Chapter 10

Stereo correspondence

(a) (b) (c)

(d) (e) (f)

(g)

Figure 10.1: *Stereo reconstruction techniques can convert a pair of images (a–b) into a depth map (c), or a sequence of images (d–e) into a 3D model (f). Figure (g) shows an image of an analytical stereo plotter, courtesy of Kenney Aerial Mapping, Inc. Images (a–c) are from* http://vision.middlebury.edu/stereo/data/scenes2003/ *and (d–f) from* http://vision.middlebury.edu/mview/data/.

In Chapters 5–6, we developed 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 of the scene. The next chapter §11.8 describes how to recover texture maps that can be painted onto the models to make them more realistic.

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 10.1a–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 distance from the observer. (You can qualitatively confirm this for yourself by looking at a scene with a range of depths and alternately closing your eyes.) While the basic physics and geometry relating visual disparity to scene structure are well understood §10.1, 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 10.1g). The development of fully automated stereo matching algorithms was a major advance in this field, enabling much more rapid and less expensive exploitation of aerial imagery (Hannah 1974, Hsieh *et al.* 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 *et al.* 2003, Seitz *et al.* 2006), and remains an active area of research. 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 *et al.* 2006), as well as view interpolation and image based rendering (Figure 10.2a–d), 3D model building (Figure 10.2e–f,h–i), and mixing live action with computer generated imagery (Figure 10.2g).

In this chapter, we describe the fundamental principles behind stereo matching, following the general taxonomy proposed by Scharstein and Szeliski (2002). We begin in §10.1 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*

[1] The word *stereo* comes from the Greek for *solid*, which is how we perceive solid shape. *[Note: Check this, and maybe look into (Koenderink 1990).]*

