi and Fulkerson 2008).

SiftGPU: A GPU Implementation of Scale Invariant Feature Transform (SIFT), <http://www.cs.unc.edu/~ccwu/siftgpu/> (Wu 2010).

SURF: Speeded Up Robust Features, <http://www.vision.ee.ethz.ch/~surf/> (Bay, Tuytelaars, and Van Gool 2006).

FAST corner detection, <http://mi.eng.cam.ac.uk/~er258/work/fast.html> (Rosten and Drummond 2005, 2006).

Linux binaries for affine region detectors and descriptors, as well as MATLAB files to compute repeatability and matching scores, <http://www.robots.ox.ac.uk/~vgg/research/affine/>.

Kanade–Lucas–Tomasi feature trackers: KLT, <http://www.ces.clemson.edu/~stb/klt/> (Shi and Tomasi 1994); GPU-KLT, <http://cs.unc.edu/~cmzach/opensource.html> (Zach, Gallup, and Frahm 2008); and Lucas–Kanade 20 Years On, http://www.ri.cmu.edu/projects/project_515.html (Baker and Matthews 2004).

Chapter 5: Segmentation

Efficient graph-based image segmentation, <http://people.cs.uchicago.edu/~pff/segment/> (Felzenszwalb and Huttenlocher 2004b).

EDISON, edge detection and image segmentation, <http://coewww.rutgers.edu/riul/research/code/EDISON/> (Meer and Georgescu 2001; Comaniciu and Meer 2002).

Normalized cuts segmentation including intervening contours, <http://www.cis.upenn.edu/~jshi/software/> (Shi and Malik 2000; Malik, Belongie, Leung *et al.* 2001).

Segmentation by weighted aggregation (SWA), <http://www.cs.weizmann.ac.il/~vision/SWA/> (Alpert, Galun, Basri *et al.* 2007).

Chapter 6: Feature-based alignment and calibration

Non-iterative PnP algorithm, <http://cvlab.epfl.ch/software/EPnP/> (Moreno-Noguer, Lepetit, and Fua 2007).

Tsai Camera Calibration Software, <http://www-2.cs.cmu.edu/~rgw/TsaiCode.html> (Tsai 1987).

Easy Camera Calibration Toolkit, <http://research.microsoft.com/en-us/um/people/zhang/Calib/> (Zhang 2000).

Camera Calibration Toolbox for MATLAB, http://www.vision.caltech.edu/bouguetj/calib_doc/; a C version is included in OpenCV.

MATLAB functions for multiple view geometry, <http://www.robots.ox.ac.uk/~vgg/hzbook/code/> (Hartley and Zisserman 2004).

Chapter 7: Structure from motion

SBA: A generic sparse bundle adjustment C/C++ package based on the Levenberg–Marquardt algorithm, <http://www.ics.forth.gr/~lourakis/sba/> (Lourakis and Argyros 2009).

Simple sparse bundle adjustment (SSBA), <http://cs.unc.edu/~cmzach/opensource.html>.

Bundler, structure from motion for unordered image collections, <http://phototour.cs.washington.edu/bundler/> (Snavely, Seitz, and Szeliski 2006).

Chapter 8: Dense motion estimation

Optical flow software, <http://www.cs.brown.edu/~black/code.html> (Black and Anandan 1996).

Optical flow using total variation and conjugate gradient descent, <http://people.csail.mit.edu/celiu/OpticalFlow/> (Liu 2009).

TV-L1 optical flow on the GPU, <http://cs.unc.edu/~cmzach/opensource.html> (Zach, Pock, and Bischof 2007a).

elastix: a toolbox for rigid and nonrigid registration of images, <http://elastix.isi.uu.nl/> (Klein, Staring, and Pluim 2007).

Deformable image registration using discrete optimization, <http://www.mrf-registration.net/deformable/index.html> (Glocker, Komodakis, Tziritas *et al.* 2008).

Chapter 9: Image stitching

Microsoft Research Image Compositing Editor for stitching images, <http://research.microsoft.com/en-us/um/redmond/groups/ivm/ice/>.

Chapter 10: Computational photography

HDRShop software for combining bracketed exposures into high-dynamic range radiance images, <http://projects.ict.usc.edu/graphics/HDRShop/>.

Super-resolution code, <http://www.robots.ox.ac.uk/~vgg/software/SR/> (Pickup 2007; Pickup, Capel, Roberts *et al.* 2007, 2009).

Chapter 11: Stereo correspondence

StereoMatcher, standalone C++ stereo matching code, <http://vision.middlebury.edu/stereo/code/> (Scharstein and Szeliski 2002).

Patch-based multi-view stereo software (PMVS Version 2), <http://grail.cs.washington.edu/software/pmv/> (Furukawa and Ponce 2011).

Chapter 12: 3D reconstruction

Scanalyze: a system for aligning and merging range data, <http://graphics.stanford.edu/software/scanalyze/> (Curless and Levoy 1996).

MeshLab: software for processing, editing, and visualizing unstructured 3D triangular meshes, <http://meshlab.sourceforge.net/>.

VRML viewers (various) are also a good way to visualize texture-mapped 3D models.

Section 12.6.4: Whole body modeling and tracking

Bayesian 3D person tracking, <http://www.cs.brown.edu/~black/code.html> (Sidenbladh, Black, and Fleet 2000; Sidenbladh and Black 2003).

HumanEva: baseline code for the tracking of articulated human motion, <http://vision.cs.brown.edu/humaneva/> (Sigal, Balan, and Black 2010).

Section 14.1.1: Face detection

Sample face detection code and evaluation tools,
<http://vision.ai.uiuc.edu/mhyang/face-detection-survey.html>.

Section 14.1.2: Pedestrian detection

A simple object detector with boosting, <http://people.csail.mit.edu/torralba/shortCourseRLOC/boosting/boosting.html> (Hastie, Tibshirani, and Friedman 2001; Torralba, Murphy, and Freeman 2007).

Discriminatively trained deformable part models, <http://people.cs.uchicago.edu/~pff/latent/> (Felzenszwalb, Girshick, McAllester *et al.* 2010).

Upper-body detector, <http://www.robots.ox.ac.uk/~vgg/software/UpperBody/> (Ferrari, Marin-Jimenez, and Zisserman 2008).

2D articulated human pose estimation software, http://www.vision.ee.ethz.ch/~calvin/articulated_human_pose_estimation_code/ (Eichner and Ferrari 2009).

Section 14.2.2: Active appearance and 3D shape models

AAMtools: An active appearance modeling toolbox, <http://cvsp.cs.ntua.gr/software/AAMtools/> (Papandreou and Maragos 2008).

Section 14.3: Instance recognition

FASTANN and FASTCLUSTER for approximate k-means (AKM), <http://www.robots.ox.ac.uk/~vgg/software/> (Philbin, Chum, Isard *et al.* 2007).

Feature matching using fast approximate nearest neighbors, <http://people.cs.ubc.ca/~mariusm/index.php/FLANN/FLANN> (Muja and Lowe 2009).

Section 14.4.1: Bag of words

Two bag of words classifiers, <http://people.csail.mit.edu/fergus/iccv2005/bagwords.html> (Fei-Fei and Perona 2005; Sivic, Russell, Efros *et al.* 2005).

Bag of features and hierarchical k-means, <http://www.vlfeat.org/> (Nistér and Stewénius 2006; Nowak, Jurie, and Triggs 2006).

Section 14.4.2: Part-based models

A simple parts and structure object detector, <http://people.csail.mit.edu/fergus/iccv2005/partsstructure.html> (Fischler and Elschlager 1973; Felzenszwalb and Huttenlocher 2005).

Section 14.5.1: Machine learning software

Support vector machines (SVM) software (http://www.support-vector-machines.org/SVM_soft.html) has pointers to lots of SVM libraries, including SVM^{light}, <http://svmlight.joachims.org/>; LIBSVM, <http://www.csie.ntu.edu.tw/~cjlin/libsvm/> (Fan, Chen, and Lin 2005); and LIBLINEAR, <http://www.csie.ntu.edu.tw/~cjlin/liblinear/> (Fan, Chang, Hsieh *et al.* 2008).

Kernel Machines: links to SVM, Gaussian processes, boosting, and other machine learning algorithms, <http://www.kernel-machines.org/software>.

Multiple kernels for image classification, <http://www.robots.ox.ac.uk/~vgg/software/MKL/> (Varma and Ray 2007; Vedaldi, Gulshan, Varma *et al.* 2009).

Appendix A.1–A.2: Matrix decompositions and linear least squares²

BLAS (Basic Linear Algebra Subprograms), <http://www.netlib.org/blas/> (Blackford, Demmel, Dongarra *et al.* 2002).

LAPACK (Linear Algebra PACKage), <http://www.netlib.org/lapack/> (Anderson, Bai, Bischof *et al.* 1999).

GotoBLAS, <http://www.tacc.utexas.edu/tacc-projects/>.

ATLAS (Automatically Tuned Linear Algebra Software), <http://math-atlas.sourceforge.net/> (Demmel, Dongarra, Eijkhout *et al.* 2005).

Intel Math Kernel Library (MKL), <http://software.intel.com/en-us/intel-mkl/>.

AMD Core Math Library (ACML), <http://developer.amd.com/cpu/Libraries/acml/Pages/default.aspx>.

Robust PCA code, <http://www.salle.url.edu/~ftorre/papers/rpca2.html> (De la Torre and Black 2003).

Appendix A.3: Non-linear least squares

MINPACK, <http://www.netlib.org/minpack/>.

levmar: Levenberg–Marquardt nonlinear least squares algorithms, <http://www.ics.forth.gr/~lourakis/levmar/> (Madsen, Nielsen, and Tingleff 2004).

Appendix A.4–A.5: Direct and iterative sparse matrix solvers

SuiteSparse (various reordering algorithms, CHOLMOD) and SuiteSparse QR, <http://www.cise.ufl.edu/research/sparse/SuiteSparse/> (Davis 2006, 2008).

PARDISO (iterative and sparse direct solution), <http://www.pardiso-project.org/>.

TAUCS (sparse direct, iterative, out of core, preconditioners), <http://www.tau.ac.il/~stoledo/taucs/>.

HSL Mathematical Software Library, <http://www.hsl.rl.ac.uk/index.html>.

² Thanks to Sameer Agarwal for suggesting and describing most of these sites.

Templates for the solution of linear systems, http://www.netlib.org/linalg/html_templates/Templates.html (Barrett, Berry, Chan *et al.* 1994). Download the PDF for instructions on how to get the software.

ITSOL, MIQR, and other sparse solvers, <http://www-users.cs.umn.edu/~saad/software/> (Saad 2003).

ILUPACK, <http://www-public.tu-bs.de/~bolle/ilupack/>.

Appendix B: Bayesian modeling and inference

Middlebury source code for MRF minimization, <http://vision.middlebury.edu/MRF/code/> (Szeliski, Zabih, Scharstein *et al.* 2008).

C++ code for efficient belief propagation for early vision, <http://people.cs.uchicago.edu/~pff/bp/> (Felzenszwalb and Huttenlocher 2006).

FastPD MRF optimization code, <http://www.csd.uoc.gr/~komod/FastPD> (Komodakis and Tziritas 2007a; Komodakis, Tziritas, and Paragios 2008)

Gaussian noise generation. A lot of basic software packages come with a uniform random noise generator (e.g., the `rand()` routine in Unix), but not all have a Gaussian random noise generator. To compute a normally distributed random variable, you can use the Box–Muller transform (Box and Muller 1958), whose C code is given in Algorithm C.1—note that this routine returns pairs of random variables. Alternative methods for generating Gaussian random numbers are given by Thomas, Luk, Leong *et al.* (2007).

Pseudocolor generation. In many applications, it is convenient to be able to visualize the set of labels assigned to an image (or to image features such as lines). One of the easiest ways to do this is to assign a unique color to each integer label. In my work, I have found it convenient to distribute these labels in a quasi-uniform fashion around the RGB color cube using the following idea.

For each (non-negative) label value, consider the bits as being split among the three color channels, e.g., for a nine-bit value, the bits could be labeled RGBRGBRGB. After collecting each of the three color values, *reverse* the bits so that the low-order bits vary the most quickly. In practice, for eight-bit color channels, this bit reverse can be stored in a table or a complete table mapping from labels to pseudocolors (say with 4092 entries) can be pre-computed. Figure 8.16 shows an example of such a pseudo-color mapping.

GPU implementation

The advent of programmable GPUs with capabilities such as pixel shaders and compute shaders has led to the development of fast computer vision algorithms for real-time applications such as segmentation, tracking, stereo, and motion estimation (Pock, Unger, Cremers *et al.* 2008; Vineet and Narayanan 2008; Zach, Gallup, and Frahm 2008). A good source for learning about such algorithms is the CVPR 2008 workshop on Visual Computer Vision on GPUs (CVGPU), http://www.cs.unc.edu/~jmf/Workshop_on_Computer_Vision_on_GPU.html, whose papers can be found on the CVPR 2008 proceedings DVD. Additional sources

```

double urand()
{
    return ((double) rand()) / ((double) RAND_MAX);
}
void grand(double& g1, double& g2)
{
#ifndef M_PI
#define M_PI 3.14159265358979323846
#endif // M_PI

    double n1 = urand();
    double n2 = urand();
    double x1 = n1 + (n1 == 0); /* guard against log(0) */
    double sqlogn1 = sqrt(-2.0 * log (x1));
    double angl = (2.0 * M_PI) * n2;
    g1 = sqlogn1 * cos(angl);
    g2 = sqlogn1 * sin(angl);
}

```

Algorithm C.1 C algorithm for Gaussian random noise generation, using the Box–Muller transform.

for GPU algorithms include the GPGPU Web site and workshops, <http://gpgpu.org/>, and the OpenVIDIA Web site, <http://openvidia.sourceforge.net/index.php/OpenVIDIA>.

C.3 Slides and lectures

As I mentioned in the preface, I hope to post slides corresponding to the material in the book. Until these are ready, your best bet is to look at the slides from the courses I have co-taught at the University of Washington, as well as related courses that have used a similar syllabus. Here is a partial list of such courses:

UW 455: Undergraduate Computer Vision, <http://www.cs.washington.edu/education/courses/455/>.

UW 576: Graduate Computer Vision, <http://www.cs.washington.edu/education/courses/576/>.

Stanford CS233B: Introduction to Computer Vision, <http://vision.stanford.edu/teaching/cs233b/>.

MIT 6.869: Advances in Computer Vision, <http://people.csail.mit.edu/torralba/courses/6.869/6.869.computervision.htm>.

Berkeley CS 280: Computer Vision, <http://www.eecs.berkeley.edu/~trevor/CS280.html>.

UNC COMP 776: Computer Vision, <http://www.cs.unc.edu/~lazebnik/spring10/>.

Middlebury CS 453: Computer Vision, <http://www.cs.middlebury.edu/~schar/courses/cs453-s10/>.

Related courses have also been taught on the topic of Computational Photography, e.g.,

CMU 15-463: Computational Photography, <http://graphics.cs.cmu.edu/courses/15-463/>.

MIT 6.815/6.865: Advanced Computational Photography, <http://stellar.mit.edu/S/course/6/sp09/6.815/>.

Stanford CS 448A: Computational photography on cell phones, <http://graphics.stanford.edu/courses/cs448a-10/>.

SIGGRAPH courses on Computational Photography, <http://web.media.mit.edu/~raskar/photo/>.

There is also an excellent set of on-line lectures available on a range of computer vision topics, such as belief propagation and graph cuts, at the UW-MSR Course of Vision Algorithms <http://www.cs.washington.edu/education/courses/577/04sp/>.

C.4 Bibliography

While a bibliography (BibTex .bib file) for all of the references cited in this book is available on the book's Web site, a much more comprehensive partially annotated bibliography of nearly *all* computer vision publications is maintained by Keith Price at <http://iris.usc.edu/Vision-Notes/bibliography/contents.html>. There is also a searchable computer graphics bibliography at <http://www.siggraph.org/publications/bibliography/>. Additional good sources for technical papers are Google Scholar and CiteSeer^X.

References

- Abdel-Hakim, A. E. and Farag, A. A. (2006). CSIFT: A SIFT descriptor with color invariant characteristics. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2006)*, pp. 1978–1983, New York City, NY.
- Adelson, E. H. and Bergen, J. (1991). The plenoptic function and the elements of early vision. In *Computational Models of Visual Processing*, pp. 3–20.
- Adelson, E. H., Simoncelli, E., and Hingorani, R. (1987). Orthogonal pyramid transforms for image coding. In *SPIE Vol. 845, Visual Communications and Image Processing II*, pp. 50–58, Cambridge, Massachusetts.
- Adiv, G. (1989). Inherent ambiguities in recovering 3-D motion and structure from a noisy flow field. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(5):477–490.
- Agarwal, A. and Triggs, B. (2006). Recovering 3D human pose from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):44–58.
- Agarwal, S. and Roth, D. (2002). Learning a sparse representation for object detection. In *Seventh European Conference on Computer Vision (ECCV 2002)*, pp. 113–127, Copenhagen.
- Agarwal, S., Snavely, N., Seitz, S. M., and Szeliski, R. (2010). Bundle adjustment in the large. In *Eleventh European Conference on Computer Vision (ECCV 2010)*, Heraklion, Crete.
- Agarwal, S., Snavely, N., Simon, I., Seitz, S. M., and Szeliski, R. (2009). Building Rome in a day. In *Twelfth IEEE International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan.
- Agarwal, S., Furukawa, Y., Snavely, N., Curless, B., Seitz, S. M., and Szeliski, R. (2010). Reconstructing Rome. *Computer*, 43(6):40–47.
- Agarwala, A. (2007). Efficient gradient-domain compositing using quadtrees. *ACM Transactions on Graphics*, 26(3).
- Agarwala, A., Hertzmann, A., Seitz, S., and Salesin, D. (2004). Keyframe-based tracking for rotoscoping and animation. *ACM Transactions on Graphics (Proc. SIGGRAPH 2004)*, 23(3):584–591.
- Agarwala, A., Agrawala, M., Cohen, M., Salesin, D., and Szeliski, R. (2006). Photographing long scenes with multi-viewpoint panoramas. *ACM Transactions on Graphics (Proc. SIGGRAPH 2006)*, 25(3):853–861.
- Agarwala, A., Dontcheva, M., Agrawala, M., Drucker, S., Colburn, A., Curless, B., Salesin, D. H., and Cohen, M. F. (2004). Interactive digital photomontage. *ACM Transactions on Graphics (Proc. SIGGRAPH 2004)*, 23(3):292–300.
- Agarwala, A., Zheng, K. C., Pal, C., Agrawala, M., Cohen, M., Curless, B., Salesin, D., and Szeliski, R. (2005). Panoramic video textures. *ACM Transactions on Graphics (Proc. SIGGRAPH 2005)*, 24(3):821–827.
- Aggarwal, J. K. and Nandhakumar, N. (1988). On the computation of motion from sequences of images—a review. *Proceedings of the IEEE*, 76(8):917–935.

- Agin, G. J. and Binford, T. O. (1976). Computer description of curved objects. *IEEE Transactions on Computers*, C-25(4):439–449.
- Ahonen, T., Hadid, A., and Pietikäinen, M. (2006). Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):2037–2041.
- Akenine-Möller, T. and Haines, E. (2002). *Real-Time Rendering*. A K Peters, Wellesley, Massachusetts, second edition.
- Al-Baali, M. and Fletcher, R. (1986). An efficient line search for nonlinear least squares. *Journal Journal of Optimization Theory and Applications*, 48(3):359–377.
- Alahari, K., Kohli, P., and Torr, P. (2011). Dynamic hybrid algorithms for discrete MAP MRF inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., and Silva, C. T. (2003). Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):3–15.
- Aliaga, D. G., Funkhouser, T., Yanovsky, D., and Carlboom, I. (2003). Sea of images. *IEEE Computer Graphics and Applications*, 23(6):22–30.
- Allen, B., Curless, B., and Popović, Z. (2003). The space of human body shapes: reconstruction and parameterization from range scans. *ACM Transactions on Graphics (Proc. SIGGRAPH 2003)*, 22(3):587–594.
- Allgower, E. L. and Georg, K. (2003). *Introduction to Numerical Continuation Methods*. Society for Industrial and Applied Mathematics.
- Aloimonos, J. (1990). Perspective approximations. *Image and Vision Computing*, 8:177–192.
- Alpert, S., Galun, M., Basri, R., and Brandt, A. (2007). Image segmentation by probabilistic bottom-up aggregation and cue integration. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Amini, A. A., Weymouth, T. E., and Jain, R. C. (1990). Using dynamic programming for solving variational problems in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9):855–867.
- Anandan, P. (1984). Computing dense displacement fields with confidence measures in scenes containing occlusion. In *Image Understanding Workshop*, pp. 236–246, New Orleans.
- Anandan, P. (1989). A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2(3):283–310.
- Anandan, P. and Irani, M. (2002). Factorization with uncertainty. *International Journal of Computer Vision*, 49(2-3):101–116.
- Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J. W. et al. (1999). *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, 3rd edition.
- Andrieu, C., de Freitas, N., Doucet, A., and Jordan, M. I. (2003). An introduction to MCMC for machine learning. *Machine Learning*, 50(1-2):5–43.
- Andriluka, M., Roth, S., and Schiele, B. (2008). People-tracking-by-detection and people-detection-by-tracking. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Andriluka, M., Roth, S., and Schiele, B. (2009). Pictorial structures revisited: People detection and articulated pose estimation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, Miami Beach, FL.

- Andriluka, M., Roth, S., and Schiele, B. (2010). Monocular 3d pose estimation and tracking by detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2010)*, San Francisco, CA.
- Anguelov, D., Srinivasan, P., Koller, D., Thrun, S., Rodgers, J., and Davis, J. (2005). SCAPE: Shape completion and animation of people. *ACM Transactions on Graphics (Proc. SIGGRAPH 2005)*, 24(3):408–416.
- Ansar, A., Castano, A., and Matthies, L. (2004). Enhanced real-time stereo using bilateral filtering. In *International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT)*.
- Antone, M. and Teller, S. (2002). Scalable extrinsic calibration of omni-directional image networks. *International Journal of Computer Vision*, 49(2-3):143–174.
- Arbeláez, P., Maire, M., Fowlkes, C., and Malik, J. (2010). *Contour Detection and Hierarchical Image Segmentation*. Technical Report UCB/EECS-2010-17, EECS Department, University of California, Berkeley. Submitted to PAMI.
- Argyriou, V. and Vlachos, T. (2003). Estimation of sub-pixel motion using gradient cross-correlation. *Electronic Letters*, 39(13):980–982.
- Arikan, O. and Forsyth, D. A. (2002). Interactive motion generation from examples. *ACM Transactions on Graphics*, 21(3):483–490.
- Arnold, R. D. (1983). *Automated Stereo Perception*. Technical Report AIM-351, Artificial Intelligence Laboratory, Stanford University.
- Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., and Wu, A. Y. (1998). An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *Journal of the ACM*, 45(6):891–923.
- Ashdown, I. (1993). Near-field photometry: A new approach. *Journal of the Illuminating Engineering Society*, 22(1):163–180.
- Atkinson, K. B. (1996). *Close Range Photogrammetry and Machine Vision*. Whittles Publishing, Scotland, UK.
- Aurich, V. and Weule, J. (1995). Non-linear Gaussian filters performing edge preserving diffusion. In *17th DAGM-Symposium*, pp. 538–545, Bielefeld.
- Avidan, S. (2001). Support vector tracking. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2001)*, pp. 283–290, Kauai, Hawaii.
- Avidan, S., Baker, S., and Shan, Y. (2010). Special issue on Internet Vision. *Proceedings of the IEEE*, 98(8):1367–1369.
- Axelsson, O. (1996). *Iterative Solution Methods*. Cambridge University Press, Cambridge.
- Ayache, N. (1989). *Vision Stéréoscopique et Perception Multisensorielle*. InterEditions, Paris.
- Azarbeyjani, A. and Pentland, A. P. (1995). Recursive estimation of motion, structure, and focal length. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6):562–575.
- Azuma, R. T., Baillot, Y., Behringer, R., Feiner, S. K., Julier, S., and MacIntyre, B. (2001). Recent advances in augmented reality. *IEEE Computer Graphics and Applications*, 21(6):34–47.
- Bab-Hadiashar, A. and Suter, D. (1998a). Robust optic flow computation. *International Journal of Computer Vision*, 29(1):59–77.
- Bab-Hadiashar, A. and Suter, D. (1998b). Robust total least squares based optic flow computation. In *Asian Conference on Computer Vision (ACCV'98)*, pp. 566–573, Hong Kong.

- Badra, F., Qumsieh, A., and Dudek, G. (1998). Rotation and zooming in image mosaicing. In *IEEE Workshop on Applications of Computer Vision (WACV'98)*, pp. 50–55, Princeton.
- Bae, S., Paris, S., and Durand, F. (2006). Two-scale tone management for photographic look. *ACM Transactions on Graphics*, 25(3):637–645.
- Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison Wesley.
- Bai, X. and Sapiro, G. (2009). Geodesic matting: A framework for fast interactive image and video segmentation and matting. *International Journal of Computer Vision*, 82(2):113–132.
- Bajcsy, R. and Kovacic, S. (1989). Multiresolution elastic matching. *Computer Vision, Graphics, and Image Processing*, 46(1):1–21.
- Baker, H. H. (1977). Three-dimensional modeling. In *Fifth International Joint Conference on Artificial Intelligence (IJCAI-77)*, pp. 649–655.
- Baker, H. H. (1982). *Depth from Edge and Intensity Based Stereo*. Technical Report AIM-347, Artificial Intelligence Laboratory, Stanford University.
- Baker, H. H. (1989). Building surfaces of evolution: The weaving wall. *International Journal of Computer Vision*, 3(1):50–71.
- Baker, H. H. and Binford, T. O. (1981). Depth from edge and intensity based stereo. In *IJCAI81*, pp. 631–636.
- Baker, H. H. and Bolles, R. C. (1989). Generalizing epipolar-plane image analysis on the spatiotemporal surface. *International Journal of Computer Vision*, 3(1):33–49.
- Baker, S. and Kanade, T. (2002). Limits on super-resolution and how to break them. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1167–1183.
- Baker, S. and Matthews, I. (2004). Lucas-Kanade 20 years on: A unifying framework: Part 1: The quantity approximated, the warp update rule, and the gradient descent approximation. *International Journal of Computer Vision*, 56(3):221–255.
- Baker, S. and Nayar, S. (1999). A theory of single-viewpoint catadioptric image formation. *International Journal of Computer Vision*, 5(2):175–196.
- Baker, S. and Nayar, S. K. (2001). Single viewpoint catadioptric cameras. In Benosman, R. and Kang, S. B. (eds), *Panoramic Vision: Sensors, Theory, and Applications*, pp. 39–71, Springer, New York.
- Baker, S., Gross, R., and Matthews, I. (2003). *Lucas-Kanade 20 Years On: A Unifying Framework: Part 3*. Technical Report CMU-RI-TR-03-35, The Robotics Institute, Carnegie Mellon University.
- Baker, S., Gross, R., and Matthews, I. (2004). *Lucas-Kanade 20 Years On: A Unifying Framework: Part 4*. Technical Report CMU-RI-TR-04-14, The Robotics Institute, Carnegie Mellon University.
- Baker, S., Szeliski, R., and Anandan, P. (1998). A layered approach to stereo reconstruction. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'98)*, pp. 434–441, Santa Barbara.
- Baker, S., Gross, R., Ishikawa, T., and Matthews, I. (2003). *Lucas-Kanade 20 Years On: A Unifying Framework: Part 2*. Technical Report CMU-RI-TR-03-01, The Robotics Institute, Carnegie Mellon University.
- Baker, S., Black, M., Lewis, J. P., Roth, S., Scharstein, D., and Szeliski, R. (2007). A database and evaluation methodology for optical flow. In *Eleventh International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil.
- Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M. J., and Szeliski, R. (2009). *A Database and Evaluation Methodology for Optical Flow*. Technical Report MSR-TR-2009-179, Microsoft Research.

- Ballard, D. H. (1981). Generalizing the Hough transform to detect arbitrary patterns. *Pattern Recognition*, 13(2):111–122.
- Ballard, D. H. and Brown, C. M. (1982). *Computer Vision*. Prentice-Hall, Englewood Cliffs, New Jersey.
- Banno, A., Masuda, T., Oishi, T., and Ikeuchi, K. (2008). Flying laser range sensor for large-scale site-modeling and its applications in Bayon digital archival project. *International Journal of Computer Vision*, 78(2-3):207–222.
- Bar-Hillel, A., Hertz, T., and Weinshall, D. (2005). Object class recognition by boosting a part based model. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 701–708, San Diego, CA.
- Bar-Joseph, Z., El-Yaniv, R., Lischinski, D., and Werman, M. (2001). Texture mixing and texture movie synthesis using statistical learning. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):120–135.
- Bar-Shalom, Y. and Fortmann, T. E. (1988). *Tracking and data association*. Academic Press, Boston.
- Barash, D. (2002). A fundamental relationship between bilateral filtering, adaptive smoothing, and the nonlinear diffusion equation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(6):844–847.
- Barash, D. and Comaniciu, D. (2004). A common framework for nonlinear diffusion, adaptive smoothing, bilateral filtering and mean shift. *Image and Vision Computing*, 22(1):73–81.
- Barbu, A. and Zhu, S.-C. (2003). Graph partition by Swendsen–Wang cuts. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 320–327, Nice, France.
- Barbu, A. and Zhu, S.-C. (2005). Generalizing Swendsen–Wang to sampling arbitrary posterior probabilities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(9):1239–1253.
- Barkans, A. C. (1997). High quality rendering using the Talisman architecture. In *Proceedings of the Eurographics Workshop on Graphics Hardware*.
- Barnard, S. T. (1989). Stochastic stereo matching over scale. *International Journal of Computer Vision*, 3(1):17–32.
- Barnard, S. T. and Fischler, M. A. (1982). Computational stereo. *Computing Surveys*, 14(4):553–572.
- Barnes, C., Jacobs, D. E., Sanders, J., Goldman, D. B., Rusinkiewicz, S., Finkelstein, A., and Agrawala, M. (2008). Video puppetry: A performative interface for cutout animation. *ACM Transactions on Graphics*, 27(5).
- Barreto, J. P. and Daniilidis, K. (2005). Fundamental matrix for cameras with radial distortion. In *Tenth International Conference on Computer Vision (ICCV 2005)*, pp. 625–632, Beijing, China.
- Barrett, R., Berry, M., Chan, T. F., Demmel, J., Donato, J. et al. (1994). *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, 2nd Edition. SIAM, Philadelphia, PA.
- Barron, J. L., Fleet, D. J., and Beauchemin, S. S. (1994). Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77.
- Barrow, H. G. and Tenenbaum, J. M. (1981). Computational vision. *Proceedings of the IEEE*, 69(5):572–595.
- Bartels, R. H., Beatty, J. C., and Barsky, B. A. (1987). *An Introduction to Splines for use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann Publishers, Los Altos.
- Bartoli, A. (2003). Towards gauge invariant bundle adjustment: A solution based on gauge dependent damping. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 760–765, Nice, France.

- Bartoli, A. and Sturm, P. (2003). Multiple-view structure and motion from line correspondences. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 207–212, Nice, France.
- Bartoli, A., Coquerelle, M., and Sturm, P. (2004). A framework for pencil-of-points structure-from-motion. In *Eighth European Conference on Computer Vision (ECCV 2004)*, pp. 28–40, Prague.
- Bascle, B., Blake, A., and Zisserman, A. (1996). Motion deblurring and super-resolution from an image sequence. In *Fourth European Conference on Computer Vision (ECCV'96)*, pp. 573–582, Cambridge, England.
- Bathe, K.-J. (2007). *Finite Element Procedures*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.
- Batra, D., Sukthankar, R., and Chen, T. (2008). Learning class-specific affinities for image labelling. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Baudisch, P., Tan, D., Steedly, D., Rudolph, E., Uyttendaele, M., Pal, C., and Szeliski, R. (2006). An exploration of user interface designs for real-time panoramic photography. *Australian Journal of Information Systems*, 13(2).
- Baumberg, A. (2000). Reliable feature matching across widely separated views. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2000)*, pp. 774–781, Hilton Head Island.
- Baumberg, A. M. and Hogg, D. C. (1996). Generating spatiotemporal models from examples. *Image and Vision Computing*, 14(8):525–532.
- Baumgart, B. G. (1974). *Geometric Modeling for Computer Vision*. Technical Report AIM-249, Artificial Intelligence Laboratory, Stanford University.
- Bay, H., Ferrari, V., and Van Gool, L. (2005). Wide-baseline stereo matching with line segments. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 329–336, San Diego, CA.
- Bay, H., Tuytelaars, T., and Van Gool, L. (2006). SURF: Speeded up robust features. In *Ninth European Conference on Computer Vision (ECCV 2006)*, pp. 404–417.
- Bayer, B. E. (1976). Color imaging array. US Patent No. 3,971,065.
- Beardsley, P., Torr, P., and Zisserman, A. (1996). 3D model acquisition from extended image sequences. In *Fourth European Conference on Computer Vision (ECCV'96)*, pp. 683–695, Cambridge, England.
- Beare, R. (2006). A locally constrained watershed transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7):1063–1074.
- Becker, S. and Bove, V. M. (1995). Semiautomatic 3-D model extraction from uncalibrated 2-D camera views. In *SPIE Vol. 2410, Visual Data Exploration and Analysis II*, pp. 447–461, San Jose.
- Beier, T. and Neely, S. (1992). Feature-based image metamorphosis. *Computer Graphics (SIGGRAPH '92)*, 26(2):35–42.
- Beis, J. S. and Lowe, D. G. (1999). Indexing without invariants in 3D object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):1000–1015.
- Belhumeur, P. N. (1996). A Bayesian approach to binocular stereopsis. *International Journal of Computer Vision*, 19(3):237–260.
- Belhumeur, P. N., Hespanha, J. P., and Kriegman, D. J. (1997). Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720.

- Belongie, S. and Malik, J. (1998). Finding boundaries in natural images: a new method using point descriptors and area completion. In *Fifth European Conference on Computer Vision (ECCV'98)*, pp. 751–766, Freiburg, Germany.
- Belongie, S., Malik, J., and Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522.
- Belongie, S., Fowlkes, C., Chung, F., and Malik, J. (2002). Spectral partitioning with indefinite kernels using the Nyström extension. In *Seventh European Conference on Computer Vision (ECCV 2002)*, pp. 531–543, Copenhagen.
- Bennett, E., Uyttendaele, M., Zitnick, L., Szeliski, R., and Kang, S. B. (2006). Video and image Bayesian demosaicing with a two color image prior. In *Ninth European Conference on Computer Vision (ECCV 2006)*, pp. 508–521, Graz.
- Benosman, R. and Kang, S. B. (eds). (2001). *Panoramic Vision: Sensors, Theory, and Applications*, Springer, New York.
- Berg, T. (2008). Internet vision. SUNY Stony Brook Course CSE 690, http://www.tamaraberg.com/teaching/Fall_08/.
- Berg, T. and Forsyth, D. (2006). Animals on the web. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2006)*, pp. 1463–1470, New York City, NY.
- Bergen, J. R., Anandan, P., Hanna, K. J., and Hingorani, R. (1992). Hierarchical model-based motion estimation. In *Second European Conference on Computer Vision (ECCV'92)*, pp. 237–252, Santa Margherita Liguere, Italy.
- Bergen, J. R., Burt, P. J., Hingorani, R., and Peleg, S. (1992). A three-frame algorithm for estimating two-component image motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(9):886–896.
- Berger, J. O. (1993). *Statistical Decision Theory and Bayesian Analysis*. Springer, New York, second edition.
- Bertalmio, M., Sapiro, G., Caselles, V., and Ballester, C. (2000). Image inpainting. In *ACM SIGGRAPH 2000 Conference Proceedings*, pp. 417–424.
- Bertalmio, M., Vese, L., Sapiro, G., and Osher, S. (2003). Simultaneous structure and texture image inpainting. *IEEE Transactions on Image Processing*, 12(8):882–889.
- Bertero, M., Poggio, T. A., and Torre, V. (1988). Ill-posed problems in early vision. *Proceedings of the IEEE*, 76(8):869–889.
- Besag, J. (1986). On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society B*, 48(3):259–302.
- Besl, P. (1989). Active optical range imaging sensors. In Sanz, J. L. (ed.), *Advances in Machine Vision*, chapter 1, pp. 1–63, Springer-Verlag.
- Besl, P. J. and Jain, R. C. (1985). Three-dimensional object recognition. *Computing Surveys*, 17(1):75–145.
- Besl, P. J. and McKay, N. D. (1992). A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256.
- Betrisey, C., Blinn, J. F., Dresevic, B., Hill, B., Hitchcock, G. et al. (2000). Displaced filtering for patterned displays. In *Society for Information Display Symposium*, pp. 296–299.
- Beymer, D. (1996). Feature correspondence by interleaving shape and texture computations. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'96)*, pp. 921–928, San Francisco.

- Bhat, D. N. and Nayar, S. K. (1998). Ordinal measures for image correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(4):415–423.
- Bickel, B., Botsch, M., Angst, R., Matusik, W., Otaduy, M., Pfister, H., and Gross, M. (2007). Multi-scale capture of facial geometry and motion. *ACM Transactions on Graphics*, 26(3).
- Billinghurst, M., Kato, H., and Poupyrev, I. (2001). The MagicBook: a transitional AR interface. *Computers & Graphics*, 25:745–753.
- Bimber, O. (2006). Computational photography—the next big step. *Computer*, 39(8):28–29.
- Birchfield, S. and Tomasi, C. (1998). A pixel dissimilarity measure that is insensitive to image sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(4):401–406.
- Birchfield, S. and Tomasi, C. (1999). Depth discontinuities by pixel-to-pixel stereo. *International Journal of Computer Vision*, 35(3):269–293.
- Birchfield, S. T., Natarajan, B., and Tomasi, C. (2007). Correspondence as energy-based segmentation. *Image and Vision Computing*, 25(8):1329–1340.
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer, New York, NY.
- Bitouk, D., Kumar, N., Dhillon, S., Belhumeur, P., and Nayar, S. K. (2008). Face swapping: Automatically replacing faces in photographs. *ACM Transactions on Graphics*, 27(3).
- Björck, A. (1996). *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics.
- Björck, A. and Dahlquist, G. (2010). *Numerical Methods in Scientific Computing*. Volume II, Society for Industrial and Applied Mathematics.
- Black, M., Yacoob, Y., Jepson, A. D., and Fleet, D. J. (1997). Learning parameterized models of image motion. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pp. 561–567, San Juan, Puerto Rico.
- Black, M. J. and Anandan, P. (1996). The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104.
- Black, M. J. and Jepson, A. D. (1996). Estimating optical flow in segmented images using variable-order parametric models with local deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):972–986.
- Black, M. J. and Jepson, A. D. (1998). EigenTracking: robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84.
- Black, M. J. and Rangarajan, A. (1996). On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *International Journal of Computer Vision*, 19(1):57–91.
- Black, M. J., Sapiro, G., Marimont, D. H., and Heeger, D. (1998). Robust anisotropic diffusion. *IEEE Transactions on Image Processing*, 7(3):421–432.
- Blackford, L. S., Demmel, J., Dongarra, J., Duff, I., Hammarling, S. et al. (2002). An updated set of basic linear algebra subprograms (BLAS). *ACM Transactions on Mathematical Software*, 28(2):135–151.
- Blake, A. and Isard, M. (1998). *Active Contours: The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion*. Springer Verlag, London.
- Blake, A. and Zisserman, A. (1987). *Visual Reconstruction*. MIT Press, Cambridge, Massachusetts.

- Blake, A., Curwen, R., and Zisserman, A. (1993). A framework for spatio-temporal control in the tracking of visual contour. *International Journal of Computer Vision*, 11(2):127–145.
- Blake, A., Kohli, P., and Rother, C. (eds). (2010). *Advances in Markov Random Fields*, MIT Press.
- Blake, A., Zimmerman, A., and Knowles, G. (1985). Surface descriptions from stereo and shading. *Image and Vision Computing*, 3(4):183–191.
- Blake, A., Rother, C., Brown, M., Perez, P., and Torr, P. (2004). Interactive image segmentation using an adaptive GMMRF model. In *Eighth European Conference on Computer Vision (ECCV 2004)*, pp. 428–441, Prague.
- Blanz, V. and Vetter, T. (1999). A morphable model for the synthesis of 3D faces. In *ACM SIGGRAPH 1999 Conference Proceedings*, pp. 187–194.
- Blanz, V. and Vetter, T. (2003). Face recognition based on fitting a 3D morphable model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25():1063–1074.
- Bleyer, M., Gelautz, M., Rother, C., and Rhemann, C. (2009). A stereo approach that handles the matting problem via image warping. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, Miami Beach, FL.
- Blinn, J. (1998). *Dirty Pixels*. Morgan Kaufmann Publishers, San Francisco.
- Blinn, J. F. (1994a). Jim Blinn’s corner: Compositing, part 1: Theory. *IEEE Computer Graphics and Applications*, 14(5):83–87.
- Blinn, J. F. (1994b). Jim Blinn’s corner: Compositing, part 2: Practice. *IEEE Computer Graphics and Applications*, 14(6):78–82.
- Blinn, J. F. and Newell, M. E. (1976). Texture and reflection in computer generated images. *Communications of the ACM*, 19(10):542–547.
- Blostein, D. and Ahuja, N. (1987). Shape from texture: Integrating texture-element extraction and surface estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(12):1233–1251.
- Bobick, A. F. (1997). Movement, activity and action: the role of knowledge in the perception of motion. *Proceedings of the Royal Society of London, B* 352:1257–1265.
- Bobick, A. F. and Intille, S. S. (1999). Large occlusion stereo. *International Journal of Computer Vision*, 33(3):181–200.
- Boden, M. A. (2006). *Mind As Machine: A History of Cognitive Science*. Oxford University Press, Oxford, England.
- Bogart, R. G. (1991). View correlation. In Arvo, J. (ed.), *Graphics Gems II*, pp. 181–190, Academic Press, Boston.
- Boiman, O., Shechtman, E., and Irani, M. (2008). In defense of nearest-neighbor based image classification. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Boissonat, J.-D. (1984). Representing 2D and 3D shapes with the Delaunay triangulation. In *Seventh International Conference on Pattern Recognition (ICPR’84)*, pp. 745–748, Montreal, Canada.
- Bolles, R. C., Baker, H. H., and Hannah, M. J. (1993). The JISCT stereo evaluation. In *Image Understanding Workshop*, pp. 263–274.

- Bolles, R. C., Baker, H. H., and Marimont, D. H. (1987). Epipolar-plane image analysis: An approach to determining structure from motion. *International Journal of Computer Vision*, 1:7–55.
- Bookstein, F. L. (1989). Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6):567–585.
- Borenstein, E. and Ullman, S. (2008). Combined top-down/bottom-up segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(12):2109–2125.
- Borgefors, G. (1986). Distance transformations in digital images. *Computer Vision, Graphics and Image Processing*, 34(3):227–248.
- Bouchard, G. and Triggs, B. (2005). Hierarchical part-based visual object categorization. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 709–714, San Diego, CA.
- Bougnoux, S. (1998). From projective to Euclidean space under any practical situation, a criticism of self-calibration. In *Sixth International Conference on Computer Vision (ICCV'98)*, pp. 790–798, Bombay.
- Bouguet, J.-Y. and Perona, P. (1999). 3D photography using shadows in dual-space geometry. *International Journal of Computer Vision*, 35(2):129–149.
- Boult, T. E. and Kender, J. R. (1986). Visual surface reconstruction using sparse depth data. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'86)*, pp. 68–76, Miami Beach.
- Bourdev, L. and Malik, J. (2009). Poselets: Body part detectors trained using 3D human pose annotations. In *Twelfth International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan.
- Bovik, A. (ed.). (2000). *Handbook of Image and Video Processing*, Academic Press, San Diego.
- Bowyer, K. W., Kranenburg, C., and Dougherty, S. (2001). Edge detector evaluation using empirical ROC curves. *Computer Vision and Image Understanding*, 84(1):77–103.
- Box, G. E. P. and Muller, M. E. (1958). A note on the generation of random normal deviates. *Annals of Mathematical Statistics*, 29(2).
- Boyer, E. and Berger, M. O. (1997). 3D surface reconstruction using occluding contours. *International Journal of Computer Vision*, 22(3):219–233.
- Boykov, Y. and Funka-Lea, G. (2006). Graph cuts and efficient N-D image segmentation. *International Journal of Computer Vision*, 70(2):109–131.
- Boykov, Y. and Jolly, M.-P. (2001). Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *Eighth International Conference on Computer Vision (ICCV 2001)*, pp. 105–112, Vancouver, Canada.
- Boykov, Y. and Kolmogorov, V. (2003). Computing geodesics and minimal surfaces via graph cuts. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 26–33, Nice, France.
- Boykov, Y. and Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137.
- Boykov, Y. and Kolmogorov, V. (2010). Basic graph cut algorithms. In Blake, A., Kohli, P., and Rother, C. (eds), *Advances in Markov Random Fields*, MIT Press.
- Boykov, Y., Veksler, O., and Zabih, R. (1998). A variable window approach to early vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1283–1294.

- Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239.
- Boykov, Y., Veksler, O., and Zabih, R. (2010). Optimizing multi-label MRFs by move making algorithms. In Blake, A., Kohli, P., and Rother, C. (eds), *Advances in Markov Random Fields*, MIT Press.
- Boykov, Y., Kolmogorov, V., Cremers, D., and Delong, A. (2006). An integral solution to surface evolution PDEs via Geo-cuts. In *Ninth European Conference on Computer Vision (ECCV 2006)*, pp. 409–422.
- Bracewell, R. N. (1986). *The Fourier Transform and its Applications*. McGraw-Hill, New York, 2nd edition.
- Bradley, D., Boubekeur, T., and Heidrich, W. (2008). Accurate multi-view reconstruction using robust binocular stereo and surface meshing. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Bradsky, G. and Kaehler, A. (2008). *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly, Sebastopol, CA.
- Brandt, A. (1986). Algebraic multigrid theory: The symmetric case. *Applied Mathematics and Computation*, 19(1-4):23–56.
- Bregler, C. and Malik, J. (1998). Tracking people with twists and exponential maps. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'98)*, pp. 8–15, Santa Barbara.
- Bregler, C., Covell, M., and Slaney, M. (1997). Video rewrite: Driving visual speech with audio. In *ACM SIGGRAPH 1997 Conference Proceedings*, pp. 353–360.
- Bregler, C., Malik, J., and Pullen, K. (2004). Twist based acquisition and tracking of animal and human kinematics. *International Journal of Computer Vision*, 56(3):179–194.
- Breu, H., Gil, J., Kirkpatrick, D., and Werman, M. (1995). Linear time Euclidean distance transform algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):529–533.
- Brice, C. R. and Fennema, C. L. (1970). Scene analysis using regions. *Artificial Intelligence*, 1(3-4):205–226.
- Briggs, W. L., Henson, V. E., and McCormick, S. F. (2000). *A Multigrid Tutorial*. Society for Industrial and Applied Mathematics, Philadelphia, second edition.
- Brillaut-O'Mahoney, B. (1991). New method for vanishing point detection. *Computer Vision, Graphics, and Image Processing*, 54(2):289–300.
- Brinkmann, R. (2008). *The Art and Science of Digital Compositing*. Morgan Kaufmann Publishers, San Francisco, 2nd edition.
- Brooks, R. A. (1981). Symbolic reasoning among 3-D models and 2-D images. *Artificial Intelligence*, 17:285–348.
- Brown, D. C. (1971). Close-range camera calibration. *Photogrammetric Engineering*, 37(8):855–866.
- Brown, L. G. (1992). A survey of image registration techniques. *Computing Surveys*, 24(4):325–376.
- Brown, M. and Lowe, D. (2002). Invariant features from interest point groups. In *British Machine Vision Conference*, pp. 656–665, Cardiff, Wales.
- Brown, M. and Lowe, D. (2003). Unsupervised 3D object recognition and reconstruction in unordered datasets. In *International Conference on 3D Imaging and Modelling*, pp. 1218–1225, Nice, France.
- Brown, M. and Lowe, D. (2007). Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1):59–73.

- Brown, M., Hartley, R., and Nistér, D. (2007). Minimal solutions for panoramic stitching. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Brown, M., Szeliski, R., and Winder, S. (2004). *Multi-Image Matching Using Multi-Scale Oriented Patches*. Technical Report MSR-TR-2004-133, Microsoft Research.
- Brown, M., Szeliski, R., and Winder, S. (2005). Multi-image matching using multi-scale oriented patches. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 510–517, San Diego, CA.
- Brown, M. Z., Burschka, D., and Hager, G. D. (2003). Advances in computational stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):993–1008.
- Brox, T., Bregler, C., and Malik, J. (2009). Large displacement optical flow. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, Miami Beach, FL.
- Brox, T., Bruhn, A., Papenberg, N., and Weickert, J. (2004). High accuracy optical flow estimation based on a theory for warping. In *Eighth European Conference on Computer Vision (ECCV 2004)*, pp. 25–36, Prague.
- Brubaker, S. C., Wu, J., Sun, J., Mullin, M. D., and Rehg, J. M. (2008). On the design of cascades of boosted ensembles for face detection. *International Journal of Computer Vision*, 77(1-3):65–86.
- Bruhn, A., Weickert, J., and Schnörr, C. (2005). Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):211–231.
- Bruhn, A., Weickert, J., Kohlberger, T., and Schnörr, C. (2006). A multigrid platform for real-time motion computation with discontinuity-preserving variational methods. *International Journal of Computer Vision*, 70(3):257–277.
- Buades, A., Coll, B., and Morel, J.-M. (2008). Nonlocal image and movie denoising. *International Journal of Computer Vision*, 76(2):123–139.
- Bălan, A. O. and Black, M. J. (2008). The naked truth: Estimating body shape under clothing. In *Tenth European Conference on Computer Vision (ECCV 2008)*, pp. 15–29, Marseilles.
- Buchanan, A. and Fitzgibbon, A. (2005). Damped Newton algorithms for matrix factorization with missing data. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 316–322, San Diego, CA.
- Buck, I., Finkelstein, A., Jacobs, C., Klein, A., Salesin, D. H., Seims, J., Szeliski, R., and Toyama, K. (2000). Performance-driven hand-drawn animation. In *Symposium on Non Photorealistic Animation and Rendering*, pp. 101–108, Annecy.
- Buehler, C., Bosse, M., McMillan, L., Gortler, S. J., and Cohen, M. F. (2001). Unstructured Lumigraph rendering. In *ACM SIGGRAPH 2001 Conference Proceedings*, pp. 425–432.
- Bugayevskiy, L. M. and Snyder, J. P. (1995). *Map Projections: A Reference Manual*. CRC Press.
- Burger, W. and Burge, M. J. (2008). *Digital Image Processing: An Algorithmic Introduction Using Java*. Springer, New York, NY.
- Burl, M. C., Weber, M., and Perona, P. (1998). A probabilistic approach to object recognition using local photometry and global geometry. In *Fifth European Conference on Computer Vision (ECCV'98)*, pp. 628–641, Freiburg, Germany.
- Burns, J. B., Hanson, A. R., and Riseman, E. M. (1986). Extracting straight lines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(4):425–455.

- Burns, P. D. and Williams, D. (1999). Using slanted edge analysis for color registration measurement. In *IS&T PICS Conference*, pp. 51–53.
- Burt, P. J. and Adelson, E. H. (1983a). The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, COM-31(4):532–540.
- Burt, P. J. and Adelson, E. H. (1983b). A multiresolution spline with applications to image mosaics. *ACM Transactions on Graphics*, 2(4):217–236.
- Burt, P. J. and Kolczynski, R. J. (1993). Enhanced image capture through fusion. In *Fourth International Conference on Computer Vision (ICCV'93)*, pp. 173–182, Berlin, Germany.
- Byrød, M. and øAström, K. (2009). Bundle adjustment using conjugate gradients with multiscale preconditioning. In *British Machine Vision Conference (BMVC 2009)*.
- Cai, D., He, X., Hu, Y., Han, J., and Huang, T. (2007). Learning a spatially smooth subspace for face recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Campbell, N. D. F., Vogiatzis, G., Hernández, C., and Cipolla, R. (2008). Using multiple hypotheses to improve depth-maps for multi-view stereo. In *Tenth European Conference on Computer Vision (ECCV 2008)*, pp. 766–779, Marseilles.
- Can, A., Stewart, C., Roysam, B., and Tanenbaum, H. (2002). A feature-based, robust, hierarchical algorithm for registering pairs of images of the curved human retina. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):347–364.
- Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698.
- Cao, Z., Yin, Q., Tang, X., and Sun, J. (2010). Face recognition with learning-based descriptor. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2010)*, San Francisco, CA.
- Capel, D. (2004). *Image Mosaicing and Super-resolution. Distinguished Dissertation Series, British Computer Society*, Springer-Verlag.
- Capel, D. and Zisserman, A. (1998). Automated mosaicing with super-resolution zoom. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'98)*, pp. 885–891, Santa Barbara.
- Capel, D. and Zisserman, A. (2000). Super-resolution enhancement of text image sequences. In *Fifteenth International Conference on Pattern Recognition (ICPR'2000)*, pp. 600–605, Barcelona, Spain.
- Capel, D. and Zisserman, A. (2003). Computer vision applied to super resolution. *IEEE Signal Processing Magazine*, 20(3):75–86.
- Capel, D. P. (2001). *Super-resolution and Image Mosaicing*. Ph.D. thesis, University of Oxford.
- Caprile, B. and Torre, V. (1990). Using vanishing points for camera calibration. *International Journal of Computer Vision*, 4(2):127–139.
- Carneiro, G. and Jepson, A. (2005). The distinctiveness, detectability, and robustness of local image features. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 296–301, San Diego, CA.
- Carneiro, G. and Lowe, D. (2006). Sparse flexible models of local features. In *Ninth European Conference on Computer Vision (ECCV 2006)*, pp. 29–43.

- Carnevali, P., Coletti, L., and Patarnello, S. (1985). Image processing by simulated annealing. *IBM Journal of Research and Development*, 29(6):569–579.
- Carranza, J., Theobalt, C., Magnor, M. A., and Seidel, H.-P. (2003). Free-viewpoint video of human actors. *ACM Transactions on Graphics (Proc. SIGGRAPH 2003)*, 22(3):569–577.
- Carroll, R., Agrawala, M., and Agarwala, A. (2009). Optimizing content-preserving projections for wide-angle images. *ACM Transactions on Graphics*, 28(3).
- Caselles, V., Kimmel, R., and Sapiro, G. (1997). Geodesic active contours. *International Journal of Computer Vision*, 21(1):61–79.
- Catmull, E. and Smith, A. R. (1980). 3-D transformations of images in scanline order. *Computer Graphics (SIGGRAPH '80)*, 14(3):279–285.
- Celniker, G. and Gossard, D. (1991). Deformable curve and surface finite-elements for free-form shape design. *Computer Graphics (SIGGRAPH '91)*, 25(4):257–266.
- Chakrabarti, A., Scharstein, D., and Zickler, T. (2009). An empirical camera model for internet color vision. In *British Machine Vision Conference (BMVC 2009)*, London, UK.
- Cham, T. J. and Cipolla, R. (1998). A statistical framework for long-range feature matching in uncalibrated image mosaicing. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'98)*, pp. 442–447, Santa Barbara.
- Cham, T.-J. and Rehg, J. M. (1999). A multiple hypothesis approach to figure tracking. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99)*, pp. 239–245, Fort Collins.
- Champeboux, G., Lavallée, S., Sautot, P., and Cinquin, P. (1992). Accurate calibration of cameras and range imaging sensors, the NPBS method. In *IEEE International Conference on Robotics and Automation*, pp. 1552–1558, Nice, France.
- Champeboux, G., Lavallée, S., Szeliski, R., and Brunie, L. (1992). From accurate range imaging sensor calibration to accurate model-based 3-D object localization. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'92)*, pp. 83–89, Champaign, Illinois.
- Chan, A. B. and Vasconcelos, N. (2009). Layered dynamic textures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(10):1862–1879.
- Chan, T. F. and Vese, L. A. (1992). Active contours without edges. *IEEE Transactions on Image Processing*, 10(2):266–277.
- Chan, T. F., Osher, S., and Shen, J. (2001). The digital TV filter and nonlinear denoising. *IEEE Transactions on Image Processing*, 10(2):231–241.
- Chang, M. M., Tekalp, A. M., and Sezan, M. I. (1997). Simultaneous motion estimation and segmentation. *IEEE Transactions on Image Processing*, 6(9):1326–1333.
- Chaudhuri, S. (2001). *Super-Resolution Imaging*. Springer.
- Chaudhuri, S. and Rajagopalan, A. N. (1999). *Depth from Defocus: A Real Aperture Imaging Approach*. Springer.
- Cheeseman, P., Kanefsky, B., Hanson, R., and Stutz, J. (1993). *Super-Resolved Surface Reconstruction From Multiple Images*. Technical Report FIA-93-02, NASA Ames Research Center, Artificial Intelligence Branch.
- Chellappa, R., Wilson, C., and Sirohey, S. (1995). Human and machine recognition of faces: A survey. *Proceedings of the IEEE*, 83(5):705–740.

- Chen, B., Neubert, B., Ofek, E., Deussen, O., and Cohen, M. F. (2009). Integrated videos and maps for driving directions. In *UIST '09: Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pp. 223–232, Victoria, BC, Canada, New York, NY, USA.
- Chen, C.-Y. and Klette, R. (1999). Image stitching - comparisons and new techniques. In *Computer Analysis of Images and Patterns (CAIP'99)*, pp. 615–622, Ljubljana.
- Chen, J. and Chen, B. (2008). Architectural modeling from sparsely scanned range data. *International Journal of Computer Vision*, 78(2-3):223–236.
- Chen, J., Paris, S., and Durand, F. (2007). Real-time edge-aware image processing with the bilateral grid. *ACM Transactions on Graphics*, 26(3).
- Chen, S. and Williams, L. (1993). View interpolation for image synthesis. In *ACM SIGGRAPH 1993 Conference Proceedings*, pp. 279–288.
- Chen, S. E. (1995). QuickTime VR – an image-based approach to virtual environment navigation. In *ACM SIGGRAPH 1995 Conference Proceedings*, pp. 29–38, Los Angeles.
- Chen, Y. and Medioni, G. (1992). Object modeling by registration of multiple range images. *Image and Vision Computing*, 10(3):145–155.
- Cheng, L., Vishwanathan, S. V. N., and Zhang, X. (2008). Consistent image analogies using semi-supervised learning. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Cheng, Y. (1995). Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799.
- Chiang, M.-C. and Boult, T. E. (1996). Efficient image warping and super-resolution. In *IEEE Workshop on Applications of Computer Vision (WACV'96)*, pp. 56–61, Sarasota.
- Chiu, K. and Raskar, R. (2009). Computer vision on tap. In *Second IEEE Workshop on Internet Vision*, Miami Beach, Florida.
- Chou, P. B. and Brown, C. M. (1990). The theory and practice of Bayesian image labeling. *International Journal of Computer Vision*, 4(3):185–210.
- Christensen, G., Joshi, S., and Miller, M. (1997). Volumetric transformation of brain anatomy. *IEEE Transactions on Medical Imaging*, 16(6):864–877.
- Christy, S. and Horaud, R. (1996). Euclidean shape and motion from multiple perspective views by affine iterations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(11):1098–1104.
- Chuang, Y.-Y., Curless, B., Salesin, D. H., and Szeliski, R. (2001). A Bayesian approach to digital matting. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2001)*, pp. 264–271, Kauai, Hawaii.
- Chuang, Y.-Y., Agarwala, A., Curless, B., Salesin, D. H., and Szeliski, R. (2002). Video matting of complex scenes. *ACM Transactions on Graphics (Proc. SIGGRAPH 2002)*, 21(3):243–248.
- Chuang, Y.-Y., Goldman, D. B., Curless, B., Salesin, D. H., and Szeliski, R. (2003). Shadow matting. *ACM Transactions on Graphics (Proc. SIGGRAPH 2003)*, 22(3):494–500.
- Chuang, Y.-Y., Goldman, D. B., Zheng, K. C., Curless, B., Salesin, D. H., and Szeliski, R. (2005). Animating pictures with stochastic motion textures. *ACM Transactions on Graphics (Proc. SIGGRAPH 2005)*, 24(3):853–860.

- Chuang, Y.-Y., Zongker, D., Hindorff, J., Curless, B., Salesin, D. H., and Szeliski, R. (2000). Environment matting extensions: Towards higher accuracy and real-time capture. In *ACM SIGGRAPH 2000 Conference Proceedings*, pp. 121–130, New Orleans.
- Chui, C. K. (1992). *Wavelet Analysis and Its Applications*. Academic Press, New York.
- Chum, O. and Matas, J. (2005). Matching with PROSAC—progressive sample consensus. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 220–226, San Diego, CA.
- Chum, O. and Matas, J. (2010). Large-scale discovery of spatially related images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(2):371–377.
- Chum, O., Philbin, J., and Zisserman, A. (2008). Near duplicate image detection: min-hash and tf-idf weighting. In *British Machine Vision Conference (BMVC 2008)*, Leeds, England.
- Chum, O., Philbin, J., Sivic, J., Isard, M., and Zisserman, A. (2007). Total recall: Automatic query expansion with a generative feature model for object retrieval. In *Eleventh International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil.
- Cipolla, R. and Blake, A. (1990). The dynamic analysis of apparent contours. In *Third International Conference on Computer Vision (ICCV'90)*, pp. 616–623, Osaka, Japan.
- Cipolla, R. and Blake, A. (1992). Surface shape from the deformation of apparent contours. *International Journal of Computer Vision*, 9(2):83–112.
- Cipolla, R. and Giblin, P. (2000). *Visual Motion of Curves and Surfaces*. Cambridge University Press, Cambridge.
- Cipolla, R., Drummond, T., and Robertson, D. P. (1999). Camera calibration from vanishing points in images of architectural scenes. In *British Machine Vision Conference (BMVC99)*.
- Claus, D. and Fitzgibbon, A. (2005). A rational function lens distortion model for general cameras. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 213–219, San Diego, CA.
- Clowes, M. B. (1971). On seeing things. *Artificial Intelligence*, 2:79–116.
- Cohen, L. D. and Cohen, I. (1993). Finite-element methods for active contour models and balloons for 2-D and 3-D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1131–1147.
- Cohen, M. and Wallace, J. (1993). *Radiosity and Realistic Image Synthesis*. Morgan Kaufmann.
- Cohen, M. F. and Szeliski, R. (2006). The Moment Camera. *Computer*, 39(8):40–45.
- Collins, R. T. (1996). A space-sweep approach to true multi-image matching. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'96)*, pp. 358–363, San Francisco.
- Collins, R. T. and Liu, Y. (2003). On-line selection of discriminative tracking features. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 346–352, Nice, France.
- Collins, R. T. and Weiss, R. S. (1990). Vanishing point calculation as a statistical inference on the unit sphere. In *Third International Conference on Computer Vision (ICCV'90)*, pp. 400–403, Osaka, Japan.
- Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619.
- Comaniciu, D. and Meer, P. (2003). An algorithm for data-driven bandwidth selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):281–288.

- Conn, A. R., Gould, N. I. M., and Toint, P. L. (2000). *Trust-Region Methods*. Society for Industrial and Applied Mathematics, Philadelphia.
- Cook, R. L. and Torrance, K. E. (1982). A reflectance model for computer graphics. *ACM Transactions on Graphics*, 1(1):7–24.
- Coorg, S. and Teller, S. (2000). Spherical mosaics with quaternions and dense correlation. *International Journal of Computer Vision*, 37(3):259–273.
- Cootes, T., Edwards, G. J., and Taylor, C. J. (2001). Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685.
- Cootes, T., Cooper, D., Taylor, C., and Graham, J. (1995). Active shape models—their training and application. *Computer Vision and Image Understanding*, 61(1):38–59.
- Cootes, T., Taylor, C., Lanitis, A., Cooper, D., and Graham, J. (1993). Building and using flexible models incorporating grey-level information. In *Fourth International Conference on Computer Vision (ICCV'93)*, pp. 242–246, Berlin, Germany.
- Cootes, T. F. and Taylor, C. J. (2001). Statistical models of appearance for medical image analysis and computer vision. In *Medical Imaging*.
- Coquillart, S. (1990). Extended free-form deformations: A sculpturing tool for 3D geometric modeling. *Computer Graphics (SIGGRAPH '90)*, 24(4):187–196.
- Cormen, T. H. (2001). *Introduction to Algorithms*. MIT Press, Cambridge, Massachusetts.
- Cornelis, N., Leibe, B., Cornelis, K., and Van Gool, L. (2008). 3D urban scene modeling integrating recognition and reconstruction. *International Journal of Computer Vision*, 78(2-3):121–141.
- Corso, J. and Hager, G. (2005). Coherent regions for concise and stable image description. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 184–190, San Diego, CA.
- Costeira, J. and Kanade, T. (1995). A multi-body factorization method for motion analysis. In *Fifth International Conference on Computer Vision (ICCV'95)*, pp. 1071–1076, Cambridge, Massachusetts.
- Costen, N., Cootes, T. F., Edwards, G. J., and Taylor, C. J. (1999). Simultaneous extraction of functional face subspaces. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99)*, pp. 492–497, Fort Collins.
- Couprise, C., Grady, L., Najman, L., and Talbot, H. (2009). Power watersheds: A new image segmentation framework extending graph cuts, random walker and optimal spanning forest. In *Twelfth International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan.
- Cour, T., Bénédit, F., and Shi, J. (2005). Spectral segmentation with multiscale graph decomposition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 1123–1130, San Diego, CA.
- Cox, D., Little, J., and O’Shea, D. (2007). *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer.
- Cox, I. J. (1994). A maximum likelihood N-camera stereo algorithm. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pp. 733–739, Seattle.
- Cox, I. J., Roy, S., and Hingorani, S. L. (1995). Dynamic histogram warping of image pairs for constant image brightness. In *IEEE International Conference on Image Processing (ICIP'95)*, pp. 366–369.

- Cox, I. J., Hingorani, S. L., Rao, S. B., and Maggs, B. M. (1996). A maximum likelihood stereo algorithm. *Computer Vision and Image Understanding*, 63(3):542–567.
- Crandall, D. and Huttenlocher, D. (2007). Composite models of objects and scenes for category recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Crandall, D., Felzenszwalb, P., and Huttenlocher, D. (2005). Spatial priors for part-based recognition using statistical models. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 10–17, San Diego, CA.
- Crandall, D., Backstrom, L., Huttenlocher, D., and Kleinberg, J. (2009). Mapping the world's photos. In *18th Int. World Wide Web Conference*, pp. 761–770, Madrid.
- Crandall, D. J. and Huttenlocher, D. P. (2006). Weakly supervised learning of part-based spatial models for visual object recognition. In *Ninth European Conference on Computer Vision (ECCV 2006)*, pp. 16–29.
- Crane, R. (1997). *A Simplified Approach to Image Processing*. Prentice Hall, Upper Saddle River, NJ.
- Craswell, N. and Szummer, M. (2007). Random walks on the click graph. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 239–246, New York, NY.
- Cremers, D. and Soatto, S. (2005). Motion competition: A variational framework for piecewise parametric motion segmentation. *International Journal of Computer Vision*, 62(3):249–265.
- Cremers, D., Rousson, M., and Deriche, R. (2007). A review of statistical approaches to level set segmentation: integrating color, texture, motion and shape. *International Journal of Computer Vision*, 72(2):195–215.
- Crevier, D. (1993). *AI: The Tumultuous Search for Artificial Intelligence*. BasicBooks, New York, NY.
- Criminisi, A., Pérez, P., and Toyama, K. (2004). Region filling and object removal by exemplar-based inpainting. *IEEE Transactions on Image Processing*, 13(9):1200–1212.
- Criminisi, A., Reid, I., and Zisserman, A. (2000). Single view metrology. *International Journal of Computer Vision*, 40(2):123–148.
- Criminisi, A., Sharp, T., and Blake, A. (2008). Geos: Geodesic image segmentation. In *Tenth European Conference on Computer Vision (ECCV 2008)*, pp. 99–112, Marseilles.
- Criminisi, A., Cross, G., Blake, A., and Kolmogorov, V. (2006). Bilayer segmentation of live video. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2006)*, pp. 53–60, New York City, NY.
- Criminisi, A., Shotton, J., Blake, A., and Torr, P. (2003). Gaze manipulation for one-to-one teleconferencing. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 191–198, Nice, France.
- Criminisi, A., Kang, S. B., Swaminathan, R., Szeliski, R., and Anandan, P. (2005). Extracting layers and analyzing their specular properties using epipolar-plane-image analysis. *Computer Vision and Image Understanding*, 97(1):51–85.
- Criminisi, A., Shotton, J., Blake, A., Rother, C., and Torr, P. H. S. (2007). Efficient dense stereo with occlusion by four-state dynamic programming. *International Journal of Computer Vision*, 71(1):89–110.
- Crow, F. C. (1984). Summed-area table for texture mapping. *Computer Graphics (SIGGRAPH '84)*, 18(3):207–212.
- Crowley, J. L. and Stern, R. M. (1984). Fast computation of the difference of low-pass transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(2):212–222.

- Csurka, G. and Perronnin, F. (2008). A simple high performance approach to semantic segmentation. In *British Machine Vision Conference (BMVC 2008)*, Leeds.
- Csurka, G., Dance, C. R., Perronnin, F., and Willamowski, J. (2006). Generic visual categorization using weak geometry. In Ponce, J., Hebert, M., Schmid, C., and Zisserman, A. (eds), *Toward Category-Level Object Recognition*, pp. 207–224, Springer, New York.
- Csurka, G., Dance, C. R., Fan, L., Willamowski, J., and Bray, C. (2004). Visual categorization with bags of keypoints. In *ECCV International Workshop on Statistical Learning in Computer Vision*, Prague.
- Cui, J., Yang, Q., Wen, F., Wu, Q., Zhang, C., Van Gool, L., and Tang, X. (2008). Transductive object cutout. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Curless, B. (1999). From range scans to 3D models. *Computer Graphics*, 33(4):38–41.
- Curless, B. and Levoy, M. (1995). Better optical triangulation through spacetime analysis. In *Fifth International Conference on Computer Vision (ICCV'95)*, pp. 987–994, Cambridge, Massachusetts.
- Curless, B. and Levoy, M. (1996). A volumetric method for building complex models from range images. In *ACM SIGGRAPH 1996 Conference Proceedings*, pp. 303–312, New Orleans.
- Cutler, R. and Davis, L. S. (2000). Robust real-time periodic motion detection, analysis, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):781–796.
- Cutler, R. and Turk, M. (1998). View-based interpretation of real-time optical flow for gesture recognition. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 416–421, Nara, Japan.
- Dai, S., Baker, S., and Kang, S. B. (2009). An MRF-based deinterlacing algorithm with exemplar-based refinement. *IEEE Transactions on Image Processing*, 18(5):956–968.
- Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 886–893, San Diego, CA.
- Dalal, N., Triggs, B., and Schmid, C. (2006). Human detection using oriented histograms of flow and appearance. In *Ninth European Conference on Computer Vision (ECCV 2006)*, pp. 428–441.
- Dana, K. J., van Ginneken, B., Nayar, S. K., and Koenderink, J. J. (1999). Reflectance and texture of real world surfaces. *ACM Transactions on Graphics*, 18(1):1–34.
- Danielsson, P. E. (1980). Euclidean distance mapping. *Computer Graphics and Image Processing*, 14(3):227–248.
- Darrell, T. and Pentland, A. (1991). Robust estimation of a multi-layered motion representation. In *IEEE Workshop on Visual Motion*, pp. 173–178, Princeton, New Jersey.
- Darrell, T. and Pentland, A. (1995). Cooperative robust estimation using layers of support. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):474–487.
- Darrell, T. and Simoncelli, E. (1993). “Nulling” filters and the separation of transparent motion. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'93)*, pp. 738–739, New York.
- Darrell, T., Gordon, G., Harville, M., and Woodfill, J. (2000). Integrated person tracking using stereo, color, and pattern detection. *International Journal of Computer Vision*, 37(2):175–185.
- Darrell, T., Baker, H., Crow, F., Gordon, G., and Woodfill, J. (1997). Magic morphin mirror: face-sensitive distortion and exaggeration. In *ACM SIGGRAPH 1997 Visual Proceedings*, Los Angeles.

- Datta, R., Joshi, D., Li, J., and Wang, J. Z. (2008). Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40(2).
- Daugman, J. (2004). How iris recognition works. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(1):21–30.
- David, P., DeMenthon, D., Duraiswami, R., and Samet, H. (2004). SoftPOSIT: Simultaneous pose and correspondence determination. *International Journal of Computer Vision*, 59(3):259–284.
- Davies, R., Twining, C., and Taylor, C. (2008). *Statistical Models of Shape*. Springer-Verlag, London.
- Davis, J. (1998). Mosaics of scenes with moving objects. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'98)*, pp. 354–360, Santa Barbara.
- Davis, J., Ramamoorthi, R., and Rusinkiewicz, S. (2003). Spacetime stereo: A unifying framework for depth from triangulation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2003)*, pp. 359–366, Madison, WI.
- Davis, J., Nahab, D., Ramamoorthi, R., and Rusinkiewicz, S. (2005). Spacetime stereo: A unifying framework for depth from triangulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(2):296–302.
- Davis, L. (1975). A survey of edge detection techniques. *Computer Graphics and Image Processing*, 4(3):248–270.
- Davis, T. A. (2006). *Direct Methods for Sparse Linear Systems*. SIAM.
- Davis, T. A. (2008). Multifrontal multithreaded rank-revealing sparse QR factorization. *ACM Trans. on Mathematical Software*, (submitted).
- Davison, A., Reid, I., Molton, N. D., and Stasse, O. (2007). MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067.
- de Agapito, L., Hayman, E., and Reid, I. (2001). Self-calibration of rotating and zooming cameras. *International Journal of Computer Vision*, 45(2):107–127.
- de Berg, M., Cheong, O., van Kreveld, M., and Overmars, M. (2006). *Computational Geometry: Algorithms and Applications*. Springer, New York, NY, third edition.
- De Bonet, J. (1997). Multiresolution sampling procedure for analysis and synthesis of texture images. In *ACM SIGGRAPH 1997 Conference Proceedings*, pp. 361–368, Los Angeles.
- De Bonet, J. S. and Viola, P. (1999). Poxels: Probabilistic voxelized volume reconstruction. In *Seventh International Conference on Computer Vision (ICCV'99)*, pp. 418–425, Kerkyra, Greece.
- De Castro, E. and Morandi, C. (1987). Registration of translated and rotated images using finite Fourier transforms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):700–703.
- de Haan, G. and Bellers, E. B. (1998). Deinterlacing—an overview. *Proceedings of the IEEE*, 86:1839–1857.
- De la Torre, F. and Black, M. J. (2003). A framework for robust subspace learning. *International Journal of Computer Vision*, 54(1/2/3):117–142.
- Debevec, P. (1998). Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *ACM SIGGRAPH 1998 Conference Proceedings*, pp. 189–198.
- Debevec, P. (2006). Virtual cinematography: Relighting through computation. *Computer*, 39(8):57–65.

- Debevec, P., Hawkins, T., Tchou, C., Duiker, H.-P., Sarokin, W., and Sagar, M. (2000). Acquiring the reflectance field of a human face. In *ACM SIGGRAPH 2000 Conference Proceedings*, pp. 145–156.
- Debevec, P., Wenger, A., Tchou, C., Gardner, A., Waese, J., and Hawkins, T. (2002). A lighting reproduction approach to live-action compositing. *ACM Transactions on Graphics (Proc. SIGGRAPH 2002)*, 21(3):547–556.
- Debevec, P. E. (1999). Image-based modeling and lighting. *Computer Graphics*, 33(4):46–50.
- Debevec, P. E. and Malik, J. (1997). Recovering high dynamic range radiance maps from photographs. In *ACM SIGGRAPH 1997 Conference Proceedings*, pp. 369–378.
- Debevec, P. E., Taylor, C. J., and Malik, J. (1996). Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *ACM SIGGRAPH 1996 Conference Proceedings*, pp. 11–20, New Orleans.
- Debevec, P. E., Yu, Y., and Borshukov, G. D. (1998). Efficient view-dependent image-based rendering with projective texture-mapping. In *Eurographics Rendering Workshop 1998*, pp. 105–116.
- DeCarlo, D. and Santella, A. (2002). Stylization and abstraction of photographs. *ACM Transactions on Graphics (Proc. SIGGRAPH 2002)*, 21(3):769–776.
- DeCarlo, D., Metaxas, D., and Stone, M. (1998). An anthropometric face model using variational techniques. In *ACM SIGGRAPH 1998 Conference Proceedings*, pp. 67–74.
- Delingette, H., Hebert, M., and Ikeuchi, K. (1992). Shape representation and image segmentation using deformable surfaces. *Image and Vision Computing*, 10(3):132–144.
- Dellaert, F. and Collins, R. (1999). Fast image-based tracking by selective pixel integration. In *ICCV Workshop on Frame-Rate Vision*, pp. 1–22.
- Delong, A., Osokin, A., Isack, H. N., and Boykov, Y. (2010). Fast approximate energy minimization with label costs. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2010)*, San Francisco, CA.
- DeMenthon, D. I. and Davis, L. S. (1995). Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15(1-2):123–141.
- Demmel, J., Dongarra, J., Eijkhout, V., Fuentes, E., Petitet, A. et al. (2005). Self-adapting linear algebra algorithms and software. *Proceedings of the IEEE*, 93(2):293–312.
- Dempster, A., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39(1):1–38.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, Miami Beach, FL.
- Deriche, R. (1987). Using Canny's criteria to derive a recursively implemented optimal edge detector. *International Journal of Computer Vision*, 1(2):167–187.
- Deriche, R. (1990). Fast algorithms for low-level vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):78–87.
- Deutscher, J. and Reid, I. (2005). Articulated body motion capture by stochastic search. *International Journal of Computer Vision*, 61(2):185–205.

- Deutscher, J., Blake, A., and Reid, I. (2000). Articulated body motion capture by annealed particle filtering. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2000)*, pp. 126–133, Hilton Head Island.
- Dev, P. (1974). *Segmentation Processes in Visual Perception: A Cooperative Neural Model*. COINS Technical Report 74C-5, University of Massachusetts at Amherst.
- Dhond, U. R. and Aggarwal, J. K. (1989). Structure from stereo—a review. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6):1489–1510.
- Dick, A., Torr, P. H. S., and Cipolla, R. (2004). Modelling and interpretation of architecture from several images. *International Journal of Computer Vision*, 60(2):111–134.
- Dickinson, S., Leonardis, A., Schiele, B., and Tarr, M. J. (eds). (2007). *Object Categorization: Computer and Human Vision Perspectives*, Cambridge University Press, New York.
- Dickmanns, E. D. and Graefe, V. (1988). Dynamic monocular machine vision. *Machine Vision and Applications*, 1:223–240.
- Diebel, J. (2006). *Representing Attitude: Euler Angles, Quaternions, and Rotation Vectors*. Technical Report, Stanford University. <http://ai.stanford.edu/~diebel/attitude.html>.
- Diebel, J. R., Thrun, S., and Brüning, M. (2006). A Bayesian method for probable surface reconstruction and decimation. *ACM Transactions on Graphics*, 25(1).
- Dimitrijevic, M., Lepetit, V., and Fua, P. (2006). Human body pose detection using Bayesian spatio-temporal templates. *Computer Vision and Image Understanding*, 104(2-3):127–139.
- Dinh, H. Q., Turk, G., and Slabaugh, G. (2002). Reconstructing surfaces by volumetric regularization using radial basis functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(10):1358–1371.
- Divvala, S., Hoiem, D., Hays, J., Efros, A. A., and Hebert, M. (2009). An empirical study of context in object detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, Miami, FL.
- Dodgson, N. A. (1992). *Image Resampling*. Technical Report TR261, Wolfson College and Computer Laboratory, University of Cambridge.
- Dollàr, P., Belongie, S., and Perona, P. (2010). The fastest pedestrian detector in the west. In *British Machine Vision Conference (BMVC 2010)*, Aberystwyth, Wales, UK.
- Dollàr, P., Wojek, C., Schiele, B., and Perona, P. (2009). Pedestrian detection: A benchmark. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, Miami Beach, FL.
- Doretto, G. and Soatto, S. (2006). Dynamic shape and appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):2006–2019.
- Doretto, G., Chiuso, A., Wu, Y. N., and Soatto, S. (2003). Dynamic textures. *International Journal of Computer Vision*, 51(2):91–109.
- Dorkó, G. and Schmid, C. (2003). Selection of scale-invariant parts for object class recognition. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 634–640, Nice, France.
- Dorsey, J., Rushmeier, H., and Sillion, F. (2007). *Digital Modeling of Material Appearance*. Morgan Kaufmann, San Francisco.
- Douglas, D. H. and Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10(2):112–122.

- Drori, I., Cohen-Or, D., and Yeshurun, H. (2003). Fragment-based image completion. *ACM Transactions on Graphics (Proc. SIGGRAPH 2003)*, 22(3):303–312.
- Duda, R. O. and Hart, P. E. (1972). Use of the Hough transform to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15.
- Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Classification*. John Wiley & Sons, New York, 2nd edition.
- Dupuis, P. and Oliensis, J. (1994). An optimal control formulation and related numerical methods for a problem in shape reconstruction. *Annals of Applied Probability*, 4(2):287–346.
- Durand, F. and Dorsey, J. (2002). Fast bilateral filtering for the display of high-dynamic-range images. *ACM Transactions on Graphics (Proc. SIGGRAPH 2002)*, 21(3):257–266.
- Durand, F. and Szeliski, R. (2007). Computational photography. *IEEE Computer Graphics and Applications*, 27(2):21–22. Guest Editors’ Introduction to Special Issue.
- Durbin, R. and Willshaw, D. (1987). An analogue approach to the traveling salesman problem using an elastic net method. *Nature*, 326:689–691.
- Durbin, R., Szeliski, R., and Yuille, A. (1989). An analysis of the elastic net approach to the travelling salesman problem. *Neural Computation*, 1(3):348–358.
- Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M., and Stuetzle, W. (1995). Multiresolution analysis of arbitrary meshes. In *ACM SIGGRAPH 1995 Conference Proceedings*, pp. 173–182, Los Angeles.
- Eden, A., Uyttendaele, M., and Szeliski, R. (2006). Seamless image stitching of scenes with large motions and exposure differences. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’2006)*, pp. 2498–2505, New York, NY.
- Efros, A. A. and Freeman, W. T. (2001). Image quilting for texture synthesis and transfer. In *ACM SIGGRAPH 2001 Conference Proceedings*, pp. 341–346.
- Efros, A. A. and Leung, T. K. (1999). Texture synthesis by non-parametric sampling. In *Seventh International Conference on Computer Vision (ICCV’99)*, pp. 1033–1038, Kerkyra, Greece.
- Efros, A. A., Berg, A. C., Mori, G., and Malik, J. (2003). Recognizing action at a distance. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 726–733, Nice, France.
- Eichner, M. and Ferrari, V. (2009). Better appearance models for pictorial structures. In *British Machine Vision Conference (BMVC 2009)*.
- Eisemann, E. and Durand, F. (2004). Flash photography enhancement via intrinsic relighting. *ACM Transactions on Graphics*, 23(3):673–678.
- Eisert, P., Steinbach, E., and Girod, B. (2000). Automatic reconstruction of stationary 3-D objects from multiple uncalibrated camera views. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(2):261–277.
- Eisert, P., Wiegand, T., and Girod, B. (2000). Model-aided coding: a new approach to incorporate facial animation into motion-compensated video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(3):344–358.
- Ekman, P. and Friesen, W. V. (1978). *Facial Action Coding System: A Technique for the Measurement of Facial Movement*. Consulting Psychologists press, Palo Alto, CA.

- El-Melegy, M. and Farag, A. (2003). Nonmetric lens distortion calibration: Closed-form solutions, robust estimation and model selection. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 554–559, Nice, France.
- Elder, J. H. (1999). Are edges incomplete? *International Journal of Computer Vision*, 34(2/3):97–122.
- Elder, J. H. and Goldberg, R. M. (2001). Image editing in the contour domain. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(3):291–296.
- Elder, J. H. and Zucker, S. W. (1998). Local scale control for edge detection and blur estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(7):699–716.
- Engels, C., Stewénius, H., and Nistér, D. (2006). Bundle adjustment rules. In *Photogrammetric Computer Vision (PCV'06)*, Bonn, Germany.
- Engl, H. W., Hanke, M., and Neubauer, A. (1996). *Regularization of Inverse Problems*. Kluwer Academic Publishers, Dordrecht.
- Enqvist, O., Josephson, K., and Kahl, F. (2009). Optimal correspondences from pairwise constraints. In *Twelfth International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan.
- Estrada, F. J. and Jepson, A. D. (2009). Benchmarking image segmentation algorithms. *International Journal of Computer Vision*, 85(2):167–181.
- Estrada, F. J., Jepson, A. D., and Chennubhotla, C. (2004). Spectral embedding and min-cut for image segmentation. In *British Machine Vision Conference (BMVC 2004)*, pp. 317–326, London.
- Evangelidis, G. D. and Psarakis, E. Z. (2008). Parametric image alignment using enhanced correlation coefficient maximization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(10):1858–1865.
- Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2008). The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results. <http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html>.
- Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):147–168.
- Ezzat, T., Geiger, G., and Poggio, T. (2002). Trainable videorealistic speech animation. *ACM Transactions on Graphics (Proc. SIGGRAPH 2002)*, 21(3):388–398.
- Fabbri, R., Costa, L. D. F., Torelli, J. C., and Bruno, O. M. (2008). 2D Euclidean distance transform algorithms: A comparative survey. *ACM Computing Surveys*, 40(1).
- Fairchild, M. D. (2005). *Color Appearance Models*. Wiley, 2nd edition.
- Fan, R.-E., Chen, P.-H., and Lin, C.-J. (2005). Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, 6:1889–1918.
- Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.
- Farbman, Z., Fattal, R., Lischinski, D., and Szeliski, R. (2008). Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Transactions on Graphics (Proc. SIGGRAPH 2008)*, 27(3).
- Farenzena, M., Fusello, A., and Gherardi, R. (2009). Structure-and-motion pipeline on a hierarchical cluster tree. In *IEEE International Workshop on 3D Digital Imaging and Modeling (3DIM 2009)*, Kyoto, Japan.
- Farin, G. (1992). From conics to NURBS: A tutorial and survey. *IEEE Computer Graphics and Applications*, 12(5):78–86.

- Farin, G. E. (1996). *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*. Academic Press, Boston, Massachusetts, 4th edition.
- Fattal, R. (2007). Image upsampling via imposed edge statistics. *ACM Transactions on Graphics*, 26(3).
- Fattal, R. (2009). Edge-avoiding wavelets and their applications. *ACM Transactions on Graphics*, 28(3).
- Fattal, R., Lischinski, D., and Werman, M. (2002). Gradient domain high dynamic range compression. *ACM Transactions on Graphics (Proc. SIGGRAPH 2002)*, 21(3):249–256.
- Faugeras, O. (1993). *Three-dimensional computer vision: A geometric viewpoint*. MIT Press, Cambridge, Massachusetts.
- Faugeras, O. and Keriven, R. (1998). Variational principles, surface evolution, PDEs, level set methods, and the stereo problem. *IEEE Transactions on Image Processing*, 7(3):336–344.
- Faugeras, O. and Luong, Q.-T. (2001). *The Geometry of Multiple Images*. MIT Press, Cambridge, MA.
- Faugeras, O. D. (1992). What can be seen in three dimensions with an uncalibrated stereo rig? In *Second European Conference on Computer Vision (ECCV'92)*, pp. 563–578, Santa Margherita Liguere, Italy.
- Faugeras, O. D. and Hebert, M. (1987). The representation, recognition and positioning of 3-D shapes from range data. In Kanade, T. (ed.), *Three-Dimensional Machine Vision*, pp. 301–353, Kluwer Academic Publishers, Boston.
- Faugeras, O. D., Luong, Q.-T., and Maybank, S. J. (1992). Camera self-calibration: Theory and experiments. In *Second European Conference on Computer Vision (ECCV'92)*, pp. 321–334, Santa Margherita Liguere, Italy.
- Favaro, P. and Soatto, S. (2006). *3-D Shape Estimation and Image Restoration: Exploiting Defocus and Motion-Blur*. Springer.
- Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874.
- Fei-Fei, L. and Perona, P. (2005). A Bayesian hierarchical model for learning natural scene categories. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 524–531, San Diego, CA.
- Fei-Fei, L., Fergus, R., and Perona, P. (2006). One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611.
- Fei-Fei, L., Fergus, R., and Torralba, A. (2009). ICCV 2009 short course on recognizing and learning object categories. In *Twelfth International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan. <http://people.csail.mit.edu/torralba/shortCourseRLOC/>.
- Feilner, M., Van De Ville, D., and Unser, M. (2005). An orthogonal family of quincunx wavelets with continuously adjustable order. *IEEE Transactions on Image Processing*, 14(4):499–520.
- Feldmar, J. and Ayache, N. (1996). Rigid, affine, and locally affine registration of free-form surfaces. *International Journal of Computer Vision*, 18(2):99–119.
- Fellbaum, C. (ed.). (1998). *WordNet: An Electronic Lexical Database*, Bradford Books.
- Felzenszwalb, P., McAllester, D., and Ramanan, D. (2008). A discriminatively trained, multiscale, deformable part model. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Felzenszwalb, P. F. and Huttenlocher, D. P. (2004a). *Distance Transforms of Sampled Functions*. Technical Report TR2004-1963, Cornell University Computing and Information Science.

- Felzenszwalb, P. F. and Huttenlocher, D. P. (2004b). Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181.
- Felzenszwalb, P. F. and Huttenlocher, D. P. (2005). Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79.
- Felzenszwalb, P. F. and Huttenlocher, D. P. (2006). Efficient belief propagation for early vision. *International Journal of Computer Vision*, 70(1):41–54.
- Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645.
- Ferencz, A., Learned-Miller, E. G., and Malik, J. (2008). Learning to locate informative features for visual identification. *International Journal of Computer Vision*, 77(1-3):3–24.
- Fergus, R. (2007a). Combined segmentation and recognition. In *CVPR 2007 Short Course on Recognizing and Learning Object Categories*. <http://people.csail.mit.edu/torralba/shortCourseRLOC/>.
- Fergus, R. (2007b). Part-based models. In *CVPR 2007 Short Course on Recognizing and Learning Object Categories*. <http://people.csail.mit.edu/torralba/shortCourseRLOC/>.
- Fergus, R. (2009). Classical methods for object recognition. In *ICCV 2009 Short Course on Recognizing and Learning Object Categories*, Kyoto, Japan. <http://people.csail.mit.edu/torralba/shortCourseRLOC/>.
- Fergus, R., Perona, P., and Zisserman, A. (2004). A visual category filter for Google images. In *Eighth European Conference on Computer Vision (ECCV 2004)*, pp. 242–256, Prague.
- Fergus, R., Perona, P., and Zisserman, A. (2005). A sparse object category model for efficient learning and exhaustive recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 380–387, San Diego, CA.
- Fergus, R., Perona, P., and Zisserman, A. (2007). Weakly supervised scale-invariant learning of models for visual recognition. *International Journal of Computer Vision*, 71(3):273–303.
- Fergus, R., Fei-Fei, L., Perona, P., and Zisserman, A. (2005). Learning object categories from Google’s image search. In *Tenth International Conference on Computer Vision (ICCV 2005)*, pp. 1816–1823, Beijing, China.
- Fergus, R., Singh, B., Hertzmann, A., Roweis, S. T., and Freeman, W. T. (2006). Removing camera shake from a single photograph. *ACM Transactions on Graphics*, 25(3):787–794.
- Ferrari, V., Marin-Jimenez, M., and Zisserman, A. (2009). Pose search: retrieving people using their pose. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, Miami Beach, FL.
- Ferrari, V., Marin-Jimenez, M. J., and Zisserman, A. (2008). Progressive search space reduction for human pose estimation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Ferrari, V., Tuytelaars, T., and Van Gool, L. (2006a). Object detection by contour segment networks. In *Ninth European Conference on Computer Vision (ECCV 2006)*, pp. 14–28.
- Ferrari, V., Tuytelaars, T., and Van Gool, L. (2006b). Simultaneous object recognition and segmentation from single or multiple model views. *International Journal of Computer Vision*, 67(2):159–188.
- Field, D. J. (1987). Relations between the statistics of natural images and the response properties of cortical cells. *Journal of the Optical Society of America A*, 4(12):2379–2394.