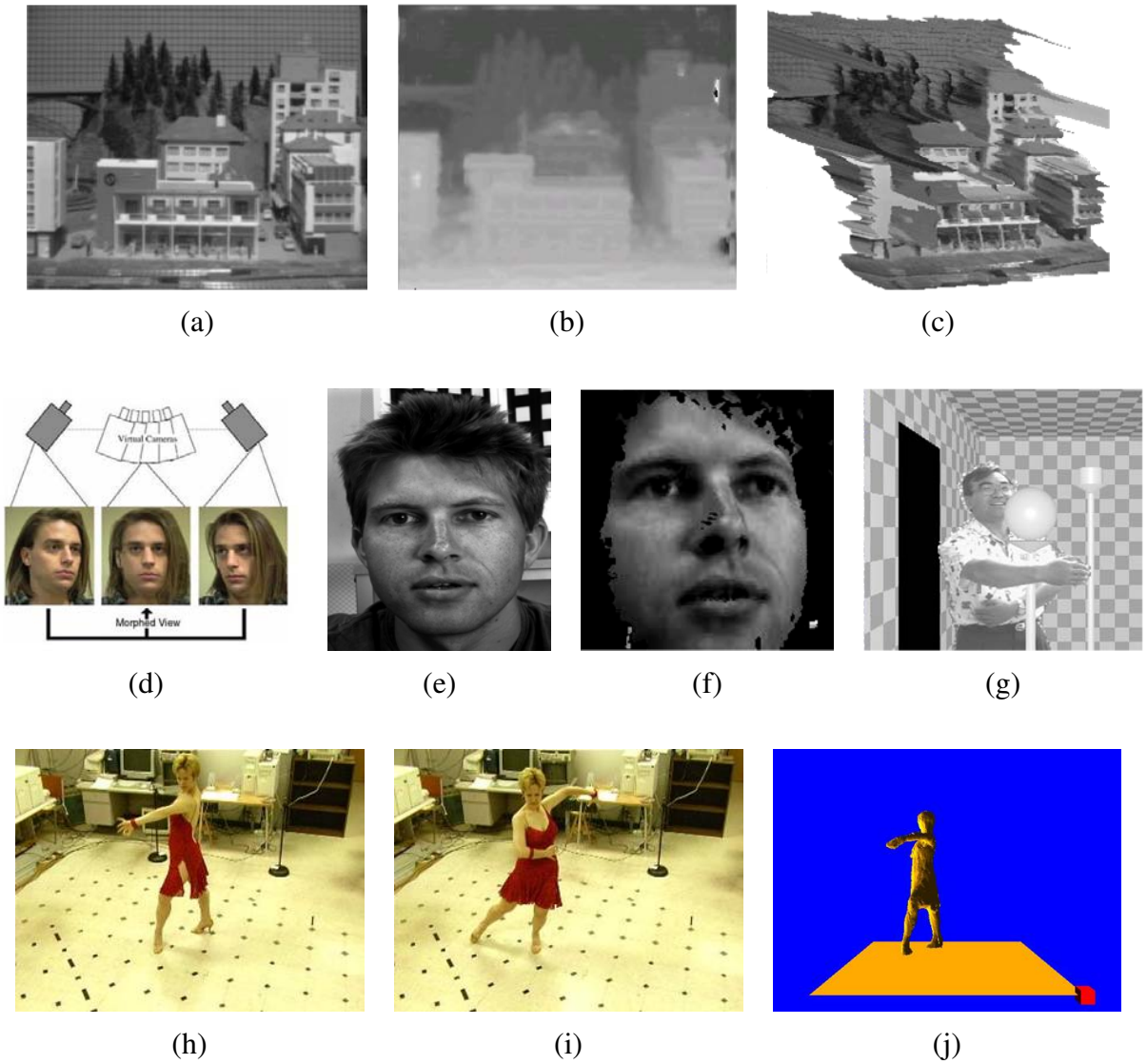


Figure 10.2: *Some applications of stereo vision: (a–c) input image, computed depth map, and new view generation from multi-view stereo (Matthies* et al. *1989); (d) view morphing between two images (Seitz and Dyer 1996); (e–f) 3D face modeling (images courtesy of Frédéric Devernay); (g)* z-keying *live and computer generated imagery (Kanade* et al. *1996); (h–j) building 3D surface models from multiple video streams in Virtualized Reality*$^{\text{TM}}$ *(Kanade* et al. *1997).*

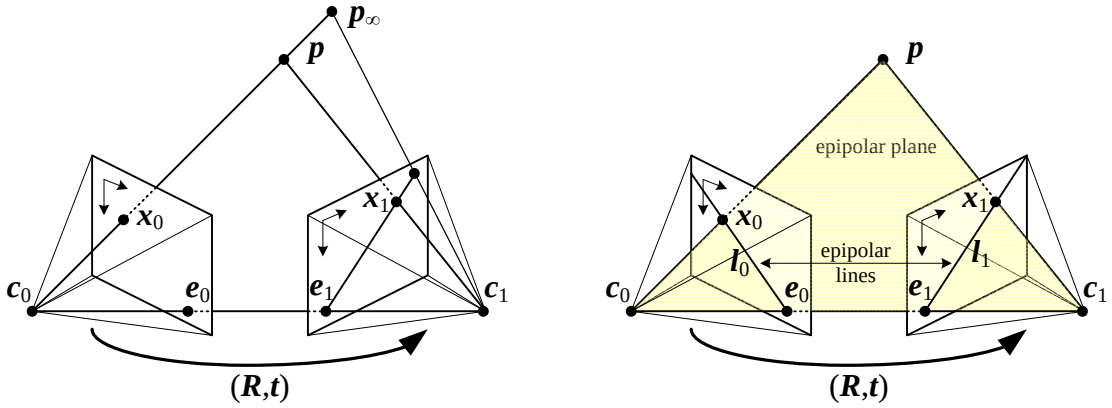


Figure 10.3: *Epipolar geometry: (a) epipolar line segment corresponding to one ray; (b) corresponding set of epipolar lines and their epipolar plane.*

line). We also describe how to pre-warp images so that corresponding epipolar lines are coincident (*rectification*), and then describe a general resampling algorithm called *plane sweep* that can be used to perform multi-image stereo matching with arbitrary camera configurations.

Next, we briefly survey techniques for *sparse* stereo matching of interest points and edge-like features §10.2. 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 10.1c). This involves first selecting a pixel matching criterion §10.3, and then using either local area-based aggregation §10.4 or global optimization §10.5 to help disambiguate potential matches. In the final part of this chapter §10.6, we discuss *multi-view stereo* methods that aim to reconstruct a complete 3D model instead of just a single disparity image (Figure 10.1d–f).

10.1 Epipolar geometry

Given a pixel in one image (say the *left* image), how can we compute its correspondence with the correct pixel in the other image? In the chapter on motion estimation §7, 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 §6.2.

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 10.3a shows how a pixel in one image $\boldsymbol{x}_0$ projects to an *epipolar line segment* in the other image. The segment is bounded

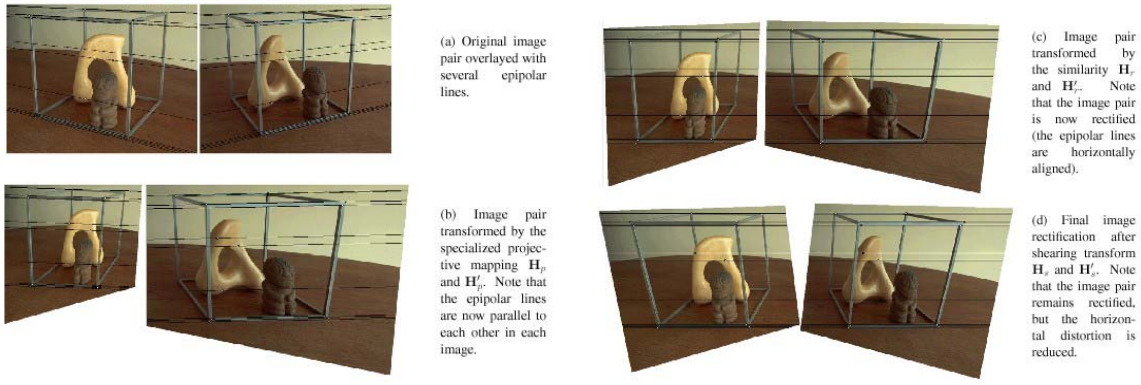


Figure 10.4: *The multi-stage stereo rectification algorithm of Loop and Zhang (1999).*
[Note: Rescan original images at better quality or ask Zhengyou]

at one end by the projection of the original viewing ray at infinity $\boldsymbol{p}_\infty$ and at the other end by the projection of the original camera center $\boldsymbol{c}_0$ into the second camera, which is known as the *epipole* $\boldsymbol{e}_1$. 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 $\boldsymbol{e}_0$. Extending both line segments to infinity, we get a pair of corresponding *epipolar lines* (Figure 10.3b), which are the intersection of the two image planes with the *epipolar plane* that passes through both camera centers $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$ as well as the point of interest $\boldsymbol{p}$ (Faugeras and Luong 2001, Hartley and Zisserman 2004).

10.1.1 Rectification

As we saw in §6.2, 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 in the calibrated essential matrix case) (Zhang 1998a, Zhang 1998b, 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 §7.4, but to only consider locations along the epipolar line (and/or to project any flow vectors that fall off the line back onto it).

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

[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 isn't *verged* too much or has too much of a scale change. In those cases, using plane sweep (below) or hypothesizing small planar patch locations in 3D (Goesele *et al.* 2007) may be preferable.

scanlines independently, or to just shift images horizontally while computing matching scores (Figure 10.4).

A simple way to rectify the two images is to first rotate both cameras so that they are looking perpendicular to line joining the two camera centers $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$. 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* (camera y axis) perpendicular to the camera center line. This ensures that corresponding epipolar lines are horizontal, and that the disparity for points at infinity is 0. Finally, re-scale the images, if necessary, to account for different focal length, magnifying the smaller image to avoid aliasing. (The full details of this procedure are in Exercise 10.1.) Note that in general, it is not possible to rectify an arbitrary collection of images simultaneously unless they 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,