- Finkelstein, A. and Salesin, D. H. (1994). Multiresolution curves. In *ACM SIGGRAPH 1994 Conference Proceedings*, pp. 261–268.
- Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.
- Fischler, M. A. and Elschlager, R. A. (1973). The representation and matching of pictorial structures. *IEEE Transactions on Computers*, 22(1):67–92.
- Fischler, M. A. and Firschein, O. (1987). *Readings in Computer Vision*. Morgan Kaufmann Publishers, Inc., Los Altos.
- Fischler, M. A., Firschein, O., Barnard, S. T., Fua, P. V., and Leclerc, Y. (1989). *The Vision Problem: Exploiting Parallel Computation*. Technical Note 458, SRI International, Menlo Park.
- Fitzgibbon, A. W. and Zisserman, A. (1998). Automatic camera recovery for closed and open image sequences. In *Fifth European Conference on Computer Vision (ECCV'98)*, pp. 311–326, Freiburg, Germany.
- Fitzgibbon, A. W., Cross, G., and Zisserman, A. (1998). Automatic 3D model construction for turn-table sequences. In *European Workshop on 3D Structure from Multiple Images of Large-Scale Environments (SMILE)*, pp. 155–170, Freiburg.
- Fleet, D. and Jepson, A. (1990). Computation of component image velocity from local phase information. *International Journal of Computer Vision*, 5(1):77–104.
- Fleuret, F. and Geman, D. (2001). Coarse-to-fine face detection. *International Journal of Computer Vision*, 41(1/2):85–107.
- Flickner, M., Sawhney, H., Niblack, W., Ashley, J., Huang, Q. et al. (1995). Query by image and video content: The QBIC system. *Computer*, 28(9):23–32.
- Foley, J. D., van Dam, A., Feiner, S. K., and Hughes, J. F. (1995). *Computer Graphics: Principles and Practice*. Addison-Wesley, Reading, MA, 2 edition.
- Förstner, W. (1986). A feature-based correspondence algorithm for image matching. *Intl. Arch. Photogrammetry & Remote Sensing*, 26(3):150–166.
- Förstner, W. (2005). Uncertainty and projective geometry. In Bayro-Corrochano, E. (ed.), *Handbook of Geometric Computing*, pp. 493–534, Springer, New York.
- Forsyth, D. and Ponce, J. (2003). *Computer Vision: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ.
- Forsyth, D. A. and Fleck, M. M. (1999). Automatic detection of human nudes. *International Journal of Computer Vision*, 32(1):63–77.
- Forsyth, D. A., Arikan, O., Ikemoto, L., O'Brien, J., and Ramanan, D. (2006). Computational studies of human motion: Part 1, tracking and motion synthesis. *Foundations and Trends in Computer Graphics and Computer Vision*, 1(2/3):77–254.
- Fossati, A., Dimitrijevic, M., Lepetit, V., and Fua, P. (2007). Bridging the gap between detection and tracking for 3D monocular video-based motion capture. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Frahm, J.-M. and Koch, R. (2003). Camera calibration with known rotation. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 1418–1425, Nice, France.
- Freeman, M. (2008). *Mastering HDR Photography*. Amphoto Books, New York.

- Freeman, W., Perona, P., and Schölkopf, B. (2008). Guest editorial: Special issue on machine learning for vision. *International Journal of Computer Vision*, 77(1-3):1.
- Freeman, W. T. (1992). *Steerable Filters and Local Analysis of Image Structure*. Ph.D. thesis, Massachusetts Institute of Technology.
- Freeman, W. T. and Adelson, E. H. (1991). The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906.
- Freeman, W. T., Jones, T. R., and Pasztor, E. C. (2002). Example-based super-resolution. *IEEE Computer Graphics and Applications*, 22(2):56–65.
- Freeman, W. T., Pasztor, E. C., and Carmichael, O. T. (2000). Learning low-level vision. *International Journal of Computer Vision*, 40(1):25–47.
- Frey, B. J. and MacKay, D. J. C. (1997). A revolution: Belief propagation in graphs with cycles. In *Advances in Neural Information Processing Systems*.
- Friedman, J., Hastie, T., and Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 38(2):337–374.
- Friskin, S. F., Perry, R. N., Rockwood, A. P., and Jones, T. R. (2000). Adaptively sampled distance fields: A general representation of shape for computer graphics. In *ACM SIGGRAPH 2000 Conference Proceedings*, pp. 249–254.
- Fritz, M. and Schiele, B. (2008). Decomposition, discovery and detection of visual categories using topic models. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Frome, A., Singer, Y., Sha, F., and Malik, J. (2007). Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *Eleventh International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil.
- Fua, P. (1993). A parallel stereo algorithm that produces dense depth maps and preserves image features. *Machine Vision and Applications*, 6(1):35–49.
- Fua, P. and Leclerc, Y. G. (1995). Object-centered surface reconstruction: Combining multi-image stereo and shading. *International Journal of Computer Vision*, 16(1):35–56.
- Fua, P. and Sander, P. (1992). Segmenting unstructured 3D points into surfaces. In *Second European Conference on Computer Vision (ECCV'92)*, pp. 676–680, Santa Margherita Liguere, Italy.
- Fuh, C.-S. and Maragos, P. (1991). Motion displacement estimation using an affine model for image matching. *Optical Engineering*, 30(7):881–887.
- Fukunaga, K. and Hostetler, L. D. (1975). The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21:32–40.
- Furukawa, Y. and Ponce, J. (2007). Accurate, dense, and robust multi-view stereopsis. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Furukawa, Y. and Ponce, J. (2008). Accurate calibration from multi-view stereo and bundle adjustment. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Furukawa, Y. and Ponce, J. (2009). Carved visual hulls for image-based modeling. *International Journal of Computer Vision*, 81(1):53–67.

- Furukawa, Y. and Ponce, J. (2011). Accurate, dense, and robust multi-view stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- Furukawa, Y., Curless, B., Seitz, S. M., and Szeliski, R. (2009a). Manhattan-world stereo. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, Miami, FL.
- Furukawa, Y., Curless, B., Seitz, S. M., and Szeliski, R. (2009b). Reconstructing building interiors from images. In *Twelfth IEEE International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan.
- Furukawa, Y., Curless, B., Seitz, S. M., and Szeliski, R. (2010). Towards internet-scale multi-view stereo. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2010)*, San Francisco, CA.
- Fusiello, A., Roberto, V., and Trucco, E. (1997). Efficient stereo with multiple windowing. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pp. 858–863, San Juan, Puerto Rico.
- Fusiello, A., Trucco, E., and Verri, A. (2000). A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, 12(1):16–22.
- Gai, J. and Kang, S. B. (2009). Matte-based restoration of vintage video. *IEEE Transactions on Image Processing*, 18:2185–2197.
- Gal, R., Wexler, Y., Ofek, E., Hoppe, H., and Cohen-Or, D. (2010). Seamless montage for texturing models. In *Proceedings of Eurographics 2010*.
- Gallagher, A. C. and Chen, T. (2008). Multi-image graph cut clothing segmentation for recognizing people. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Gallup, D., Frahm, J.-M., Mordohai, P., and Pollefeys, M. (2008). Variable baseline/resolution stereo. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Gamble, E. and Poggio, T. (1987). *Visual integration and detection of discontinuities: the key role of intensity edges*. A. I. Memo 970, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.
- Gammeter, S., Bossard, L., Quack, T., and Van Gool, L. (2009). I know what you did last summer: Object-level auto-annotation of holiday snaps. In *Twelfth International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan.
- Gao, W., Chen, Y., Wang, R., Shan, S., and Jiang, D. (2003). Learning and synthesizing MPEG-4 compatible 3-D face animation from video sequence. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(11):1119–1128.
- Garding, J. (1992). Shape from texture for smooth curved surfaces in perspective projection. *Journal of Mathematical Imaging and Vision*, 2:329–352.
- Gargallo, P., Prados, E., and Sturm, P. (2007). Minimizing the reprojection error in surface reconstruction from images. In *Eleventh International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil.
- Gavrila, D. M. (1999). The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1):82–98.
- Gavrila, D. M. and Davis, L. S. (1996). 3D model-based tracking of humans in action: A multi-view approach. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'96)*, pp. 73–80, San Francisco.

- Gavrila, D. M. and Philomin, V. (1999). Real-time object detection for smart vehicles. In *Seventh International Conference on Computer Vision (ICCV'99)*, pp. 87–93, Kerkyra, Greece.
- Geiger, D. and Girosi, F. (1991). Parallel and deterministic algorithms for MRFs: Surface reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):401–412.
- Geiger, D., Ladendorf, B., and Yuille, A. (1992). Occlusions and binocular stereo. In *Second European Conference on Computer Vision (ECCV'92)*, pp. 425–433, Santa Margherita Liguere, Italy.
- Gelb, A. (ed.). (1974). *Applied Optimal Estimation*. MIT Press, Cambridge, Massachusetts.
- Geller, T. (2008). Overcoming the uncanny valley. *IEEE Computer Graphics and Applications*, 28(4):11–17.
- Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741.
- Gennert, M. A. (1988). Brightness-based stereo matching. In *Second International Conference on Computer Vision (ICCV'88)*, pp. 139–143, Tampa.
- Gersho, A. and Gray, R. M. (1991). *Vector Quantization and Signal Compression*. Springer.
- Gershun, A. (1939). The light field. *Journal of Mathematics and Physics*, XVIII:51–151.
- Gevers, T., van de Weijer, J., and Stokman, H. (2006). Color feature detection. In Lukac, R. and Plataniotis, K. N. (eds), *Color Image Processing: Methods and Applications*, CRC Press.
- Giblin, P. and Weiss, R. (1987). Reconstruction of surfaces from profiles. In *First International Conference on Computer Vision (ICCV'87)*, pp. 136–144, London, England.
- Gionis, A., Indyk, P., and Motwani, R. (1999). Similarity search in high dimensions via hashing. In *25th International Conference on Very Large Data Bases (VLDB'99)*, pp. 518–529.
- Girod, B., Greiner, G., and Niemann, H. (eds). (2000). *Principles of 3D Image Analysis and Synthesis*, Kluwer, Boston.
- Glassner, A. S. (1995). *Principles of Digital Image Synthesis*. Morgan Kaufmann Publishers, San Francisco.
- Gleicher, M. (1995). Image snapping. In *ACM SIGGRAPH 1995 Conference Proceedings*, pp. 183–190.
- Gleicher, M. (1997). Projective registration with difference decomposition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pp. 331–337, San Juan, Puerto Rico.
- Gleicher, M. and Witkin, A. (1992). Through-the-lens camera control. *Computer Graphics (SIGGRAPH '92)*, 26(2):331–340.
- Glocker, B., Komodakis, N., Tziritas, G., Navab, N., and Paragios, N. (2008). Dense image registration through MRFs and efficient linear programming. *Medical Image Analysis*, 12(6):731–741.
- Glocker, B., Paragios, N., Komodakis, N., Tziritas, G., and Navab, N. (2008). Optical flow estimation with uncertainties through dynamic MRFs. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Gluckman, J. (2006a). Higher order image pyramids. In *Ninth European Conference on Computer Vision (ECCV 2006)*, pp. 308–320.
- Gluckman, J. (2006b). Scale variant image pyramids. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2006)*, pp. 1069–1075, New York City, NY.
- Goesele, M., Curless, B., and Seitz, S. (2006). Multi-view stereo revisited. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2006)*, pp. 2402–2409, New York City, NY.

- Goesele, M., Fuchs, C., and Seidel, H.-P. (2003). Accuracy of 3D range scanners by measurement of the slanted edge modulation transfer function. In *Fourth International Conference on 3-D Digital Imaging and Modeling*, Banff.
- Goesele, M., Snavely, N., Curless, B., Hoppe, H., and Seitz, S. M. (2007). Multi-view stereo for community photo collections. In *Eleventh International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil.
- Gold, S., Rangarajan, A., Lu, C., Pappu, S., and Mjolsness, E. (1998). New algorithms for 2D and 3D point matching: Pose estimation and correspondence. *Pattern Recognition*, 31(8):1019–1031.
- Goldberg, A. V. and Tarjan, R. E. (1988). A new approach to the maximum-flow problem. *Journal of the ACM*, 35(4):921–940.
- Goldluecke, B. and Cremers, D. (2009). Superresolution texture maps for multiview reconstruction. In *Twelfth International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan.
- Goldman, D. B. (2011). Vignette and exposure calibration and compensation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- Golovinskiy, A., Matusik, W., ster, H. P., Rusinkiewicz, S., and Funkhouser, T. (2006). A statistical model for synthesis of detailed facial geometry. *ACM Transactions on Graphics*, 25(3):1025–1034.
- Golub, G. and Van Loan, C. F. (1996). *Matrix Computation, third edition*. The John Hopkins University Press, Baltimore and London.
- Gomes, J. and Velho, L. (1997). *Image Processing for Computer Graphics*. Springer-Verlag, New York.
- Gomes, J., Darsa, L., Costa, B., and Velho, L. (1999). *Warping and Morphing of Graphical Objects*. Morgan Kaufmann Publishers, San Francisco.
- Gong, M., Yang, R., Wang, L., and Gong, M. (2007). A performance study on different cost aggregation approaches used in realtime stereo matching. *International Journal of Computer Vision*, 75(2):283–296.
- Gonzales, R. C. and Woods, R. E. (2008). *Digital Image Processing*. Prentice-Hall, Upper Saddle River, NJ, 3rd edition.
- Gooch, B. and Gooch, A. (2001). *Non-Photorealistic Rendering*. A K Peters, Ltd, Natick, Massachusetts.
- Gordon, I. and Lowe, D. G. (2006). What and where: 3D object recognition with accurate pose. In Ponce, J., Hebert, M., Schmid, C., and Zisserman, A. (eds), *Toward Category-Level Object Recognition*, pp. 67–82, Springer, New York.
- Gorelick, L., Blank, M., Shechtman, E., Irani, M., and Basri, R. (2007). Actions as space-time shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2247–2253.
- Gortler, S. J. and Cohen, M. F. (1995). Hierarchical and variational geometric modeling with wavelets. In *Symposium on Interactive 3D Graphics*, pp. 35–43, Monterey, CA.
- Gortler, S. J., Grzeszczuk, R., Szeliski, R., and Cohen, M. F. (1996). The Lumigraph. In *ACM SIGGRAPH 1996 Conference Proceedings*, pp. 43–54, New Orleans.
- Goshtasby, A. (1989). Correction of image deformation from lens distortion using Bézier patches. *Computer Vision, Graphics, and Image Processing*, 47(4):385–394.
- Goshtasby, A. (2005). *2-D and 3-D Image Registration*. Wiley, New York.
- Gotchev, A. and Rosenhahn, B. (eds). (2009). *Proceedings of the 3DTV Conference: The True Vision—Capture, Transmission and Display of 3D Video*, IEEE Computer Society Press.

- Govindu, V. M. (2006). Revisiting the brightness constraint: Probabilistic formulation and algorithms. In *Ninth European Conference on Computer Vision (ECCV 2006)*, pp. 177–188.
- Grady, L. (2006). Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1768–1783.
- Grady, L. (2008). A lattice-preserving multigrid method for solving the inhomogeneous Poisson equations used in image analysis. In *Tenth European Conference on Computer Vision (ECCV 2008)*, pp. 252–264, Marseilles.
- Grady, L. and Ali, S. (2008). Fast approximate random walker segmentation using eigenvector precomputation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Grady, L. and Alvino, C. (2008). Reformulating and optimizing the Mumford–Shah functional on a graph — a faster, lower energy solution. In *Tenth European Conference on Computer Vision (ECCV 2008)*, pp. 248–261, Marseilles.
- Grauman, K. and Darrell, T. (2005). Efficient image matching with distributions of local invariant features. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 627–634, San Diego, CA.
- Grauman, K. and Darrell, T. (2007a). Pyramid match hashing: Sub-linear time indexing over partial correspondences. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Grauman, K. and Darrell, T. (2007b). The pyramid match kernel: Efficient learning with sets of features. *Journal of Machine Learning Research*, 8:725–760.
- Grauman, K., Shakhnarovich, G., and Darrell, T. (2003). Inferring 3D structure with a statistical image-based shape model. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 641–648, Nice, France.
- Greene, N. (1986). Environment mapping and other applications of world projections. *IEEE Computer Graphics and Applications*, 6(11):21–29.
- Greene, N. and Heckbert, P. (1986). Creating raster Omnimax images from multiple perspective views using the elliptical weighted average filter. *IEEE Computer Graphics and Applications*, 6(6):21–27.
- Greig, D., Porteous, B., and Seheult, A. (1989). Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society, Series B*, 51(2):271–279.
- Greban, K. D., Thorpe, C. E., and Kanade, T. (1988). Geometric camera calibration using systems of linear equations. In *IEEE International Conference on Robotics and Automation*, pp. 562–567, Philadelphia.
- Griffin, G., Holub, A., and Perona, P. (2007). *Caltech-256 Object Category Dataset*. Technical Report 7694, California Institute of Technology.
- Grimson, W. E. L. (1983). An implementation of a computational theory of visual surface interpolation. *Computer Vision, Graphics, and Image Processing*, 22:39–69.
- Grimson, W. E. L. (1985). Computational experiments with a feature based stereo algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7(1):17–34.
- Gross, R., Matthews, I., and Baker, S. (2006). Active appearance models with occlusion. *Image and Vision Computing*, 24(6):593–604.

- Gross, R., Shi, J., and Cohn, J. F. (2005). Quo vadis face recognition? In *IEEE Workshop on Empirical Evaluation Methods in Computer Vision*, San Diego.
- Gross, R., Baker, S., Matthews, I., and Kanade, T. (2005). Face recognition across pose and illumination. In Li, S. Z. and Jain, A. K. (eds), *Handbook of Face Recognition*, Springer.
- Gross, R., Sweeney, L., De la Torre, F., and Baker, S. (2008). Semi-supervised learning of multi-factor models for face de-identification. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Gross, R., Matthews, I., Cohn, J., Kanade, T., and Baker, S. (2010). Multi-PIE. *Image and Vision Computing*, 28(5):807–813.
- Grossberg, M. D. and Nayar, S. K. (2001). A general imaging model and a method for finding its parameters. In *Eighth International Conference on Computer Vision (ICCV 2001)*, pp. 108–115, Vancouver, Canada.
- Grossberg, M. D. and Nayar, S. K. (2004). Modeling the space of camera response functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1272–1282.
- Gu, C., Lim, J., Arbelaez, P., and Malik, J. (2009). Recognition using regions. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, Miami Beach, FL.
- Gu, X., Gortler, S. J., and Hoppe, H. (2002). Geometry images. *ACM Transactions on Graphics*, 21(3):355–361.
- Guan, P., Weiss, A., Bălan, A. O., and Black, M. J. (2009). Estimating human shape and pose from a single image. In *Twelfth International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan.
- Guennebaud, G. and Gross, M. (2007). Algebraic point set surfaces. *ACM Transactions on Graphics*, 26(3).
- Guennebaud, G., Germann, M., and Gross, M. (2008). Dynamic sampling and rendering of algebraic point set surfaces. *Computer Graphics Forum*, 27(2):653–662.
- Guenter, B., Grimm, C., Wood, D., Malvar, H., and Pighin, F. (1998). Making faces. In *ACM SIGGRAPH 1998 Conference Proceedings*, pp. 55–66.
- Guillaumin, M., Verbeek, J., and Schmid, C. (2009). Is that you? Metric learning approaches for face identification. In *Twelfth International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan.
- Gulbins, J. and Gulbins, R. (2009). *Photographic Multishot Techniques: High Dynamic Range, Super-Resolution, Extended Depth of Field, Stitching*. Rocky Nook.
- Habbecke, M. and Kobbelt, L. (2007). A surface-growing approach to multi-view stereo reconstruction. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Hager, G. D. and Belhumeur, P. N. (1998). Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039.
- Hall, R. (1989). *Illumination and Color in Computer Generated Imagery*. Springer-Verlag, New York.
- Haller, M., Billinghurst, M., and Thomas, B. (2007). *Emerging Technologies of Augmented Reality: Interfaces and Design*. IGI Publishing.
- Hampel, F. R., Ronchetti, E. M., Rousseeuw, P. J., and Stahel, W. A. (1986). *Robust Statistics : The Approach Based on Influence Functions*. Wiley, New York.
- Han, F. and Zhu, S.-C. (2005). Bottom-up/top-down image parsing by attribute graph grammar. In *Tenth International Conference on Computer Vision (ICCV 2005)*, pp. 1778–1785, Beijing, China.

- Hanna, K. J. (1991). Direct multi-resolution estimation of ego-motion and structure from motion. In *IEEE Workshop on Visual Motion*, pp. 156–162, Princeton, New Jersey.
- Hannah, M. J. (1974). *Computer Matching of Areas in Stereo Images*. Ph.D. thesis, Stanford University.
- Hannah, M. J. (1988). Test results from SRI's stereo system. In *Image Understanding Workshop*, pp. 740–744, Cambridge, Massachusetts.
- Hansen, M., Anandan, P., Dana, K., van der Wal, G., and Burt, P. (1994). Real-time scene stabilization and mosaic construction. In *IEEE Workshop on Applications of Computer Vision (WACV'94)*, pp. 54–62, Sarasota.
- Hanson, A. R. and Riseman, E. M. (eds). (1978). *Computer Vision Systems*, Academic Press, New York.
- Haralick, R. M. and Shapiro, L. G. (1985). Image segmentation techniques. *Computer Vision, Graphics, and Image Processing*, 29(1):100–132.
- Haralick, R. M. and Shapiro, L. G. (1992). *Computer and Robot Vision*. Addison-Wesley, Reading, MA.
- Haralick, R. M., Lee, C.-N., Ottenberg, K., and Nölle, M. (1994). Review and analysis of solutions of the three point perspective pose estimation problem. *International Journal of Computer Vision*, 13(3):331–356.
- Hardie, R. C., Barnard, K. J., and Armstrong, E. E. (1997). Joint MAP registration and high-resolution image estimation using a sequence of undersampled images. *IEEE Transactions on Image Processing*, 6(12):1621–1633.
- Haritaoglu, I., Harwood, D., and Davis, L. S. (2000). W⁴: Real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):809–830.
- Harker, M. and O'Leary, P. (2008). Least squares surface reconstruction from measured gradient fields. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Harris, C. and Stephens, M. J. (1988). A combined corner and edge detector. In *Alvey Vision Conference*, pp. 147–152.
- Hartley, R. and Kang, S. B. (2007). Parameter-free radial distortion correction with center of distortion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(8):1309–1321.
- Hartley, R., Gupta, R., and Chang, T. (1992). Estimation of relative camera positions for uncalibrated cameras. In *Second European Conference on Computer Vision (ECCV'92)*, pp. 579–587, Santa Margherita Liguere, Italy.
- Hartley, R. I. (1994a). Projective reconstruction and invariants from multiple images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(10):1036–1041.
- Hartley, R. I. (1994b). Self-calibration from multiple views of a rotating camera. In *Third European Conference on Computer Vision (ECCV'94)*, pp. 471–478, Stockholm, Sweden.
- Hartley, R. I. (1997a). In defense of the 8-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593.
- Hartley, R. I. (1997b). Self-calibration of stationary cameras. *International Journal of Computer Vision*, 22(1):5–23.
- Hartley, R. I. (1998). Chirality. *International Journal of Computer Vision*, 26(1):41–61.
- Hartley, R. I. and Kang, S. B. (2005). Parameter-free radial distortion correction with centre of distortion estimation. In *Tenth International Conference on Computer Vision (ICCV 2005)*, pp. 1834–1841, Beijing, China.

- Hartley, R. I. and Sturm, P. (1997). Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157.
- Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry*. Cambridge University Press, Cambridge, UK.
- Hartley, R. I., Hayman, E., de Agapito, L., and Reid, I. (2000). Camera calibration and the search for infinity. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2000)*, pp. 510–517, Hilton Head Island.
- Hasinoff, S. W. and Kutulakos, K. N. (2008). Light-efficient photography. In *Tenth European Conference on Computer Vision (ECCV 2008)*, pp. 45–59, Marseilles.
- Hasinoff, S. W., Durand, F., and Freeman, W. T. (2010). Noise-optimal capture for high dynamic range photography. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2010)*, San Francisco, CA.
- Hasinoff, S. W., Kang, S. B., and Szeliski, R. (2006). Boundary matting for view synthesis. *Computer Vision and Image Understanding*, 103(1):22–32.
- Hasinoff, S. W., Kutulakos, K. N., Durand, F., and Freeman, W. T. (2009). Time-constrained photography. In *Twelfth International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan.
- Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, New York.
- Hayes, B. (2008). Computational photography. *American Scientist*, 96:94–99.
- Hays, J. and Efros, A. A. (2007). Scene completion using millions of photographs. *ACM Transactions on Graphics*, 26(3).
- Hays, J., Leordeanu, M., Efros, A. A., and Liu, Y. (2006). Discovering texture regularity as a higher-order correspondence problem. In *Ninth European Conference on Computer Vision (ECCV 2006)*, pp. 522–535.
- He, L.-W. and Zhang, Z. (2005). Real-time whiteboard capture and processing using a video camera for teleconferencing. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2005)*, pp. 1113–1116, Philadelphia.
- He, X. and Zemel, R. S. (2008). Learning hybrid models for image annotation with partially labeled data. In *Advances in Neural Information Processing Systems*.
- He, X., Zemel, R. S., and Carreira-Perpiñán, M. A. (2004). Multiscale conditional random fields for image labeling. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2004)*, pp. 695–702, Washington, DC.
- He, X., Zemel, R. S., and Ray, D. (2006). Learning and incorporating top-down cues in image segmentation. In *Ninth European Conference on Computer Vision (ECCV 2006)*, pp. 338–351.
- Healey, G. E. and Kondepudy, R. (1994). Radiometric CCD camera calibration and noise estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(3):267–276.
- Healey, G. E. and Shafer, S. A. (1992). *Color. Physics-Based Vision: Principles and Practice*, Jones & Bartlett, Cambridge, MA.
- Heath, M. D., Sarkar, S., Sanocki, T., and Bowyer, K. W. (1998). Comparison of edge detectors. *Computer Vision and Image Understanding*, 69(1):38–54.
- Hebert, M. (2000). Active and passive range sensing for robotics. In *IEEE International Conference on Robotics and Automation*, pp. 102–110, San Francisco.

- Hecht, E. (2001). *Optics*. Pearson Addison Wesley, Reading, MA, 4th edition.
- Heckbert, P. (1986). Survey of texture mapping. *IEEE Computer Graphics and Applications*, 6(11):56–67.
- Heckbert, P. (1989). *Fundamentals of Texture Mapping and Image Warping*. Master's thesis, The University of California at Berkeley.
- Heeger, D. J. (1988). Optical flow using spatiotemporal filters. *International Journal of Computer Vision*, 1(1):279–302.
- Heeger, D. J. and Bergen, J. R. (1995). Pyramid-based texture analysis/synthesis. In *ACM SIGGRAPH 1995 Conference Proceedings*, pp. 229–238.
- Heisele, B., Serre, T., and Poggio, T. (2007). A component-based framework for face detection and identification. *International Journal of Computer Vision*, 74(2):167–181.
- Heisele, B., Ho, P., Wu, J., and Poggio, T. (2003). Face recognition: component-based versus global approaches. *Computer Vision and Image Understanding*, 91(1-2):6–21.
- Herley, C. (2005). Automatic occlusion removal from minimum number of images. In *International Conference on Image Processing (ICIP 2005)*, pp. 1046–1049–16, Genova.
- Hernandez, C. and Schmitt, F. (2004). Silhouette and stereo fusion for 3D object modeling. *Computer Vision and Image Understanding*, 96(3):367–392.
- Hernandez, C. and Vogiatzis, G. (2010). Self-calibrating a real-time monocular 3d facial capture system. In *Fifth International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT'10)*, Paris.
- Hernandez, C., Vogiatzis, G., and Cipolla, R. (2007). Probabilistic visibility for multi-view stereo. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Hernandez, C., Vogiatzis, G., Brostow, G. J., Stenger, B., and Cipolla, R. (2007). Non-rigid photometric stereo with colored lights. In *Eleventh International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil.
- Hershberger, J. and Snoeyink, J. (1992). *Speeding Up the Douglas-Peucker Line-Simplification Algorithm*. Technical Report TR-92-07, Computer Science Department, The University of British Columbia.
- Hertzmann, A., Jacobs, C. E., Oliver, N., Curless, B., and Salesin, D. H. (2001). Image analogies. In *ACM SIGGRAPH 2001 Conference Proceedings*, pp. 327–340.
- Hiep, V. H., Keriven, R., Pons, J.-P., and Labatut, P. (2009). Towards high-resolution large-scale multi-view stereo. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, Miami Beach, FL.
- Hillman, P., Hannah, J., and Renshaw, D. (2001). Alpha channel estimation in high resolution images and image sequences. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2001)*, pp. 1063–1068, Kauai, Hawaii.
- Hilton, A., Fua, P., and Ronfard, R. (2006). Modeling people: Vision-based understanding of a person's shape, appearance, movement, and behaviour. *Computer Vision and Image Understanding*, 104(2-3):87–89.
- Hilton, A., Stoddart, A. J., Illingworth, J., and Windeatt, T. (1996). Reliable surface reconstruction from multiple range images. In *Fourth European Conference on Computer Vision (ECCV'96)*, pp. 117–126, Cambridge, England.

- Hinckley, K., Sinclair, M., Hanson, E., Szeliski, R., and Conway, M. (1999). The VideoMouse: a camera-based multi-degree-of-freedom input device. In *12th annual ACM symposium on User interface software and technology*, pp. 103–112.
- Hinterstoisser, S., Benhimane, S., Navab, N., Fua, P., and Lepetit, V. (2008). Online learning of patch perspective rectification for efficient object detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Hinton, G. E. (1977). *Relaxation and its Role in Vision*. Ph.D. thesis, University of Edinburgh.
- Hirschmüller, H. (2008). Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341.
- Hirschmüller, H. and Scharstein, D. (2009). Evaluation of stereo matching costs on images with radiometric differences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(9):1582–1599.
- Hjaltason, G. R. and Samet, H. (2003). Index-driven similarity search in metric spaces. *ACM Transactions on Database Systems*, 28(4):517–580.
- Hofmann, T. (1999). Probabilistic latent semantic indexing. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 50–57, Berkeley, CA.
- Hogg, D. (1983). Model-based vision: A program to see a walking person. *Image and Vision Computing*, 1(1):5–20.
- Hoiem, D., Efros, A. A., and Hebert, M. (2005a). Automatic photo pop-up. *ACM Transactions on Graphics (Proc. SIGGRAPH 2005)*, 24(3):577–584.
- Hoiem, D., Efros, A. A., and Hebert, M. (2005b). Geometric context from a single image. In *Tenth International Conference on Computer Vision (ICCV 2005)*, pp. 654–661, Beijing, China.
- Hoiem, D., Efros, A. A., and Hebert, M. (2008a). Closing the loop in scene interpretation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Hoiem, D., Efros, A. A., and Hebert, M. (2008b). Putting objects in perspective. *International Journal of Computer Vision*, 80(1):3–15.
- Hoiem, D., Rother, C., and Winn, J. (2007). 3D LayoutCRF for multi-view object class recognition and segmentation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Hoover, A., Jean-Baptiste, G., Jiang, X., Flynn, P. J., Bunke, H. et al. (1996). An experimental comparison of range image segmentation algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7):673–689.
- Hoppe, H. (1996). Progressive meshes. In *ACM SIGGRAPH 1996 Conference Proceedings*, pp. 99–108, New Orleans.
- Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. (1992). Surface reconstruction from unorganized points. *Computer Graphics (SIGGRAPH '92)*, 26(2):71–78.
- Horn, B. K. P. (1974). Determining lightness from an image. *Computer Graphics and Image Processing*, 3(1):277–299.
- Horn, B. K. P. (1975). Obtaining shape from shading information. In Winston, P. H. (ed.), *The Psychology of Computer Vision*, pp. 115–155, McGraw-Hill, New York.
- Horn, B. K. P. (1977). Understanding image intensities. *Artificial Intelligence*, 8(2):201–231.

- Horn, B. K. P. (1986). *Robot Vision*. MIT Press, Cambridge, Massachusetts.
- Horn, B. K. P. (1987). Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642.
- Horn, B. K. P. (1990). Height and gradient from shading. *International Journal of Computer Vision*, 5(1):37–75.
- Horn, B. K. P. and Brooks, M. J. (1986). The variational approach to shape from shading. *Computer Vision, Graphics, and Image Processing*, 33:174–208.
- Horn, B. K. P. and Brooks, M. J. (eds). (1989). *Shape from Shading*, MIT Press, Cambridge, Massachusetts.
- Horn, B. K. P. and Schunck, B. G. (1981). Determining optical flow. *Artificial Intelligence*, 17:185–203.
- Horn, B. K. P. and Weldon Jr., E. J. (1988). Direct methods for recovering motion. *International Journal of Computer Vision*, 2(1):51–76.
- Hornung, A., Zeng, B., and Kobbelt, L. (2008). Image selection for improved multi-view stereo. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Horowitz, S. L. and Pavlidis, T. (1976). Picture segmentation by a tree traversal algorithm. *Journal of the ACM*, 23(2):368–388.
- Horry, Y., Anjyo, K.-I., and Arai, K. (1997). Tour into the picture: Using a spidery mesh interface to make animation from a single image. In *ACM SIGGRAPH 1997 Conference Proceedings*, pp. 225–232.
- Hough, P. V. C. (1962). Method and means for recognizing complex patterns. *U. S. Patent*, 3,069,654.
- Houhou, N., Thiran, J.-P., and Bresson, X. (2008). Fast texture segmentation using the shape operator and active contour. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Howe, N. R., Leventon, M. E., and Freeman, W. T. (2000). Bayesian reconstruction of 3D human motion from single-camera video. In *Advances in Neural Information Processing Systems*.
- Hsieh, Y. C., McKeown, D., and Perlant, F. P. (1992). Performance evaluation of scene registration and stereo matching for cartographic feature extraction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):214–238.
- Hu, W., Tan, T., Wang, L., and Maybank, S. (2004). A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 34(3):334–352.
- Hua, G., Brown, M., and Winder, S. (2007). Discriminant embedding for local image descriptors. In *Eleventh International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil.
- Huang, G. B., Ramesh, M., Berg, T., and Learned-Miller, E. (2007). *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. Technical Report 07-49, University of Massachusetts, Amherst.
- Huang, T. S. (1981). *Image Sequence Analysis*. Springer-Verlag, Berlin, Heidelberg.
- Huber, P. J. (1981). *Robust Statistics*. John Wiley & Sons, New York.
- Huffman, D. A. (1971). Impossible objects and nonsense sentences. *Machine Intelligence*, 8:295–323.
- Huguet, F. and Devernay, F. (2007). A variational method for scene flow estimation from stereo sequences. In *Eleventh International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil.

- Huttenlocher, D. P., Klanderman, G., and Rucklidge, W. (1993). Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863.
- Huynh, D. Q., Hartley, R., and Heyden, A. (2003). Outlier correcton in image sequences for the affine camera. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 585–590, Nice, France.
- Iddan, G. J. and Yahav, G. (2001). 3D imaging in the studio (and elsewhere...). In *Three-Dimensional Image Capture and Applications IV*, pp. 48–55.
- Igarashi, T., Nishino, K., and Nayar, S. (2007). The appearance of human skin: A survey. *Foundations and Trends in Computer Graphics and Computer Vision*, 3(1):1–95.
- Ikeuchi, K. (1981). Shape from regular patterns. *Artificial Intelligence*, 22(1):49–75.
- Ikeuchi, K. and Horn, B. K. P. (1981). Numerical shape from shading and occluding boundaries. *Artificial Intelligence*, 17:141–184.
- Ikeuchi, K. and Miyazaki, D. (eds). (2007). *Digitally Archiving Cultural Objects*, Springer, Boston, MA.
- Ikeuchi, K. and Sato, Y. (eds). (2001). *Modeling From Reality*, Kluwer Academic Publishers, Boston.
- Illingworth, J. and Kittler, J. (1988). A survey of the Hough transform. *Computer Vision, Graphics, and Image Processing*, 44:87–116.
- Intille, S. S. and Bobick, A. F. (1994). Disparity-space images and large occlusion stereo. In *Third European Conference on Computer Vision (ECCV'94)*, Stockholm, Sweden.
- Irani, M. and Anandan, P. (1998). Video indexing based on mosaic representations. *Proceedings of the IEEE*, 86(5):905–921.
- Irani, M. and Peleg, S. (1991). Improving resolution by image registration. *Graphical Models and Image Processing*, 53(3):231–239.
- Irani, M., Hsu, S., and Anandan, P. (1995). Video compression using mosaic representations. *Signal Processing: Image Communication*, 7:529–552.
- Irani, M., Roussel, B., and Peleg, S. (1994). Computing occluding and transparent motions. *International Journal of Computer Vision*, 12(1):5–16.
- Irani, M., Roussel, B., and Peleg, S. (1997). Recovery of ego-motion using image stabilization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):268–272.
- Isaksen, A., McMillan, L., and Gortler, S. J. (2000). Dynamically reparameterized light fields. In *ACM SIGGRAPH 2000 Conference Proceedings*, pp. 297–306.
- Isard, M. and Blake, A. (1998). CONDENSATION—conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28.
- Ishiguro, H., Yamamoto, M., and Tsuji, S. (1992). Omni-directional stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):257–262.
- Ishikawa, H. (2003). Exact optimization for Markov random fields with convex priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1333–1336.
- Ishikawa, H. and Veksler, O. (2010). Convex and truncated convex priors for multi-label MRFs. In Blake, A., Kohli, P., and Rother, C. (eds), *Advances in Markov Random Fields*, MIT Press.
- Isidoro, J. and Sclaroff, S. (2003). Stochastic refinement of the visual hull to satisfy photometric and silhouette consistency constraints. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 1335–1342, Nice, France.

- Ivanchenko, V., Shen, H., and Coughlan, J. (2009). Elevation-based stereo implemented in real-time on a GPU. In *IEEE Workshop on Applications of Computer Vision (WACV 2009)*, Snowbird, Utah.
- Jacobs, C. E., Finkelstein, A., and Salesin, D. H. (1995). Fast multiresolution image querying. In *ACM SIGGRAPH 1995 Conference Proceedings*, pp. 277–286.
- Jähne, B. (1997). *Digital Image Processing*. Springer-Verlag, Berlin.
- Jain, A. K. and Dubes, R. C. (1988). *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, New Jersey.
- Jain, A. K., Bolle, R. M., and Pankanti, S. (eds). (1999). *Biometrics: Personal Identification in Networked Society*. Kluwer.
- Jain, A. K., Duin, R. P. W., and Mao, J. (2000). Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37.
- Jain, A. K., Topchy, A., Law, M. H. C., and Buhmann, J. M. (2004). Landscape of clustering algorithms. In *International Conference on Pattern Recognition (ICPR 2004)*, pp. 260–263.
- Jenkin, M. R. M., Jepson, A. D., and Tsotsos, J. K. (1991). Techniques for disparity measurement. *CVGIP: Image Understanding*, 53(1):14–30.
- Jensen, H. W., Marschner, S. R., Levoy, M., and Hanrahan, P. (2001). A practical model for subsurface light transport. In *ACM SIGGRAPH 2001 Conference Proceedings*, pp. 511–518.
- Jeong, Y., Nistér, D., Steedly, D., Szeliski, R., and Kweon, I.-S. (2010). Pushing the envelope of modern methods for bundle adjustment. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2010)*, San Francisco, CA.
- Jia, J. and Tang, C.-K. (2003). Image registration with global and local luminance alignment. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 156–163, Nice, France.
- Jia, J., Sun, J., Tang, C.-K., and Shum, H.-Y. (2006). Drag-and-drop pasting. *ACM Transactions on Graphics*, 25(3):631–636.
- Jiang, Z., Wong, T.-T., and Bao, H. (2003). Practical super-resolution from dynamic video sequences. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2003)*, pp. 549–554, Madison, WI.
- Johnson, A. E. and Hebert, M. (1999). Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–448.
- Johnson, A. E. and Kang, S. B. (1997). Registration and integration of textured 3-D data. In *International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, pp. 234–241, Ottawa.
- Jojic, N. and Frey, B. J. (2001). Learning flexible sprites in video layers. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2001)*, pp. 199–206, Kauai, Hawaii.
- Jones, D. G. and Malik, J. (1992). A computational framework for determining stereo correspondence from a set of linear spatial filters. In *Second European Conference on Computer Vision (ECCV'92)*, pp. 397–410, Santa Margherita Liguere, Italy.
- Jones, M. J. and Rehg, J. M. (2001). Statistical color models with application to skin detection. *International Journal of Computer Vision*, 46(1):81–96.
- Joshi, N., Matusik, W., and Avidan, S. (2006). Natural video matting using camera arrays. *ACM Transactions on Graphics*, 25(3):779–786.

- Joshi, N., Szeliski, R., and Kriegman, D. J. (2008). PSF estimation using sharp edge prediction. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Joshi, N., Zitnick, C. L., Szeliski, R., and Kriegman, D. J. (2009). Image deblurring and denoising using color priors. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, Miami, FL.
- Ju, S. X., Black, M. J., and Jepson, A. D. (1996). Skin and bones: Multi-layer, locally affine, optical flow and regularization with transparency. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'96)*, pp. 307–314, San Francisco.
- Ju, S. X., Black, M. J., and Yacoob, Y. (1996). Cardboard people: a parameterized model of articulated image motion. In *2nd International Conference on Automatic Face and Gesture Recognition*, pp. 38–44, Killington, VT.
- Juan, O. and Boykov, Y. (2006). Active graph cuts. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2006)*, pp. 1023–1029, New York City, NY.
- Jurie, F. and Dhome, M. (2002). Hyperplane approximation for template matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):996–1000.
- Jurie, F. and Schmid, C. (2004). Scale-invariant shape features for recognition of object categories. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2004)*, pp. 90–96, Washington, DC.
- Kadir, T., Zisserman, A., and Brady, M. (2004). An affine invariant salient region detector. In *Eighth European Conference on Computer Vision (ECCV 2004)*, pp. 228–241, Prague.
- Kaftory, R., Schechner, Y., and Zeevi, Y. (2007). Variational distance-dependent image restoration. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Kahl, F. and Hartley, R. (2008). Multiple-view geometry under the l_∞ -norm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1603–1617.
- Kakadiaris, I. and Metaxas, D. (2000). Model-based estimation of 3D human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1453–1459.
- Kakumanu, P., Makrogiannis, S., and Bourbakis, N. (2007). A survey of skin-color modeling and detection methods. *Pattern Recognition*, 40(3):1106–1122.
- Kamvar, S. D., Klein, D., and Manning, C. D. (2002). Interpreting and extending classical agglomerative clustering algorithms using a model-based approach. In *International Conference on Machine Learning*, pp. 283–290.
- Kanade, T. (1977). *Computer Recognition of Human Faces*. Birkhauser, Basel.
- Kanade, T. (1980). A theory of the origami world. *Artificial Intelligence*, 13:279–311.
- Kanade, T. (ed.). (1987). *Three-Dimensional Machine Vision*, Kluwer Academic Publishers, Boston.
- Kanade, T. (1994). Development of a video-rate stereo machine. In *Image Understanding Workshop*, pp. 549–557, Monterey.
- Kanade, T. and Okutomi, M. (1994). A stereo matching algorithm with an adaptive window: Theory and experiment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):920–932.
- Kanade, T., Rander, P. W., and Narayanan, P. J. (1997). Virtualized reality: constructing virtual worlds from real scenes. *IEEE MultiMedia Magazine*, 4(1):34–47.