$$d = f\frac{B}{Z}, \tag{10.1}$$

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

$$x' = x + d(x, y), \quad y' = y \tag{10.2}$$

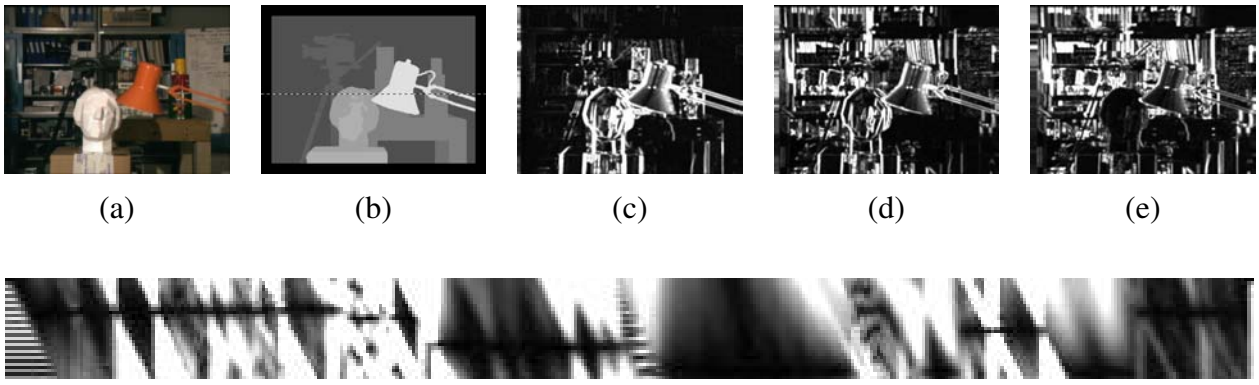
describes the relationship between corresponding pixel coordinates in the left and right images (Bolles *et al.* 1987, Okutomi and Kanade 1993, Scharstein and Szeliski 2002).[3] The task of extracting depth from a set of images then becomes one of estimating the *disparity map* $d(x, y)$.

After rectification, we can easily compare the similarity of pixels at corresponding locations (x, y) and $(x', y') = (x + d, y)$ and store these in a *disparity space image* (DSI) $C(x, y, d)$ for further processing (Figure 10.5). 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 *et al.* (1993) and Intille and Bobick (1994).

10.1.2 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 10.6), which is commonly known as the *plane sweep* algorithm (Collins 1996, Szeliski and Golland 1999, Saito and Kanade 1999).

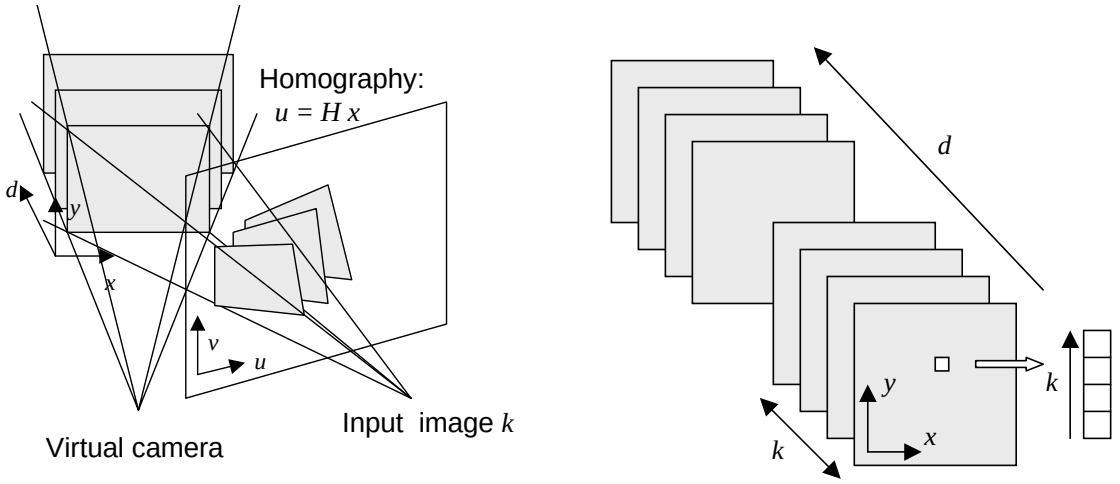
[3] The term *disparity* was first introduced in the human vision literature to describe the difference in location of corresponding features seen by the left and right eyes (Marr 1982). Horizontal disparity is the most commonly studied phenomenon, but vertical disparity is possible if the eyes are verged.



(a) (b) (c) (d) (e)

(f)

Figure 10.5: *Slices through a typical disparity space image (DSI) (Scharstein and Szeliski 2002): (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)). Different dark (matching) regions are visible in (c–e), e.g., the bookshelves, table and cans, and head statue, while three different disparity levels can be seen as horizontal lines in the (x, d) slice (f). Note the dark bands in the various DSIs, which indicate regions that match at this disparity. (Smaller dark regions are often the result of textureless regions.) Additional examples of DSIs can be found in (Bobick and Intille 1999).*



(a) (b)

Figure 10.6: *Sweeping a set of planes through a scene (Szeliski and Golland 1999): (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.*

As we saw in §2.1.4, where we introduced projective depth (also known as *plane plus parallax* (Kumar *et al.* 1994b, Sawhney 1994, Szeliski and Coughlan 1997)), the last row of a full-rank 4×4 projection matrix $\tilde{\boldsymbol{P}}$ can be set to an arbitrary plane equation $\boldsymbol{p}_3 = s_3[\hat{\boldsymbol{n}}_0|c_0]$. The resulting 4-dimensional projective transform (*collineation*) (2.68) maps 3D world points $\boldsymbol{p} = (X, Y, Z, 1)$ into screen coordinates $\boldsymbol{x}_s = (x_s, y_s, 1, d)$, 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 10.6a, corresponds to mapping each input image into the *virtual camera* $\tilde{\boldsymbol{P}}$ (defining the disparity space) through a series of homographies (2.68–2.71),

$$\tilde{\boldsymbol{x}}_k \sim \tilde{\boldsymbol{P}}_k \tilde{\boldsymbol{P}}^{-1} \boldsymbol{x}_s = \tilde{\boldsymbol{H}}_k \tilde{\boldsymbol{x}} + \boldsymbol{t}_k d = (\tilde{\boldsymbol{H}}_k + \boldsymbol{t}_k [0\ 0\ d]) \tilde{\boldsymbol{x}}. \tag{10.3}$$

(Szeliski and Golland 1999) as shown in Figure 2.12b, where $\tilde{\boldsymbol{x}}_k$ and $\tilde{\boldsymbol{x}}$ are the homogeneous pixel coordinates in the source and virtual (reference) images. The family of homographies $\tilde{\boldsymbol{H}}_k(d) = \tilde{\boldsymbol{H}}_k + \boldsymbol{t}_k[0\ 0\ d]$, which is parametererized by the addition of a rank-1 matrix, is known as a *homology* (Hartley and Zisserman 2004). *[Note: check that this is correct]*

The choise 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 §12.1–12.2. In other applications, such as view interpolation for gaze correction in video-conferencing §10.4.2 (Ott *et al.* 1993, Criminisi *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 dependant, and is usually set to bracket the range of interest 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, *[Note: find reference to this configuration, probably some CMU Navlab stereo; Can Martial help me find this?]* it is most convenient to set up disparity to measure per-pixel elevation above the ground.

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 10.6b) (Szeliski and Golland 1999). In most stereo algorithms, the photoconsistency (e.g., sum of squared or robust differences) w.r.t. the reference image I_r is calculated and stored in the DSI

$$C(x, y, d) = \sum_k \rho(\tilde{I}(x, y, d, k) - I_r(x, y)). \tag{10.4}$$

However, it is also possible to compute alternative statistics such as robust variance, focus, or entropy §10.3.1 (Vaish *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 §10.6.

Of course, planes are not the only surfaces that can be used to define a 3D sweep of the space of interest. Cylindrical surfaces, especially when coupled with panoramic photography §8, are often used (Ishiguro *et al.* 1992, Kang and Szeliski 1997, Shum and Szeliski 1999, Li *et al.* 2004a, Zheng *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 *et al.* 1993). Figure 10.5 shows examples of slices through a typical DSI. More figures of this kind can be found in (Bobick and Intille 1999).

10.2 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 (Baker and Binford 1981, Grimson 1985, Ohta and Kanade 1985, Hsieh *et al.* 1992, Bolles *et al.* 1993).[4] 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 algorithm such as those discussed in §3.6.1 and §11.5.1.