- Kanade, T., Yoshida, A., Oda, K., Kano, H., and Tanaka, M. (1996). A stereo machine for video-rate dense depth mapping and its new applications. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'96)*, pp. 196–202, San Francisco.
- Kanatani, K. and Morris, D. D. (2001). Gauges and gauge transformations for uncertainty description of geometric structure with indeterminacy. *IEEE Transactions on Information Theory*, 47(5):2017–2028.
- Kang, S. B. (1998). *Depth Painting for Image-based Rendering Applications*. Technical Report, Compaq Computer Corporation, Cambridge Research Lab.
- Kang, S. B. (1999). A survey of image-based rendering techniques. In *Videometrics VI*, pp. 2–16, San Jose.
- Kang, S. B. (2001). Radial distortion snakes. *IEICE Trans. Inf. & Syst.*, E84-D(12):1603–1611.
- Kang, S. B. and Jones, M. (2002). Appearance-based structure from motion using linear classes of 3-D models. *International Journal of Computer Vision*, 49(1):5–22.
- Kang, S. B. and Szeliski, R. (1997). 3-D scene data recovery using omnidirectional multibaseline stereo. *International Journal of Computer Vision*, 25(2):167–183.
- Kang, S. B. and Szeliski, R. (2004). Extracting view-dependent depth maps from a collection of images. *International Journal of Computer Vision*, 58(2):139–163.
- Kang, S. B. and Weiss, R. (1997). Characterization of errors in compositing panoramic images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pp. 103–109, San Juan, Puerto Rico.
- Kang, S. B. and Weiss, R. (1999). Characterization of errors in compositing panoramic images. *Computer Vision and Image Understanding*, 73(2):269–280.
- Kang, S. B. and Weiss, R. (2000). Can we calibrate a camera using an image of a flat, textureless Lambertian surface? In *Sixth European Conference on Computer Vision (ECCV 2000)*, pp. 640–653, Dublin, Ireland.
- Kang, S. B., Szeliski, R., and Anandan, P. (2000). The geometry-image representation tradeoff for rendering. In *International Conference on Image Processing (ICIP-2000)*, pp. 13–16, Vancouver.
- Kang, S. B., Szeliski, R., and Chai, J. (2001). Handling occlusions in dense multi-view stereo. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2001)*, pp. 103–110, Kauai, Hawaii.
- Kang, S. B., Szeliski, R., and Shum, H.-Y. (1997). A parallel feature tracker for extended image sequences. *Computer Vision and Image Understanding*, 67(3):296–310.
- Kang, S. B., Szeliski, R., and Uyttendaele, M. (2004). *Seamless Stitching using Multi-Perspective Plane Sweep*. Technical Report MSR-TR-2004-48, Microsoft Research.
- Kang, S. B., Li, Y., Tong, X., and Shum, H.-Y. (2006). Image-based rendering. *Foundations and Trends in Computer Graphics and Computer Vision*, 2(3):173–258.
- Kang, S. B., Uyttendaele, M., Winder, S., and Szeliski, R. (2003). High dynamic range video. *ACM Transactions on Graphics (Proc. SIGGRAPH 2003)*, 22(3):319–325.
- Kang, S. B., Webb, J., Zitnick, L., and Kanade, T. (1995). A multibaseline stereo system with active illumination and real-time image acquisition. In *Fifth International Conference on Computer Vision (ICCV'95)*, pp. 88–93, Cambridge, Massachusetts.
- Kannala, J., Rahtu, E., Brandt, S. S., and Heikkila, J. (2008). Object recognition and segmentation by non-rigid quasi-dense matching. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.

- Kass, M. (1988). Linear image features in stereopsis. *International Journal of Computer Vision*, 1(4):357–368.
- Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331.
- Kato, H., Billinghurst, M., Poupyrev, I., Imamoto, K., and Tachibana, K. (2000). Virtual object manipulation on a table-top AR environment. In *International Symposium on Augmented Reality (ISAR 2000)*.
- Kaufman, L. and Rousseeuw, P. J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, Hoboken.
- Kazhdan, M., Bolitho, M., and Hoppe, H. (2006). Poisson surface reconstruction. In *Eurographics Symposium on Geometry Processing*, pp. 61–70.
- Ke, Y. and Sukthankar, R. (2004). PCA-SIFT: a more distinctive representation for local image descriptors. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2004)*, pp. 506–513, Washington, DC.
- Kehl, R. and Van Gool, L. (2006). Markerless tracking of complex human motions from multiple views. *Computer Vision and Image Understanding*, 104(2-3):190–209.
- Kenney, C., Zuliani, M., and Manjunath, B. (2005). An axiomatic approach to corner detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 191–197, San Diego, CA.
- Keren, D., Peleg, S., and Brada, R. (1988). Image sequence enhancement using sub-pixel displacements. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'88)*, pp. 742–746, Ann Arbor, Michigan.
- Kim, J., Kolmogorov, V., and Zabih, R. (2003). Visual correspondence using energy minimization and mutual information. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 1033–1040, Nice, France.
- Kimmel, R. (1999). Demosaicing: image reconstruction from color CCD samples. *IEEE Transactions on Image Processing*, 8(9):1221–1228.
- Kimura, S., Shinbo, T., Yamaguchi, H., Kawamura, E., and Nakano, K. (1999). A convolver-based real-time stereo machine (SAZAN). In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99)*, pp. 457–463, Fort Collins.
- Kindermann, R. and Snell, J. L. (1980). *Markov Random Fields and Their Applications*. American Mathematical Society.
- King, D. (1997). *The Commissar Vanishes*. Henry Holt and Company.
- Kirby, M. and Sirovich, L. (1990). Application of the Karhunen–Lòeve procedure for the characterization of human faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):103–108.
- Kirkpatrick, S., Gelatt, C. D. J., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220:671–680.
- Kirovski, D., Jojic, N., and Jancke, G. (2004). Tamper-resistant biometric IDs. In *ISSE 2004 - Securing Electronic Business Processes: Highlights of the Information Security Solutions Europe 2004 Conference*, pp. 160–175.
- Kittler, J. and Föglein, J. (1984). Contextual classification of multispectral pixel data. *Image and Vision Computing*, 2(1):13–29.

- Klaus, A., Sormann, M., and Karner, K. (2006). Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In *International Conference on Pattern Recognition (ICPR 2006)*, pp. 15–18.
- Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. In *International Symposium on Mixed and Augmented Reality (ISMAR 2007)*, Nara.
- Klein, G. and Murray, D. (2008). Improving the agility of keyframe-based slam. In *Tenth European Conference on Computer Vision (ECCV 2008)*, pp. 802–815, Marseilles.
- Klein, S., Staring, M., and Pluim, J. P. W. (2007). Evaluation of optimization methods for nonrigid medical image registration using mutual information and B-splines. *IEEE Transactions on Image Processing*, 16(12):2879–2890.
- Klinker, G. J. (1993). *A Physical Approach to Color Image Understanding*. A K Peters, Wellesley, Massachusetts.
- Klinker, G. J., Shafer, S. A., and Kanade, T. (1990). A physical approach to color image understanding. *International Journal of Computer Vision*, 4(1):7–38.
- Koch, R., Pollefeys, M., and Van Gool, L. J. (2000). Realistic surface reconstruction of 3D scenes from uncalibrated image sequences. *Journal Visualization and Computer Animation*, 11:115–127.
- Koenderink, J. J. (1990). *Solid Shape*. MIT Press, Cambridge, Massachusetts.
- Koethe, U. (2003). Integrated edge and junction detection with the boundary tensor. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 424–431, Nice, France.
- Kohli, P. (2007). *Minimizing Dynamic and Higher Order Energy Functions using Graph Cuts*. Ph.D. thesis, Oxford Brookes University.
- Kohli, P. and Torr, P. H. S. (2005). Efficiently solving dynamic Markov random fields using graph cuts. In *Tenth International Conference on Computer Vision (ICCV 2005)*, pp. 922–929, Beijing, China.
- Kohli, P. and Torr, P. H. S. (2007). Dynamic graph cuts for efficient inference in markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2079–2088.
- Kohli, P. and Torr, P. H. S. (2008). Measuring uncertainty in graph cut solutions. *Computer Vision and Image Understanding*, 112(1):30–38.
- Kohli, P., Kumar, M. P., and Torr, P. H. S. (2009). \mathcal{P}^3 & beyond: Move making algorithms for solving higher order functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(9):1645–1656.
- Kohli, P., Ladický, L., and Torr, P. H. S. (2009). Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324.
- Kokaram, A. (2004). On missing data treatment for degraded video and film archives: a survey and a new Bayesian approach. *IEEE Transactions on Image Processing*, 13(3):397–415.
- Kolev, K. and Cremers, D. (2008). Integration of multiview stereo and silhouettes via convex functionals on convex domains. In *Tenth European Conference on Computer Vision (ECCV 2008)*, pp. 752–765, Marseilles.
- Kolev, K. and Cremers, D. (2009). Continuous ratio optimization via convex relaxation with applications to multiview 3D reconstruction. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, Miami Beach, FL.
- Kolev, K., Klodt, M., Brox, T., and Cremers, D. (2009). Continuous global optimization in multiview 3D reconstruction. *International Journal of Computer Vision*, 84(1):80–96.

- Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, Cambridge, Massachusetts.
- Kolmogorov, V. (2006). Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1568–1583.
- Kolmogorov, V. and Boykov, Y. (2005). What metrics can be approximated by geo-cuts, or global optimization of length/area and flux. In *Tenth International Conference on Computer Vision (ICCV 2005)*, pp. 564–571, Beijing, China.
- Kolmogorov, V. and Zabih, R. (2002). Multi-camera scene reconstruction via graph cuts. In *Seventh European Conference on Computer Vision (ECCV 2002)*, pp. 82–96, Copenhagen.
- Kolmogorov, V. and Zabih, R. (2004). What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159.
- Kolmogorov, V., Criminisi, A., Blake, A., Cross, G., and Rother, C. (2006). Probabilistic fusion of stereo with color and contrast for bi-layer segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1480–1492.
- Komodakis, N. and Paragios, N. (2008). Beyond loose LP-relaxations: Optimizing MRFs by repairing cycles. In *Tenth European Conference on Computer Vision (ECCV 2008)*, pp. 806–820, Marseilles.
- Komodakis, N. and Tziritas, G. (2007a). Approximate labeling via graph cuts based on linear programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8):1436–1453.
- Komodakis, N. and Tziritas, G. (2007b). Image completion using efficient belief propagation via priority scheduling and dynamic pruning. *IEEE Transactions on Image Processing*, 29(11):2649–2661.
- Komodakis, N., Paragios, N., and Tziritas, G. (2007). MRF optimization via dual decomposition: Message-passing revisited. In *Eleventh International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil.
- Komodakis, N., Tziritas, G., and Paragios, N. (2007). Fast, approximately optimal solutions for single and dynamic MRFs. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Komodakis, N., Tziritas, G., and Paragios, N. (2008). Performance vs computational efficiency for optimizing single and dynamic MRFs: Setting the state of the art with primal dual strategies. *Computer Vision and Image Understanding*, 112(1):14–29.
- Konolige, K. (1997). Small vision systems: Hardware and implementation. In *Eighth International Symposium on Robotics Research*, pp. 203–212, Hayama, Japan.
- Kopf, J., Cohen, M. F., Lischinski, D., and Uyttendaele, M. (2007). Joint bilateral upsampling. *ACM Transactions on Graphics*, 26(3).
- Kopf, J., Uyttendaele, M., Deussen, O., and Cohen, M. F. (2007). Capturing and viewing gigapixel images. *ACM Transactions on Graphics*, 26(3).
- Kopf, J., Lischinski, D., Deussen, O., Cohen-Or, D., and Cohen, M. (2009). Locally adapted projections to reduce panorama distortions. *Computer Graphics Forum (Proceedings of EGSR 2009)*, 28(4).
- Koutis, I. (2007). *Combinatorial and algebraic tools for optimal multilevel algorithms*. Ph.D. thesis, Carnegie Mellon University. Technical Report CMU-CS-07-131.

- Koutis, I. and Miller, G. L. (2008). Graph partitioning into isolated, high conductance clusters: theory, computation and applications to preconditioning. In *Symposium on Parallel Algorithms and Architectures*, pp. 137–145, Munich.
- Koutis, I., Miller, G. L., and Tolliver, D. (2009). Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing. In *5th International Symposium on Visual Computing (ISVC09)*, Las Vegas.
- Kovar, L., Gleicher, M., and Pighin, F. (2002). Motion graphs. *ACM Transactions on Graphics*, 21(3):473–482.
- Košecká, J. and Zhang, W. (2005). Extraction, matching and pose recovery based on dominant rectangular structures. *Computer Vision and Image Understanding*, 100(3):174–293.
- Kraus, K. (1997). *Photogrammetry*. Dümmler, Bonn.
- Krishnan, D. and Fergus, R. (2009). Fast image deconvolution using hyper-Laplacian priors. In *Advances in Neural Information Processing Systems*.
- Kuglin, C. D. and Hines, D. C. (1975). The phase correlation image alignment method. In *IEEE 1975 Conference on Cybernetics and Society*, pp. 163–165, New York.
- Kulis, B. and Grauman, K. (2009). Kernelized locality-sensitive hashing for scalable image search. In *Twelfth International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan.
- Kumar, M. P. (2008). *Combinatorial and Convex Optimization for Probabilistic Models in Computer Vision*. Ph.D. thesis, Oxford Brookes University.
- Kumar, M. P. and Torr, P. H. S. (2006). Fast memory-efficient generalized belief propagation. In *Ninth European Conference on Computer Vision (ECCV 2006)*, pp. 451–463.
- Kumar, M. P. and Torr, P. H. S. (2008). Improved moves for truncated convex models. In *Advances in Neural Information Processing Systems*.
- Kumar, M. P., Torr, P. H. S., and Zisserman, A. (2008). Learning layered motion segmentations of video. *International Journal of Computer Vision*, 76(3):301–319.
- Kumar, M. P., Torr, P. H. S., and Zisserman, A. (2010). OBJCUT: Efficient segmentation using top-down and bottom-up cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3).
- Kumar, M. P., Veksler, O., and Torr, P. H. S. (2010). Improved moves for truncated convex models. *Journal of Machine Learning Research*, (submitted).
- Kumar, M. P., Zisserman, A., and H.S.Torr, P. (2009). Efficient discriminative learning of parts-based models. In *Twelfth International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan.
- Kumar, R., Anandan, P., and Hanna, K. (1994). Direct recovery of shape from multiple views: A parallax based approach. In *Twelfth International Conference on Pattern Recognition (ICPR'94)*, pp. 685–688, Jerusalem, Israel.
- Kumar, R., Anandan, P., Irani, M., Bergen, J., and Hanna, K. (1995). Representation of scenes from collections of images. In *IEEE Workshop on Representations of Visual Scenes*, pp. 10–17, Cambridge, Massachusetts.
- Kumar, S. and Hebert, M. (2003). Discriminative random fields: A discriminative framework for contextual interaction in classification. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 1150–1157, Nice, France.
- Kumar, S. and Hebert, M. (2006). Discriminative random fields. *International Journal of Computer Vision*, 68(2):179–202.

- Kundur, D. and Hatzinakos, D. (1996). Blind image deconvolution. *IEEE Signal Processing Magazine*, 13(3):43–64.
- Kutulakos, K. N. (2000). Approximate N-view stereo. In *Sixth European Conference on Computer Vision (ECCV 2000)*, pp. 67–83, Dublin, Ireland.
- Kutulakos, K. N. and Seitz, S. M. (2000). A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):199–218.
- Kwatra, V., Essa, I., Bobick, A., and Kwatra, N. (2005). Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics (Proc. SIGGRAPH 2005)*, 24(5):795–802.
- Kwatra, V., Schödl, A., Essa, I., Turk, G., and Bobick, A. (2003). Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics (Proc. SIGGRAPH 2003)*, 22(3):277–286.
- Kybík, J. and Unser, M. (2003). Fast parametric elastic image registration. *IEEE Transactions on Image Processing*, 12(11):1427–1442.
- Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*.
- Lafourcade, E. P. F., Foo, S.-C., Torrance, K. E., and Greenberg, D. P. (1997). Non-linear approximation of reflectance functions. In *ACM SIGGRAPH 1997 Conference Proceedings*, pp. 117–126, Los Angeles.
- Lai, S.-H. and Vemuri, B. C. (1997). Physically based adaptive preconditioning for early vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):594–607.
- Lalonde, J.-F., Hoiem, D., Efros, A. A., Rother, C., Winn, J., and Criminisi, A. (2007). Photo clip art. *ACM Transactions on Graphics*, 26(3).
- Lampert, C. H. (2008). Kernel methods in computer vision. *Foundations and Trends in Computer Graphics and Computer Vision*, 4(3):193–285.
- Langer, M. S. and Zucker, S. W. (1994). Shape from shading on a cloudy day. *Journal Optical Society America, A*, 11(2):467–478.
- Lanitis, A., Taylor, C. J., and Cootes, T. F. (1997). Automatic interpretation and coding of face images using flexible models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):742–756.
- Larlus, D. and Jurie, F. (2008). Combining appearance models and Markov random fields for category level object segmentation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Larson, G. W. (1998). LogLuv encoding for full-gamut, high-dynamic range images. *Journal of Graphics Tools*, 3(1):15–31.
- Larson, G. W., Rushmeier, H., and Piatko, C. (1997). A visibility matching tone reproduction operator for high dynamic range scenes. *IEEE Transactions on Visualization and Computer Graphics*, 3(4):291–306.
- Laurentini, A. (1994). The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162.
- Lavallée, S. and Szeliski, R. (1995). Recovering the position and orientation of free-form objects from image contours using 3-D distance maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(4):378–390.
- Laveau, S. and Faugeras, O. D. (1994). 3-D scene representation as a collection of images. In *Twelfth International Conference on Pattern Recognition (ICPR'94)*, pp. 689–691, Jerusalem, Israel.

- Lazebnik, S., Schmid, C., and Ponce, J. (2005). A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1265–1278.
- Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2006)*, pp. 2169–2176, New York City, NY.
- Le Gall, D. (1991). MPEG: A video compression standard for multimedia applications. *Communications of the ACM*, 34(4):46–58.
- Leclerc, Y. G. (1989). Constructing simple stable descriptions for image partitioning. *International Journal of Computer Vision*, 3(1):73–102.
- LeCun, Y., Huang, F. J., and Bottou, L. (2004). Learning methods for generic object recognition with invariance to pose and lighting. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2004)*, pp. 97–104, Washington, DC.
- Lee, J., Chai, J., Reitsma, P. S. A., Hodgins, J. K., and Pollard, N. S. (2002). Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics*, 21(3):491–500.
- Lee, M.-C., ge Chen, W., lung Bruce Lin, C., Gu, C., Markoc, T., Zabinsky, S. I., and Szeliski, R. (1997). A layered video object coding system using sprite and affine motion model. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(1):130–145.
- Lee, M. E. and Redner, R. A. (1990). A note on the use of nonlinear filtering in computer graphics. *IEEE Computer Graphics and Applications*, 10(3):23–29.
- Lee, M. W. and Cohen, I. (2006). A model-based approach for estimating human 3D poses in static images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(6):905–916.
- Lee, S., Wolberg, G., and Shin, S. Y. (1996). Data interpolation using multilevel b-splines. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):228–244.
- Lee, S., Wolberg, G., Chwa, K.-Y., and Shin, S. Y. (1996). Image metamorphosis with scattered feature constraints. *IEEE Transactions on Visualization and Computer Graphics*, 2(4):337–354.
- Lee, Y. D., Terzopoulos, D., and Waters, K. (1995). Realistic facial modeling for animation. In *ACM SIGGRAPH 1995 Conference Proceedings*, pp. 55–62.
- Lei, C. and Yang, Y.-H. (2009). Optical flow estimation on coarse-to-fine region-trees using discrete optimization. In *Twelfth International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan.
- Leibe, B. and Schiele, B. (2003). Analyzing appearance and contour based methods for object categorization. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2003)*, pp. 409–415, Madison, WI.
- Leibe, B., Leonardis, A., and Schiele, B. (2008). Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 77(1-3):259–289.
- Leibe, B., Seemann, E., and Schiele, B. (2005). Pedestrian detection in crowded scenes. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 878–885, San Diego, CA.
- Leibe, B., Cornelis, N., Cornelis, K., and Van Gool, L. (2007). Dynamic 3D scene analysis from a moving vehicle. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.

- Leibowitz, D. (2001). *Camera Calibration and Reconstruction of Geometry from Images*. Ph.D. thesis, University of Oxford.
- Lempitsky, V. and Boykov, Y. (2007). Global optimization for shape fitting. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Lempitsky, V. and Ivanov, D. (2007). Seamless mosaicing of image-based texture maps. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Lempitsky, V., Blake, A., and Rother, C. (2008). Image segmentation by branch-and-mincut. In *Tenth European Conference on Computer Vision (ECCV 2008)*, pp. 15–29, Marseilles.
- Lempitsky, V., Roth, S., and Rother, C. (2008). FlowFusion: Discrete-continuous optimization for optical flow estimation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Lempitsky, V., Rother, C., and Blake, A. (2007). Logcut - efficient graph cut optimization for Markov random fields. In *Eleventh International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil.
- Lengyel, J. and Snyder, J. (1997). Rendering with coherent layers. In *ACM SIGGRAPH 1997 Conference Proceedings*, pp. 233–242, Los Angeles.
- Lensch, H. P. A., Kautz, J., Goesele, M., Heidrich, W., and Seidel, H.-P. (2003). Image-based reconstruction of spatial appearance and geometric detail. *ACM Transactions on Graphics*, 22(2):234–257.
- Leonardis, A. and Bischof, H. (2000). Robust recognition using eigenimages. *Computer Vision and Image Understanding*, 78(1):99–118.
- Leonardis, A., Jaklič, A., and Solina, F. (1997). Superquadrics for segmenting and modeling range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1289–1295.
- Lepetit, V. and Fua, P. (2005). Monocular model-based 3D tracking of rigid objects. *Foundations and Trends in Computer Graphics and Computer Vision*, 1(1).
- Lepetit, V., Pilet, J., and Fua, P. (2004). Point matching as a classification problem for fast and robust object pose estimation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2004)*, pp. 244–250, Washington, DC.
- Lepetit, V., Pilet, J., and Fua, P. (2006). Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1465–1479.
- Leung, T. K., Burl, M. C., and Perona, P. (1995). Finding faces in cluttered scenes using random labeled graph matching. In *Fifth International Conference on Computer Vision (ICCV'95)*, pp. 637–644, Cambridge, Massachusetts.
- Levenberg, K. (1944). A method for the solution of certain problems in least squares. *Quarterly of Applied Mathematics*, 2:164–168.
- Levin, A. (2006). Blind motion deblurring using image statistics. In *Advances in Neural Information Processing Systems*.
- Levin, A. and Szeliski, R. (2004). Visual odometry and map correlation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2004)*, pp. 611–618, Washington, DC.
- Levin, A. and Szeliski, R. (2006). *Motion Uncertainty and Field of View*. Technical Report MSR-TR-2006-37, Microsoft Research.
- Levin, A. and Weiss, Y. (2006). Learning to combine bottom-up and top-down segmentation. In *Ninth European Conference on Computer Vision (ECCV 2006)*, pp. 581–594.

- Levin, A. and Weiss, Y. (2007). User assisted separation of reflections from a single image using a sparsity prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(9):1647–1654.
- Levin, A., Acha, A. R., and Lischinski, D. (2008). Spectral matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(10):1699–1712.
- Levin, A., Lischinski, D., and Weiss, Y. (2004). Colorization using optimization. *ACM Transactions on Graphics*, 23(3):689–694.
- Levin, A., Lischinski, D., and Weiss, Y. (2008). A closed form solution to natural image matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):228–242.
- Levin, A., Zomet, A., and Weiss, Y. (2004). Separating reflections from a single image using local features. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2004)*, pp. 306–313, Washington, DC.
- Levin, A., Fergus, R., Durand, F., and Freeman, W. T. (2007). Image and depth from a conventional camera with a coded aperture. *ACM Transactions on Graphics*, 26(3).
- Levin, A., Weiss, Y., Durand, F., and Freeman, B. (2009). Understanding and evaluating blind deconvolution algorithms. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, Miami Beach, FL.
- Levin, A., Zomet, A., Peleg, S., and Weiss, Y. (2004). Seamless image stitching in the gradient domain. In *Eighth European Conference on Computer Vision (ECCV 2004)*, pp. 377–389, Prague.
- Levoy, M. (1988). Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37.
- Levoy, M. (2006). Light fields and computational imaging. *Computer*, 39(8):46–55.
- Levoy, M. (2008). Technical perspective: Computational photography on large collections of images. *Communications of the ACM*, 51(10):86.
- Levoy, M. and Hanrahan, P. (1996). Light field rendering. In *ACM SIGGRAPH 1996 Conference Proceedings*, pp. 31–42, New Orleans.
- Levoy, M. and Whitted, T. (1985). *The Use of Points as a Display Primitive*. Technical Report 85-022, University of North Carolina at Chapel Hill.
- Levoy, M., Ng, R., Adams, A., Footer, M., and Horowitz, M. (2006). Light field microscopy. *ACM Transactions on Graphics*, 25(3):924–934.
- Levoy, M., Pulli, K., Curless, B., Rusinkiewicz, S., Koller, D. et al. (2000). The digital Michelangelo project: 3D scanning of large statues. In *ACM SIGGRAPH 2000 Conference Proceedings*, pp. 131–144.
- Lew, M. S., Sebe, N., Djeraba, C., and Jain, R. (2006). Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications and Applications*, 2(1):1–19.
- Leyvand, T., Cohen-Or, D., Dror, G., and Lischinski, D. (2008). Data-driven enhancement of facial attractiveness. *ACM Transactions on Graphics*, 27(3).
- Lhuillier, M. and Quan, L. (2002). Match propagation for image-based modeling and rendering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1140–1146.
- Lhuillier, M. and Quan, L. (2005). A quasi-dense approach to surface reconstruction from uncalibrated images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):418–433.

- Li, H. and Hartley, R. (2007). The 3D–3D registration problem revisited. In *Eleventh International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil.
- Li, L.-J. and Fei-Fei, L. (2010). Optimol: Automatic object picture collection via incremental model learning. *International Journal of Computer Vision*, 88(2):147–168.
- Li, S. (1995). *Markov Random Field Modeling in Computer Vision*. Springer-Verlag.
- Li, S. Z. and Jain, A. K. (eds). (2005). *Handbook of Face Recognition*, Springer.
- Li, X., Wu, C., Zach, C., Lazebnik, S., and Frahm, J.-M. (2008). Modeling and recognition of landmark image collections using iconic scene graphs. In *Tenth European Conference on Computer Vision (ECCV 2008)*, pp. 427–440, Marseilles.
- Li, Y. and Huttenlocher, D. P. (2008). Learning for optical flow using stochastic optimization. In *Tenth European Conference on Computer Vision (ECCV 2008)*, pp. 379–391, Marseilles.
- Li, Y., Crandall, D. J., and Huttenlocher, D. P. (2009). Landmark classification in large-scale image collections. In *Twelfth International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan.
- Li, Y., Wang, T., and Shum, H.-Y. (2002). Motion texture: a two-level statistical model for character motion synthesis. *ACM Transactions on Graphics*, 21(3):465–472.
- Li, Y., Shum, H.-Y., Tang, C.-K., and Szeliski, R. (2004). Stereo reconstruction from multiperspective panoramas. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1):44–62.
- Li, Y., Sun, J., Tang, C.-K., and Shum, H.-Y. (2004). Lazy snapping. *ACM Transactions on Graphics (Proc. SIGGRAPH 2004)*, 23(3):303–308.
- Liang, L., Xiao, R., Wen, F., and Sun, J. (2008). Face alignment via component-based discriminative search. In *Tenth European Conference on Computer Vision (ECCV 2008)*, pp. 72–85, Marseilles.
- Liang, L., Liu, C., Xu, Y.-Q., Guo, B., and Shum, H.-Y. (2001). Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics*, 20(3):127–150.
- Liebowitz, D. and Zisserman, A. (1998). Metric rectification for perspective images of planes. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'98)*, pp. 482–488, Santa Barbara.
- Lim, J. (1990). *Two-Dimensional Signal and Image Processing*. Prentice-Hall, Englewood, NJ.
- Lim, J. J., Arbeláez, P., Gu, C., and Malik, J. (2009). Context by region ancestry. In *Twelfth International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan.
- Lin, D., Kapoor, A., Hua, G., and Baker, S. (2010). Joint people, event, and location recognition in personal photo collections using cross-domain context. In *Eleventh European Conference on Computer Vision (ECCV 2010)*, Heraklion, Crete.
- Lin, W.-C., Hays, J., Wu, C., Kwatra, V., and Liu, Y. (2006). Quantitative evaluation of near regular texture synthesis algorithms. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2006)*, pp. 427–434, New York City, NY.
- Lindeberg, T. (1990). Scale-space for discrete signals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(3):234–254.
- Lindeberg, T. (1993). Detecting salient blob-like image structures and their scales with a scale-space primal sketch: a method for focus-of-attention. *International Journal of Computer Vision*, 11(3):283–318.

- Lindeberg, T. (1994). Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics*, 21(2):224–270.
- Lindeberg, T. (1998a). Edge detection and ridge detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):116–154.
- Lindeberg, T. (1998b). Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):79–116.
- Lindeberg, T. and Garding, J. (1997). Shape-adapted smoothing in estimation of 3-D shape cues from affine deformations of local 2-D brightness structure. *Image and Vision Computing*, 15(6):415–434.
- Lippman, A. (1980). Movie maps: An application of the optical videodisc to computer graphics. *Computer Graphics (SIGGRAPH '80)*, 14(3):32–43.
- Lischinski, D., Farbman, Z., Uyttendaele, M., and Szeliski, R. (2006a). Interactive local adjustment of tonal values. *ACM Transactions on Graphics (Proc. SIGGRAPH 2006)*, 25(3):646–653.
- Lischinski, D., Farbman, Z., Uyttendaele, M., and Szeliski, R. (2006b). Interactive local adjustment of tonal values. *ACM Transactions on Graphics*, 25(3):646–653.
- Litvinov, A. and Schechner, Y. Y. (2005). Radiometric framework for image mosaicking. *Journal of the Optical Society of America A*, 22(5):839–848.
- Litwinowicz, P. (1997). Processing images and video for an impressionist effect. In *ACM SIGGRAPH 1997 Conference Proceedings*, pp. 407–414.
- Litwinowicz, P. and Williams, L. (1994). Animating images with drawings. In *ACM SIGGRAPH 1994 Conference Proceedings*, pp. 409–412.
- Liu, C. (2009). *Beyond Pixels: Exploring New Representations and Applications for Motion Analysis*. Ph.D. thesis, Massachusetts Institute of Technology.
- Liu, C., Yuen, J., and Torralba, A. (2009). Nonparametric scene parsing: Label transfer via dense scene alignment. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, Miami Beach, FL.
- Liu, C., Freeman, W. T., Adelson, E., and Weiss, Y. (2008). Human-assisted motion annotation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Liu, C., Szeliski, R., Kang, S. B., Zitnick, C. L., and Freeman, W. T. (2008). Automatic estimation and removal of noise from a single image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):299–314.
- Liu, F., Gleicher, M., Jin, H., and Agarwala, A. (2009). Content-preserving warps for 3d video stabilization. *ACM Transactions on Graphics*, 28(3).
- Liu, X., Chen, T., and Kumar, B. V. (2003). Face authentication for multiple subjects using eigenflow. *Pattern Recognition*, 36(2):313–328.
- Liu, Y., Collins, R. T., and Tsin, Y. (2004). A computational model for periodic pattern perception based on frieze and wallpaper groups. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):354–371.
- Liu, Y., Lin, W.-C., and Hays, J. (2004). Near-regular texture analysis and manipulation. *ACM Transactions on Graphics*, 23(3):368–376.
- Livingstone, M. (2008). *Vision and Art: The Biology of Seeing*. Abrams, New York.

- Lobay, A. and Forsyth, D. A. (2006). Shape from texture without boundaries. *International Journal of Computer Vision*, 67(1):71–91.
- Lombaert, H., Sun, Y., Grady, L., and Xu, C. (2005). A multilevel banded graph cuts method for fast image segmentation. In *Tenth International Conference on Computer Vision (ICCV 2005)*, pp. 259–265, Beijing, China.
- Longere, P., Delahunt, P. B., Zhang, X., and Brainard, D. H. (2002). Perceptual assessment of demosaicing algorithm performance. *Proceedings of the IEEE*, 90(1):123–132.
- Longuet-Higgins, H. C. (1981). A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135.
- Loop, C. and Zhang, Z. (1999). Computing rectifying homographies for stereo vision. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99)*, pp. 125–131, Fort Collins.
- Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics (SIGGRAPH '87)*, 21(4):163–169.
- Lorusso, A., Eggert, D., and Fisher, R. B. (1995). A comparison of four algorithms for estimating 3-D rigid transformations. In *British Machine Vision Conference (BMVC95)*, pp. 237–246, Birmingham, England.
- Lourakis, M. I. A. and Argyros, A. A. (2009). SBA: A software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software*, 36(1).
- Lowe, D. G. (1988). Organization of smooth image curves at multiple scales. In *Second International Conference on Computer Vision (ICCV'88)*, pp. 558–567, Tampa.
- Lowe, D. G. (1989). Organization of smooth image curves at multiple scales. *International Journal of Computer Vision*, 3(2):119–130.
- Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Seventh International Conference on Computer Vision (ICCV'99)*, pp. 1150–1157, Kerkyra, Greece.
- Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.
- Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application in stereo vision. In *Seventh International Joint Conference on Artificial Intelligence (IJCAI-81)*, pp. 674–679, Vancouver.
- Luong, Q.-T. and Faugeras, O. D. (1996). The fundamental matrix: Theory, algorithms, and stability analysis. *International Journal of Computer Vision*, 17(1):43–75.
- Luong, Q.-T. and Viéville, T. (1996). Canonical representations for the geometries of multiple projective views. *Computer Vision and Image Understanding*, 64(2):193–229.
- Lyu, S. and Simoncelli, E. (2008). Nonlinear image representation using divisive normalization. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Lyu, S. and Simoncelli, E. (2009). Modeling multiscale subbands of photographic images with fields of Gaussian scale mixtures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):693–706.
- Ma, W.-C., Jones, A., Chiang, J.-Y., Hawkins, T., Frederiksen, S., Peers, P., Vukovic, M., Ouhyoung, M., andDebevec, P. (2008). Facial performance synthesis using deformation-driven polynomial displacement maps. *ACM Transactions on Graphics*, 27(5).

- Ma, Y., Derksen, H., Hong, W., and Wright, J. (2007). Segmentation of multivariate mixed data via lossy data coding and compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(9):1546–1562.
- MacDonald, L. (ed.). (2006). *Digital Heritage: Applying Digital Imaging to Cultural Heritage*, Butterworth-Heinemann.
- Madsen, K., Nielsen, H. B., and Tingleff, O. (2004). Methods for non-linear least squares problems. *Informatics and Mathematical Modelling*, Technical University of Denmark (DTU).
- Maes, F., Collignon, A., Vandermeulen, D., Marchal, G., and Suetens, P. (1997). Multimodality image registration by maximization of mutual information. *IEEE Transactions on Medical Imaging*, 16(2):187–198.
- Magnor, M. (2005). *Video-Based Rendering*. A. K. Peters, Wellesley, MA.
- Magnor, M. and Girod, B. (2000). Data compression for light-field rendering. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(3):338–343.
- Magnor, M., Ramanathan, P., and Girod, B. (2003). Multi-view coding for image-based rendering using 3-D scene geometry. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(11):1092–1106.
- Mahajan, D., Huang, F.-C., Matusik, W., Ramamoorthi, R., and Belhumeur, P. (2009). Moving gradients: A path-based method for plausible image interpolation. *ACM Transactions on Graphics*, 28(3).
- Maimone, M., Cheng, Y., and Matthies, L. (2007). Two years of visual odometry on the Mars exploration rovers. *Journal of Field Robotics*, 24(3).
- Maire, M., Arbelaez, P., Fowlkes, C., and Malik, J. (2008). Using contours to detect and localize junctions in natural images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Maitin-Shepard, J., Cusumano-Towner, M., Lei, J., and Abbeel, P. (2010). Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In *IEEE International Conference on Robotics and Automation*, Anchorage, AK.
- Maitre, M., Shinagawa, Y., and Do, M. N. (2008). Symmetric multi-view stereo reconstruction from planar camera arrays. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Maji, S., Berg, A., and Malik, J. (2008). Classification using intersection kernel support vector machines is efficient. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Malik, J. and Rosenholtz, R. (1997). Computing local surface orientation and shape from texture for curved surfaces. *International Journal of Computer Vision*, 23(2):149–168.
- Malik, J., Belongie, S., Leung, T., and Shi, J. (2001). Contour and texture analysis for image segmentation. *International Journal of Computer Vision*, 43(1):7–27.
- Malisiewicz, T. and Efros, A. A. (2008). Recognition by association via learning per-exemplar distances. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Malladi, R., Sethian, J. A., and Vemuri, B. C. (1995). Shape modeling with front propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):158–176.
- Mallat, S. G. (1989). A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-11(7):674–693.

- Malvar, H. S. (1990). Lapped transforms for efficient transform/subband coding. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(6):969–978.
- Malvar, H. S. (1998). Biorthogonal and nonuniform lapped transforms for transform coding with reduced blocking and ringing artifacts. *IEEE Transactions on Signal Processing*, 46(4):1043–1053.
- Malvar, H. S. (2000). Fast progressive image coding without wavelets. In *IEEE Data Compressions Conference*, pp. 243–252, Snowbird, UT.
- Malvar, H. S., He, L.-W., and Cutler, R. (2004). High-quality linear interpolation for demosaicing of Bayer-patterned color images. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'04)*, pp. 485–488, Montreal.
- Mancini, T. A. and Wolff, L. B. (1992). 3D shape and light source location from depth and reflectance. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'92)*, pp. 707–709, Champaign, Illinois.
- Manjunath, B. S. and Ma, W. Y. (1996). Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):837–842.
- Mann, S. and Picard, R. W. (1994). Virtual bellows: Constructing high-quality images from video. In *First IEEE International Conference on Image Processing (ICIP-94)*, pp. 363–367, Austin.
- Mann, S. and Picard, R. W. (1995). On being ‘undigital’ with digital cameras: Extending dynamic range by combining differently exposed pictures. In *IS&T’s 48th Annual Conference*, pp. 422–428, Washington, D. C.
- Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.
- Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441.
- Marr, D. (1982). *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. W. H. Freeman, San Francisco.
- Marr, D. and Hildreth, E. (1980). Theory of edge detection. *Proceedings of the Royal Society of London, B* 207:187–217.
- Marr, D. and Nishihara, H. K. (1978). Representation and recognition of the spatial organization of three-dimensional shapes. *Proc. Roy. Soc. London, B*, 200:269–294.
- Marr, D. and Poggio, T. (1976). Cooperative computation of stereo disparity. *Science*, 194:283–287.
- Marr, D. C. and Poggio, T. (1979). A computational theory of human stereo vision. *Proceedings of the Royal Society of London, B* 204:301–328.
- Marroquin, J., Mitter, S., and Poggio, T. (1985). Probabilistic solution of ill-posed problems in computational vision. In *Image Understanding Workshop*, pp. 293–309, Miami Beach.
- Marroquin, J., Mitter, S., and Poggio, T. (1987). Probabilistic solution of ill-posed problems in computational vision. *Journal of the American Statistical Association*, 82(397):76–89.
- Marroquin, J. L. (1983). *Design of Cooperative Networks*. Working Paper 253, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.
- Martin, D., Fowlkes, C., and Malik, J. (2004). Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):530–549.

- Martin, D., Fowlkes, C., Tal, D., and Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Eighth International Conference on Computer Vision (ICCV 2001)*, pp. 416–423, Vancouver, Canada.
- Martin, W. N. and Aggarwal, J. K. (1983). Volumetric description of objects from multiple views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(2):150–158.
- Martinec, D. and Pajdla, T. (2007). Robust rotation and translation estimation in multiview reconstruction. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Massey, M. and Bender, W. (1996). Salient stills: Process and practice. *IBM Systems Journal*, 35(3&4):557–573.
- Matas, J., Chum, O., Urban, M., and Pajdla, T. (2004). Robust wide baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767.
- Matei, B. C. and Meer, P. (2006). Estimation of nonlinear errors-in-variables models for computer vision applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1537–1552.
- Matsushita, Y. and Lin, S. (2007). Radiometric calibration from noise distributions. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Matsushita, Y., Ofek, E., Ge, W., Tang, X., and Shum, H.-Y. (2006). Full-frame video stabilization with motion inpainting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7):1150–1163.
- Matthews, I. and Baker, S. (2004). Active appearance models revisited. *International Journal of Computer Vision*, 60(2):135–164.
- Matthews, I., Xiao, J., and Baker, S. (2007). 2D vs. 3D deformable face models: Representational power, construction, and real-time fitting. *International Journal of Computer Vision*, 75(1):93–113.
- Matthies, L., Kanade, T., and Szeliski, R. (1989). Kalman filter-based algorithms for estimating depth from image sequences. *International Journal of Computer Vision*, 3(3):209–236.
- Matusik, W., Buehler, C., and McMillan, L. (2001). Polyhedral visual hulls for real-time rendering. In *12th Eurographics Workshop on Rendering Techniques*, pp. 115–126, London.
- Matusik, W., Buehler, C., Raskar, R., Gortler, S. J., and McMillan, L. (2000). Image-based visual hulls. In *ACM SIGGRAPH 2000 Conference Proceedings*, pp. 369–374.
- Mayhew, J. E. W. and Frisby, J. P. (1980). The computation of binocular edges. *Perception*, 9:69–87.
- Mayhew, J. E. W. and Frisby, J. P. (1981). Psychophysical and computational studies towards a theory of human stereopsis. *Artificial Intelligence*, 17(1-3):349–408.
- McCamy, C. S., Marcus, H., and Davidson, J. G. (1976). A color-rendition chart. *Journal of Applied Photogrammetric Engineering*, 2(3):95–99.
- McCane, B., Novins, K., Crannitch, D., and Galvin, B. (2001). On benchmarking optical flow. *Computer Vision and Image Understanding*, 84(1):126–143.
- McGuire, M., Matusik, W., Pfister, H., Hughes, J. F., and Durand, F. (2005). Defocus video matting. *ACM Transactions on Graphics (Proc. SIGGRAPH 2005)*, 24(3):567–576.
- McInerney, T. and Terzopoulos, D. (1993). A finite element model for 3D shape reconstruction and nonrigid motion tracking. In *Fourth International Conference on Computer Vision (ICCV'93)*, pp. 518–523, Berlin, Germany.

- McInerney, T. and Terzopoulos, D. (1996). Deformable models in medical image analysis: A survey. *Medical Image Analysis*, 1(2):91–108.
- McInerney, T. and Terzopoulos, D. (1999). Topology adaptive deformable surfaces for medical image volume segmentation. *IEEE Transactions on Medical Imaging*, 18(10):840–850.
- McInerney, T. and Terzopoulos, D. (2000). T-snakes: Topology adaptive snakes. *Medical Image Analysis*, 4:73–91.
- McLauchlan, P. F. (2000). A batch/recursive algorithm for 3D scene reconstruction. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2000)*, pp. 738–743, Hilton Head Island.
- McLauchlan, P. F. and Jaenicke, A. (2002). Image mosaicing using sequential bundle adjustment. *Image and Vision Computing*, 20(9-10):751–759.
- McLean, G. F. and Kotturi, D. (1995). Vanishing point detection by line clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(11):1090–1095.
- McMillan, L. and Bishop, G. (1995). Plenoptic modeling: An image-based rendering system. In *ACM SIGGRAPH 1995 Conference Proceedings*, pp. 39–46.
- McMillan, L. and Gortler, S. (1999). Image-based rendering: A new interface between computer vision and computer graphics. *Computer Graphics*, 33(4):61–64.
- Meehan, J. (1990). *Panoramic Photography*. Watson-Guptill.
- Meer, P. and Georgescu, B. (2001). Edge detection with embedded confidence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12):1351–1365.
- Meilă, M. and Shi, J. (2000). Learning segmentation by random walks. In *Advances in Neural Information Processing Systems*.
- Meilă, M. and Shi, J. (2001). A random walks view of spectral segmentation. In *Workshop on Artificial Intelligence and Statistics*, pp. 177–182, Key West, FL.
- Meltzer, J. and Soatto, S. (2008). Edge descriptors for robust wide-baseline correspondence. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Meltzer, T., Yanover, C., and Weiss, Y. (2005). Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation. In *Tenth International Conference on Computer Vision (ICCV 2005)*, pp. 428–435, Beijing, China.
- Mémin, E. and Pérez, P. (2002). Hierarchical estimation and segmentation of dense motion fields. *International Journal of Computer Vision*, 44(2):129–155.
- Menet, S., Saint-Marc, P., and Medioni, G. (1990a). Active contour models: overview, implementation and applications. In *IEEE International Conference on Systems, Man and Cybernetics*, pp. 194–199, Los Angeles.
- Menet, S., Saint-Marc, P., and Medioni, G. (1990b). B-snakes: implementation and applications to stereo. In *Image Understanding Workshop*, pp. 720–726, Pittsburgh.
- Merrell, P., Akbarzadeh, A., Wang, L., Mordohai, P., Frahm, J.-M., Yang, R., Nister, D., and Pollefeys, M. (2007). Real-time visibility-based fusion of depth maps. In *Eleventh International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil.
- Mertens, T., Kautz, J., and Reeth, F. V. (2007). Exposure fusion. In *Proceedings of Pacific Graphics 2007*, pp. 382–390.

- Metaxas, D. and Terzopoulos, D. (2002). Dynamic deformation of solid primitives with constraints. *ACM Transactions on Graphics (Proc. SIGGRAPH 2002)*, 21(3):309–312.
- Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1091.
- Meyer, C. D. (2000). *Matrix Analysis and Applied Linear Algebra*. Society for Industrial and Applied Mathematics, Philadelphia.
- Meyer, Y. (1993). *Wavelets: Algorithms and Applications*. Society for Industrial and Applied Mathematics, Philadelphia.
- Mikolajczyk, K. and Schmid, C. (2004). Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86.
- Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630.
- Mikolajczyk, K., Schmid, C., and Zisserman, A. (2004). Human detection based on a probabilistic assembly of robust part detectors. In *Eighth European Conference on Computer Vision (ECCV 2004)*, pp. 69–82, Prague.
- Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., and Van Gool, L. J. (2005). A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1-2):43–72.
- Milgram, D. L. (1975). Computer methods for creating photomosaics. *IEEE Transactions on Computers*, C-24(11):1113–1119.
- Milgram, D. L. (1977). Adaptive techniques for photomosaicking. *IEEE Transactions on Computers*, C-26(11):1175–1180.
- Miller, I., Campbell, M., Huttenlocher, D., Kline, F.-R., Nathan, A. et al. (2008). Team Cornell’s Skynet: Robust perception and planning in an urban environment. *Journal of Field Robotics*, 25(8):493–527.
- Mitiche, A. and Boutheny, P. (1996). Computation and analysis of image motion: A synopsis of current problems and methods. *International Journal of Computer Vision*, 19(1):29–55.
- Mitsunaga, T. and Nayar, S. K. (1999). Radiometric self calibration. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’99)*, pp. 374–380, Fort Collins.
- Mittal, A. and Davis, L. S. (2003). M₂ tracker: A multi-view approach to segmenting and tracking people in a cluttered scene. *International Journal of Computer Vision*, 51(3):189–203.
- Mičušík, B. and Košecká, J. (2009). Piecewise planar city 3D modeling from street view panoramic sequences. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, Miami Beach, FL.
- Mičušík, B., Wildenauer, H., and Košecká, J. (2008). Detection and matching of rectilinear structures. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Moeslund, T. B. and Granum, E. (2001). A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3):231–268.
- Moeslund, T. B., Hilton, A., and Krüger, V. (2006). A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2-3):90–126.
- Moezzi, S., Katkere, A., Kuramura, D., and Jain, R. (1996). Reality modeling and visualization from multiple video sequences. *IEEE Computer Graphics and Applications*, 16(6):58–63.

- Moghaddam, B. and Pentland, A. (1997). Probabilistic visual learning for object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):696–710.
- Moghaddam, B., Jebara, T., and Pentland, A. (2000). Bayesian face recognition. *Pattern Recognition*, 33(11):1771–1782.
- Mohan, A., Papageorgiou, C., and Poggio, T. (2001). Example-based object detection in images by components. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(4):349–361.
- Möller, K. D. (1988). *Optics*. University Science Books, Mill Valley, CA.
- Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D. et al. (2008). Junior: The Stanford entry in the Urban Challenge. *Journal of Field Robotics*, 25(9):569–597.
- Moon, P. and Spencer, D. E. (1981). *The Photic Field*. MIT Press, Cambridge, Massachusetts.
- Moons, T., Van Gool, L., and Vergauwen, M. (2010). 3D reconstruction from multiple images. *Foundations and Trends in Computer Graphics and Computer Vision*, 4(4).
- Moosmann, F., Nowak, E., and Jurie, F. (2008). Randomized clustering forests for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(9):1632–1646.
- Moravec, H. (1977). Towards automatic visual obstacle avoidance. In *Fifth International Joint Conference on Artificial Intelligence (IJCAI'77)*, p. 584, Cambridge, Massachusetts.
- Moravec, H. (1983). The Stanford cart and the CMU rover. *Proceedings of the IEEE*, 71(7):872–884.
- Moreno-Noguer, F., Lepetit, V., and Fua, P. (2007). Accurate non-iterative $O(n)$ solution to the PnP problem. In *Eleventh International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil.
- Mori, G. (2005). Guiding model search using segmentation. In *Tenth International Conference on Computer Vision (ICCV 2005)*, pp. 1417–1423, Beijing, China.
- Mori, G., Ren, X., Efros, A., and Malik, J. (2004). Recovering human body configurations: Combining segmentation and recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2004)*, pp. 326–333, Washington, DC.
- Mori, M. (1970). The uncanny valley. *Energy*, 7(4):33–35. <http://www.androidscience.com/theuncannyvalley/proceedings2005/uncannyvalley.html>.
- Morimoto, C. and Chellappa, R. (1997). Fast 3D stabilization and mosaic construction. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pp. 660–665, San Juan, Puerto Rico.
- Morita, T. and Kanade, T. (1997). A sequential factorization method for recovering shape and motion from image streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(8):858–867.
- Morris, D. D. and Kanade, T. (1998). A unified factorization algorithm for points, line segments and planes with uncertainty models. In *Sixth International Conference on Computer Vision (ICCV'98)*, pp. 696–702, Bombay.
- Morrone, M. and Burr, D. (1988). Feature detection in human vision: A phase dependent energy model. *Proceedings of the Royal Society of London B*, 235:221–245.
- Mortensen, E. N. (1999). Vision-assisted image editing. *Computer Graphics*, 33(4):55–57.
- Mortensen, E. N. and Barrett, W. A. (1995). Intelligent scissors for image composition. In *ACM SIGGRAPH 1995 Conference Proceedings*, pp. 191–198.

- Mortensen, E. N. and Barrett, W. A. (1998). Interactive segmentation with intelligent scissors. *Graphical Models and Image Processing*, 60(5):349–384.
- Mortensen, E. N. and Barrett, W. A. (1999). Toboggan-based intelligent scissors with a four parameter edge model. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99)*, pp. 452–458, Fort Collins.
- Mueller, P., Zeng, G., Wonka, P., and Van Gool, L. (2007). Image-based procedural modeling of facades. *ACM Transactions on Graphics*, 26(3).
- Mühllich, M. and Mester, R. (1998). The role of total least squares in motion analysis. In *Fifth European Conference on Computer Vision (ECCV'98)*, pp. 305–321, Freiburg, Germany.
- Muja, M. and Lowe, D. G. (2009). Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Applications (VISAPP)*, Lisbon, Portugal.
- Mumford, D. and Shah, J. (1989). Optimal approximations by piecewise smooth functions and variational problems. *Comm. Pure Appl. Math.*, XLII(5):577–685.
- Munder, S. and Gavrila, D. M. (2006). An experimental study on pedestrian classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1863–1868.
- Mundy, J. L. (2006). Object recognition in the geometric era: A retrospective. In Ponce, J., Hebert, M., Schmid, C., and Zisserman, A. (eds), *Toward Category-Level Object Recognition*, pp. 3–28, Springer, New York.
- Mundy, J. L. and Zisserman, A. (eds). (1992). *Geometric Invariance in Computer Vision*. MIT Press, Cambridge, Massachusetts.
- Murase, H. and Nayar, S. K. (1995). Visual learning and recognition of 3-D objects from appearance. *International Journal of Computer Vision*, 14(1):5–24.
- Murphy, E. P. (2005). *A Testing Procedure to Characterize Color and Spatial Quality of Digital Cameras Used to Image Cultural Heritage*. Master's thesis, Rochester Institute of Technology.
- Murphy, K., Torralba, A., and Freeman, W. T. (2003). Using the forest to see the trees: A graphical model relating features, objects, and scenes. In *Advances in Neural Information Processing Systems*.
- Murphy-Chutorian, E. and Trivedi, M. M. (2009). Head pose estimation in computer vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):607–626.
- Murray, R. M., Li, Z. X., and Sastry, S. S. (1994). *A Mathematical Introduction to Robotic Manipulation*. CRC Press.
- Mutch, J. and Lowe, D. G. (2008). Object class recognition and localization using sparse features with limited receptive fields. *International Journal of Computer Vision*, 80(1):45–57.
- Nagel, H. H. (1986). Image sequences—ten (octal) years—from phenomenology towards a theoretical foundation. In *Eighth International Conference on Pattern Recognition (ICPR'86)*, pp. 1174–1185, Paris.
- Nagel, H.-H. and Enkelmann, W. (1986). An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(5):565–593.
- Nakamura, Y., Matsuura, T., Satoh, K., and Ohta, Y. (1996). Occlusion detectable stereo—occlusion patterns in camera matrix. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'96)*, pp. 371–378, San Francisco.
- Nakao, T., Kashitani, A., and Kaneyoshi, A. (1998). Scanning a document with a small camera attached to a mouse. In *IEEE Workshop on Applications of Computer Vision (WACV'98)*, pp. 63–68, Princeton.

- Nalwa, V. S. (1987). Edge-detector resolution improvement by image interpolation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(3):446–451.
- Nalwa, V. S. (1993). *A Guided Tour of Computer Vision*. Addison-Wesley, Reading, MA.
- Nalwa, V. S. and Binford, T. O. (1986). On detecting edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):699–714.
- Narasimhan, S. G. and Nayar, S. K. (2005). Enhancing resolution along multiple imaging dimensions using assorted pixels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):518–530.
- Narayanan, P., Rander, P., and Kanade, T. (1998). Constructing virtual worlds using dense stereo. In *Sixth International Conference on Computer Vision (ICCV'98)*, pp. 3–10, Bombay.
- Nayar, S., Watanabe, M., and Noguchi, M. (1995). Real-time focus range sensor. In *Fifth International Conference on Computer Vision (ICCV'95)*, pp. 995–1001, Cambridge, Massachusetts.
- Nayar, S. K. (2006). Computational cameras: Redefining the image. *Computer*, 39(8):30–38.
- Nayar, S. K. and Branzoi, V. (2003). Adaptive dynamic range imaging: Optical control of pixel exposures over space and time. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 1168–1175, Nice, France.
- Nayar, S. K. and Mitsunaga, T. (2000). High dynamic range imaging: Spatially varying pixel exposures. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2000)*, pp. 472–479, Hilton Head Island.
- Nayar, S. K. and Nakagawa, Y. (1994). Shape from focus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(8):824–831.
- Nayar, S. K., Ikeuchi, K., and Kanade, T. (1991). Shape from interreflections. *International Journal of Computer Vision*, 6(3):173–195.
- Nayar, S. K., Watanabe, M., and Noguchi, M. (1996). Real-time focus range sensor. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(12):1186–1198.
- Negahdaripour, S. (1998). Revised definition of optical flow: Integration of radiometric and geometric cues for dynamic scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(9):961–979.
- Nehab, D., Rusinkiewicz, S., Davis, J., and Ramamoorthi, R. (2005). Efficiently combining positions and normals for precise 3d geometry. *ACM Transactions on Graphics (Proc. SIGGRAPH 2005)*, 24(3):536–543.
- Nene, S. and Nayar, S. K. (1997). A simple algorithm for nearest neighbor search in high dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(9):989–1003.
- Nene, S. A., Nayar, S. K., and Murase, H. (1996). *Columbia Object Image Library (COIL-100)*. Technical Report CU-CS-006-96, Department of Computer Science, Columbia University.
- Netravali, A. and Robbins, J. (1979). Motion-compensated television coding: Part 1. *Bell System Tech.*, 58(3):631–670.
- Nevatia, R. (1977). A color edge detector and its use in scene segmentation. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-7(11):820–826.
- Nevatia, R. and Binford, T. (1977). Description and recognition of curved objects. *Artificial Intelligence*, 8:77–98.

- Ng, A. Y., Jordan, M. I., and Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, pp. 849–854.
- Ng, R. (2005). Fourier slice photography. *ACM Transactions on Graphics (Proc. SIGGRAPH 2005)*, 24(3):735–744.
- Ng, R., Levoy, M., Brédif, M., Duval, G., Horowitz, M., and Hanrahan, P. (2005). *Light Field Photography with a Hand-held Plenoptic Camera*. Technical Report CSTR 2005-02, Stanford University.
- Nielsen, M., Florack, L. M. J., and Deriche, R. (1997). Regularization, scale-space, and edge-detection filters. *Journal of Mathematical Imaging and Vision*, 7(4):291–307.
- Nielson, G. M. (1993). Scattered data modeling. *IEEE Computer Graphics and Applications*, 13(1):60–70.
- Nir, T., Bruckstein, A. M., and Kimmel, R. (2008). Over-parameterized variational optical flow. *International Journal of Computer Vision*, 76(2):205–216.
- Nishihara, H. K. (1984). Practical real-time imaging stereo matcher. *OptEng*, 23(5):536–545.
- Nistér, D. (2003). Preemptive RANSAC for live structure and motion estimation. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 199–206, Nice, France.
- Nistér, D. (2004). An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–777.
- Nistér, D. and Stewénius, H. (2006). Scalable recognition with a vocabulary tree. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2006)*, pp. 2161–2168, New York City, NY.
- Nistér, D. and Stewénius, H. (2008). Linear time maximally stable extremal regions. In *Tenth European Conference on Computer Vision (ECCV 2008)*, pp. 183–196, Marseilles.
- Nistér, D., Naroditsky, O., and Bergen, J. (2006). Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3–20.
- Noborio, H., Fukada, S., and Arimoto, S. (1988). Construction of the octree approximating three-dimensional objects by using multiple views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-10(6):769–782.
- Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer, New York, second edition.
- Nomura, Y., Zhang, L., and Nayar, S. K. (2007). Scene collages and flexible camera arrays. In *Eurographics Symposium on Rendering*.
- Nordström, N. (1990). Biased anisotropic diffusion: A unified regularization and diffusion approach to edge detection. *Image and Vision Computing*, 8(4):318–327.
- Nowak, E., Jurie, F., and Triggs, B. (2006). Sampling strategies for bag-of-features image classification. In *Ninth European Conference on Computer Vision (ECCV 2006)*, pp. 490–503.
- Obdržálek, S. and Matas, J. (2006). Object recognition using local affine frames on maximally stable extremal regions. In Ponce, J., Hebert, M., Schmid, C., and Zisserman, A. (eds), *Toward Category-Level Object Recognition*, pp. 83–104, Springer, New York.
- Oh, B. M., Chen, M., Dorsey, J., and Durand, F. (2001). Image-based modeling and photo editing. In *ACM SIGGRAPH 2001 Conference Proceedings*, pp. 433–442.
- Ohlander, R., Price, K., and Reddy, D. R. (1978). Picture segmentation using a recursive region splitting method. *Computer Graphics and Image Processing*, 8(3):313–333.

- Ohta, Y. and Kanade, T. (1985). Stereo by intra- and inter-scanline search using dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7(2):139–154.
- Otake, Y., Belyaev, A., Alexa, M., Turk, G., and Seidel, H.-P. (2003). Multi-level partition of unity implicits. *ACM Transactions on Graphics (Proc. SIGGRAPH 2003)*, 22(3):463–470.
- Okutomi, M. and Kanade, T. (1992). A locally adaptive window for signal matching. *International Journal of Computer Vision*, 7(2):143–162.
- Okutomi, M. and Kanade, T. (1993). A multiple baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4):353–363.
- Okutomi, M. and Kanade, T. (1994). A stereo matching algorithm with an adaptive window: Theory and experiment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):920–932.
- Oliensis, J. (2005). The least-squares error for structure from infinitesimal motion. *International Journal of Computer Vision*, 61(3):259–299.
- Oliensis, J. and Hartley, R. (2007). Iterative extensions of the Sturm/Triggs algorithm: Convergence and nonconvergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2217–2233.
- Oliva, A. and Torralba, A. (2001). Modeling the shape of the scene: a holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175.
- Oliva, A. and Torralba, A. (2007). The role of context in object recognition. *Trends in Cognitive Sciences*, 11(12):520–527.
- Olsson, C., Eriksson, A. P., and Kahl, F. (2008). Improved spectral relaxation methods for binary quadratic optimization problems. *Computer Vision and Image Understanding*, 112(1):3–13.
- Omer, I. and Werman, M. (2004). Color lines: Image specific color representation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2004)*, pp. 946–953, Washington, DC.
- Ong, E.-J., Micilotta, A. S., Bowden, R., and Hilton, A. (2006). Viewpoint invariant exemplar-based 3D human tracking. *Computer Vision and Image Understanding*, 104(2-3):178–189.
- Opelt, A., Pinz, A., and Zisserman, A. (2006). A boundary-fragment-model for object detection. In *Ninth European Conference on Computer Vision (ECCV 2006)*, pp. 575–588.
- Opelt, A., Pinz, A., Fussenegger, M., and Auer, P. (2006). Generic object recognition with boosting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3):614–641.
- OpenGL-ARB. (1997). *OpenGL Reference Manual: The Official Reference Document to OpenGL, Version 1.1*. Addison-Wesley, Reading, MA, 2nd edition.
- Oppenheim, A. V. and Schafer, R. W. (1996). *Signals and Systems*. Prentice Hall, Englewood Cliffs, New Jersey, 2nd edition.
- Oppenheim, A. V., Schafer, R. W., and Buck, J. R. (1999). *Discrete-Time Signal Processing*. Prentice Hall, Englewood Cliffs, New Jersey, 2nd edition.
- Oren, M. and Nayar, S. (1997). A theory of specular surface geometry. *International Journal of Computer Vision*, 24(2):105–124.
- O'Rourke, J. and Badler, N. I. (1980). Model-based image analysis of human motion using constraint propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(6):522–536.
- Osher, S. and Paragios, N. (eds.). (2003). *Geometric Level Set Methods in Imaging, Vision, and Graphics*, Springer.

- Osuna, E., Freund, R., and Girosi, F. (1997). Training support vector machines: An application to face detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pp. 130–136, San Juan, Puerto Rico.
- O'Toole, A. J., Jiang, F., Roark, D., and Abdi, H. (2006). Predicting human face recognition. In Zhao, W.-Y. and Chellappa, R. (eds), *Face Processing: Advanced Methods and Models*, Elsevier.
- O'Toole, A. J., Phillips, P. J., Jiang, F., Ayyad, J., Pénard, N., and Abdi, H. (2009). Face recognition algorithms surpass humans matching faces over changes in illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(9):1642–1646.
- Ott, M., Lewis, J. P., and Cox, I. J. (1993). Teleconferencing eye contact using a virtual camera. In *INTERACT'93 and CHI'93 conference companion on Human factors in computing systems*, pp. 109–110, Amsterdam.
- Otte, M. and Nagel, H.-H. (1994). Optical flow estimation: advances and comparisons. In *Third European Conference on Computer Vision (ECCV'94)*, pp. 51–60, Stockholm, Sweden.
- Oztireli, C., Guennebaud, G., and Gross, M. (2008). Feature preserving point set surfaces. *Computer Graphics Forum*, 28(2):493–501.
- Özuyşal, M., Calonder, M., Lepetit, V., and Fua, P. (2010). Fast keypoint recognition using random ferns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3).
- Pagliaroni, D. W. (1992). Distance transforms: Properties and machine vision applications. *Graphical Models and Image Processing*, 54(1):56–74.
- Pal, C., Szeliski, R., Uyttendaele, M., and Jojic, N. (2004). Probability models for high dynamic range imaging. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2004)*, pp. 173–180, Washington, DC.
- Palmer, S. E. (1999). *Vision Science: Photons to Phenomenology*. The MIT Press, Cambridge, Massachusetts.
- Pankanti, S., Bolle, R. M., and Jain, A. K. (2000). Biometrics: The future of identification. *Computer*, 21(2):46–49.
- Papageorgiou, C. and Poggio, T. (2000). A trainable system for object detection. *International Journal of Computer Vision*, 38(1):15–33.
- Papandreou, G. and Maragos, P. (2008). Adaptive and constrained algorithms for inverse compositional active appearance model fitting. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Papenberg, N., Bruhn, A., Brox, T., Didas, S., and Weickert, J. (2006). Highly accurate optic flow computation with theoretically justified warping. *International Journal of Computer Vision*, 67(2):141–158.
- Papert, S. (1966). *The Summer Vision Project*. Technical Report AIM-100, Artificial Intelligence Group, Massachusetts Institute of Technology. <http://hdl.handle.net/1721.1/6125>.
- Paragios, N. and Deriche, R. (2000). Geodesic active contours and level sets for the detection and tracking of moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(3):266–280.
- Paragios, N. and Sgallari, F. (2009). Special issue on scale space and variational methods in computer vision. *International Journal of Computer Vision*, 84(2).
- Paragios, N., Faugeras, O. D., Chan, T., and Schnörr, C. (eds). (2005). *Third International Workshop on Variational, Geometric, and Level Set Methods in Computer Vision (VLSM 2005)*, Springer.

- Paris, S. and Durand, F. (2006). A fast approximation of the bilateral filter using a signal processing approach. In *Ninth European Conference on Computer Vision (ECCV 2006)*, pp. 568–580.
- Paris, S. and Durand, F. (2007). A topological approach to hierarchical segmentation using mean shift. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Paris, S., Kornprobst, P., Tumblin, J., and Durand, F. (2008). Bilateral filtering: Theory and applications. *Foundations and Trends in Computer Graphics and Computer Vision*, 4(1):1–73.
- Park, M., Brocklehurst, K., Collins, R. T., and Liu, Y. (2009). Deformed lattice detection in real-world images using mean-shift belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(10):1804–1816.
- Park, S. C., Park, M. K., and Kang, M. G. (2003). Super-resolution image reconstruction: A technical overview. *IEEE Signal Processing Magazine*, 20:21–36.
- Parke, F. I. and Waters, K. (1996). *Computer Facial Animation*. A K Peters, Wellesley, Massachusetts.
- Parker, J. A., Kenyon, R. V., and Troxel, D. E. (1983). Comparison of interpolating methods for image resampling. *IEEE Transactions on Medical Imaging*, MI-2(1):31–39.
- Pattanaik, S. N., Ferwerda, J. A., Fairchild, M. D., and Greenberg, D. P. (1998). A multiscale model of adaptation and spatial vision for realistic image display. In *ACM SIGGRAPH 1998 Conference Proceedings*, pp. 287–298, Orlando.
- Pauly, M., Keiser, R., Kobbett, L. P., and Gross, M. (2003). Shape modeling with point-sampled geometry. *ACM Transactions on Graphics (Proc. SIGGRAPH 2003)*, 21(3):641–650.
- Pavlidis, T. (1977). *Structural Pattern Recognition*. Springer-Verlag, Berlin; New York.
- Pavlidis, T. and Liow, Y.-T. (1990). Integrating region growing and edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(3):225–233.
- Pavlović, V., Sharma, R., and Huang, T. S. (1997). Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):677–695.
- Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers, Los Altos.
- Peleg, R., Ben-Ezra, M., and Pritch, Y. (2001). Omnistereo: Panoramic stereo imaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(3):279–290.
- Peleg, S. (1981). Elimination of seams from photomosaics. *Computer Vision, Graphics, and Image Processing*, 16(1):1206–1210.
- Peleg, S. and Herman, J. (1997). Panoramic mosaics by manifold projection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pp. 338–343, San Juan, Puerto Rico.
- Peleg, S. and Rav-Acha, A. (2006). Lucas-Kanade without iterative warping. In *International Conference on Image Processing (ICIP-2006)*, pp. 1097–1100, Atlanta.
- Peleg, S., Rousso, B., Rav-Acha, A., and Zomet, A. (2000). Mosaicing on adaptive manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1144–1154.
- Penev, P. and Atick, J. (1996). Local feature analysis: A general statistical theory for object representation. *Network Computation and Neural Systems*, 7:477–500.

- Pentland, A. P. (1984). Local shading analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(2):170–179.
- Pentland, A. P. (1986). Perceptual organization and the representation of natural form. *Artificial Intelligence*, 28(3):293–331.
- Pentland, A. P. (1987). A new sense for depth of field. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(4):523–531.
- Pentland, A. P. (1994). Interpolation using wavelet bases. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(4):410–414.
- Pérez, P., Blake, A., and Gangnet, M. (2001). JetStream: Probabilistic contour extraction with particles. In *Eighth International Conference on Computer Vision (ICCV 2001)*, pp. 524–531, Vancouver, Canada.
- Pérez, P., Gangnet, M., and Blake, A. (2003). Poisson image editing. *ACM Transactions on Graphics (Proc. SIGGRAPH 2003)*, 22(3):313–318.
- Perona, P. (1995). Deformable kernels for early vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):488–499.
- Perona, P. and Malik, J. (1990a). Detecting and localizing edges composed of steps, peaks and roofs. In *Third International Conference on Computer Vision (ICCV'90)*, pp. 52–57, Osaka, Japan.
- Perona, P. and Malik, J. (1990b). Scale space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639.
- Peters, J. and Reif, U. (2008). *Subdivision Surfaces*. Springer.
- Petschnigg, G., Agrawala, M., Hoppe, H., Szeliski, R., Cohen, M., and Toyama, K. (2004). Digital photography with flash and no-flash image pairs. *ACM Transactions on Graphics (Proc. SIGGRAPH 2004)*, 23(3):664–672.
- Pfister, H., Zwicker, M., van Baar, J., and Gross, M. (2000). Surfels: Surface elements as rendering primitives. In *ACM SIGGRAPH 2000 Conference Proceedings*, pp. 335–342.
- Pflugfelder, R. (2008). *Self-calibrating Cameras in Video Surveillance*. Ph.D. thesis, Graz University of Technology.
- Philbin, J. and Zisserman, A. (2008). Object mining using a matching graph on very large image collections. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Bhubaneswar, India.
- Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2007). Object retrieval with large vocabularies and fast spatial matching. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Philbin, J., Chum, O., Sivic, J., Isard, M., and Zisserman, A. (2008). Lost in quantization: Improving particular object retrieval in large scale image databases. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Phillips, P. J., Moon, H., Rizvi, S. A., and Rauss, P. J. (2000). The FERET evaluation methodology for face recognition algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1090–1104.
- Phillips, P. J., Scruggs, W. T., O'Toole, A. J., Flynn, P. J., Bowyer, K. W. et al. (2010). FRVT 2006 and ICE 2006 large-scale experimental results. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):831–846.

- Phong, B. T. (1975). Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317.
- Pickup, L. C. (2007). *Machine Learning in Multi-frame Image Super-resolution*. Ph.D. thesis, University of Oxford.
- Pickup, L. C. and Zisserman, A. (2009). Automatic retrieval of visual continuity errors in movies. In *ACM International Conference on Image and Video Retrieval*, Santorini, Greece.
- Pickup, L. C., Capel, D. P., Roberts, S. J., and Zisserman, A. (2007). Overcoming registration uncertainty in image super-resolution: Maximize or marginalize? *EURASIP Journal on Advances in Signal Processing*, 2010(Article ID 23565).
- Pickup, L. C., Capel, D. P., Roberts, S. J., and Zisserman, A. (2009). Bayesian methods for image super-resolution. *The Computer Journal*, 52.
- Pighin, F., Szeliski, R., and Salesin, D. H. (2002). Modeling and animating realistic faces from images. *International Journal of Computer Vision*, 50(2):143–169.
- Pighin, F., Hecker, J., Lischinski, D., Salesin, D. H., and Szeliski, R. (1998). Synthesizing realistic facial expressions from photographs. In *ACM SIGGRAPH 1998 Conference Proceedings*, pp. 75–84, Orlando.
- Pilet, J., Lepetit, V., and Fua, P. (2008). Fast non-rigid surface detection, registration, and realistic augmentation. *International Journal of Computer Vision*, 76(2).
- Pinz, A. (2005). Object categorization. *Foundations and Trends in Computer Graphics and Computer Vision*, 1(4):255–353.
- Pizer, S. M., Amburn, E. P., Austin, J. D., Cromartie, R., Geselowitz, A. et al. (1987). Adaptive histogram equalization and its variations. *Computer Vision, Graphics, and Image Processing*, 39(3):355–368.
- Platel, B., Balmachnova, E., Florack, L., and ter Haar Romeny, B. (2006). Top-points as interest points for image matching. In *Ninth European Conference on Computer Vision (ECCV 2006)*, pp. 418–429.
- Platt, J. C. (2000). Optimal filtering for patterned displays. *IEEE Signal Processing Letters*, 7(7):179–180.
- Pock, T., Unger, M., Cremers, D., and Bischof, H. (2008). Fast and exact solution of total variation models on the GPU. In *CVPR 2008 Workshop on Visual Computer Vision on GPUs (CVGPU)*, Anchorage, AK.
- Poelman, C. J. and Kanade, T. (1997). A paraperspective factorization method for shape and motion recovery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):206–218.
- Poggio, T. and Koch, C. (1985). Ill-posed problems in early vision: from computational theory to analogue networks. *Proceedings of the Royal Society of London*, B 226:303–323.
- Poggio, T., Gamble, E., and Little, J. (1988). Parallel integration of vision modules. *Science*, 242(4877):436–440.
- Poggio, T., Torre, V., and Koch, C. (1985). Computational vision and regularization theory. *Nature*, 317(6035):314–319.
- Poggio, T., Little, J., Gamble, E., Gillet, W., Geiger, D. et al. (1988). The MIT vision machine. In *Image Understanding Workshop*, pp. 177–198, Boston.
- Polana, R. and Nelson, R. C. (1997). Detection and recognition of periodic, nonrigid motion. *International Journal of Computer Vision*, 23(3):261–282.
- Pollard, S. B., Mayhew, J. E. W., and Frisby, J. P. (1985). PMF: A stereo correspondence algorithm using a disparity gradient limit. *Perception*, 14:449–470.

- Pollefeys, M. and Van Gool, L. (2002). From images to 3D models. *Communications of the ACM*, 45(7):50–55.
- Pollefeys, M., Nistér, D., Frahm, J.-M., Akbarzadeh, A., Mordohai, P. et al. (2008). Detailed real-time urban 3D reconstruction from video. *International Journal of Computer Vision*, 78(2-3):143–167.
- Ponce, J., Hebert, M., Schmid, C., and Zisserman, A. (eds). (2006). *Toward Category-Level Object Recognition*, Springer, New York.
- Ponce, J., Berg, T., Everingham, M., Forsyth, D., Hebert, M. et al. (2006). Dataset issues in object recognition. In Ponce, J., Hebert, M., Schmid, C., and Zisserman, A. (eds), *Toward Category-Level Object Recognition*, pp. 29–48, Springer, New York.
- Pons, J.-P., Keriven, R., and Faugeras, O. (2005). Modelling dynamic scenes by registering multi-view image sequences. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 822–827, San Diego, CA.
- Pons, J.-P., Keriven, R., and Faugeras, O. (2007). Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. *International Journal of Computer Vision*, 72(2):179–193.
- Porter, T. and Duff, T. (1984). Compositing digital images. *Computer Graphics (SIGGRAPH '84)*, 18(3):253–259.
- Portilla, J. and Simoncelli, E. P. (2000). A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40(1):49–71.
- Portilla, J., Strela, V., Wainwright, M., and Simoncelli, E. P. (2003). Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Transactions on Image Processing*, 12(11):1338–1351.
- Potetz, B. and Lee, T. S. (2008). Efficient belief propagation for higher-order cliques using linear constraint nodes. *Computer Vision and Image Understanding*, 112(1):39–54.
- Potmesil, M. (1987). Generating octree models of 3D objects from their silhouettes in a sequence of images. *Computer Vision, Graphics, and Image Processing*, 40:1–29.
- Pratt, W. K. (2007). *Digital Image Processing*. Wiley-Interscience, Hoboken, NJ, 4th edition.
- Prazdny, K. (1985). Detection of binocular disparities. *Biological Cybernetics*, 52:93–99.
- Pritchett, P. and Zisserman, A. (1998). Wide baseline stereo matching. In *Sixth International Conference on Computer Vision (ICCV'98)*, pp. 754–760, Bombay.
- Proesmans, M., Van Gool, L., and Defoort, F. (1998). Reading between the lines – a method for extracting dynamic 3D with texture. In *Sixth International Conference on Computer Vision (ICCV'98)*, pp. 1081–1086, Bombay.
- Protter, M. and Elad, M. (2009). Super resolution with probabilistic motion estimation. *IEEE Transactions on Image Processing*, 18(8):1899–1904.
- Pullen, K. and Bregler, C. (2002). Motion capture assisted animation: texturing and synthesis. *ACM Transactions on Graphics*, 21(3):501–508.
- Pulli, K. (1999). Multiview registration for large data sets. In *Second International Conference on 3D Digital Imaging and Modeling (3DIM'99)*, pp. 160–168, Ottawa, Canada.
- Pulli, K., Abi-Rached, H., Duchamp, T., Shapiro, L., and Stuetzle, W. (1998). Acquisition and visualization of colored 3D objects. In *International Conference on Pattern Recognition (ICPR'98)*, pp. 11–15.
- Quack, T., Leibe, B., and Van Gool, L. (2008). World-scale mining of objects and events from community photo collections. In *Conference on Image and Video Retrieval*, pp. 47–56, Niagara Falls.

- Quam, L. H. (1984). Hierarchical warp stereo. In *Image Understanding Workshop*, pp. 149–155, New Orleans.
- Quan, L. and Lan, Z. (1999). Linear N-point camera pose determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):774–780.
- Quan, L. and Mohr, R. (1989). Determining perspective structures using hierarchical Hough transform. *Pattern Recognition Letters*, 9(4):279–286.
- Rabinovich, A., Vedaldi, A., Galleguillos, C., Wiewiora, E., and Belongie, S. (2007). Objects in context. In *Eleventh International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil.
- Rademacher, P. and Bishop, G. (1998). Multiple-center-of-projection images. In *ACM SIGGRAPH 1998 Conference Proceedings*, pp. 199–206, Orlando.
- Raginsky, M. and Lazebnik, S. (2009). Locality-sensitive binary codes from shift-invariant kernels. In *Advances in Neural Information Processing Systems*.
- Raman, S. and Chaudhuri, S. (2007). A matte-less, variational approach to automatic scene compositing. In *Eleventh International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil.
- Raman, S. and Chaudhuri, S. (2009). Bilateral filter based compositing for variable exposure photography. In *Proceedings of Eurographics 2009*.
- Ramanan, D. and Baker, S. (2009). Local distance functions: A taxonomy, new algorithms, and an evaluation. In *Twelfth International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan.
- Ramanan, D., Forsyth, D., and Zisserman, A. (2005). Strike a pose: Tracking people by finding stylized poses. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 271–278, San Diego, CA.
- Ramanarayanan, G. and Bala, K. (2007). Constrained texture synthesis via energy minimization. *IEEE Transactions on Visualization and Computer Graphics*, 13(1):167–178.
- Ramer, U. (1972). An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3):244–256.
- Ramnath, K., Koterba, S., Xiao, J., Hu, C., Matthews, I., Baker, S., Cohn, J., and Kanade, T. (2008). Multi-view AAM fitting and construction. *International Journal of Computer Vision*, 76(2):183–204.
- Raskar, R. and Tumblin, J. (2010). *Computational Photography: Mastering New Techniques for Lenses, Lighting, and Sensors*. A K Peters, Wellesley, Massachusetts.
- Raskar, R., Tan, K.-H., Feris, R., Yu, J., and Turk, M. (2004). Non-photorealistic camera: Depth edge detection and stylized rendering using multi-flash imaging. *ACM Transactions on Graphics*, 23(3):679–688.
- Rav-Acha, A., Kohli, P., Fitzgibbon, A., and Rother, C. (2008). Unwrap mosaics: A new representation for video editing. *ACM Transactions on Graphics*, 27(3).
- Rav-Acha, A., Pritch, Y., Lischinski, D., and Peleg, S. (2005). Dynamosaics: Video mosaics with non-chronological time. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 58–65, San Diego, CA.
- Ravikumar, P., Agarwal, A., and Wainwright, M. J. (2008). Message-passing for graph-structured linear programs: Proximal projections, convergence and rounding schemes. In *International Conference on Machine Learning*, pp. 800–807.
- Ray, S. F. (2002). *Applied Photographic Optics*. Focal Press, Oxford, 3rd edition.

- Rehg, J. and Kanade, T. (1994). Visual tracking of high DOF articulated structures: an application to human hand tracking. In *Third European Conference on Computer Vision (ECCV'94)*, pp. 35–46, Stockholm, Sweden.
- Rehg, J. and Witkin, A. (1991). Visual tracking with deformation models. In *IEEE International Conference on Robotics and Automation*, pp. 844–850, Sacramento.
- Rehg, J., Morris, D. D., and Kanade, T. (2003). Ambiguities in visual tracking of articulated objects using two- and three-dimensional models. *International Journal of Robotics Research*, 22(6):393–418.
- Reichenbach, S. E., Park, S. K., and Narayanswamy, R. (1991). Characterizing digital image acquisition devices. *Optical Engineering*, 30(2):170–177.
- Reinhard, E., Stark, M., Shirley, P., and Ferwerda, J. (2002). Photographic tone reproduction for digital images. *ACM Transactions on Graphics (Proc. SIGGRAPH 2002)*, 21(3):267–276.
- Reinhard, E., Ward, G., Pattanaik, S., andDebevec, P. (2005). *High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting*. Morgan Kaufmann.
- Rhemann, C., Rother, C., and Gelautz, M. (2008). Improving color modeling for alpha matting. In *British Machine Vision Conference (BMVC 2008)*, Leeds.
- Rhemann, C., Rother, C., Rav-Acha, A., and Sharp, T. (2008). High resolution matting via interactive trimap segmentation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Rhemann, C., Rother, C., Wang, J., Gelautz, M., Kohli, P., and Rott, P. (2009). A perceptually motivated online benchmark for image matting. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, Miami Beach, FL.
- Richardson, I. E. G. (2003). *H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia*. Wiley.
- Rioual, O. and Vetterli, M. (1991). Wavelets and signal processing. *IEEE Signal Processing Magazine*, 8(4):14–38.
- Rioux, M. and Bird, T. (1993). White laser, synced scan. *IEEE Computer Graphics and Applications*, 13(3):15–17.
- Rioux, M., Bechthold, G., Taylor, D., and Duggan, M. (1987). Design of a large depth of view three-dimensional camera for robot vision. *Optical Engineering*, 26(12):1245–1250.
- Riseman, E. M. and Arbib, M. A. (1977). Computational techniques in the visual segmentation of static scenes. *Computer Graphics and Image Processing*, 6(3):221–276.
- Ritter, G. X. and Wilson, J. N. (2000). *Handbook of Computer Vision Algorithms in Image Algebra*. CRC Press, Boca Raton, 2nd edition.
- Robert, C. P. (2007). *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation*. Springer-Verlag, New York.
- Roberts, L. G. (1965). Machine perception of three-dimensional solids. In Tippett, J. T., Borkowitz, D. A., Clapp, L. C., Koester, C. J., and Vanderburgh Jr., A. (eds), *Optical and Electro-Optical Information Processing*, pp. 159–197, MIT Press, Cambridge, Massachusetts.
- Robertson, D. and Cipolla, R. (2004). An image-based system for urban navigation. In *British Machine Vision Conference*, pp. 656–665, Kingston.

- Robertson, D. P. and Cipolla, R. (2002). Building architectural models from many views using map constraints. In *Seventh European Conference on Computer Vision (ECCV 2002)*, pp. 155–169, Copenhagen.
- Robertson, D. P. and Cipolla, R. (2009). Architectural modelling. In Varga, M. (ed.), *Practical Image Processing and Computer Vision*, John Wiley.
- Robertson, N. and Reid, I. (2006). A general method for human activity recognition in video. *Computer Vision and Image Understanding*, 104(2-3):232–248.
- Roble, D. (1999). Vision in film and special effects. *Computer Graphics*, 33(4):58–60.
- Roble, D. and Zafar, N. B. (2009). Don't trust your eyes: cutting-edge visual effects. *Computer*, 42(7):35–41.
- Rogez, G., Rihan, J., Ramalingam, S., Orrite, C., and Torr, P. H. S. (2008). Randomized trees for human pose detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Rogmans, S., Lu, J., Bekaert, P., and Lafruit, G. (2009). Real-time stereo-based views synthesis algorithms: A unified framework and evaluation on commodity GPUs. *Signal Processing: Image Communication*, 24:49–64.
- Rohr, K. (1994). Towards model-based recognition of human movements in image sequences. *Computer Vision, Graphics, and Image Processing*, 59(1):94–115.
- Román, A. and Lensch, H. P. A. (2006). Automatic multiperspective images. In *Eurographics Symposium on Rendering*, pp. 83–92.
- Román, A., Garg, G., and Levoy, M. (2004). Interactive design of multi-perspective images for visualizing urban landscapes. In *IEEE Visualization 2004*, pp. 537–544, Minneapolis.
- Romdhani, S. and Vetter, T. (2003). Efficient, robust and accurate fitting of a 3D morphable model. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 59–66, Nice, France.
- Romdhani, S., Torr, P. H. S., Schölkopf, B., and Blake, A. (2001). Computationally efficient face detection. In *Eighth International Conference on Computer Vision (ICCV 2001)*, pp. 695–700, Vancouver, Canada.
- Rosales, R. and Sclaroff, S. (2000). Inferring body pose without tracking body parts. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2000)*, pp. 721–727, Hilton Head Island.
- Rosenfeld, A. (1980). Quadtrees and pyramids for pattern recognition and image processing. In *Fifth International Conference on Pattern Recognition (ICPR'80)*, pp. 802–809, Miami Beach.
- Rosenfeld, A. (ed.). (1984). *Multiresolution Image Processing and Analysis*, Springer-Verlag, New York.
- Rosenfeld, A. and Davis, L. S. (1979). Image segmentation and image models. *Proceedings of the IEEE*, 67(5):764–772.
- Rosenfeld, A. and Kak, A. C. (1976). *Digital Picture Processing*. Academic Press, New York.
- Rosenfeld, A. and Pfaltz, J. L. (1966). Sequential operations in digital picture processing. *Journal of the ACM*, 13(4):471–494.
- Rosenfeld, A., Hummel, R. A., and Zucker, S. W. (1976). Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6:420–433.
- Rosten, E. and Drummond, T. (2005). Fusing points and lines for high performance tracking. In *Tenth International Conference on Computer Vision (ICCV 2005)*, pp. 1508–1515, Beijing, China.
- Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *Ninth European Conference on Computer Vision (ECCV 2006)*, pp. 430–443.

- Roth, S. and Black, M. J. (2007a). On the spatial statistics of optical flow. *International Journal of Computer Vision*, 74(1):33–50.
- Roth, S. and Black, M. J. (2007b). Steerable random fields. In *Eleventh International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil.
- Roth, S. and Black, M. J. (2009). Fields of experts. *International Journal of Computer Vision*, 82(2):205–229.
- Rother, C. (2002). A new approach for vanishing point detection in architectural environments. *Image and Vision Computing*, 20(9-10):647–656.
- Rother, C. (2003). Linear multi-view reconstruction of points, lines, planes and cameras using a reference plane. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 1210–1217, Nice, France.
- Rother, C. and Carlsson, S. (2002). Linear multi view reconstruction and camera recovery using a reference plane. *International Journal of Computer Vision*, 49(2/3):117–141.
- Rother, C., Kolmogorov, V., and Blake, A. (2004). “GrabCut”—interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (Proc. SIGGRAPH 2004)*, 23(3):309–314.
- Rother, C., Bordeaux, L., Hamadi, Y., and Blake, A. (2006). Autocollage. *ACM Transactions on Graphics*, 25(3):847–852.
- Rother, C., Kohli, P., Feng, W., and Jia, J. (2009). Minimizing sparse higher order energy functions of discrete variables. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, Miami Beach, FL.
- Rother, C., Kolmogorov, V., Lempitsky, V., and Szummer, M. (2007). Optimizing binary MRFs via extended roof duality. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Rother, C., Kumar, S., Kolmogorov, V., and Blake, A. (2005). Digital tapestry. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’2005)*, pp. 589–596, San Diego, CA.
- Rothganger, F., Lazebnik, S., Schmid, C., and Ponce, J. (2006). 3D object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *International Journal of Computer Vision*, 66(3):231–259.
- Rousseeuw, P. J. (1984). Least median of squares regression. *Journal of the American Statistical Association*, 79:871–880.
- Rousseeuw, P. J. and Leroy, A. M. (1987). *Robust Regression and Outlier Detection*. Wiley, New York.
- Rousson, M. and Paragios, N. (2008). Prior knowledge, level set representations, and visual grouping. *International Journal of Computer Vision*, 76(3):231–243.
- Roweis, S. (1998). EM algorithms for PCA and SPCA. In *Advances in Neural Information Processing Systems*, pp. 626–632.
- Rowland, D. A. and Perrett, D. I. (1995). Manipulating facial appearance through shape and color. *IEEE Computer Graphics and Applications*, 15(5):70–76.
- Rowley, H. A., Baluja, S., and Kanade, T. (1998a). Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38.
- Rowley, H. A., Baluja, S., and Kanade, T. (1998b). Rotation invariant neural network-based face detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’98)*, pp. 38–44, Santa Barbara.

- Roy, S. and Cox, I. J. (1998). A maximum-flow formulation of the N-camera stereo correspondence problem. In *Sixth International Conference on Computer Vision (ICCV'98)*, pp. 492–499, Bombay.
- Rozenfeld, S., Shimshoni, I., and Lindenbaum, M. (2007). Dense mirroring surface recovery from 1d homographies and sparse correspondences. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Rubner, Y., Tomasi, C., and Guibas, L. J. (2000). The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E., McClelland, J. L., and the PDP research group (eds), *Parallel distributed processing: Explorations in the microstructure of cognition*, pp. 318–362, Bradford Books, Cambridge, Massachusetts.
- Rusinkiewicz, S. and Levoy, M. (2000). Qsplat: A multiresolution point rendering system for large meshes. In *ACM SIGGRAPH 2000 Conference Proceedings*, pp. 343–352.
- Russ, J. C. (2007). *The Image Processing Handbook*. CRC Press, Boca Raton, 5th edition.
- Russell, B., Efros, A., Sivic, J., Freeman, W., and Zisserman, A. (2006). Using multiple segmentations to discover objects and their extent in image collections. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2006)*, pp. 1605–1612, New York City, NY.
- Russell, B. C., Torralba, A., Murphy, K. P., and Freeman, W. T. (2008). LabelMe: A database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1-3):157–173.
- Russell, B. C., Torralba, A., Liu, C., Fergus, R., and Freeman, W. T. (2007). Object recognition by scene alignment. In *Advances in Neural Information Processing Systems*.
- Ruzon, M. A. and Tomasi, C. (2000). Alpha estimation in natural images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2000)*, pp. 18–25, Hilton Head Island.
- Ruzon, M. A. and Tomasi, C. (2001). Edge, junction, and corner detection using color distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1281–1295.
- Ryan, T. W., Gray, R. T., and Hunt, B. R. (1980). Prediction of correlation errors in stereo-pair images. *Optical Engineering*, 19(3):312–322.
- Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, second edition.
- Saint-Marc, P., Chen, J. S., and Medioni, G. (1991). Adaptive smoothing: A general tool for early vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):514–529.
- Saito, H. and Kanade, T. (1999). Shape reconstruction in projective grid space from large number of images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99)*, pp. 49–54, Fort Collins.
- Samet, H. (1989). *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Reading, Massachusetts.
- Sander, P. T. and Zucker, S. W. (1990). Inferring surface trace and differential structure from 3-D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9):833–854.
- Sapiro, G. (2001). *Geometric Partial Differential Equations and Image Analysis*. Cambridge University Press.
- Sato, Y. and Ikeuchi, K. (1996). Reflectance analysis for 3D computer graphics model generation. *Graphical Models and Image Processing*, 58(5):437–451.