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 *et al.* 2003, Goesele *et al.* 2007) and/or free-space reasoning (Taylor 2003), as described in more detail in the sections on multi-view stereo §10.6, volumetric reconstruction §11.3, and model-based reconstruction §11.7.1.

[4] We explore the topic of matching extended silhouette curves in §11.2.

10.3 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 existing algorithms can easily be constructed. It is based on the observation that stereo algorithms generally perform (subsets of) the following four steps:

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

The actual sequence of steps taken depends on the specific algorithm.

For example, *local* (window-based) algorithms §10.4, 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 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 *et al.* 1993) and the rank transform (Zabih and Woodfill 1994). (This can also be viewed as a preprocessing step; see Section 10.3.1.)

Global algorithms, on the other hand, make explicit smoothness assumptions and then solve an optimization problem §10.5. Such algorithms typically do not perform an aggregation step, but rather seek a disparity assignment (step 3) that minimizes a global cost function that combines data

(step 1) and smoothness terms. The main distinction between these algorithms is the minimization procedure used, e.g., simulated annealing (Marroquin *et al.* 1987, Barnard 1989), probabilistic (mean-field) diffusion (Scharstein and Szeliski 1998), expectation maximimization (EM) (Birchfield *et al.* 2007), graph cuts (Boykov *et al.* 2001), or loopy belief propagation (Sun *et al.* 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, Scharstein and Szeliski 1998, 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 *et al.* 1987, Quam 1984, Bergen *et al.* 1992a).

10.3.1 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 §7.1 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 *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 (7.2), 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 *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 *et al.* 2004), which is particularly useful in multi-view stereo.

Other traditional matching costs include normalized cross-correlation (7.11) (Hannah 1974, Bolles *et al.* 1993), which behaves similarly to sum-of-squared-differences (SSD), and binary matching costs (i.e., match / 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 *et al.* 1991, Jones and Malik 1992). , filters that remove a regular or robust (bilaterally filtered) means (Ansar *et al.* 2004, Hirschmüller and Scharstein 2009), and non-parametric measures such as rank and census transforms (Zabih and Woodfill 1994) or entropy (Zitnick *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 *et al.* 1995), or mutual information (Kim *et al.* 2003, Hirschmüller 2008). Local, smoothly varying compensation fields have also been proposed (Strecha *et al.* 2003, Zhang *et al.* 2006).

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), bilateral subtraction (Ansar *et al.* 2004), or hierarchical mutual inforamtion (Hirschmüller 2008), should be used.

[Note: Say something about how disparity measures map to measurement probabilities? Ask Antonio about this.]

10.4 Local methods

[Note: This whole chapter is somewhat thin on figures and images. Can I think of how to add some more?]

Local and window-based methods aggregate the matching cost by summing or averaging over a *support region* in the DSI $C(x, y, d)$. (For two recent surveys and comparisons of such techniques, please see (Gong *et al.* 2007, Tombari *et al.* 2008).) 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

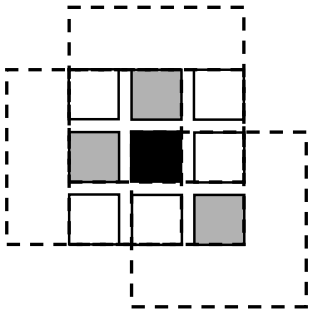


Figure 10.7: *Shiftable window (Scharstein and Szeliski 2002). The effect of trying all* 3×3 *shifted windows around the black pixel is the same as taking the minimum matching score across all* centered *(non-shifted) windows in the same neighborhood. (Only 3 of the neighboring shifted windows are shown here for clarity.)*

using square windows or Gaussian convolution (traditional), multiple windows anchored at different points, i.e., shiftable windows (Arnold 1983, Bobick and Intille 1999), windows with adaptive sizes (Okutomi and Kanade 1992, Kanade and Okutomi 1994, Veksler 2001, Veksler 2003, Kang *et al.* 2001), windows based on connected components of constant disparity (Boykov *et al.* 1998), or the results of color-based segmentation (Yoon and Kweon 2006, Tombari *et al.* 2008). Three-dimensional support functions that have been proposed include limited disparity difference (Grimson 1985), limited disparity gradient (Pollard *et al.* 1985), Prazdny's coherence principle (Prazdny 1985), and the more recent work (which includes visibility and occlusion reasoning) by Zitnick and Kanade (2000).