- Sato, Y., Wheeler, M., and Ikeuchi, K. (1997). Object shape and reflectance modeling from observation. In *ACM SIGGRAPH 1997 Conference Proceedings*, pp. 379–387, Los Angeles.
- Savarese, S. and Fei-Fei, L. (2007). 3D generic object categorization, localization and pose estimation. In *Eleventh International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil.
- Savarese, S. and Fei-Fei, L. (2008). View synthesis for recognizing unseen poses of object classes. In *Tenth European Conference on Computer Vision (ECCV 2008)*, pp. 602–615, Marseilles.
- Savarese, S., Chen, M., and Perona, P. (2005). Local shape from mirror reflections. *International Journal of Computer Vision*, 64(1):31–67.
- Savarese, S., Andreetto, M., Rushmeier, H. E., Bernardini, F., and Perona, P. (2007). 3D reconstruction by shadow carving: Theory and practical evaluation. *International Journal of Computer Vision*, 71(3):305–336.
- Sawhney, H. S. (1994). Simplifying motion and structure analysis using planar parallax and image warping. In *Twelfth International Conference on Pattern Recognition (ICPR'94)*, pp. 403–408, Jerusalem, Israel.
- Sawhney, H. S. and Ayer, S. (1996). Compact representation of videos through dominant multiple motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):814–830.
- Sawhney, H. S. and Hanson, A. R. (1991). Identification and 3D description of ‘shallow’ environmental structure over a sequence of images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'91)*, pp. 179–185, Maui, Hawaii.
- Sawhney, H. S. and Kumar, R. (1999). True multi-image alignment and its application to mosaicing and lens distortion correction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(3):235–243.
- Sawhney, H. S., Kumar, R., Gendel, G., Bergen, J., Dixon, D., and Paragano, V. (1998). VideoBrush: Experiences with consumer video mosaicing. In *IEEE Workshop on Applications of Computer Vision (WACV'98)*, pp. 56–62, Princeton.
- Sawhney, H. S., Arpa, A., Kumar, R., Samarasekera, S., Aggarwal, M., Hsu, S., Nister, D., and Hanna, K. (2002). Video flashlights: real time rendering of multiple videos for immersive model visualization. In *Proceedings of the 13th Eurographics Workshop on Rendering*, pp. 157–168, Pisa, Italy.
- Saxena, A., Sun, M., and Ng, A. Y. (2009). Make3D: Learning 3D scene structure from a single still image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):824–840.
- Schaffalitzky, F. and Zisserman, A. (2000). Planar grouping for automatic detection of vanishing lines and points. *Image and Vision Computing*, 18:647–658.
- Schaffalitzky, F. and Zisserman, A. (2002). Multi-view matching for unordered image sets, or “How do I organize my holiday snaps?”. In *Seventh European Conference on Computer Vision (ECCV 2002)*, pp. 414–431, Copenhagen.
- Scharr, H., Black, M. J., and Haussecker, H. W. (2003). Image statistics and anisotropic diffusion. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 840–847, Nice, France.
- Scharstein, D. (1994). Matching images by comparing their gradient fields. In *Twelfth International Conference on Pattern Recognition (ICPR'94)*, pp. 572–575, Jerusalem, Israel.
- Scharstein, D. (1999). *View Synthesis Using Stereo Vision*. Volume 1583, Springer-Verlag.
- Scharstein, D. and Pal, C. (2007). Learning conditional random fields for stereo. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.

- Scharstein, D. and Szeliski, R. (1998). Stereo matching with nonlinear diffusion. *International Journal of Computer Vision*, 28(2):155–174.
- Scharstein, D. and Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1):7–42.
- Scharstein, D. and Szeliski, R. (2003). High-accuracy stereo depth maps using structured light. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2003)*, pp. 195–202, Madison, WI.
- Schechner, Y. Y., Nayar, S. K., and Belhumeur, P. N. (2009). Multiplexing for optimal lighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8):1339–1354.
- Schindler, G., Brown, M., and Szeliski, R. (2007). City-scale location recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Schindler, G., Krishnamurthy, P., Lublinerman, R., Liu, Y., and Dellaert, F. (2008). Detecting and matching repeated patterns for automatic geo-tagging in urban environments. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Schlesinger, D. and Flach, B. (2006). *Transforming an arbitrary minsum problem into a binary one*. Technical Report TUD-FI06-01, Dresden University of Technology.
- Schlesinger, M. I. (1976). Syntactic analysis of two-dimensional visual signals in noisy conditions. *Kibernetika*, 4:113–130.
- Schlesinger, M. I. and Giginyak, V. V. (2007a). Solution to structural recognition (max,+)-problems by their equivalent transformations – part 1. *Control Systems and Computers*, 2007(1):3–15.
- Schlesinger, M. I. and Giginyak, V. V. (2007b). Solution to structural recognition (max,+)-problems by their equivalent transformations – part 2. *Control Systems and Computers*, 2007(2):3–18.
- Schmid, C. and Mohr, R. (1997). Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):530–534.
- Schmid, C. and Zisserman, A. (1997). Automatic line matching across views. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pp. 666–671, San Juan, Puerto Rico.
- Schmid, C., Mohr, R., and Bauckhage, C. (2000). Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151–172.
- Schneiderman, H. and Kanade, T. (2004). Object detection using the statistics of parts. *International Journal of Computer Vision*, 56(3):151–177.
- Schödl, A. and Essa, I. (2002). Controlled animation of video sprites. In *ACM Symposium on Computer Animation*, San Antonio.
- Schödl, A., Szeliski, R., Salesin, D. H., and Essa, I. (2000). Video textures. In *ACM SIGGRAPH 2000 Conference Proceedings*, pp. 489–498, New Orleans.
- Schoenemann, T. and Cremers, D. (2008). High resolution motion layer decomposition using dual-space graph cuts. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Schölkopf, B. and Smola, A. (eds). (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, Cambridge, Massachusetts.
- Schraudolph, N. N. (2010). Polynomial-time exact inference in NP-hard binary MRFs via reweighted perfect matching. In *13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 717–724.

- Schröder, P. and Sweldens, W. (1995). Spherical wavelets: Efficiently representing functions on the sphere. In *ACM SIGGRAPH 1995 Conference Proceedings*, pp. 161–172.
- Schultz, R. R. and Stevenson, R. L. (1996). Extraction of high-resolution frames from video sequences. *IEEE Transactions on Image Processing*, 5(6):996–1011.
- Sclaroff, S. and Isidoro, J. (2003). Active blobs: region-based, deformable appearance models. *Computer Vision and Image Understanding*, 89(2-3):197–225.
- Scott, G. L. and Longuet-Higgins, H. C. (1990). Feature grouping by relocalization of eigenvectors of the proximity matrix. In *British Machine Vision Conference*, pp. 103–108.
- Sebastian, T. B. and Kimia, B. B. (2005). Curves vs. skeletons in object recognition. *Signal Processing*, 85(2):246–263.
- Sederberg, T. W. and Parry, S. R. (1986). Free-form deformations of solid geometric models. *Computer Graphics (SIGGRAPH '86)*, 20(4):151–160.
- Sederberg, T. W., Gao, P., Wang, G., and Mu, H. (1993). 2D shape blending: An intrinsic solution to the vertex path problem. In *ACM SIGGRAPH 1993 Conference Proceedings*, pp. 15–18.
- Seitz, P. (1989). Using local orientation information as image primitive for robust object recognition. In *SPIE Vol. 1199, Visual Communications and Image Processing IV*, pp. 1630–1639.
- Seitz, S. (2001). The space of all stereo images. In *Eighth International Conference on Computer Vision (ICCV 2001)*, pp. 26–33, Vancouver, Canada.
- Seitz, S. and Szeliski, R. (1999). Applications of computer vision to computer graphics. *Computer Graphics*, 33(4):35–37. Guest Editors' introduction to the Special Issue.
- Seitz, S., Curless, B., Diebel, J., Scharstein, D., and Szeliski, R. (2006). A comparison and evaluation of multi-view stereo reconstruction algorithms. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2006)*, pp. 519–526, New York, NY.
- Seitz, S. M. and Baker, S. (2009). Filter flow. In *Twelfth International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan.
- Seitz, S. M. and Dyer, C. M. (1996). View morphing. In *ACM SIGGRAPH 1996 Conference Proceedings*, pp. 21–30, New Orleans.
- Seitz, S. M. and Dyer, C. M. (1997). Photorealistic scene reconstruction by voxel coloring. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pp. 1067–1073, San Juan, Puerto Rico.
- Seitz, S. M. and Dyer, C. M. (1999). Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35(2):151–173.
- Seitz, S. M. and Dyer, C. R. (1997). View invariant analysis of cyclic motion. *International Journal of Computer Vision*, 25(3):231–251.
- Serra, J. (1982). *Image Analysis and Mathematical Morphology*. Academic Press, New York.
- Serra, J. and Vincent, L. (1992). An overview of morphological filtering. *Circuits, Systems and Signal Processing*, 11(1):47–108.
- Serre, T., Wolf, L., and Poggio, T. (2005). Object recognition with features inspired by visual cortex. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 994–1000, San Diego, CA.

- Sethian, J. (1999). *Level Set Methods and Fast Marching Methods*. Cambridge University Press, Cambridge, 2nd edition.
- Shade, J., Gortler, S., He, L., and Szeliski, R. (1998). Layered depth images. In *ACM SIGGRAPH 1998 Conference Proceedings*, pp. 231–242, Orlando.
- Shade, J., Lischinski, D., Salesin, D., DeRose, T., and Snyder, J. (1996). Hierarchical images caching for accelerated walkthroughs of complex environments. In *ACM SIGGRAPH 1996 Conference Proceedings*, pp. 75–82, New Orleans.
- Shafer, S. A. (1985). Using color to separate reflection components. *COLOR Research and Applications*, 10(4):210–218.
- Shafer, S. A., Healey, G., and Wolff, L. (1992). *Physics-Based Vision: Principles and Practice*. Jones & Bartlett, Cambridge, MA.
- Shafique, K. and Shah, M. (2005). A noniterative greedy algorithm for multiframe point correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):51–65.
- Shah, J. (1993). A nonlinear diffusion model for discontinuous disparity and half-occlusion in stereo. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'93)*, pp. 34–40, New York.
- Shakhnarovich, G., Darrell, T., and Indyk, P. (eds). (2006). *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*, MIT Press.
- Shakhnarovich, G., Viola, P., and Darrell, T. (2003). Fast pose estimation with parameter-sensitive hashing. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 750–757, Nice, France.
- Shan, Y., Liu, Z., and Zhang, Z. (2001). Model-based bundle adjustment with application to face modeling. In *Eighth International Conference on Computer Vision (ICCV 2001)*, pp. 644–641, Vancouver, Canada.
- Sharon, E., Galun, M., Sharon, D., Basri, R., and Brandt, A. (2006). Hierarchy and adaptivity in segmenting visual scenes. *Nature*, 442(7104):810–813.
- Shashua, A. and Toelg, S. (1997). The quadric reference surface: Theory and applications. *International Journal of Computer Vision*, 23(2):185–198.
- Shashua, A. and Wexler, Y. (2001). Q-warping: Direct computation of quadratic reference surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(8):920–925.
- Shaw, D. and Barnes, N. (2006). Perspective rectangle detection. In *Workshop on Applications of Computer Vision at ECCV'2006*.
- Shewchuk, J. R. (1994). An introduction to the conjugate gradient method without the agonizing pain. Unpublished manuscript, available on author's homepage (<http://www.cs.berkeley.edu/~jrs/>). An earlier version appeared as a Carnegie Mellon University Technical Report, CMU-CS-94-125.
- Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(22):888–905.
- Shi, J. and Tomasi, C. (1994). Good features to track. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pp. 593–600, Seattle.
- Shimizu, M. and Okutomi, M. (2001). Precise sub-pixel estimation on area-based matching. In *Eighth International Conference on Computer Vision (ICCV 2001)*, pp. 90–97, Vancouver, Canada.
- Shirley, P. (2005). *Fundamentals of Computer Graphics*. A K Peters, Wellesley, Massachusetts, second edition.

- Shizawa, M. and Mase, K. (1991). A unified computational theory of motion transparency and motion boundaries based on eigenenergy analysis. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'91)*, pp. 289–295, Maui, Hawaii.
- Shoemake, K. (1985). Animating rotation with quaternion curves. *Computer Graphics (SIGGRAPH '85)*, 19(3):245–254.
- Shotton, J., Blake, A., and Cipolla, R. (2005). Contour-based learning for object detection. In *Tenth International Conference on Computer Vision (ICCV 2005)*, pp. 503–510, Beijing, China.
- Shotton, J., Johnson, M., and Cipolla, R. (2008). Semantic texton forests for image categorization and segmentation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Shotton, J., Winn, J., Rother, C., and Criminisi, A. (2009). Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling appearance, shape and context. *International Journal of Computer Vision*, 81(1):2–23.
- Shufelt, J. (1999). Performance evaluation and analysis of vanishing point detection techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(3):282–288.
- Shum, H.-Y. and He, L.-W. (1999). Rendering with concentric mosaics. In *ACM SIGGRAPH 1999 Conference Proceedings*, pp. 299–306, Los Angeles.
- Shum, H.-Y. and Szeliski, R. (1999). Stereo reconstruction from multiperspective panoramas. In *Seventh International Conference on Computer Vision (ICCV'99)*, pp. 14–21, Kerkyra, Greece.
- Shum, H.-Y. and Szeliski, R. (2000). Construction of panoramic mosaics with global and local alignment. *International Journal of Computer Vision*, 36(2):101–130. Erratum published July 2002, 48(2):151–152.
- Shum, H.-Y., Chan, S.-C., and Kang, S. B. (2007). *Image-Based Rendering*. Springer, New York, NY.
- Shum, H.-Y., Han, M., and Szeliski, R. (1998). Interactive construction of 3D models from panoramic mosaics. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'98)*, pp. 427–433, Santa Barbara.
- Shum, H.-Y., Ikeuchi, K., and Reddy, R. (1995). Principal component analysis with missing data and its application to polyhedral modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(9):854–867.
- Shum, H.-Y., Kang, S. B., and Chan, S.-C. (2003). Survey of image-based representations and compression techniques. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(11):1020–1037.
- Shum, H.-Y., Wang, L., Chai, J.-X., and Tong, X. (2002). Rendering by manifold hopping. *International Journal of Computer Vision*, 50(2):185–201.
- Shum, H.-Y., Sun, J., Yamazaki, S., Li, Y., and Tang, C.-K. (2004). Pop-up light field: An interactive image-based modeling and rendering system. *ACM Transactions on Graphics*, 23(2):143–162.
- Sidenbladh, H. and Black, M. J. (2003). Learning the statistics of people in images and video. *International Journal of Computer Vision*, 54(1):189–209.
- Sidenbladh, H., Black, M. J., and Fleet, D. J. (2000). Stochastic tracking of 3D human figures using 2D image motion. In *Sixth European Conference on Computer Vision (ECCV 2000)*, pp. 702–718, Dublin, Ireland.
- Sigal, L. and Black, M. J. (2006). Predicting 3D people from 2D pictures. In *AMDO 2006 - IV Conference on Articulated Motion and Deformable Objects*, pp. 185–195, Mallorca, Spain.

- Sigal, L., Balan, A., and Black, M. J. (2010). Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal of Computer Vision*, 87(1-2):4–27.
- Sigal, L., Bhatia, S., Roth, S., Black, M. J., and Isard, M. (2004). Tracking loose-limbed people. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2004)*, pp. 421–428, Washington, DC.
- Sillion, F. and Puech, C. (1994). *Radiosity and Global Illumination*. Morgan Kaufmann.
- Sim, T., Baker, S., and Bsat, M. (2003). The CMU pose, illumination, and expression database. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1615–1618.
- Simard, P. Y., Bottou, L., Haffner, P., and Cun, Y. L. (1998). Boxlets: a fast convolution algorithm for signal processing and neural networks. In *Advances in Neural Information Processing Systems 13*, pp. 571–577.
- Simon, I. and Seitz, S. M. (2008). Scene segmentation using the wisdom of crowds. In *Tenth European Conference on Computer Vision (ECCV 2008)*, pp. 541–553, Marseilles.
- Simon, I., Snavely, N., and Seitz, S. M. (2007). Scene summarization for online image collections. In *Eleventh International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil.
- Simoncelli, E. P. (1999). Bayesian denoising of visual images in the wavelet domain. In Müller, P. and Vidakovic, B. (eds), *Bayesian Inference in Wavelet Based Models*, pp. 291–308, Springer-Verlag, New York.
- Simoncelli, E. P. and Adelson, E. H. (1990a). Non-separable extensions of quadrature mirror filters to multiple dimensions. *Proceedings of the IEEE*, 78(4):652–664.
- Simoncelli, E. P. and Adelson, E. H. (1990b). Subband transforms. In Woods, J. (ed.), *Subband Coding*, pp. 143–191, Kluwer Academic Press, Norwell, MA.
- Simoncelli, E. P., Adelson, E. H., and Heeger, D. J. (1991). Probability distributions of optic flow. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'91)*, pp. 310–315, Maui, Hawaii.
- Simoncelli, E. P., Freeman, W. T., Adelson, E. H., and Heeger, D. J. (1992). Shiftable multiscale transforms. *IEEE Transactions on Information Theory*, 38(3):587–607.
- Singaraju, D., Grady, L., and Vidal, R. (2008). Interactive image segmentation via minimization of quadratic energies on directed graphs. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Singaraju, D., Rother, C., and Rhemann, C. (2009). New appearance models for natural image matting. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, Miami Beach, FL.
- Singaraju, D., Grady, L., Sinop, A. K., and Vidal, R. (2010). A continuous valued MRF for image segmentation. In Blake, A., Kohli, P., and Rother, C. (eds), *Advances in Markov Random Fields*, MIT Press.
- Sinha, P., Balas, B., Ostrovsky, Y., and Russell, R. (2006). Face recognition by humans: Nineteen results all computer vision researchers should know about. *Proceedings of the IEEE*, 94(11):1948–1962.
- Sinha, S., Mordohai, P., and Pollefeys, M. (2007). Multi-view stereo via graph cuts on the dual of an adaptive tetrahedral mesh. In *Eleventh International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil.

- Sinha, S. N. and Pollefeys, M. (2005). Multi-view reconstruction using photo-consistency and exact silhouette constraints: A maximum-flow formulation. In *Tenth International Conference on Computer Vision (ICCV 2005)*, pp. 349–356, Beijing, China.
- Sinha, S. N., Steedly, D., and Szeliski, R. (2009). Piecewise planar stereo for image-based rendering. In *Twelfth IEEE International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan.
- Sinha, S. N., Steedly, D., Szeliski, R., Agrawala, M., and Pollefeys, M. (2008). Interactive 3D architectural modeling from unordered photo collections. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2008)*, 27(5).
- Sinop, A. K. and Grady, L. (2007). A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. In *Eleventh International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil.
- Sivic, J. and Zisserman, A. (2003). Video Google: A text retrieval approach to object matching in videos. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 1470–1477, Nice, France.
- Sivic, J. and Zisserman, A. (2009). Efficient visual search of videos cast as text retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):591–606.
- Sivic, J., Everingham, M., and Zisserman, A. (2009). “Who are you?”—Learning person specific classifiers from video. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, Miami Beach, FL.
- Sivic, J., Zitnick, C. L., and Szeliski, R. (2006). Finding people in repeated shots of the same scene. In *British Machine Vision Conference (BMVC 2006)*, pp. 909–918, Edinburgh.
- Sivic, J., Russell, B., Zisserman, A., Freeman, W. T., and Efros, A. A. (2008). Unsupervised discovery of visual object class hierarchies. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Sivic, J., Russell, B. C., Efros, A. A., Zisserman, A., and Freeman, W. T. (2005). Discovering objects and their localization in images. In *Tenth International Conference on Computer Vision (ICCV 2005)*, pp. 370–377, Beijing, China.
- Slabaugh, G. G., Culbertson, W. B., Slabaugh, T. G., Culbertson, B., Malzbender, T., and Stevens, M. (2004). Methods for volumetric reconstruction of visual scenes. *International Journal of Computer Vision*, 57(3):179–199.
- Slama, C. C. (ed.). (1980). *Manual of Photogrammetry*. American Society of Photogrammetry, Falls Church, Virginia, fourth edition.
- Smelyanskiy, V. N., Cheeseman, P., Maluf, D. A., and Morris, R. D. (2000). Bayesian super-resolved surface reconstruction from images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2000)*, pp. 375–382, Hilton Head Island.
- Smeulders, A. W. M., Worring, M., Santini, S., Gupta, A., and Jain, R. C. (2000). Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):477–490.
- Sminchisescu, C. and Triggs, B. (2001). Covariance scaled sampling for monocular 3D body tracking. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2001)*, pp. 447–454, Kauai, Hawaii.
- Sminchisescu, C., Kanaujia, A., and Metaxas, D. (2006). Conditional models for contextual human motion recognition. *Computer Vision and Image Understanding*, 104(2-3):210–220.

- Sminchisescu, C., Kanaujia, A., Li, Z., and Metaxas, D. (2005). Discriminative density propagation for 3D human motion estimation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 390–397, San Diego, CA.
- Smith, A. R. and Blinn, J. F. (1996). Blue screen matting. In *ACM SIGGRAPH 1996 Conference Proceedings*, pp. 259–268, New Orleans.
- Smith, B. M., Zhang, L., Jin, H., and Agarwala, A. (2009). Light field video stabilization. In *Twelfth International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan.
- Smith, S. M. and Brady, J. M. (1997). SUSAN—a new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78.
- Smolic, A. and Kauff, P. (2005). Interactive 3-D video representation and coding technologies. *Proceedings of the IEEE*, 93(1):98–110.
- Snavely, N., Seitz, S. M., and Szeliski, R. (2006). Photo tourism: Exploring photo collections in 3D. *ACM Transactions on Graphics (Proc. SIGGRAPH 2006)*, 25(3):835–846.
- Snavely, N., Seitz, S. M., and Szeliski, R. (2008a). Modeling the world from Internet photo collections. *International Journal of Computer Vision*, 80(2):189–210.
- Snavely, N., Seitz, S. M., and Szeliski, R. (2008b). Skeletal graphs for efficient structure from motion. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Snavely, N., Garg, R., Seitz, S. M., and Szeliski, R. (2008). Finding paths through the world’s photos. *ACM Transactions on Graphics (Proc. SIGGRAPH 2008)*, 27(3).
- Snavely, N., Simon, I., Goesele, M., Szeliski, R., and Seitz, S. M. (2010). Scene reconstruction and visualization from community photo collections. *Proceedings of the IEEE*, 98(8):1370–1390.
- Soatto, S., Yezzi, A. J., and Jin, H. (2003). Tales of shape and radiance in multiview stereo. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 974–981, Nice, France.
- Soille, P. (2006). Morphological image compositing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(5):673–683.
- Solina, F. and Bajcsy, R. (1990). Recovery of parametric models from range images: The case for superquadrics with global deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):131–147.
- Sontag, D. and Jaakkola, T. (2007). New outer bounds on the marginal polytope. In *Advances in Neural Information Processing Systems*.
- Sontag, D., Meltzer, T., Globerson, A., Jaakkola, T., and Weiss, Y. (2008). Tightening LP relaxations for MAP using message passing. In *Uncertainty in Artificial Intelligence (UAI)*.
- Soucy, M. and Laurendeau, D. (1992). Multi-resolution surface modeling from multiple range views. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'92)*, pp. 348–353, Champaign, Illinois.
- Srinivasan, S., Chellappa, R., Veeraraghavan, A., and Aggarwal, G. (2005). Electronic image stabilization and mosaicking algorithms. In Bovik, A. (ed.), *Handbook of Image and Video Processing*, Academic Press.
- Srivasan, P., Liang, P., and Hackwood, S. (1990). Computational geometric methods in volumetric intersections for 3D reconstruction. *Pattern Recognition*, 23(8):843–857.
- Stamos, I., Liu, L., Chen, C., Wolberg, G., Yu, G., and Zokai, S. (2008). Integrating automated range registration with multiview geometry for the photorealistic modeling of large-scale scenes. *International Journal of Computer Vision*, 78(2-3):237–260.

- Stark, J. A. (2000). Adaptive image contrast enhancement using generalizations of histogram equalization. *IEEE Transactions on Image Processing*, 9(5):889–896.
- Stauffer, C. and Grimson, W. (1999). Adaptive background mixture models for real-time tracking. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99)*, pp. 246–252, Fort Collins.
- Steedly, D. and Essa, I. (2001). Propagation of innovative information in non-linear least-squares structure from motion. In *Eighth International Conference on Computer Vision (ICCV 2001)*, pp. 223–229, Vancouver, Canada.
- Steedly, D., Essa, I., and Dellaert, F. (2003). Spectral partitioning for structure from motion. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 996–1003, Nice, France.
- Steedly, D., Pal, C., and Szeliski, R. (2005). Efficiently registering video into panoramic mosaics. In *Tenth International Conference on Computer Vision (ICCV 2005)*, pp. 1300–1307, Beijing, China.
- Steele, R. and Jaynes, C. (2005). Feature uncertainty arising from covariant image noise. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 1063–1070, San Diego, CA.
- Steele, R. M. and Jaynes, C. (2006). Overconstrained linear estimation of radial distortion and multi-view geometry. In *Ninth European Conference on Computer Vision (ECCV 2006)*, pp. 253–264.
- Stein, A., Hoiem, D., and Hebert, M. (2007). Learning to extract object boundaries using motion cues. In *Eleventh International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil.
- Stein, F. and Medioni, G. (1992). Structural indexing: Efficient 3-D object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):125–145.
- Stein, G. (1995). Accurate internal camera calibration using rotation, with analysis of sources of error. In *Fifth International Conference on Computer Vision (ICCV'95)*, pp. 230–236, Cambridge, Massachusetts.
- Stein, G. (1997). Lens distortion calibration using point correspondences. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pp. 602–608, San Juan, Puerto Rico.
- Stenger, B., Thayanathan, A., Torr, P. H. S., and Cipolla, R. (2006). Model-based hand tracking using a hierarchical bayesian filter. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1372–1384.
- Stewart, C. V. (1999). Robust parameter estimation in computer vision. *SIAM Reviews*, 41(3):513–537.
- Stiller, C. and Konrad, J. (1999). Estimating motion in image sequences: A tutorial on modeling and computation of 2D motion. *IEEE Signal Processing Magazine*, 16(4):70–91.
- Stollnitz, E. J., DeRose, T. D., and Salesin, D. H. (1996). *Wavelets for Computer Graphics: Theory and Applications*. Morgan Kaufmann, San Francisco.
- Strang, G. (1988). *Linear Algebra and its Applications*. Harcourt, Brace, Jovanovich, Publishers, San Diego, 3rd edition.
- Strang, G. (1989). Wavelets and dilation equations: A brief introduction. *SIAM Reviews*, 31(4):614–627.
- Strecha, C., Fransens, R., and Van Gool, L. (2006). Combined depth and outlier estimation in multi-view stereo. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2006)*, pp. 2394–2401, New York City, NY.
- Strecha, C., Tuytelaars, T., and Van Gool, L. (2003). Dense matching of multiple wide-baseline views. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 1194–1201, Nice, France.

- Strecha, C., von Hansen, W., Van Gool, L., Fua, P., and Thoennessen, U. (2008). On benchmarking camera calibration and multi-view stereo. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Sturm, P. (2005). Multi-view geometry for general camera models. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 206–212, San Diego, CA.
- Sturm, P. and Ramalingam, S. (2004). A generic concept for camera calibration. In *Eighth European Conference on Computer Vision (ECCV 2004)*, pp. 1–13, Prague.
- Sturm, P. and Triggs, W. (1996). A factorization based algorithm for multi-image projective structure and motion. In *Fourth European Conference on Computer Vision (ECCV'96)*, pp. 709–720, Cambridge, England.
- Su, H., Sun, M., Fei-Fei, L., and Savarese, S. (2009). Learning a dense multi-view representation for detection, viewpoint classification and synthesis of object categories. In *Twelfth International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan.
- Sudderth, E. B., Torralba, A., Freeman, W. T., and Willsky, A. S. (2008). Describing visual scenes using transformed objects and parts. *International Journal of Computer Vision*, 77(1-3):291–330.
- Sullivan, S. and Ponce, J. (1998). Automatic model construction and pose estimation from photographs using triangular splines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1091–1096.
- Sun, D., Roth, S., Lewis, J. P., and Black, M. J. (2008). Learning optical flow. In *Tenth European Conference on Computer Vision (ECCV 2008)*, pp. 83–97, Marseilles.
- Sun, J., Zheng, N., and Shum, H. (2003). Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):787–800.
- Sun, J., Jia, J., Tang, C.-K., and Shum, H.-Y. (2004). Poisson matting. *ACM Transactions on Graphics (Proc. SIGGRAPH 2004)*, 23(3):315–321.
- Sun, J., Li, Y., Kang, S. B., and Shum, H.-Y. (2006). Flash matting. *ACM Transactions on Graphics*, 25(3):772–778.
- Sun, J., Yuan, L., Jia, J., and Shum, H.-Y. (2004). Image completion with structure propagation. *ACM Transactions on Graphics (Proc. SIGGRAPH 2004)*, 24(3):861–868.
- Sun, M., Su, H., Savarese, S., and Fei-Fei, L. (2009). A multi-view probabilistic model for 3D object classes. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, Miami Beach, FL.
- Sung, K.-K. and Poggio, T. (1998). Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):39–51.
- Sutherland, I. E. (1974). Three-dimensional data input by tablet. *Proceedings of the IEEE*, 62(4):453–461.
- Swain, M. J. and Ballard, D. H. (1991). Color indexing. *International Journal of Computer Vision*, 7(1):11–32.
- Swaminathan, R., Kang, S. B., Szeliski, R., Criminisi, A., and Nayar, S. K. (2002). On the motion and appearance of specularities in image sequences. In *Seventh European Conference on Computer Vision (ECCV 2002)*, pp. 508–523, Copenhagen.
- Sweldens, W. (1996). Wavelets and the lifting scheme: A 5 minute tour. *Z. Angew. Math. Mech.*, 76 (Suppl. 2):41–44.
- Sweldens, W. (1997). The lifting scheme: A construction of second generation wavelets. *SIAM J. Math. Anal.*, 29(2):511–546.

- Swendsen, R. H. and Wang, J.-S. (1987). Nonuniversal critical dynamics in Monte Carlo simulations. *Physical Review Letters*, 58(2):86–88.
- Szeliski, R. (1986). *Cooperative Algorithms for Solving Random-Dot Stereograms*. Technical Report CMU-CS-86-133, Computer Science Department, Carnegie Mellon University.
- Szeliski, R. (1989). *Bayesian Modeling of Uncertainty in Low-Level Vision*. Kluwer Academic Publishers, Boston.
- Szeliski, R. (1990a). Bayesian modeling of uncertainty in low-level vision. *International Journal of Computer Vision*, 5(3):271–301.
- Szeliski, R. (1990b). Fast surface interpolation using hierarchical basis functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):513–528.
- Szeliski, R. (1991a). Fast shape from shading. *CVGIP: Image Understanding*, 53(2):129–153.
- Szeliski, R. (1991b). Shape from rotation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'91)*, pp. 625–630, Maui, Hawaii.
- Szeliski, R. (1993). Rapid octree construction from image sequences. *CVGIP: Image Understanding*, 58(1):23–32.
- Szeliski, R. (1994). Image mosaicing for tele-reality applications. In *IEEE Workshop on Applications of Computer Vision (WACV'94)*, pp. 44–53, Sarasota.
- Szeliski, R. (1996). Video mosaics for virtual environments. *IEEE Computer Graphics and Applications*, 16(2):22–30.
- Szeliski, R. (1999). A multi-view approach to motion and stereo. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99)*, pp. 157–163, Fort Collins.
- Szeliski, R. (2006a). Image alignment and stitching: A tutorial. *Foundations and Trends in Computer Graphics and Computer Vision*, 2(1):1–104.
- Szeliski, R. (2006b). Locally adapted hierarchical basis preconditioning. *ACM Transactions on Graphics (Proc. SIGGRAPH 2006)*, 25(3):1135–1143.
- Szeliski, R. and Coughlan, J. (1997). Spline-based image registration. *International Journal of Computer Vision*, 22(3):199–218.
- Szeliski, R. and Golland, P. (1999). Stereo matching with transparency and matting. *International Journal of Computer Vision*, 32(1):45–61. Special Issue for Marr Prize papers.
- Szeliski, R. and Hinton, G. (1985). Solving random-dot stereograms using the heat equation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'85)*, pp. 284–288, San Francisco.
- Szeliski, R. and Ito, M. R. (1986). New Hermite cubic interpolator for two-dimensional curve generation. *IEE Proceedings E*, 133(6):341–347.
- Szeliski, R. and Kang, S. B. (1994). Recovering 3D shape and motion from image streams using nonlinear least squares. *Journal of Visual Communication and Image Representation*, 5(1):10–28.
- Szeliski, R. and Kang, S. B. (1995). Direct methods for visual scene reconstruction. In *IEEE Workshop on Representations of Visual Scenes*, pp. 26–33, Cambridge, Massachusetts.
- Szeliski, R. and Kang, S. B. (1997). Shape ambiguities in structure from motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):506–512.

- Szeliski, R. and Lavallée, S. (1996). Matching 3-D anatomical surfaces with non-rigid deformations using octree-splines. *International Journal of Computer Vision*, 18(2):171–186.
- Szeliski, R. and Scharstein, D. (2004). Sampling the disparity space image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):419–425.
- Szeliski, R. and Shum, H.-Y. (1996). Motion estimation with quadtree splines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(12):1199–1210.
- Szeliski, R. and Shum, H.-Y. (1997). Creating full view panoramic image mosaics and texture-mapped models. In *ACM SIGGRAPH 1997 Conference Proceedings*, pp. 251–258, Los Angeles.
- Szeliski, R. and Tonnesen, D. (1992). Surface modeling with oriented particle systems. *Computer Graphics (SIGGRAPH '92)*, 26(2):185–194.
- Szeliski, R. and Torr, P. (1998). Geometrically constrained structure from motion: Points on planes. In *European Workshop on 3D Structure from Multiple Images of Large-Scale Environments (SMILE)*, pp. 171–186, Freiburg, Germany.
- Szeliski, R. and Weiss, R. (1998). Robust shape recovery from occluding contours using a linear smoother. *International Journal of Computer Vision*, 28(1):27–44.
- Szeliski, R., Avidan, S., and Anandan, P. (2000). Layer extraction from multiple images containing reflections and transparency. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2000)*, pp. 246–253, Hilton Head Island.
- Szeliski, R., Tonnesen, D., and Terzopoulos, D. (1993a). Curvature and continuity control in particle-based surface models. In *SPIE Vol. 2031, Geometric Methods in Computer Vision II*, pp. 172–181, San Diego.
- Szeliski, R., Tonnesen, D., and Terzopoulos, D. (1993b). Modeling surfaces of arbitrary topology with dynamic particles. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'93)*, pp. 82–87, New York.
- Szeliski, R., Uyttendaele, M., and Steedly, D. (2008). *Fast Poisson Blending using Multi-Splines*. Technical Report MSR-TR-2008-58, Microsoft Research.
- Szeliski, R., Winder, S., and Uyttendaele, M. (2010). *High-quality multi-pass image resampling*. Technical Report MSR-TR-2010-10, Microsoft Research.
- Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., and Rother, C. (2008). A comparative study of energy minimization methods for Markov random fields with smoothness-based priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(6):1068–1080.
- Szummer, M. and Picard, R. W. (1996). Temporal texture modeling. In *IEEE International Conference on Image Processing (ICIP-96)*, pp. 823–826, Lausanne.
- Tabb, M. and Ahuja, N. (1997). Multiscale image segmentation by integrated edge and region detection. *IEEE Transactions on Image Processing*, 6(5):642–655.
- Taguchi, Y., Wilburn, B., and Zitnick, C. L. (2008). Stereo reconstruction with mixed pixels using adaptive over-segmentation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Tanaka, M. and Okutomi, M. (2008). Locally adaptive learning for translation-variant MRF image priors. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.

- Tao, H., Sawhney, H. S., and Kumar, R. (2001). A global matching framework for stereo computation. In *Eighth International Conference on Computer Vision (ICCV 2001)*, pp. 532–539, Vancouver, Canada.
- Tappen, M. F. (2007). Utilizing variational optimization to learn Markov random fields. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Tappen, M. F. and Freeman, W. T. (2003). Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 900–907, Nice, France.
- Tappen, M. F., Freeman, W. T., and Adelson, E. H. (2005). Recovering intrinsic images from a single image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(9):1459–1472.
- Tappen, M. F., Russell, B. C., and Freeman, W. T. (2003). Exploiting the sparse derivative prior for super-resolution and image demosaicing. In *Third International Workshop on Statistical and Computational Theories of Vision*, Nice, France.
- Tappen, M. F., Liu, C., Freeman, W., and Adelson, E. (2007). Learning Gaussian conditional random fields for low-level vision. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Tardif, J.-P. (2009). Non-iterative approach for fast and accurate vanishing point detection. In *Twelfth International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan.
- Tardif, J.-P., Sturm, P., and Roy, S. (2007). Plane-based self-calibration of radial distortion. In *Eleventh International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil.
- Tardif, J.-P., Sturm, P., Trudeau, M., and Roy, S. (2009). Calibration of cameras with radially symmetric distortion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(9):1552–1566.
- Taubin, G. (1995). Curve and surface smoothing without shrinkage. In *Fifth International Conference on Computer Vision (ICCV'95)*, pp. 852–857, Cambridge, Massachusetts.
- Taubman, D. S. and Marcellin, M. W. (2002). Jpeg2000: standard for interactive imaging. *Proceedings of the IEEE*, 90(8):1336–1357.
- Taylor, C. J. (2003). Surface reconstruction from feature based stereo. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 184–190, Nice, France.
- Taylor, C. J., Debevec, P. E., and Malik, J. (1996). Reconstructing polyhedral models of architectural scenes from photographs. In *Fourth European Conference on Computer Vision (ECCV'96)*, pp. 659–668, Cambridge, England.
- Taylor, C. J., Kriegman, D. J., and Anandan, P. (1991). Structure and motion in two dimensions from multiple images: A least squares approach. In *IEEE Workshop on Visual Motion*, pp. 242–248, Princeton, New Jersey.
- Taylor, P. (2009). *Text-to-Speech Synthesis*. Cambridge University Press, Cambridge.
- Tek, K. and Kimia, B. B. (2003). Symmetry maps of free-form curve segments via wave propagation. *International Journal of Computer Vision*, 54(1-3):35–81.
- Tekalp, M. (1995). *Digital Video Processing*. Prentice Hall, Upper Saddle River, NJ.
- Telea, A. (2004). An image inpainting technique based on fast marching method. *Journal of Graphics Tools*, 9(1):23–34.
- Teller, S., Antone, M., Bodnar, Z., Bosse, M., Coorg, S., Jethwa, M., and Master, N. (2003). Calibrated, registered images of an extended urban area. *International Journal of Computer Vision*, 53(1):93–107.

- Teodosio, L. and Bender, W. (1993). Salient video stills: Content and context preserved. In *ACM Multimedia 93*, pp. 39–46, Anaheim, California.
- Terzopoulos, D. (1983). Multilevel computational processes for visual surface reconstruction. *Computer Vision, Graphics, and Image Processing*, 24:52–96.
- Terzopoulos, D. (1986a). Image analysis using multigrid relaxation methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(2):129–139.
- Terzopoulos, D. (1986b). Regularization of inverse visual problems involving discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(4):413–424.
- Terzopoulos, D. (1988). The computation of visible-surface representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-10(4):417–438.
- Terzopoulos, D. (1999). Visual modeling for computer animation: Graphics with a vision. *Computer Graphics*, 33(4):42–45.
- Terzopoulos, D. and Fleischer, K. (1988). Deformable models. *The Visual Computer*, 4(6):306–331.
- Terzopoulos, D. and Metaxas, D. (1991). Dynamic 3D models with local and global deformations: Deformable superquadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):703–714.
- Terzopoulos, D. and Szeliski, R. (1992). Tracking with Kalman snakes. In Blake, A. and Yuille, A. L. (eds), *Active Vision*, pp. 3–20, MIT Press, Cambridge, Massachusetts.
- Terzopoulos, D. and Waters, K. (1990). Analysis of facial images using physical and anatomical models. In *Third International Conference on Computer Vision (ICCV'90)*, pp. 727–732, Osaka, Japan.
- Terzopoulos, D. and Witkin, A. (1988). Physically-based models with rigid and deformable components. *IEEE Computer Graphics and Applications*, 8(6):41–51.
- Terzopoulos, D., Witkin, A., and Kass, M. (1987). Symmetry-seeking models and 3D object reconstruction. *International Journal of Computer Vision*, 1(3):211–221.
- Terzopoulos, D., Witkin, A., and Kass, M. (1988). Constraints on deformable models: Recovering 3D shape and nonrigid motion. *Artificial Intelligence*, 36(1):91–123.
- Thayananthan, A., Iwasaki, M., and Cipolla, R. (2008). Principled fusion of high-level model and low-level cues for motion segmentation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Thirthala, S. and Pollefeys, M. (2005). The radial trifocal tensor: A tool for calibrating the radial distortion of wide-angle cameras. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 321–328, San Diego, CA.
- Thomas, D. B., Luk, W., Leong, P. H., and Villasenor, J. D. (2007). Gaussian random number generators. *ACM Computing Surveys*, 39(4).
- Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. The MIT Press, Cambridge, Massachusetts.
- Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A. et al. (2006). Stanley, the robot that won the DARPA Grand Challenge. *Journal of Field Robotics*, 23(9):661–692.
- Tian, Q. and Huhns, M. N. (1986). Algorithms for subpixel registration. *Computer Vision, Graphics, and Image Processing*, 35:220–233.
- Tikhonov, A. N. and Arsenin, V. Y. (1977). *Solutions of Ill-Posed Problems*. V. H. Winston, Washington, D. C.

- Tipping, M. E. and Bishop, C. M. (1999). Probabilistic principal components analysis. *Journal of the Royal Statistical Society, Series B*, 61(3):611–622.
- Toint, P. L. (1987). On large scale nonlinear least squares calculations. *SIAM J. Sci. Stat. Comput.*, 8(3):416–435.
- Tola, E., Lepetit, V., and Fua, P. (2010). DAISY: An efficient dense descriptor applied to wide-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):815–830.
- Tolliver, D. and Miller, G. (2006). Graph partitioning by spectral rounding: Applications in image segmentation and clustering. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2006)*, pp. 1053–1060, New York City, NY.
- Tomasi, C. and Kanade, T. (1992). Shape and motion from image streams under orthography: A factorization method. *International Journal of Computer Vision*, 9(2):137–154.
- Tomasi, C. and Manduchi, R. (1998). Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision (ICCV'98)*, pp. 839–846, Bombay.
- Tombari, F., Mattoccia, S., and Di Stefano, L. (2007). Segmentation-based adaptive support for accurate stereo correspondence. In *Pacific-Rim Symposium on Image and Video Technology*.
- Tombari, F., Mattoccia, S., Di Stefano, L., and Addimanda, E. (2008). Classification and evaluation of cost aggregation methods for stereo correspondence. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Tommasini, T., Fusillo, A., Trucco, E., and Roberto, V. (1998). Making good features track better. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'98)*, pp. 178–183, Santa Barbara.
- Torborg, J. and Kajiya, J. T. (1996). Talisman: Commodity realtime 3D graphics for the PC. In *ACM SIGGRAPH 1996 Conference Proceedings*, pp. 353–363, New Orleans.
- Torr, P. H. S. (2002). Bayesian model estimation and selection for epipolar geometry and generic manifold fitting. *International Journal of Computer Vision*, 50(1):35–61.
- Torr, P. H. S. and Fitzgibbon, A. W. (2004). Invariant fitting of two view geometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):648–650.
- Torr, P. H. S. and Murray, D. (1997). The development and comparison of robust methods for estimating the fundamental matrix. *International Journal of Computer Vision*, 24(3):271–300.
- Torr, P. H. S., Szeliski, R., and Anandan, P. (1999). An integrated Bayesian approach to layer extraction from image sequences. In *Seventh International Conference on Computer Vision (ICCV'99)*, pp. 983–990, Kerkyra, Greece.
- Torr, P. H. S., Szeliski, R., and Anandan, P. (2001). An integrated Bayesian approach to layer extraction from image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(3):297–303.
- Torralba, A. (2003). Contextual priming for object detection. *International Journal of Computer Vision*, 53(2):169–191.
- Torralba, A. (2007). Classifier-based methods. In *CVPR 2007 Short Course on Recognizing and Learning Object Categories*. <http://people.csail.mit.edu/torralba/shortCourseRLOC/>.
- Torralba, A. (2008). Object recognition and scene understanding. MIT Course 6.870, <http://people.csail.mit.edu/torralba/courses/6.870/6.870.recognition.htm>.

- Torralba, A., Freeman, W. T., and Fergus, R. (2008). 80 million tiny images: a large dataset for non-parametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970.
- Torralba, A., Murphy, K. P., and Freeman, W. T. (2004). Contextual models for object detection using boosted random fields. In *Advances in Neural Information Processing Systems*.
- Torralba, A., Murphy, K. P., and Freeman, W. T. (2007). Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5):854–869.
- Torralba, A., Weiss, Y., and Fergus, R. (2008). Small codes and large databases of images for object recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Torralba, A., Murphy, K. P., Freeman, W. T., and Rubin, M. A. (2003). Context-based vision system for place and object recognition. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 273–280, Nice, France.
- Torrance, K. E. and Sparrow, E. M. (1967). Theory for off-specular reflection from roughened surfaces. *Journal of the Optical Society of America A*, 57(9):1105–1114.
- Torresani, L., Hertzmann, A., and Bregler, C. (2008). Non-rigid structure-from-motion: Estimating shape and motion with hierarchical priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5):878–892.
- Toyama, K. (1998). *Prolegomena for Robust Face Tracking*. Technical Report MSR-TR-98-65, Microsoft Research.
- Toyama, K., Krumm, J., Brumitt, B., and Meyers, B. (1999). Wallflower: Principles and practice of background maintenance. In *Seventh International Conference on Computer Vision (ICCV'99)*, pp. 255–261, Kerkyra, Greece.
- Tran, S. and Davis, L. (2002). 3D surface reconstruction using graph cuts with surface constraints. In *Seventh European Conference on Computer Vision (ECCV 2002)*, pp. 219–231, Copenhagen.
- Trefethen, L. N. and Bau, D. (1997). *Numerical Linear Algebra*. SIAM.
- Treisman, A. (1985). Preattentive processing in vision. *Computer Vision, Graphics, and Image Processing*, 31(2):156–177.
- Triggs, B. (1996). Factorization methods for projective structure and motion. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'96)*, pp. 845–851, San Francisco.
- Triggs, B. (2004). Detecting keypoints with stable position, orientation, and scale under illumination changes. In *Eighth European Conference on Computer Vision (ECCV 2004)*, pp. 100–113, Prague.
- Triggs, B., McLauchlan, P. F., Hartley, R. I., and Fitzgibbon, A. W. (1999). Bundle adjustment — a modern synthesis. In *International Workshop on Vision Algorithms*, pp. 298–372, Kerkyra, Greece.
- Trobin, W., Pock, T., Cremers, D., and Bischof, H. (2008). Continuous energy minimization via repeated binary fusion. In *Tenth European Conference on Computer Vision (ECCV 2008)*, pp. 677–690, Marseilles.
- Troccoli, A. and Allen, P. (2008). Building illumination coherent 3D models of large-scale outdoor scenes. *International Journal of Computer Vision*, 78(2-3):261–280.
- Trottenberg, U., Oosterlee, C. W., and Schuller, A. (2000). *Multigrid*. Academic Press.
- Trucco, E. and Verri, A. (1998). *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, Upper Saddle River, NJ.

- Tsai, P. S. and Shah, M. (1994). Shape from shading using linear approximation. *Image and Vision Computing*, 12:487–498.
- Tsai, R. Y. (1987). A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal of Robotics and Automation*, RA-3(4):323–344.
- Tschumperlé, D. (2006). Curvature-preserving regularization of multi-valued images using PDEs. In *Ninth European Conference on Computer Vision (ECCV 2006)*, pp. 295–307.
- Tschumperlé, D. and Deriche, R. (2005). Vector-valued image regularization with PDEs: A common framework for different applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:506–517.
- Tsin, Y., Kang, S. B., and Szeliski, R. (2006). Stereo matching with linear superposition of layers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(2):290–301.
- Tsin, Y., Ramesh, V., and Kanade, T. (2001). Statistical calibration of CCD imaging process. In *Eighth International Conference on Computer Vision (ICCV 2001)*, pp. 480–487, Vancouver, Canada.
- Tu, Z., Chen, X., Yuille, A. L., and Zhu, S.-C. (2005). Image parsing: Unifying segmentation, detection, and recognition. *International Journal of Computer Vision*, 63(2):113–140.
- Tumblin, J. and Rushmeier, H. E. (1993). Tone reproduction for realistic images. *IEEE Computer Graphics and Applications*, 13(6):42–48.
- Tumblin, J. and Turk, G. (1999). LCIS: A boundary hierarchy for detail-preserving contrast reduction. In *ACM SIGGRAPH 1999 Conference Proceedings*, pp. 83–90, Los Angeles.
- Tumblin, J., Agrawal, A., and Raskar, R. (2005). Why I want a gradient camera. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 103–110, San Diego, CA.
- Turcot, P. and Lowe, D. G. (2009). Better matching with fewer features: The selection of useful features in large database recognition problems. In *ICCV Workshop on Emergent Issues in Large Amounts of Visual Data (WS-LAVD)*, Kyoto, Japan.
- Turk, G. and Levoy, M. (1994). Zippered polygonal meshes from range images. In *ACM SIGGRAPH 1994 Conference Proceedings*, pp. 311–318.
- Turk, G. and O'Brien, J. (2002). Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics*, 21(4):855–873.
- Turk, M. and Pentland, A. (1991a). Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86.
- Turk, M. and Pentland, A. (1991b). Face recognition using eigenfaces. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'91)*, pp. 586–591, Maui, Hawaii.
- Tuytelaars, T. and Mikolajczyk, K. (2007). Local invariant feature detectors. *Foundations and Trends in Computer Graphics and Computer Vision*, 3(1).
- Tuytelaars, T. and Van Gool, L. (2004). Matching widely separated views based on affine invariant regions. *International Journal of Computer Vision*, 59(1):61–85.
- Tuytelaars, T., Van Gool, L., and Proesmans, M. (1997). The cascaded Hough transform. In *International Conference on Image Processing (ICIP'97)*, pp. 736–739.
- Ullman, S. (1979). The interpretation of structure from motion. *Proceedings of the Royal Society of London*, B-203:405–426.

- Unnikrishnan, R., Pantofaru, C., and Hebert, M. (2007). Toward objective evaluation of image segmentation algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):828–944.
- Unser, M. (1999). Splines: A perfect fit for signal and image processing. *IEEE Signal Processing Magazine*, 16(6):22–38.
- Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R. et al. (2008). Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466.
- Urtasun, R., Fleet, D. J., and Fua, P. (2006). Temporal motion models for monocular and multiview 3D human body tracking. *Computer Vision and Image Understanding*, 104(2-3):157–177.
- Uyttendaele, M., Eden, A., and Szeliski, R. (2001). Eliminating ghosting and exposure artifacts in image mosaics. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2001)*, pp. 509–516, Kauai, Hawaii.
- Uyttendaele, M., Criminisi, A., Kang, S. B., Winder, S., Hartley, R., and Szeliski, R. (2004). Image-based interactive exploration of real-world environments. *IEEE Computer Graphics and Applications*, 24(3):52–63.
- Vaillant, R. and Faugeras, O. D. (1992). Using extremal boundaries for 3-D object modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):157–173.
- Vaish, V., Szeliski, R., Zitnick, C. L., Kang, S. B., and Levoy, M. (2006). Reconstructing occluded surfaces using synthetic apertures: Shape from focus vs. shape from stereo. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2006)*, pp. 2331–2338, New York, NY.
- van de Weijer, J. and Schmid, C. (2006). Coloring local feature extraction. In *Ninth European Conference on Computer Vision (ECCV 2006)*, pp. 334–348.
- van den Hengel, A., Dick, A., Thormhlen, T., Ward, B., and Torr, P. H. S. (2007). Videotrace: Rapid interactive scene modeling from video. *ACM Transactions on Graphics*, 26(3).
- Van Huffel, S. and Lemmerling, P. (eds). (2002). *Total Least Squares and Errors-in-Variables Modeling*, Springer.
- Van Huffel, S. and Vandewalle, J. (1991). *The Total Least Squares Problem: Computational Aspects and Analysis*. Society for Industrial and Applied Mathematics, Philadelphia.
- van Ouwerkerk, J. D. (2006). Image super-resolution survey. *Image and Vision Computing*, 24(10):1039–1052.
- Varma, M. and Ray, D. (2007). Learning the discriminative power-invariance trade-off. In *Eleventh International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil.
- Vasconcelos, N. (2007). From pixels to semantic spaces: Advances in content-based image retrieval. *Computer*, 40(7):20–26.
- Vasilescu, M. A. O. and Terzopoulos, D. (2007). Multilinear (tensor) image synthesis, analysis, and recognition. *IEEE Signal Processing Magazine*, 24(6):118–123.
- Vedaldi, A. and Fulkerson, B. (2008). VLFeat: An open and portable library of computer vision algorithms. <http://www.vlfeat.org/>.
- Vedaldi, A., Gulshan, V., Varma, M., and Zisserman, A. (2009). Multiple kernels for object detection. In *Twelfth International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan.
- Vedula, S., Baker, S., and Kanade, T. (2005). Image-based spatio-temporal modeling and view interpolation of dynamic events. *ACM Transactions on Graphics*, 24(2):240–261.

- Vedula, S., Baker, S., Rander, P., Collins, R., and Kanade, T. (2005). Three-dimensional scene flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):475–480.
- Veeraraghavan, A., Raskar, R., Agrawal, A., Mohan, A., and Tumblin, J. (2007). Dappled photography: Mask enhanced cameras for heterodyned light fields and coded aperture refocusing. *ACM Transactions on Graphics*, 26(3).
- Veksler, O. (1999). *Efficient Graph-based Energy Minimization Methods in Computer Vision*. Ph.D. thesis, Cornell University.
- Veksler, O. (2001). Stereo matching by compact windows via minimum ratio cycle. In *Eighth International Conference on Computer Vision (ICCV 2001)*, pp. 540–547, Vancouver, Canada.
- Veksler, O. (2003). Fast variable window for stereo correspondence using integral images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2003)*, pp. 556–561, Madison, WI.
- Veksler, O. (2007). Graph cut based optimization for MRFs with truncated convex priors. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Verbeek, J. and Triggs, B. (2007). Region classification with Markov field aspect models. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Vergauwen, M. and Van Gool, L. (2006). Web-based 3D reconstruction service. *Machine Vision and Applications*, 17(2):321–329.
- Vetter, T. and Poggio, T. (1997). Linear object classes and image synthesis from a single example image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):733–742.
- Vezhnevets, V., Sazonov, V., and Andreeva, A. (2003). A survey on pixel-based skin color detection techniques. In *GRAPHICON03*, pp. 85–92.
- Vicente, S., Kolmogorov, V., and Rother, C. (2008). Graph cut based image segmentation with connectivity priors. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Vidal, R., Ma, Y., and Sastry, S. S. (2010). *Generalized Principal Component Analysis*. Springer.
- Viéville, T. and Faugeras, O. D. (1990). Feedforward recovery of motion and structure from a sequence of 2D-lines matches. In *Third International Conference on Computer Vision (ICCV'90)*, pp. 517–520, Osaka, Japan.
- Vincent, L. and Soille, P. (1991). Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–596.
- Vineet, V. and Narayanan, P. J. (2008). CUDA cuts: Fast graph cuts on the GPU. In *CVPR 2008 Workshop on Visual Computer Vision on GPUs (CVGPU)*, Anchorage, AK.
- Viola, P. and Wells III, W. (1997). Alignment by maximization of mutual information. *International Journal of Computer Vision*, 24(2):137–154.
- Viola, P., Jones, M. J., and Snow, D. (2003). Detecting pedestrians using patterns of motion and appearance. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 734–741, Nice, France.
- Viola, P. A. and Jones, M. J. (2004). Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154.
- Vlasic, D., Baran, I., Matusik, W., and Popović, J. (2008). Articulated mesh animation from multi-view silhouettes. *ACM Transactions on Graphics*, 27(3).

- Vlasic, D., Brand, M., Pfister, H., and Popović, J. (2005). Face transfer with multilinear models. *ACM Transactions on Graphics (Proc. SIGGRAPH 2005)*, 24(3):426–433.
- Vogiatzis, G., Torr, P., and Cipolla, R. (2005). Multi-view stereo via volumetric graph-cuts. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 391–398, San Diego, CA.
- Vogiatzis, G., Hernandez, C., Torr, P., and Cipolla, R. (2007). Multi-view stereo via volumetric graph-cuts and occlusion robust photo-consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2241–2246.
- von Ahn, L. and Dabbish, L. (2004). Labeling images with a computer game. In *CHI'04: SIGCHI Conference on Human Factors in Computing Systems*, pp. 319–326, Vienna, Austria.
- von Ahn, L., Liu, R., and Blum, M. (2006). Peekaboom: A game for locating objects in images. In *CHI'06: SIGCHI Conference on Human Factors in Computing Systems*, pp. 55–64, Montréal, Québec, Canada.
- Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305.
- Wainwright, M. J., Jaakkola, T. S., and Willsky, A. S. (2005). MAP estimation via agreement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory*, 51(11):3697–3717.
- Waite, P. and Ferrie, F. (1991). From uncertainty to visual exploration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):1038–1049.
- Walker, E. L. and Herman, M. (1988). Geometric reasoning for constructing 3D scene descriptions from images. *Artificial Intelligence*, 37:275–290.
- Wallace, G. K. (1991). The JPEG still picture compression standard. *Communications of the ACM*, 34(4):30–44.
- Wallace, J. R., Cohen, M. F., and Greenberg, D. P. (1987). A two-pass solution to the rendering equation: A synthesis of ray tracing and radiosity methods. *Computer Graphics (SIGGRAPH '87)*, 21(4):311–320.
- Waltz, D. L. (1975). Understanding line drawings of scenes with shadows. In Winston, P. H. (ed.), *The Psychology of Computer Vision*, McGraw-Hill, New York.
- Wang, H. and Oliensis, J. (2010). Shape matching by segmentation averaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):619–635.
- Wang, J. and Cohen, M. F. (2005). An iterative optimization approach for unified image segmentation and matting. In *Tenth International Conference on Computer Vision (ICCV 2005)*, Beijing, China.
- Wang, J. and Cohen, M. F. (2007a). Image and video matting: A survey. *Foundations and Trends in Computer Graphics and Computer Vision*, 3(2).
- Wang, J. and Cohen, M. F. (2007b). Optimized color sampling for robust matting. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Wang, J. and Cohen, M. F. (2007c). Simultaneous matting and compositing. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Wang, J., Agrawala, M., and Cohen, M. F. (2007). Soft scissors: An interactive tool for realtime high quality matting. *ACM Transactions on Graphics*, 26(3).
- Wang, J., Thiesson, B., Xu, Y., and Cohen, M. (2004). Image and video segmentation by anisotropic kernel mean shift. In *Eighth European Conference on Computer Vision (ECCV 2004)*, pp. 238–249, Prague.

- Wang, J., Bhat, P., Colburn, R. A., Agrawala, M., and Cohen, M. F. (2005). Video cutout. *ACM Transactions on Graphics (Proc. SIGGRAPH 2005)*, 24(3):585–594.
- Wang, J. Y. A. and Adelson, E. H. (1994). Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638.
- Wang, L., Kang, S. B., Szeliski, R., and Shum, H.-Y. (2001). Optimal texture map reconstruction from multiple views. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2001)*, pp. 347–354, Kauai, Hawaii.
- Wang, Y. and Zhu, S.-C. (2003). Modeling textured motion: Particle, wave and sketch. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 213–220, Nice, France.
- Wang, Z., Bovik, A. C., and Simoncelli, E. P. (2005). Structural approaches to image quality assessment. In Bovik, A. C. (ed.), *Handbook of Image and Video Processing*, pp. 961–974, Elsevier Academic Press.
- Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612.
- Wang, Z.-F. and Zheng, Z.-G. (2008). A region based stereo matching algorithm using cooperative optimization. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Ward, G. (1992). Measuring and modeling anisotropic reflection. *Computer Graphics (SIGGRAPH '92)*, 26(4):265–272.
- Ward, G. (1994). The radiance lighting simulation and rendering system. In *ACM SIGGRAPH 1994 Conference Proceedings*, pp. 459–472.
- Ward, G. (2003). Fast, robust image registration for compositing high dynamic range photographs from hand-held exposures. *Journal of Graphics Tools*, 8(2):17–30.
- Ward, G. (2004). High dynamic range image encodings. http://www.anyhere.com/gward/hdrenc/hdr_encodings.html.
- Ware, C., Arthur, K., and Booth, K. S. (1993). Fish tank virtual reality. In *INTERCHI'03*, pp. 37–42, Amsterdam.
- Warren, J. and Weimer, H. (2001). *Subdivision Methods for Geometric Design: A Constructive Approach*. Morgan Kaufmann.
- Watanabe, M. and Nayar, S. K. (1998). Rational filters for passive depth from defocus. *International Journal of Computer Vision*, 27(3):203–225.
- Watt, A. (1995). *3D Computer Graphics*. Addison-Wesley, Harlow, England, third edition.
- Weber, J. and Malik, J. (1995). Robust computation of optical flow in a multi-scale differential framework. *International Journal of Computer Vision*, 14(1):67–81.
- Weber, M., Welling, M., and Perona, P. (2000). Unsupervised learning of models for recognition. In *Sixth European Conference on Computer Vision (ECCV 2000)*, pp. 18–32, Dublin, Ireland.
- Wedel, A., Cremers, D., Pock, T., and Bischof, H. (2009). Structure- and motion-adaptive regularization for high accuracy optic flow. In *Twelfth International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan.
- Wedel, A., Rabe, C., Vaudrey, T., Brox, T., Franke, U., and Cremers, D. (2008). Efficient dense scene flow from sparse or dense stereo data. In *Tenth European Conference on Computer Vision (ECCV 2008)*, pp. 739–751, Marseilles.

- Wei, C. Y. and Quan, L. (2004). Region-based progressive stereo matching. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2005)*, pp. 106–113, Washington, D. C.
- Wei, L.-Y. and Levoy, M. (2000). Fast texture synthesis using tree-structured vector quantization. In *ACM SIGGRAPH 2000 Conference Proceedings*, pp. 479–488.
- Weickert, J. (1998). *Anisotropic Diffusion in Image Processing*. Tübner, Stuttgart.
- Weickert, J., ter Haar Romeny, B. M., and Viergever, M. A. (1998). Efficient and reliable schemes for nonlinear diffusion filtering. *IEEE Transactions on Image Processing*, 7(3):398–410.
- Weinland, D., Ronfard, R., and Boyer, E. (2006). Free viewpoint action recognition using motion history volumes. *Computer Vision and Image Understanding*, 104(2-3):249–257.
- Weiss, Y. (1997). Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pp. 520–526, San Juan, Puerto Rico.
- Weiss, Y. (1999). Segmentation using eigenvectors: A unifying view. In *Seventh International Conference on Computer Vision (ICCV'99)*, pp. 975–982, Kerkyra, Greece.
- Weiss, Y. (2001). Deriving intrinsic images from image sequences. In *Eighth International Conference on Computer Vision (ICCV 2001)*, pp. 7–14, Vancouver, Canada.
- Weiss, Y. and Adelson, E. H. (1996). A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'96)*, pp. 321–326, San Francisco.
- Weiss, Y. and Freeman, B. (2007). What makes a good model of natural images? In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Weiss, Y. and Freeman, W. T. (2001a). Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Neural Computation*, 13(10):2173–2200.
- Weiss, Y. and Freeman, W. T. (2001b). On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47(2):736–744.
- Weiss, Y., Torralba, A., and Fergus, R. (2008). Spectral hashing. In *Advances in Neural Information Processing Systems*.
- Weiss, Y., Yanover, C., and Meltzer, T. (2010). Linear programming and variants of belief propagation. In Blake, A., Kohli, P., and Rother, C. (eds), *Advances in Markov Random Fields*, MIT Press.
- Wells, III, W. M. (1986). Efficient synthesis of Gaussian filters by cascaded uniform filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(2):234–239.
- Weng, J., Ahuja, N., and Huang, T. S. (1993). Optimal motion and structure estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):864–884.
- Wenger, A., Gardner, A., Tchou, C., Unger, J., Hawkins, T., and Debevec, P. (2005). Performance relighting and reflectance transformation with time-multiplexed illumination. *ACM Transactions on Graphics (Proc. SIGGRAPH 2005)*, 24(3):756–764.
- Werlberger, M., Trobin, W., Pock, T., Bischof, H., Wedel, A., and Cremers, D. (2009). Anisotropic Huber-L1 optical flow. In *British Machine Vision Conference (BMVC 2009)*, London.
- Werner, T. (2007). A linear programming approach to max-sum problem: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1165–1179.

- Werner, T. and Zisserman, A. (2002). New techniques for automated architectural reconstruction from photographs. In *Seventh European Conference on Computer Vision (ECCV 2002)*, pp. 541–555, Copenhagen.
- Westin, S. H., Arvo, J. R., and Torrance, K. E. (1992). Predicting reflectance functions from complex surfaces. *Computer Graphics (SIGGRAPH '92)*, 26(4):255–264.
- Westover, L. (1989). Interactive volume rendering. In *Workshop on Volume Visualization*, pp. 9–16, Chapel Hill.
- Wexler, Y., Fitzgibbon, A., and Zisserman, A. (2002). Bayesian estimation of layers from multiple images. In *Seventh European Conference on Computer Vision (ECCV 2002)*, pp. 487–501, Copenhagen.
- Wexler, Y., Shechtman, E., and Irani, M. (2007). Space-time completion of video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):463–476.
- Weyrich, T., Lawrence, J., Lensch, H. P. A., Rusinkiewicz, S., and Zickler, T. (2008). Principles of appearance acquisition and representation. *Foundations and Trends in Computer Graphics and Computer Vision*, 4(2):75–191.
- Weyrich, T., Matusik, W., Pfister, H., Bickel, B., Donner, C. et al. (2006). Analysis of human faces using a measurement-based skin reflectance model. *ACM Transactions on Graphics*, 25(3):1013–1024.
- Wheeler, M. D., Sato, Y., and Ikeuchi, K. (1998). Consensus surfaces for modeling 3D objects from multiple range images. In *Sixth International Conference on Computer Vision (ICCV'98)*, pp. 917–924, Bombay.
- White, R. and Forsyth, D. (2006). Combining cues: Shape from shading and texture. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2006)*, pp. 1809–1816, New York City, NY.
- White, R., Crane, K., and Forsyth, D. A. (2007). Capturing and animating occluded cloth. *ACM Transactions on Graphics*, 26(3).
- Wiejak, J. S., Buxton, H., and Buxton, B. F. (1985). Convolution with separable masks for early image processing. *Computer Vision, Graphics, and Image Processing*, 32(3):279–290.
- Wilburn, B., Joshi, N., Vaish, V., Talvala, E.-V., Antunez, E. et al. (2005). High performance imaging using large camera arrays. *ACM Transactions on Graphics (Proc. SIGGRAPH 2005)*, 24(3):765–776.
- Wilczkowiak, M., Brostow, G. J., Tordoff, B., and Cipolla, R. (2005). Hole filling through photomontage. In *British Machine Vision Conference (BMVC 2005)*, pp. 492–501, Oxford Brookes.
- Williams, D. and Burns, P. D. (2001). Diagnostics for digital capture using MTF. In *IS&T PICS Conference*, pp. 227–232.
- Williams, D. J. and Shah, M. (1992). A fast algorithm for active contours and curvature estimation. *Computer Vision, Graphics, and Image Processing*, 55(1):14–26.
- Williams, L. (1983). Pyramidal parametrics. *Computer Graphics (SIGGRAPH '83)*, 17(3):1–11.
- Williams, L. (1990). Performance driven facial animation. *Computer Graphics (SIGGRAPH '90)*, 24(4):235–242.
- Williams, O., Blake, A., and Cipolla, R. (2003). A sparse probabilistic learning algorithm for real-time tracking. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 353–360, Nice, France.
- Williams, T. L. (1999). *The Optical Transfer Function of Imaging Systems*. Institute of Physics Publishing, London.

- Winder, S. and Brown, M. (2007). Learning local image descriptors. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Winkenbach, G. and Salesin, D. H. (1994). Computer-generated pen-and-ink illustration. In *ACM SIGGRAPH 1994 Conference Proceedings*, pp. 91–100, Orlando, Florida.
- Winn, J. and Shotton, J. (2006). The layout consistent random field for recognizing and segmenting partially occluded objects. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2006)*, pp. 37–44, New York City, NY.
- Winnemöller, H., Olsen, S. C., and Gooch, B. (2006). Real-time video abstraction. *ACM Transactions on Graphics*, 25(3):1221–1226.
- Winston, P. H. (ed.). (1975). *The Psychology of Computer Vision*, McGraw-Hill, New York.
- Wiskott, L., Fellous, J.-M., Krüger, N., and von der Malsburg, C. (1997). Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):775–779.
- Witkin, A. (1981). Recovering surface shape and orientation from texture. *Artificial Intelligence*, 17(1-3):17–45.
- Witkin, A. (1983). Scale-space filtering. In *Eighth International Joint Conference on Artificial Intelligence (IJCAI-83)*, pp. 1019–1022.
- Witkin, A., Terzopoulos, D., and Kass, M. (1986). Signal matching through scale space. In *Fifth National Conference on Artificial Intelligence (AAAI-86)*, pp. 714–719, Philadelphia.
- Witkin, A., Terzopoulos, D., and Kass, M. (1987). Signal matching through scale space. *International Journal of Computer Vision*, 1:133–144.
- Wolberg, G. (1990). *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos.
- Wolberg, G. and Pavlidis, T. (1985). Restoration of binary images using stochastic relaxation with annealing. *Pattern Recognition Letters*, 3:375–388.
- Wolff, L. B., Shafer, S. A., and Healey, G. E. (eds). (1992a). *Radiometry. Physics-Based Vision: Principles and Practice*, Jones & Bartlett, Cambridge, MA.
- Wolff, L. B., Shafer, S. A., and Healey, G. E. (eds). (1992b). *Shape Recovery. Physics-Based Vision: Principles and Practice*, Jones & Bartlett, Cambridge, MA.
- Wood, D. N., Finkelstein, A., Hughes, J. F., Thayer, C. E., and Salesin, D. H. (1997). Multiperspective panoramas for cel animation. In *ACM SIGGRAPH 1997 Conference Proceedings*, pp. 243–250, Los Angeles.
- Wood, D. N., Azuma, D. I., Aldinger, K., Curless, B., Duchamp, T., Salesin, D. H., and Stuetzle, W. (2000). Surface light fields for 3D photography. In *ACM SIGGRAPH 2000 Conference Proceedings*, pp. 287–296.
- Woodford, O., Reid, I., Torr, P. H., and Fitzgibbon, A. (2008). Global stereo reconstruction under second order smoothness priors. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Woodham, R. J. (1981). Analysing images of curved surfaces. *Artificial Intelligence*, 17:117–140.
- Woodham, R. J. (1994). Gradient and curvature from photometric stereo including local confidence estimation. *Journal of the Optical Society of America A*, 11:3050–3068.
- Wren, C. R., Azarbayejani, A., Darrell, T., and Pentland, A. P. (1997). Pfnder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785.
- Wright, S. (2006). *Digital Compositing for Film and Video*. Focal Press, 2nd edition.

- Wu, C. (2010). SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). <http://www.cs.unc.edu/~ccwu/siftgpu/>.
- Wyszecki, G. and Stiles, W. S. (2000). *Color Science: Concepts and Methods, Quantitative Data and Formulae*. John Wiley & Sons, New York, 2nd edition.
- Xiao, J. and Shah, M. (2003). Two-frame wide baseline matching. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 603–609, Nice, France.
- Xiao, J. and Shah, M. (2005). Motion layer extraction in the presence of occlusion using graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1644–1659.
- Xiong, Y. and Turkowski, K. (1997). Creating image-based VR using a self-calibrating fisheye lens. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pp. 237–243, San Juan, Puerto Rico.
- Xiong, Y. and Turkowski, K. (1998). Registration, calibration and blending in creating high quality panoramas. In *IEEE Workshop on Applications of Computer Vision (WACV'98)*, pp. 69–74, Princeton.
- Xu, L., Chen, J., and Jia, J. (2008). A segmentation based variational model for accurate optical flow estimation. In *Tenth European Conference on Computer Vision (ECCV 2008)*, pp. 671–684, Marseilles.
- Yang, D., El Gamal, A., Fowler, B., and Tian, H. (1999). A 640x512 CMOS image sensor with ultra-wide dynamic range floating-point pixel level ADC. *IEEE Journal of Solid State Circuits*, 34(12):1821–1834.
- Yang, L. and Albregtsen, F. (1996). Fast and exact computation of Cartesian geometric moments using discrete Green's theorem. *Pattern Recognition*, 29(7):1061–1073.
- Yang, L., Meer, P., and Foran, D. (2007). Multiple class segmentation using a unified framework over mean-shift patches. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Yang, L., Jin, R., Sukthankar, R., and Jurie, F. (2008). Unifying discriminative visual codebook generation with classifier training for object category recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Yang, M.-H., Ahuja, N., and Tabb, M. (2002). Extraction of 2D motion trajectories and its application to hand gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1061–1074.
- Yang, M.-H., Kriegman, D. J., and Ahuja, N. (2002). Detecting faces in images: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):34–58.
- Yang, Q., Wang, L., Yang, R., Stewénius, H., and Nistér, D. (2009). Stereo matching with color-weighted correlation, hierarchical belief propagation and occlusion handling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(3):492–504.
- Yang, Y., Yuille, A., and Lu, J. (1993). Local, global, and multilevel stereo matching. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'93)*, pp. 274–279, New York.
- Yanover, C., Meltzer, T., and Weiss, Y. (2006). Linear programming relaxations and belief propagation — an empirical study. *Journal of Machine Learning Research*, 7:1887–1907.
- Yao, B. Z., Yang, X., Lin, L., Lee, M. W., and Zhu, S.-C. (2010). I2T: Image parsing to text description. *Proceedings of the IEEE*, 98(8):1485–1508.
- Yaou, M.-H. and Chang, W.-T. (1994). Fast surface interpolation using multiresolution wavelets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(7):673–689.

- Yatziv, L. and Sapiro, G. (2006). Fast image and video colorization using chrominance blending. *IEEE Transactions on Image Processing*, 15(5):1120–1129.
- Yedidia, J. S., Freeman, W. T., and Weiss, Y. (2001). Understanding belief propagation and its generalization. In *International Joint Conference on Artificial Intelligence (IJCAI 2001)*.
- Yezzi, Jr., A. J., Kichenassamy, S., Kumar, A., Olver, P., and Tannenbaum, A. (1997). A geometric snake model for segmentation of medical imagery. *IEEE Transactions on Medical Imaging*, 16(2):199–209.
- Yilmaz, A. and Shah, M. (2006). Matching actions in presence of camera motion. *Computer Vision and Image Understanding*, 104(2-3):221–231.
- Yilmaz, A., Javed, O., and Shah, M. (2006). Object tracking: A survey. *ACM Computing Surveys*, 38(4).
- Yin, P., Criminisi, A., Winn, J., and Essa, I. (2007). Tree-based classifiers for bilayer video segmentation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Yoon, K.-J. and Kweon, I.-S. (2006). Adaptive support-weight approach for correspondence search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):650–656.
- Yserentant, H. (1986). On the multi-level splitting of finite element spaces. *Numerische Mathematik*, 49:379–412.
- Yu, S. X. and Shi, J. (2003). Multiclass spectral clustering. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 313–319, Nice, France.
- Yu, Y. and Malik, J. (1998). Recovering photometric properties of architectural scenes from photographs. In *ACM SIGGRAPH 1998 Conference Proceedings*, pp. 207–218, Orlando.
- Yu, Y.,Debevec, P., Malik, J., and Hawkins, T. (1999). Inverse global illumination: Recovering reflectance models of real scenes from photographs. In *ACM SIGGRAPH 1999 Conference Proceedings*, pp. 215–224.
- Yuan, L., Sun, J., Quan, L., and Shum, H.-Y. (2007). Image deblurring with blurred/noisy image pairs. *ACM Transactions on Graphics*, 26(3).
- Yuan, L., Sun, J., Quan, L., and Shum, H.-Y. (2008). Progressive inter-scale and intra-scale non-blind image deconvolution. *ACM Transactions on Graphics*, 27(3).
- Yuan, L., Wen, F., Liu, C., and Shum, H.-Y. (2004). Synthesizing dynamic texture with closed-loop linear dynamic system. In *Eighth European Conference on Computer Vision (ECCV 2004)*, pp. 603–616, Prague.
- Yuille, A. (1991). Deformable templates for face recognition. *Journal of Cognitive Neuroscience*, 3(1):59–70.
- Yuille, A. (2002). CCCP algorithms to minimize the Bethe and Kikuchi free energies: Convergent alternatives to belief propagation. *Neural Computation*, 14(7):1691–1722.
- Yuille, A. (2010). Loopy belief propagation, mean-field and Bethe approximations. In Blake, A., Kohli, P., and Rother, C. (eds), *Advances in Markov Random Fields*, MIT Press.
- Yuille, A. and Poggio, T. (1984). A Generalized Ordering Constraint for Stereo Correspondence. A. I. Memo 777, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.
- Yuille, A., Vincent, L., and Geiger, D. (1992). Statistical morphology and Bayesian reconstruction. *Journal of Mathematical Imaging and Vision*, 1(3):223–238.
- Zabih, R. and Woodfill, J. (1994). Non-parametric local transforms for computing visual correspondence. In *Third European Conference on Computer Vision (ECCV'94)*, pp. 151–158, Stockholm, Sweden.

- Zach, C. (2008). Fast and high quality fusion of depth maps. In *Fourth International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT'08)*, Atlanta.
- Zach, C., Gallup, D., and Frahm, J.-M. (2008). Fast gain-adaptive KLT tracking on the GPU. In *CVPR 2008 Workshop on Visual Computer Vision on GPUs (CVGPU)*, Anchorage, AK.
- Zach, C., Klopschitz, M., and Pollefeys, M. (2010). Disambiguating visual relations using loop constraints. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2010)*, San Francisco, CA.
- Zach, C., Pock, T., and Bischof, H. (2007a). A duality based approach for realtime TV-L1 optical flow. In *Pattern Recognition (DAGM 2007)*.
- Zach, C., Pock, T., and Bischof, H. (2007b). A globally optimal algorithm for robust TV- L^1 range image integration. In *Eleventh International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil.
- Zanella, V. and Fuentes, O. (2004). An approach to automatic morphing of face images in frontal view. In *Mexican International Conference on Artificial Intelligence (MICAI 2004)*, pp. 679–687, Mexico City.
- Zebedin, L., Bauer, J., Karner, K., and Bischof, H. (2008). Fusion of feature- and area-based information for urban buildings modeling from aerial imagery. In *Tenth European Conference on Computer Vision (ECCV 2008)*, pp. 873–886, Marseilles.
- Zelnik-Manor, L. and Perona, P. (2007). Automating joiners. In *Symposium on Non Photorealistic Animation and Rendering*, Annecy.
- Zhang, G., Jia, J., Wong, T.-T., and Bao, H. (2008). Recovering consistent video depth maps via bundle optimization. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Zhang, J., McMillan, L., and Yu, J. (2006). Robust tracking and stereo matching under variable illumination. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2006)*, pp. 871–878, New York City, NY.
- Zhang, J., Marszalek, M., Lazebnik, S., and Schmid, C. (2007). Local features and kernels for classification of texture and object categories: a comprehensive study. *International Journal of Computer Vision*, 73(2):213–238.
- Zhang, L., Curless, B., and Seitz, S. (2003). Spacetime stereo: Shape recovery for dynamic scenes. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2003)*, pp. 367–374, Madison, WI.
- Zhang, L., Dugas-Phocion, G., Samson, J.-S., and Seitz, S. M. (2002). Single view modeling of free-form scenes. *Journal of Visualization and Computer Animation*, 13(4):225–235.
- Zhang, L., Snavely, N., Curless, B., and Seitz, S. M. (2004). Spacetime faces: High resolution capture for modeling and animation. *ACM Transactions on Graphics*, 23(3):548–558.
- Zhang, R., Tsai, P.-S., Cryer, J. E., and Shah, M. (1999). Shape from shading: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):690–706.
- Zhang, Y. and Kambhamettu, C. (2003). On 3D scene flow and structure recovery from multiview image sequences. *IEEE Transactions on Systems, Man, and Cybernetics*, 33(4):592–606.
- Zhang, Z. (1994). Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2):119–152.

- Zhang, Z. (1998a). Determining the epipolar geometry and its uncertainty: A review. *International Journal of Computer Vision*, 27(2):161–195.
- Zhang, Z. (1998b). On the optimization criteria used in two-view motion analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(7):717–729.
- Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334.
- Zhang, Z. and He, L.-W. (2007). Whiteboard scanning and image enhancement. *Digital Signal Processing*, 17(2):414–432.
- Zhang, Z. and Shan, Y. (2000). A progressive scheme for stereo matching. In *Second European Workshop on 3D Structure from Multiple Images of Large-Scale Environments (SMILE 2000)*, pp. 68–85, Dublin, Ireland.
- Zhang, Z., Deriche, R., Faugeras, O., and Luong, Q. (1995). A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence*, 78:87–119.
- Zhao, G. and Pietikäinen, M. (2007). Dynamic texture recognition using local binary patterns with an application to facial expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):915–928.
- Zhao, W., Chellappa, R., Phillips, P. J., and Rosenfeld, A. (2003). Face recognition: A literature survey. *ACM Computing Surveys*, 35(4):399–358.
- Zheng, J. Y. (1994). Acquiring 3-D models from sequences of contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):163–178.
- Zheng, K. C., Kang, S. B., Cohen, M., and Szeliski, R. (2007). Layered depth panoramas. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, Minneapolis, MN.
- Zheng, Y., Lin, S., and Kang, S. B. (2006). Single-image vignetting correction. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2006)*, pp. 461–468, New York City, NY.
- Zheng, Y., Yu, J., Kang, S.-B., Lin, S., and Kambhamettu, C. (2008). Single-image vignetting correction using radial gradient symmetry. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, Anchorage, AK.
- Zheng, Y., Zhou, X. S., Georgescu, B., Zhou, S. K., and Comaniciu, D. (2006). Example based non-rigid shape detection. In *Ninth European Conference on Computer Vision (ECCV 2006)*, pp. 423–436.
- Zheng, Y.-T., Zhao, M., Song, Y., Adam, H., Buddemeier, U., Bissacco, A., Brucher, F., Chua, T.-S., and Neven, H. (2009). Tour the world: building a web-scale landmark recognition engine. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, Miami Beach, FL.
- Zhong, J. and Sclaroff, S. (2003). Segmenting foreground objects from a dynamic, textured background via a robust Kalman filter. In *Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 44–50, Nice, France.
- Zhou, C., Lin, S., and Nayar, S. (2009). Coded aperture pairs for depth from defocus. In *Twelfth International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan.
- Zhou, Y., Gu, L., and Zhang, H.-J. (2003). Bayesian tangent shape model: Estimating shape and pose parameters via Bayesian inference. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2003)*, pp. 109–116, Madison, WI.
- Zhu, L., Chen, Y., Lin, Y., Lin, C., and Yuille, A. (2008). Recursive segmentation and recognition templates for 2D parsing. In *Advances in Neural Information Processing Systems*.

- Zhu, S.-C. and Mumford, D. (2006). A stochastic grammar of images. *Foundations and Trends in Computer Graphics and Computer Vision*, 2(4).
- Zhu, S. C. and Yuille, A. L. (1996). Region competition: Unifying snakes, region growing, and Bayes/MDL for multiband image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9):884–900.
- Zhu, Z. and Kanade, T. (2008). Modeling and representations of large-scale 3D scenes. *International Journal of Computer Vision*, 78(2-3):119–120.
- Zisserman, A., Giblin, P. J., and Blake, A. (1989). The information available to a moving observer from specularities. *Image and Vision Computing*, 7(1):38–42.
- Zitnick, C. L. and Kanade, T. (2000). A cooperative algorithm for stereo matching and occlusion detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7):675–684.
- Zitnick, C. L. and Kang, S. B. (2007). Stereo for image-based rendering using image over-segmentation. *International Journal of Computer Vision*, 75(1):49–65.
- Zitnick, C. L., Jojic, N., and Kang, S. B. (2005). Consistent segmentation for optical flow estimation. In *Tenth International Conference on Computer Vision (ICCV 2005)*, pp. 1308–1315, Beijing, China.
- Zitnick, C. L., Kang, S. B., Uyttendaele, M., Winder, S., and Szeliski, R. (2004). High-quality video view interpolation using a layered representation. *ACM Transactions on Graphics (Proc. SIGGRAPH 2004)*, 23(3):600–608.
- Zitov'aa, B. and Flusser, J. (2003). Image registration methods: A survey. *Image and Vision Computing*, 21:997–1000.
- Zoghalmi, I., Faugeras, O., and Deriche, R. (1997). Using geometric corners to build a 2D mosaic from a set of images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pp. 420–425, San Juan, Puerto Rico.
- Zongker, D. E., Werner, D. M., Curless, B., and Salesin, D. H. (1999). Environment matting and compositing. In *ACM SIGGRAPH 1999 Conference Proceedings*, pp. 205–214.
- Zorin, D., Schröder, P., and Sweldens, W. (1996). Interpolating subdivision for meshes with arbitrary topology. In *ACM SIGGRAPH 1997 Conference Proceedings*, pp. 189–192, New Orleans.