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

$$C(x, y, d) = w(x, y, d) * C_0(x, y, d), \tag{10.5}$$

or, in the case of rectangular windows, using efficient (moving average) box-filters §3.2.1 (Kanade *et al.* 1996, Kimura *et al.* 1999). Shiftable windows can also be implemented efficiently using a separable sliding min-filter (Figure 10.7) (Scharstein and Szeliski 2002, §4.2). Selecting among windows of different shape and sizes can be performed more efficiently by first computing a *summed area table* §3.2.1 (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 don't straddle depth discontinuities (Figure 10.8). An alternative method for aggregation is *iterative diffusion*, i.e., an aggregation (or averaging) operation that is implemented by 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 *et al.* (2007) and Tombari *et al.* (2008), the fast variable window approach of (Veksler 2003) and the locally weighting approach developed by

(a) (b) (c) (d)

Figure 10.8: *Aggregation window sizes and weights adapted to image content (Tombari et al. 2008): (a) original image with selected evaluation points; (b) variable windows (Veksler 2003); (c) adaptive weights (Yoon and Kweon 2006); (d) segmentation-based (Tombari et al. 2007). Notice how the adaptive weights and segmentation-based techniques adapt their support to similarly-colored pixels.*

(Yoon and Kweon 2006) consistently stood out as having the best performance to speed tradeoff.[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 10.8c). (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 *et al.* (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 correspondence algorithms) is that uniqueness of matches is only enforced for one image (the *reference image*), while points in the other image might get matched to multiple points, unless cross-checking and subsequent hole filling is used (Fua 1993, Hirschmüller and Scharstein 2009).

10.4.1 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 *et al.* 1992a) or splines (Szeliski and Coughlan 1997)). For applications such as robot navigation or people tracking, these may be perfectly adequate. However for image-based

[5] More recent and extensive results from Tombari *et al.* (2008) can be found at `http://www.vision.deis.unibo.it/spe`.

rendering, such quantized maps lead to very unappealing view synthesis results (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 *et al.* 1980, Lucas and Kanade 1981, Tian and Huhns 1986, Matthies *et al.* 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 in (Scharstein and Szeliski 2002, Szeliski and Scharstein 2004).

Besides sub-pixel computations, there are of course other ways of post-processing the computed disparities. Occluded areas can be detected using cross-checking (comparing left-to-right and right-to-left disparity maps) (Cochran and Medioni 1992, 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 latter processing stages, is to associate *confidences* with per-pixel depth estimates (Figure 10.9), which can be done by looking at the curvature of the correlation surface, i.e., how strong the minimim in the DSI image as the winning disparity compared to its neighbors. Matthies *et al.* (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

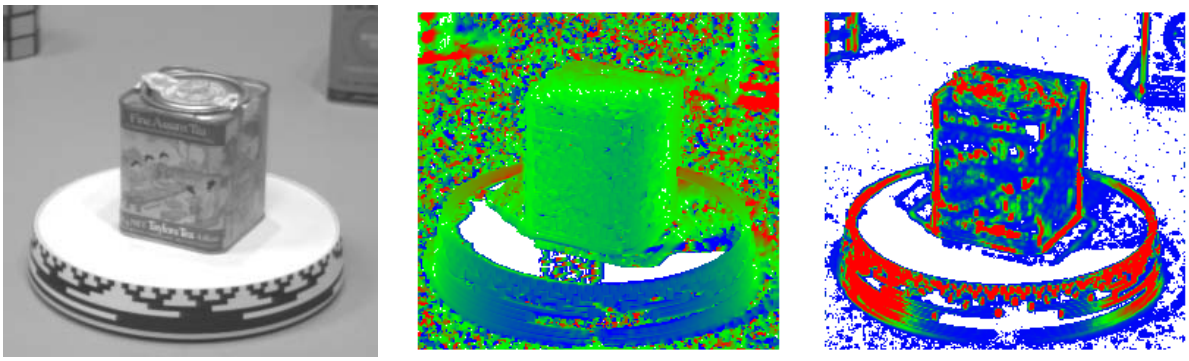
$$Var(d) = \frac{\sigma_I^2}{a}, \tag{10.6}$$

where a is the curvature of the DSI as a function of d, which can be measures 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 §5.1.4, (7.43), and Appendix B.7.) *[Note: May need to rationalize the above cross-references later.]*