Index

- 3D Rotations, *see* Rotations
- 3D alignment, 283
 - absolute orientation, 283, 515
 - orthogonal Procrustes, 283
- 3D photography, 537
- 3D video, 564

- Absolute orientation, 283, 515
- Active appearance model (AAM), 598
- Active contours, 238
- Active illumination, 512
- Active rangefinding, 512
- Active shape model (ASM), 243, 598
- Activity recognition, 534
- Adaptive smoothing, 111
- Affine transforms, 34, 37
- Affinities (segmentation), 260
 - normalizing, 262
- Algebraic multigrid, 254
- Algorithms
 - testing, viii
- Aliasing, 69, 417
- Alignment, *see* Image alignment
- Alpha
 - opacity, 93
 - pre-multiplied, 93
- Alpha matte, 93
- Ambient illumination, 58
- Analog to digital conversion (ADC), 68
- Anisotropic diffusion, 111
- Anisotropic filtering, 148
- Anti-aliasing filter, 70, 417
- Aperture, 62
- Aperture problem, 347

- Applications, 5
 - 3D model reconstruction, 319, 327
 - 3D photography, 537
 - augmented reality, 287, 325
 - automotive safety, 5
 - background replacement, 489
 - biometrics, 588
 - colorization, 442
 - de-interlacing, 364
 - digital heritage, 517
 - document scanning, 379
 - edge editing, 219
 - facial animation, 528
 - flash photography, 434
 - frame interpolation, 368
 - gaze correction, 483
 - head tracking, 483
 - hole filling, 457
 - image restoration, 169
 - image search, 630
 - industrial, 5
 - intelligent photo editing, 621
 - Internet photos, 327
 - location recognition, 609
 - machine inspection, 5
 - match move, 324
 - medical imaging, 5, 268, 358
 - morphing, 152
 - mosaic-based video compression, 383
 - non-photorealistic rendering, 458
 - Optical character recognition (OCR), 5
 - panography, 277
 - performance-driven animation, 209
 - photo pop-up, 623

- Photo Tourism, 548
- Photomontage, 403
- planar pattern tracking, 287
- rotoscoping, 249
- scene completion, 621
- scratch removal, 457
- single view reconstruction, 292
- tonal adjustment, 97
- video denoising, 364
- video stabilization, 354
- video summarization, 383
- video-based walkthroughs, 566
- VideoMouse, 288
- view morphing, 315
- visual effects, 5
- whiteboard scanning, 379
- z-keying, 489
- Arc length parameterization of a curve, 217
- Architectural reconstruction, 524
- Area statistics, 115
 - mean (centroid), 115
 - perimeter, 115
 - second moment (inertia), 115
- Aspect ratio, 47, 48
- Augmented reality, 287, 298, 325
- Auto-calibration, 313
- Automatic gain control (AGC), 67
- Axis/angle representation of rotations, 37
- B-snake, 241
- B-spline, 151, 152, 220, 241, 246, 359
 - cubic, 128
 - multilevel, 518
 - octree, 523
- Background plate, 454
- Background subtraction (maintenance), 531
- Bag of words (keypoints), 612, 639
 - distance metrics, 613
- Band-pass filter, 104
- Bartlett filter, *see* Bilinear kernel
- Bayer pattern (RGB sensor mosaic), 76
 - demosaicing, 76, 440
- Bayes' rule, 124, 159, 667
 - MAP (maximum a posteriori) estimate, 668
 - posterior distribution, 667
- Bayesian modeling, 158, 667
 - MAP estimate, 159, 668
 - matting, 449
 - posterior distribution, 159, 667
 - prior distribution, 159, 667
 - uncertainty, 159
- Belief propagation (BP), 163, 672
 - update rule, 673
- Bias, 91, 339
- Bidirectional Reflectance Distribution Function, *see* BRDF
- Bilateral filter, 110
 - joint, 435
 - range kernel, 110
 - tone mapping, 428
- Bilinear blending, 97
- Bilinear kernel, 103
- Biometrics, 588
- Bipartite problem, 322
- Blind image deconvolution, 437
- Block-based motion estimation
 - (block matching), 341
- Blocks world, 10
- Blue screen matting, 94, 171, 445
- Blur kernel, 62
 - estimation, 416, 463
- Blur removal, 126, 174
- Body color, 57
- Boltzmann distribution, 159, 668
- Boosting, 582
 - AdaBoost algorithm, 584
 - decision stump, 582
 - weak learner, 582
- Border (boundary) effects, 101, 173
- Boundary detection, 215
- Box filter, 103
- Boxlet, 107
- BRDF, 55
 - anisotropic, 56
 - isotropic, 56
 - recovery, 536
 - spatially varying (SVBRDF), 536
- Brightness, 91
- Brightness constancy, 3, 338
- Brightness constancy constraint, 338, 345, 360
- Bundle adjustment, 320

Calibration, *see* Camera calibration
Calibration matrix, 46
Camera calibration, 45, 86
 accuracy, 299
 aliasing, 417
 extrinsic, 46, 284
 intrinsic, 45, 288
 optical blur, 416, 463
 patterns, 289
 photometric, 412
 plumb-line method, 295, 300
 point spread function, 416, 463
 radial distortion, 295
 radiometric, 412, 421, 461
 rotational motion, 293, 298
 slant edge, 417
 vanishing points, 290
 vignetting, 416
Camera matrix, 46, 49
Catadioptric optics, 64
Category-level recognition, 611
 bag of words, 612, 639
 data sets, 631
 part-based, 615
 segmentation, 620
 surveys, 635
CCD, 65
 blooming, 65
Central difference, 104
Chained transformations, 287, 321
Chamfer matching, 113
Characteristic function, 115, 248, 516, 522
Characteristic polynomial, 649
Chirality, 306, 310
Cholesky factorization, 650
 algorithm, 650
 incomplete, 659
 sparse, 657
Chromatic aberration, 63, 301
Chromaticity coordinates, 73
CIE L*a*b*, *see* Color
CIE L*u*v*, *see* Color
CIE XYZ, *see* Color
Circle of confusion, 62
CLAHE, *see* Histogram equalization

Clustering
 agglomerative, 251
 cluster analysis, 237, 268
 divisive, 251
CMOS, 66
Co-vector, 34
Coefficient matrix, 156
Collineation, 37
Color, 71
 balance, 76, 85, 171
 camera, 75
 demosaicing, 76, 440
 fringing, 442
 hue, saturation, value (HSV), 79
 L*a*b*, 74
 L*u*v*, 74, 254
 primaries, 71
 profile, 414
 ratios, 79
 RGB, 72
 transform, 92
 twist, 76, 92
 XYZ, 72
 YIQ, 78
 YUV, 78
Color filter array (CFA), 76, 440
Color line model, 450
ColorChecker chart, 414
Colorization, 442
Compositing, 92, 169, 171
 image stitching, 396
 opacity, 93
 over operator, 93
 surface, 396
 transparency, 93
Compression, 80
Computational photography, 409
 active illumination, 436
 flash and non-flash, 434
 high dynamic range, 419
 references, 411, 460
 tone mapping, 427
Concentric mosaic, 384, 556
CONDENSATION, 246
Condition number, 657

- Conditional random field (CRF), 165, 484, 621
 Confusion matrix (table), 201
 Conic section, 31
 Conjugate gradient descent (CG), 657
 algorithm, 658
 non-linear, 658
 preconditioned, 659
 Connected components, 115, 174
 Constellation model, 618
 Content based image retrieval (CBIR), 630
 Continuation method, 158
 Contour
 arc length parameterization, 217
 chain code, 217
 matching, 218, 231
 smoothing, 218
 Contrast, 91
 Controlled-continuity spline, 155
 Convolution, 100
 kernel, 98
 mask, 98
 superposition, 100
 Coring, 134, 177
 Correlation, 98, 340
 windowed, 342
 Correspondence map, 350
 Cramer–Rao lower bound, 282, 349, 678
 Cube map
 Hough transform, 223
 image stitching, 396
 Curve
 arc length parameterization, 217
 evolution, 218
 matching, 218
 smoothing, 218
 Cylindrical coordinates, 385
 Data energy (term), 159, 668
 Data sets and test databases, 680
 recognition, 631
 De-interlacing, 364
 Decimation, 130
 Decimation kernels
 bicubic, 131
 binomial, 130, 132
 QMF, 131
 windowed sinc, 130
 Demosaicing (Bayer), 76, 440
 Depth from defocus, 511
 Depth map, *see* Disparity map
 Depth of field, 62, 83
 Di-chromatic reflection model, 60
 Difference matting (keying), 94, 172, 446, 531
 Difference of Gaussians (DoG), 135
 Difference of low-pass (DOLP), 135
 Diffuse reflection, 57
 Diffusion
 anisotropic, 111
 Digital camera, 65
 color, 75
 color filter array (CFA), 76
 compression, 80
 Direct current (DC), 81
 Direct linear transform (DLT), 284
 Direct sparse matrix techniques, 655
 Directional derivative, 105
 selectivity, 106
 Discrete cosine transform (DCT), 81, 125
 Discrete Fourier transform (DFT), 118
 Discriminative random field (DRF), 167
 Disparity, 45, 473
 Disparity map, 473, 492
 multiple, 491
 Disparity space image (DSI), 473
 generalized, 475
 Displaced frame difference (DFD), 338
 Displacement field, 150
 Distance from face space (DFFS), 590
 Distance in face space (DIFS), 590
 Distance map, *see* Distance transform
 Distance transform, 113, 174
 Euclidean, 114
 image stitching, 398
 Manhattan (city block), 113
 signed, 114
 Domain (of a function), 91
 Domain scaling law, 147
 Downsampling, *see* Decimation
 Dynamic programming (DP), 485, 670
 monotonicity, 487
 ordering constraint, 487

- scanline optimization, 487
- Dynamic snake, 243
- Dynamic texture, 563
- Earth mover's distance (EMD), 613
- Edge detection, 210, 230
 - boundary detection, 215
 - Canny, 211
 - chain code, 217
 - color, 214
 - Difference of Gaussian, 212
 - edgel (edge element), 212
 - hysteresis, 217
 - Laplacian of Gaussian, 212
 - linking, 215, 231
 - marching cubes, 213
 - scale selection, 213
 - steerable filter, 213
 - zero crossing, 212
- Eigenface, 589
- Eigenvalue decomposition, 242, 589, 647
- Eigenvector, 647
- Elastic deformations, 358
 - image registration, 358
- Elastic nets, 239
- Elliptical weighted average (EWA), 148
- Environment map, 55, 555
- Environment matte, 556
- Epanechnikov kernel, 259
- Epipolar constraint, 307
- Epipolar geometry, 307, 471
 - pure rotation, 311
 - pure translation, 311
- Epipolar line, 471
- Epipolar plane, 471, 477
 - image (EPI), 490, 552
- Epipolar volume, 552
- Epipole, 308, 471
- Error rates
 - accuracy (ACC), 202
 - false negative (FN), 201
 - false positive (FP), 201
 - positive predictive value (PPV), 202
 - precision, 202
 - recall, 202
 - ROC curve, 202
- true negative (TN), 201
- true positive (TP), 201
- Errors-in-variable model, 389, 653
 - heteroscedastic, 654
- Essential matrix, 308
 - 5-point algorithm, 310
 - eight-point algorithm, 308
 - re-normalization, 309
 - seven-point algorithm, 309
 - twisted pair, 310
- Estimation theory, 662
- Euclidean transformation, 33, 36
- Euler angles, 37
- Expectation maximization (EM), 256
- Exponential twist, 39
- Exposure bracketing, 421
- Exposure value (EV), 62, 412
- F-number (stop), 62, 84
- Face detection, 578
 - boosting, 582
 - cascade of classifiers, 583
 - clustering and PCA, 580
 - data sets, 631
 - neural networks, 580
 - support vector machines, 582
- Face modeling, 526
- Face recognition, 588
 - active appearance model, 598
 - data sets, 631
 - eigenface, 589
 - elastic bunch graph matching, 596
 - local binary patterns (LBP), 635
 - local feature analysis, 596
- Face transfer, 561
- Facial motion capture, 528, 530, 561
- Factor graph, 160, 669, 672
- Factorization, 14, 315
 - missing data, 318
 - projective, 318
- Fast Fourier transform (FFT), 118
- Fast marching method (FMM), 248
- Feature descriptor, 196, 229
 - bias and gain normalization, 196
 - GLOH, 198
 - patch, 196

- PCA-SIFT, 197
- performance (evaluation), 198
- quantization, 207, 607, 612
- SIFT, 197
- steerable filter, 198
- Feature detection, 183, 185, 228
 - Adaptive non-maximal suppression, 189
 - affine invariance, 194
 - auto-correlation, 185
 - Förstner, 188
 - Harris, 188
 - Laplacian of Gaussian, 191
 - MSER, 195
 - region, 195
 - repeatability, 190
 - rotation invariance, 193
 - scale invariance, 191
- Feature matching, 183, 200, 229
 - densification, 207
 - efficiency, 205
 - error rates, 201
 - hashing, 205
 - indexing structure, 205
 - k-d trees, 206
 - locality sensitive hashing, 205
 - nearest neighbor, 203
 - strategy, 200
 - verification, 207
- Feature tracking, 207, 230
 - affine, 208
 - learning, 209
- Feature tracks, 315, 327
- Feature-based alignment, 275
 - 2D, 275
 - 3D, 283
 - iterative, 278
 - Jacobian, 276
 - least squares, 275
 - match verification, 603
 - RANSAC, 281
 - robust, 281
- Field of Experts (FoE), 163
- Fill factor, 67
- Fill-in, 322, 656
- Filter
 - adaptive, 111
 - band-pass, 104
 - bilateral, 110
 - directional derivative, 105
 - edge-preserving, 109, 111
 - Laplacian of Gaussian, 104
 - median, 108
 - moving average, 103
 - non-linear, 108
 - separable, 102, 173
 - steerable, 105, 174
- Filter coefficients, 98
- Filter kernel, *see* Kernel
- Finding faces, *see* Face detection
- Finite element analysis, 155
 - stiffness matrix, 156
- Finite impulse response (FIR) filter, 98, 107
- Fisher information matrix, 277, 282, 664, 678
- Fisher's linear discriminant (FLD), 593
- Fisheye lens, 53
- Flash and non-flash merging, 434
- Flash matting, 454
- Flip-book animation, 296
- Flying spot scanner, 514
- Focal length, 47, 48, 61
- Focus, 62
 - shape-from, 511, 539
- Focus of expansion (FOE), 311
- Form factor, 60
- Forward mapping, *see* Forward warping
- Forward warping, 145, 177
- Fourier transform, 116, 174
 - discrete, 118
 - examples, 119
 - magnitude (gain), 117
 - pairs, 119
 - Parseval's Theorem, 119
 - phase (shift), 117
 - power spectrum, 123
 - properties, 118
 - two-dimensional, 123
- Fourier-based motion estimation, 341
 - rotations and scale, 344
- Frame interpolation, 368
- Free-viewpoint video, 564

- Fundamental matrix, 312
estimation, *see* Essential matrix
- Fundamental radiometric relation, 65
- Gain, 91, 339
- Gamma, 92
- Gamma correction, 77, 85
- Gap closing (image stitching), 382
- Garbage matte, 454
- Gaussian kernel, 103
- Gaussian Markov random field (GMRF), 163, 168, 438
- Gaussian mixtures, *see* Mixture of Gaussians
- Gaussian pyramid, 132
- Gaussian scale mixtures (GSM), 163
- Gaze correction, 483
- Geman–McClure function, 338
- Generalized cylinders, 11, 515, 519
- Geodesic active contour, 248
- Geodesic distance (segmentation), 267
- Geometric image formation, 29
- Geometric lens aberrations, 63
- Geometric primitives, 29
homogeneous coordinates, 30
lines, 30, 31
normal vector, 30
normal vectors, 31
planes, 31
points, 30, 31
- Geometric transformations
2D, 33, 145
3D, 36
3D perspective, 37
3D rotations, 37
affine, 34, 37
bilinear, 35
calibration matrix, 46
collineation, 37
Euclidean, 33, 36
forward warping, 145, 177
hierarchy, 34
homography, 34, 37, 50, 379
inverse warping, 146
perspective, 34
projections, 42
projective, 34
rigid body, 33, 36
- scaled rotation, 34, 37
similarity, 34, 37
translation, 33, 36
- Geometry image, 520
- Gesture recognition, 530
- Gibbs distribution, 159, 668
- Gibbs sampler, 670
- Gimbal lock, 37
- Gist (of a scene), 623, 626
- Global illumination, 60
- Global optimization, 153
- GPU algorithms, 688
- Gradient location-orientation histogram (GLOH), 198
- Graduated non-convexity (GNC), 158
- Graph cuts
MRF inference, 161, 674
normalized cuts, 260
- Graph-based segmentation, 252
- Grassfire transform, 114, 219, 398
- Ground control points, 309, 377
- Hammersley–Clifford theorem, 159, 668
- Hann window, 121
- Harris corner detector, *see* Feature detection
- Head tracking, 483
active appearance model (AAM), 598
- Helmholtz reciprocity, 56
- Hessian, 156, 189, 277, 279, 282, 346, 350, 652
eigenvalues, 349
image, 346, 361
inverse, 282, 349, 352
local, 360
patch-based, 351
rank-deficient, 326
reduced motion, 322
sparse, 322, 334, 655
- Heteroscedastic, 277, 654
- Hidden Markov model (HMM), 563
- Hierarchical motion estimation, 341
- High dynamic range (HDR) imaging, 419
formats, 426
tone mapping, 427
- Highest confidence first, 670
- Highest confidence first (HCF), 161
- Hilbert transform pair, 106
- Histogram equalization, 94, 172

- locally adaptive, 96, 172
- Histogram intersection, 613
- Histogram of oriented gradients (HOG), 585
- History of computer vision, 10
- Hole filling, 457
- Homogeneous coordinates, 30, 306
- Homography, 34, 50, 379
- Hough transform, 221, 233
 - cascaded, 224
 - cube map, 223
 - generalized, 222
- Human body shape modeling, 533
- Human motion tracking, 530
 - activity recognition, 534
 - adaptive shape modeling, 533
 - background subtraction, 531
 - flow-based, 531
 - initialization, 531
 - kinematic models, 532
 - particle filtering, 533
 - probabilistic models, 533
- Hyper-Laplacian, 158, 162, 164
- Ideal points, 30
- Ill-posed (ill-conditioned) problems, 154
- Illusions, 3
- Image alignment
 - feature-based, 275, 475
 - intensity-based, 337
 - intensity-based vs. feature-based, 393
- Image analogies, 458
- Image blending
 - feathering, 400
 - GIST, 405
 - gradient domain, 404
 - image stitching, 398
 - Poisson, 404
 - pyramid, 140, 403
- Image compositing, *see* Compositing
- Image compression, 80
- Image decimation, 130
- Image deconvolution, *see* Blur removal
- Image filtering, 98
- Image formation
 - geometric, 29
 - photometric, 54
- Image gradient, 104, 112, 345
 - constraint, 156
- Image interpolation, 127
- Image matting, 443, 464
- Image processing, 89
 - textbooks, 89, 169
- Image pyramid, 127, 175
- Image resampling, 145, 175
 - test images, 176
- Image restoration, 126, 169
 - blur removal, 126, 174, 175
 - deblocking, 179
 - inpainting, 169
 - noise removal, 126, 174, 178
 - using MRFs, 169
- Image search, 630
- Image segmentation, *see* Segmentation
- Image sensing, *see* Sensing
- Image statistics, 115
- Image stitching, 375
 - automatic, 392
 - bundle adjustment, 388
 - compositing, 396
 - coordinate transformations, 397
 - cube map, 396
 - cylindrical, 385, 407
 - de-ghosting, 392, 401, 408
 - direct vs. feature-based, 393
 - exposure compensation, 405
 - feathering, 400
 - gap closing, 382
 - global alignment, 387
 - homography, 379
 - motion models, 378
 - panography, 277
 - parallax removal, 391
 - photogrammetry, 377
 - pixel selection, 398
 - planar perspective motion, 379
 - recognizing panoramas, 392
 - rotational motion, 380
 - seam selection, 400
 - spherical, 385
 - up vector selection, 390
- Image warping, 145, 177, 341

- Image-based modeling, 547
Image-based rendering, 543
 concentric mosaic, 556
 environment matte, 556
 impostors, 549
 layered depth image, 549
 layers, 549
 light field, 551
 Lumigraph, 551
 modeling vs. rendering continuum, 559
 sprites, 549
 surface light field, 555
 unstructured Lumigraph, 554
 view interpolation, 545
 view-dependent texture maps, 547
Image-based visual hull, 499
ImageNet, 629
Implicit surface, 522
Impostors, *see* Sprites
Impulse response, 100
Incremental refinement
 motion estimation, 341, 345
Incremental rotation, 41
Indexing structure, 205
Indicator function, 522
Industrial applications, 5
Infinite impulse response (IIR) filter, 107
Influence function, 158, 281, 666
Information matrix, 277, 282, 326, 664, 678
Inpainting, 457
Instance recognition, 602
 algorithm, 606
 data sets, 631
 geometric alignment, 603
 inverted index, 604
 large scale, 604
 match verification, 603
 query expansion, 608
 stop list, 605
 visual words, 605
 vocabulary tree, 607
Integrability constraint, 509
Integral image, 106
Integrating sphere, 414
Intelligent scissors, 247
Interaction potential, 159, 160, 668, 672
Interactive computer vision, 537
International Color Consortium (ICC), 414
Internet photos, 327
Interpolation, 127
Interpolation kernels
 bicubic, 128
 bilinear, 127
 binomial, 127
 sinc, 130
 spline, 130
Intrinsic camera calibration, 288
Intrinsic images, 11
Inverse kinematics (IK), 532
Inverse mapping, *see* Inverse warping
Inverse problems, 3, 154
Inverse warping, 146
ISO setting, 67
Iterated closest point (ICP), 239, 283, 515
Iterated conditional modes (ICM), 161, 670
Iterative back projection (IBP), 437
Iterative feature-based alignment, 278
Iterative sparse matrix techniques, 656
 conjugate gradient, 657
Iteratively reweighted least squares
 (IRLS), 281, 286, 350, 666
Jacobian, 276, 287, 321, 345, 654
 image, 346
 motion, 350
 sparse, 322, 334, 655
Joint bilateral filter, 435
Joint domain (feature space), 259
K-d trees, 206
K-means, 256
Kalman snakes, 243
Kanade–Lucas–Tomasi (KLT) tracker, 208
Karhunen–Loëve transform, 125, 589
Kernel, 103
 bilinear, 103
 Gaussian, 103
 low-pass, 103
 Sobel operator, 104
 unsharp mask, 103
Kernel basis function, 155

- Kernel density estimation, 257
 Keypoint detection, *see* Feature detection
 Kinematic model (chain), 532
 Kruppa equations, 314
- $L^*a^*b^*$, *see* Color
 $L^*u^*v^*$, *see* Color
 L_1 norm, 158, 338, 361, 523
 L_∞ norm, 324
 Lambertian reflection, 57
 Laplacian matting, 451
 Laplacian of Gaussian (LoG) filter, 104
 Laplacian pyramid, 135
 - blending, 141, 176, 403
 - perfect reconstruction, 135
 Latent Dirichlet process (LDP), 626
 Layered depth image (LDI), 549
 Layered depth panorama, 556
 Layered motion estimation, 365
 - transparent, 368
 Layers
 - image-based rendering, 549
 Layout consistent random field, 621
 Learning in computer vision, 627
 Least median of squares (LMS), 281
 Least squares
 - iterative solvers, 286, 656
 - linear, 83, 275, 283, 337, 648, 651, 662, 665, 687
 - non-linear, 278, 286, 306, 654, 666, 687
 - robust, *see* Robust least squares
 - sparse, 322, 656, 687
 - total, 653
 - weighted, 277, 433, 436, 443
 Lens
 - compound, 63
 - nodal point, 63
 - thin, 61
 Lens distortions, 52
 - calibration, 295
 - decentering, 53
 - radial, 52
 - spline-based, 53
 - tangential, 53
 Lens law, 61
 Level of detail (LOD), 520
 Level sets, 248, 249
- fast marching method, 248
 geodesic active contour, 248
 Levenberg–Marquardt, 279, 326, 334, 655, 684
 Lifting, *see* Wavelets
 Light field
 - higher dimensional, 558
 - light slab, 552
 - ray space, 553
 - rendering, 551
 - surface, 555
 Lightness, 74
 Line at infinity, 30
 Line detection, 220
 - Hough transform, 221, 233
 - RANSAC, 224
 - simplification, 220, 233
 - successive approximation, 221, 233
 Line equation, 30, 31
 Line fitting, 83, 233
 - uncertainty, 233
 Line hull, *see* Visual hull
 Line labeling, 11
 Line process, 170, 484, 669
 Line spread function (LSF), 417
 Line-based structure from motion, 330
 Linear algebra, 645
 - least squares, 651
 - matrix decompositions, 646
 - references, 646
 Linear blend, 91
 Linear discriminant analysis (LDA), 593
 Linear filtering, 98
 Linear operator, 91
 - superposition, 91
 Linear shift invariant (LSI) filter, 100
 Live-wire, 247
 Local distance functions, 596
 Local operator, 98
 Locality sensitive hashing (LSH), 205
 Locally adaptive histogram equalization, 96
 Location recognition, 609
 Loopy belief propagation (LBP), 163, 673
 Low-pass filter, 103
 - sinc, 103
 Lumigraph, 551

- unstructured, 554
- Luminance, 73
- Lumisphere, 555
- M-estimator, 281, 338, 666
- Mahalanobis distance, 256, 591, 594, 663
- Manifold mosaic, 400, 569
- Markov chain Monte Carlo (MCMC), 665, 670
- Markov random field, 158, 668
 - cliques, 160, 669
 - directed edges, 266
 - dynamic, 675
 - flux, 266
 - inference, *see* MRF inference
 - layout consistent, 621
 - learning parameters, 158
 - line process, 170, 484, 669
 - neighborhood, 160, 668
 - order, 160, 669
 - random walker, 267
 - stereo matching, 484
- Marr's framework, 12
 - computational theory, 12
 - hardware implementation, 12
 - representations and algorithms, 12
- Match move, 324
- Matrix decompositions, 646
 - Cholesky, 650
 - eigenvalue (ED), 647
 - QR, 649
 - singular value (SVD), 646
 - square root, 650
- Matte reflection, 57
- Matting, 92, 94, 443, 464
 - alpha matte, 93
 - Bayesian, 449
 - blue screen, 94, 171, 445
 - difference, 94, 172, 446, 531
 - flash, 454
 - GrabCut, 450
 - Laplacian, 451
 - natural, 446
 - optimization-based, 450
 - Poisson, 450
 - shadow, 452
 - smoke, 452
- triangulation, 445, 454
- trimap, 446
- two screen, 445
- video, 454
- Maximally stable extremal region (MSER), 195
- Maximum a posteriori (MAP) estimate, 159, 668
- Mean absolute difference (MAD), 479
- Mean average precision, 202
- Mean shift, 254, 258
 - bandwidth selection, 259
- Mean square error (MSE), 81, 479
- Measurement equation (model), 306, 662
- Measurement matrix, 316
- Measurement model, *see* Bayesian model
- Medial axis transform (MAT), 114
- Median absolute deviation (MAD), 338
- Median filter, 108
 - weighted, 109
- Medical image registration, 358
- Medical image segmentation, 268
- Membrane, 155
- Mesh-based warping, 149, 177
- Metamer, 72
- Metric learning, 596
- Metric tree, 207
- MIP-mapping, 147
 - trilinear, 148
- Mixture of Gaussians, 239, 243, 256
 - color model, 447
 - expectation maximization (EM), 256
 - mixing coefficient, 256
 - soft assignment, 256
- Model selection, 378, 668
- Model-based reconstruction, 523
 - architecture, 524
 - heads and faces, 526
 - human body, 530
- Model-based stereo, 524, 547
- Models
 - Bayesian, 158, 667
 - forward, 3
 - physically based, 13
 - physics-based, 3
 - probabilistic, 3
- Modular eigenspace, 595

- Modulation transfer function (MTF), 70, 417
- Morphable model
- body, 533
 - face, 528, 561
 - multidimensional, 561
- Morphing, 152, 178, 545, 546
- 3D body, 533
 - 3D face, 528
 - automated, 372
 - facial feature, 561
 - feature-based, 152, 178
 - flow-based, 372
 - video textures, 563
 - view morphing, 546, 570
- Morphological operator, 112
- closing, 112
 - dilation, 112
 - erosion, 112
 - opening, 112
- Morphology, 112
- Mosaic, *see* Image stitching
- Mosaics
- motion models, 378
 - video compression, 383
 - whiteboard and document scanning, 379
- Motion compensated video compression, 341, 370
- Motion compensation, 81
- Motion estimation, 337
- affine, 350
 - aperture problem, 347
 - compositional, 351
 - Fourier-based, 341
 - frame interpolation, 368
 - hierarchical, 341
 - incremental refinement, 345
 - layered, 365
 - learning, 354, 361
 - linear appearance variation, 349
 - optical flow, 360
 - parametric, 350
 - patch-based, 337, 351
 - phase correlation, 343
 - quadtree spline-based, 358
 - reflections, 369
 - spline-based, 355
 - translational, 337
 - transparent, 368
 - uncertainty modeling, 347
- Motion field, 350
- Motion models
- learned, 354
- Motion segmentation, 373
- Motion stereo, 491
- Motion-based user interaction, 373
- Moving least squares (MLS), 522
- MRF inference, 161, 669
- alpha expansion, 163, 675
 - belief propagation, 163, 672
 - dynamic programming, 670
 - expansion move, 163, 675
 - gradient descent, 670
 - graph cuts, 161, 674
 - highest confidence first, 161
 - highest confidence first (HCF), 670
 - iterated conditional modes, 161, 670
 - linear programming (LP), 676
 - loopy belief propagation, 163, 673
 - Markov chain Monte Carlo, 670
 - simulated annealing, 161, 670
 - stochastic gradient descent, 161, 670
 - swap move (alpha-beta), 163, 675
 - Swendsen–Wang, 670
- Multi-frame motion estimation, 363
- Multi-pass transforms, 149
- Multi-perspective panoramas, 384
- Multi-perspective plane sweep (MPPS), 391
- Multi-view stereo, 489
- epipolar plane image, 490
 - evaluation, 496
 - initialization requirements, 496
 - reconstruction algorithm, 495
 - scene representation, 493
 - shape priors, 495
 - silhouettes, 497
 - space carving, 496
 - spatio-temporally shiftable window, 491
 - taxonomy, 493
 - visibility, 495
 - volumetric, 492
 - voxel coloring, 495

- Multigrid, 660
algebraic (AMG), 254, 660
- Multiple hypothesis tracking, 243
- Multiple-center-of-projection images, 384, 569
- Multiresolution representation, 132
- Mutual information, 340, 358
- Natural image matting, 446
- Nearest neighbor
distance ratio (NNDR), 203
matching, *see* Feature matching
- Negative posterior log likelihood, 159, 664, 667
- Neighborhood operator, 98, 108
- Neural networks, 580
- Nintendo Wii, 288
- Nodal point, 63
- Noise
sensor, 67, 415
- Noise level function (NLF), 68, 84, 415, 462
- Noise removal, 126, 174, 178
- Non-linear filter, 108, 169
- Non-linear least squares
see Least squares, 278
- Non-maximal suppression, *see* Feature detection
- Non-parametric density modeling, 257
- Non-photorealistic rendering (NPR), 458
- Non-rigid motion, 332
- Normal equations, 277, 346, 652, 654
- Normal map (geometry image), 520
- Normal vector, 31
- Normalized cross-correlation (NCC), 340, 371, 479
- Normalized cuts, 260
intervening contour, 262
- Normalized device coordinates (NDC), 44, 48
- Normalized sum of squared differences
(NSSD), 340
- Norms
 L_1 , 158, 338, 361, 523
 L_∞ , 324
- Nyquist rate / frequency, 69
- Object detection, 578
car, 585, 634
face, 578
part-based, 586
pedestrian, 585, 601
- Object-centered projection, 51
- Occluding contours, 476
- Octree reconstruction, 498
- Octree spline, 359
- Omnidirectional vision systems, 566
- Opacity, 93
- Operator
linearity, 91
- Optic flow, *see* Optical flow
- Optical center, 47
- Optical flow, 360
anisotropic smoothness, 361
evaluation, 363
fusion move, 363
global and local, 360
Markov random field, 361
multi-frame, 363
normal flow, 347
patch-based, 360
region-based, 367
regularization, 360
robust regularization, 361
smoothness, 360
total variation, 361
- Optical flow constraint equation, 345
- Optical illusions, 3
- Optical transfer function (OTF), 70, 416
- Optical triangulation, 513
- Optics, 61
chromatic aberration, 63
Seidel aberrations, 63
vignetting, 64, 462
- Optimal motion estimation, 320
- Oriented particles (points), 521
- Orthogonal Procrustes, 283
- Orthographic projection, 42
- Osculating circle, 477
- Over operator, 93
- Overview, 17
- Padding, 101, 173
- Panography, 277, 297
- Panorama, *see* Image stitching
- Panorama with depth, 384, 475, 556
- Para-perspective projection, 44
- Parallel tracking and mapping (PTAM), 325

- Parameter sensitive hashing, 205
 Parametric motion estimation, 350
 Parametric surface, 519
 Parametric transformation, 145, 177
 Parseval's Theorem, *see* Fourier transform
 Part-based recognition, 615
 constellation model, 618
 Particle filtering, 243, 533, 665
 Parzen window, 257
 PASCAL Visual Object Classes Challenge (VOC), 631
 Patch-based motion estimation, 337
 Peak signal-to-noise Ratio (PSNR), 81, 126
 Pedestrian detection, 585
 Penumbra, 55
 Performance-driven animation, 209, 530, 561
 Perspective n-point problem (PnP), 285
 Perspective projection, 44
 Perspective transform (2D), 34
 Phase correlation, 343, 371
 Phong shading, 58
 Photo pop-up, 623
 Photo Tourism, 548
 Photo-mosaic, 377
 Photoconsistency, 474, 494
 Photometric image formation, 54
 calibration, 412
 global illumination, 60
 lighting, 54
 optics, 61
 radiosity, 60
 reflectance, 55
 shading, 58
 Photometric stereo, 510
 Photometry, 54
 Photomontage, 403
 Physically based models, 13
 Physics-based vision, 15
 Pictorial structures, 11, 17, 616
 Pixel transform, 91
 Plücker coordinates, 32
 Planar pattern tracking, 287
 Plane at infinity, 31
 Plane equation, 31
 Plane plus parallax, 49, 356, 366, 474, 549
 Plane sweep, 474, 501
 Plane-based structure from motion, 331
 Plenoptic function, 551
 Plenoptic modeling, 546
 Plumb-line calibration method, 295, 300
 Point distribution model, 242
 Point operator, 89
 Point process, 89
 Point spread function (PSF), 70
 estimation, 416, 463
 Point-based representations, 521
 Points at infinity, 30
 Poisson
 blending, 404
 equations, 523
 matting, 450
 noise, 68
 surface reconstruction, 523
 Polar coordinates, 30
 Polar projection, 53, 387
 Polyphase filter, 127
 Pop-out effect, 4
 Pose estimation, 284
 iterative, 286
 Power spectrum, 123
 Precision, *see* Error rates
 mean average, 202
 Preconditioning, 659
 Principal component analysis (PCA), 242, 580, 589, 648, 664
 face modeling, 526
 generalized, 649
 missing data, 318, 649
 Prior energy (term), 159, 668
 Prior model, *see* Bayesian model
 Profile curves, 476
 Progressive mesh (PM), 520
 Projections
 object-centered, 51
 orthographic, 42
 para-perspective, 44
 perspective, 44
 Projective (uncalibrated) reconstruction, 312
 Projective depth, 49, 474
 Projective disparity, 49, 474
 Projective space, 30

- PROSAC (PROgressive SAmple Consensus), 282
PSNR, *see* Peak signal-to-noise ratio
Pyramid, 127, 175
 blending, 141, 176
 Gaussian, 132
 half-octave, 135
 Laplacian, 135
 motion estimation, 341
 octave, 132
 radial frequency implementation, 140
 steerable, 140
Pyramid match kernel, 613

QR factorization, 649
Quadratic form, 156
Quadrature mirror filter (QMF), 131
Quadric equation, 31, 32
Quadtree spline
 motion estimation, 358
 restricted, 358
Quaternions, 39
 antipodal, 39
 multiplication, 40
Query by image content (QBIC), 630
Query expansion, 608
Quincunx sampling, 135

Radial basis function, 151, 155, 518
Radial distortion, 52
 barrel, 52
 calibration, 295
 parameters, 52
 pincushion, 52
Radiance map, 424
Radiometric image formation, 54
Radiometric response function, 412
Radiometry, 54
Radiosity, 60
Random walker, 267, 675
Range (of a function), 91
Range data, *see* Range scan
Range image, *see* Range scan
Range scan
 alignment, 515, 540
 large scenes, 517
 merging, 516
 registration, 515, 540
 segmentation, 515
 volumetric, 516
Range sensing (rangefinding), 512
 coded pattern, 513
 light stripe, 513
 shadow stripe, 513, 540
 spacetime stereo, 515
 stereo, 514
 texture pattern (checkerboard), 514
 time of flight, 514
RANSAC
 (RAmdom SAmple Consensus), 281
 inliers, 281
 preemptive, 282
 progressive (PROSAC), 282
RAW image format, 68
Ray space (light field), 553
Ray tracing, 60
Rayleigh quotient, 262
Recall, *see* Error rates
Receiver Operating Characteristic
 area under the curve (AUC), 202
 mean average precision, 202
 ROC curve, 202, 229
Recognition, 575
 3D models, 637
 category (class), 611
 color similarity, 630
 context, 625
 contour-based, 636
 data sets, 631
 face, 588
 instance, 602
 large scale, 628
 learning, 627
 part-based, 615
 scene understanding, 625
 segmentation, 620
 shape context, 636
Rectangle detection, 226
Rectification, 472, 500
 standard rectified geometry, 473
Recursive filter, 107
Reference plane, 49

- Reflectance, 55
- Reflectance map, 509
- Reflectance modeling, 535
- Reflection
 - di-chromatic, 60
 - diffuse, 57
 - specular, 58
- Region
 - merging, 251
 - splitting, 251
- Region segmentation, *see* Segmentation
- Registration, *see* Image Alignment
 - feature-based, 275
 - intensity-based, 337
 - medical image, 358
- Regularization, 154, 356
 - robust, 157
- Regularization parameter, 155
- Residual error, 276, 281, 306, 320, 338, 346, 350, 361, 651, 658
- RGB (red green blue), *see* Color
- Rigid body transformation, 33, 36
- Robust error metric, *see* Robust penalty function
- Robust least squares, 224, 226, 281, 338, 666
 - iteratively reweighted, 281, 286, 350, 666
- Robust penalty function, 157, 338, 349, 437, 475, 479, 480, 484, 666
- Robust regularization, 157
- Robust statistics, 338, 666
 - inliers, 281
 - M-estimator, 281, 338, 666
- Rodriguez's formula, 38
- Root mean square error (RMS), 81, 339
- Rotations, 37
 - Euler angles, 37
 - axis/angle, 37
 - exponential twist, 39
 - incremental, 41
 - interpolation, 41
 - quaternions, 39
 - Rodriguez's formula, 38
- Sampling, 69
- Scale invariant feature transform (SIFT), 197
- Scale-space, 12, 104, 135, 249
- Scatter matrix, 589
- between-class, 593
- within-class, 592
- Scattered data interpolation, 151, 518
- Scene completion, 621
- Scene flow, 492, 565
- Scene understanding, 625
 - gist, 623, 626
 - scene alignment, 628
- Schur complement, 322, 656
- Scratch removal, 457
- Seam selection
 - image stitching, 400
- Second-order cone programming (SOCP), 324
- Seed and grow
 - stereo, 476
 - structure from motion, 328
- Segmentation, 235
 - active contours, 238
 - affinities, 260
 - binary MRF, 160, 264
 - CONDENSATION, 246
 - connected components, 115, 174
 - energy-based, 264
 - for recognition, 620
 - geodesic active contour, 248
 - geodesic distance, 267
 - GrabCut, 266, 450
 - graph cuts, 264
 - graph-based, 252
 - hierarchical, 251, 254
 - intelligent scissors, 247
 - joint feature space, 259
 - k-means, 256
 - level sets, 248
 - mean shift, 254, 258
 - medical image, 268
 - merging, 251
 - minimum description length (MDL), 264
 - mixture of Gaussians, 256
 - Mumford–Shah, 264
 - non-parametric, 257
 - normalized cuts, 260
 - probabilistic aggregation, 253
 - random walker, 267
 - snakes, 238

- splitting, 251
- stereo matching, 487
- thresholding, 112
- tobogganing, 247, 251
- watershed, 251
- weighted aggregation (SWA), 263
- Seidel aberrations, 63
- Self-calibration, 313
 - bundle adjustment, 315
 - Kruppa equations, 314
- Sensing, 65
 - aliasing, 69, 417
 - color, 71
 - color balance, 76
 - gamma, 77
 - pipeline, 66, 413
 - sampling, 69
 - sampling pitch, 67
- Sensor noise, 67, 415
 - amplifier, 67
 - dark current, 67
 - fixed pattern, 67
 - shot noise, 67
- Separable filtering, 102, 173
- Shading, 58
 - equation, 57
 - shape-from, 508
- Shadow matting, 452
- Shape context, 219, 636
- Shape from
 - focus, 511, 539
 - photometric stereo, 510
 - profiles, 476
 - shading, 508
 - silhouettes, 497
 - specularities, 511
 - stereo, 467
 - texture, 510
- Shape parameters, 242, 598
- Shape-from-X, 12
 - focus, 12
 - photometric stereo, 12
 - shading, 12
 - texture, 12
- Shift invariance, 100
- Shiftable multi-scale transform, 140
- Shutter speed, 66
- Signed distance function, 248, 515, 521, 522
- Silhouette-based reconstruction, 497
 - octree, 498
 - visual hull, 497
- Similarity transform, 34, 37
- Simulated annealing, 161, 670
- Simultaneous localization and mapping (SLAM), 324
- Sinc filter
 - interpolation, 130
 - low-pass, 103
 - windowed, 130
- Single view metrology, 292, 300
- Singular value decomposition (SVD), 646
- Skeletal set, 324, 328
- Skeleton, 114, 219
- Skew, 46, 47
- Skin color detection, 85
- Slant edge calibration, 417
- Slippery spring, 240
- Smoke matting, 452
- Smoothness constraint, 155
- Smoothness penalty, 155
- Snakes, 238
 - ballooning, 239
 - dynamic, 243
 - internal energy, 238
 - Kalman, 243
 - shape priors, 241
 - slippery spring, 240
- Soft assignment, 256
- Software, 682
- Space carving
 - multi-view stereo, 496
- Spacetime stereo, 515
- Sparse flexible model, 617
- Sparse matrices, 655, 687
 - compressed sparse row (CSR), 655
 - skyline storage, 655
- Sparse methods
 - direct, 655, 687
 - iterative, 656, 687
- Spatial pyramid matching, 614
- Spectral response function, 75

- Spectral sensitivity, 75
 Specular flow, 511
 Specular reflection, 58
 Spherical coordinates, 31, 223, 225, 385
 Spherical linear interpolation, 41
 Spin image, 515
 Splatting, *see* Forward warping
 volumetric, 521
 Spline
 controlled continuity, 155
 octree, 359
 quadtree, 358
 thin plate, 155
 Spline-based motion estimation, 355
 Splining images, *see* Laplacian pyramid blending
 Sprites
 image-based rendering, 549
 motion estimation, 365
 video, 563
 video compression, 383
 with depth, 550
 Statistical decision theory, 662, 665
 Steerable filter, 105, 174
 Steerable pyramid, 140
 Steerable random field, 162
 Stereo, 467
 aggregation methods, 481, 501
 coarse-to-fine, 485
 cooperative algorithms, 485
 correspondence, 469
 curve-based, 476
 dense correspondence, 477
 depth map, 469
 dynamic programming, 485
 edge-based, 475
 epipolar geometry, 471
 feature-based, 475
 global optimization, 484, 502
 graph cut, 485
 layers, 488
 local methods, 480
 model-based, 524, 547
 multi-view, 489
 non-parametric similarity measures, 479
 photoconsistency, 474
 plane sweep, 474, 501
 rectification, 472, 500
 region-based, 480
 scanline optimization, 487
 seed and grow, 476
 segmentation-based, 480, 487
 semi-global optimization, 487
 shiftable window, 491
 similarity measure, 479
 spacetime, 515
 sparse correspondence, 475
 sub-pixel refinement, 482
 support region, 480
 taxonomy, 469, 478
 uncertainty, 482
 window-based, 480, 501
 winner-take-all (WTA), 481
 Stereo-based head tracking, 483
 Stiffness matrix, 156
 Stitching, *see* Image stitching
 Stochastic gradient descent, 161
 Structural Similarity (SSIM) index, 126
 Structure from motion, 305
 affine, 317
 bas-relief ambiguity, 326
 bundle adjustment, 320
 constrained, 329
 factorization, 315
 feature tracks, 327
 iterative factorization, 318
 line-based, 330
 multi-frame, 315
 non-rigid, 332
 orthographic, 315
 plane-based, 319, 331
 projective factorization, 318
 seed and grow, 328
 self-calibration, 313
 skeletal set, 324, 328
 two-frame, 307
 uncertainty, 326
 Subdivision surface, 519
 subdivision connectivity, 520
 Subspace learning, 596
 Sum of absolute differences (SAD), 338, 371, 479

- Sum of squared differences (SSD), 337, 371, 479
 bias and gain, 339
Fourier-based computation, 342
 normalized, 340
 surface, 186, 348
 weighted, 339
 windowed, 339
Sum of sum of squared differences (SSSD), 489
Summed area table, 106
Super-resolution, 436, 463
 example-based, 438
 faces, 439
 hallucination, 438
 prior, 437
Superposition principle, 91
Superquadric, 522
Support vector machine (SVM), 582, 585
Surface element (surfel), 521
Surface interpolation, 518
Surface light field, 555
Surface representations, 518
 non-parametric, 519
 parametric, 519
 point-based, 521
 simplification, 520
 splines, 519
 subdivision surface, 519
 symmetry-seeking, 519
 triangle mesh, 519
Surface simplification, 520
Swendsen–Wang algorithm, 670
- Telecentric lens, 42, 512
Temporal derivative, 346, 360
Temporal texture, 563
Term frequency-inverse document frequency (TF-IDF), 605
Testing algorithms, viii
TextonBoost, 621
Texture
 shape-from, 510
Texture addressing mode, 102
Texture map
 recovery, 534
 view-dependent, 535, 547
Texture mapping
 anisotropic filtering, 148
 MIP-mapping, 147
 multi-pass, 149
 trilinear interpolation, 148
Texture synthesis, 455, 465
 by numbers, 459
 hole filling, 457
 image quilting, 456
 non-parametric, 455
 transfer, 458
Thin lens, 61
Thin-plate spline, 155
Thresholding, 112
Through-the-lens camera control, 287, 324
Tobogganing, 247, 251
Tonal adjustment, 97, 172
Tone mapping, 427
 adaptive, 427
 bilateral filter, 428
 global, 427
 gradient domain, 430
 halos, 428
 interactive, 431
 local, 427
 scale selection, 431
Total least squares (TLS), 233, 349, 653
Total variation, 158, 361, 523
Tracking
 feature, 207
 head, 483
 human motion, 530
 multiple hypothesis, 243
 planar pattern, 287
 PTAM, 325
Translational motion estimation, 337
 bias and gain, 339
Transparency, 93
Travelling salesman problem (TSP), 239
Tri-chromatic sensing, 72
Tri-stimulus values, 72, 75
Triangulation, 305
Trilinear interpolation, *see* MIP-mapping
Trimap (matting), 446
Trust region method, 655
Two-dimensional Fourier transform, 123

- Uncanny valley, 3
- Uncertainty
 - correspondence, 277
 - modeling, 282, 678
 - weighting, 277
- Unsharp mask, 103
- Upsampling, *see* Interpolation
- Vanishing point
 - detection, 224, 234
 - Hough, 224
 - least squares, 226
 - modeling, 524
 - uncertainty, 234
- Variable reordering, 656
 - minimum degree, 656
 - multi-frontal, 656
 - nested dissection, 656
- Variable state dimension filter (VSDF), 323
- Variational method, 154
- Video compression
 - motion compensated, 341
- Video compression (coding), 370
- Video denoising, 364
- Video matting, 454
- Video objects (coding), 365
- Video sprites, 563
- Video stabilization, 354, 372
- Video texture, 561
- Video-based animation, 560
- Video-based rendering, 560
 - 3D video, 564
 - animating pictures, 564
 - sprites, 563
 - video texture, 561
 - virtual viewpoint video, 564
 - walkthroughs, 566
- VideoMouse, 288
- View correlation, 324
- View interpolation, 315, 545, 570
- View morphing, 315, 546, 563
- View-based eigenspace, 595
- View-dependent texture maps, 547
- Vignetting, 64, 339, 416, 462
 - mechanical, 65
 - natural, 64
- Virtual viewpoint video, 564
- Visual hull, 497
 - image-based, 499
- Visual illusions, 3
- Visual odometry, 324
- Visual words, 207, 605, 612
- Vocabulary tree, 207, 607
- Volumetric 3D reconstruction, 492
- Volumetric range image processing (VRIP), 516
- Volumetric representations, 522
- Voronoi diagram, 400
- Voxel coloring
 - multi-view stereo, 495
- Watershed, 251, 257
 - basins, 251, 257
 - oriented, 251
- Wavelets, 136, 176
 - compression, 176
 - lifting, 138
 - overcomplete, 137, 140
 - second generation, 139
 - self-inverting, 140
 - tight frame, 137
 - weighted, 139
- Weaving wall, 477
- Weighted least squares (WLS), 431, 443
- Weighted prediction (bias and gain), 339
- White balance, 76, 85
- Whitening transform, 591
- Wiener filter, 123, 124, 174
- Wire removal, 457
- Wrapping mode, 102
- XYZ, *see* Color
- Zippering, 516