10.4.2 *Application*: Stereo-based head tracking

[Note: not sure yet what to do with this section ... revisit later]

Use rough head position for "Fishtank VR" (3D parallax).



(a) (b) (c)

Figure 10.9: *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.*

Gaze correction (not gaze tracking) for desktop teleconferencing: interpolate mid-point view. But this requires accurate 3D model. Describe Cox's original idea (Ott *et al.* 1993), and more recent work by Criminisi *et al.* (2003).

Related to this is background replacement in the i2i real-time stereo system with color segmentation (Kolmogorov *et al.* 2006) (see application §10.5.2 below on z-keying)

Another application is people tracking (EasyLiving, in (Toyama *et al.* 1999)? Also, Woodfill et al, many others...)

10.5 Global optimization

[Note: Is this whole survey thing too much for a textbook? Try to think of the main points, and cite the essentials?]

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 §3.6 (3.97) and §7.4, the objective is to find a solution d that minimizes a global energy,

$$E(d) = E_d(d) + \lambda E_s(d). \tag{10.7}$$

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

$$E_d(d) = \sum_{(x,y)} C(x, y, d(x, y)), \tag{10.8}$$

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 only measuring the differences between neighboring pixels' disparities,

$$E_s(d) = \sum_{(x,y)} \rho(d(x,y) - d(x+1,y)) + \rho(d(x,y) - d(x,y+1)), \tag{10.9}$$

where ρ is some monotonically increasing function of disparity difference.[6] An alternative to smoothness functionals is to use a lower-dimensional representation such as splines (Szeliski and Coughlan 1997).

In standard regularization §3.6.1, 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 1986, Black and Rangarajan 1996, Scharstein and Szeliski 1998). Geman and Geman's seminal paper (Geman and Geman 1984) gave a Bayesian interpretation of these kinds of energy functions (Szeliski 1989) and proposed a discontinuity-preserving energy function based on Markov Random Fields (MRFs) and additional *line processes*. Black and Rangarajan (1996) show how line processes can be often be subsumed by a robust regularization framework.

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

$$\rho_d(d(x,y) - d(x+1,y)) \cdot \rho_I(\|I(x,y) - I(x+1,y)\|), \tag{10.10}$$

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 *et al.* 2001) encourages disparity discontinuities to coincide with intensity/color edges and appears to account for some of the good performance of global optimization approaches. While most researcher set these functions heuristically, Scharstein and Pal (2007) show how the free parameters in such *conditional random fields* (§3.6.2 (3.117)) 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 *et al.* 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, Ishikawa and Geiger 1998, Boykov *et al.* 2001,

[6] It is also possible to use larger neighborhoods such as $\mathcal{N}_8$, which can lead to better boundaries, but at the cost of more complex optimization (Boykov and Kolmogorov 2003).

Veksler 1999, Kolmogorov and Zabih 2001). Such methods are more efficient than simulated annealing and have produced good results, as have techniques based on loopy belief propagation (Sun *et al.* 2003, Tappen and Freeman 2003). Appendix B.6 and a recent survey paper on MRF inference (Szeliski *et al.* 2008c) 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 *et al.* 1992a, Barron *et al.* 1994, Szeliski and Coughlan 1997).

Dynamic programming. A different class of global optimization algorithms are those based on *dynamic programming*. While the 2D-optimization of Equation (10.7) 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 *et al.* 1992, Cox *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 handled explicitly by assigning a group of pixels in one image to a single pixel in the other image. Figure 10.10 schematically shows how DP works, while Figure 10.5f shows a real DSI slice overwhich the DP is applied.

To implement dynamic programming for a scanline y, each entry (state) in a 2D cost matrix

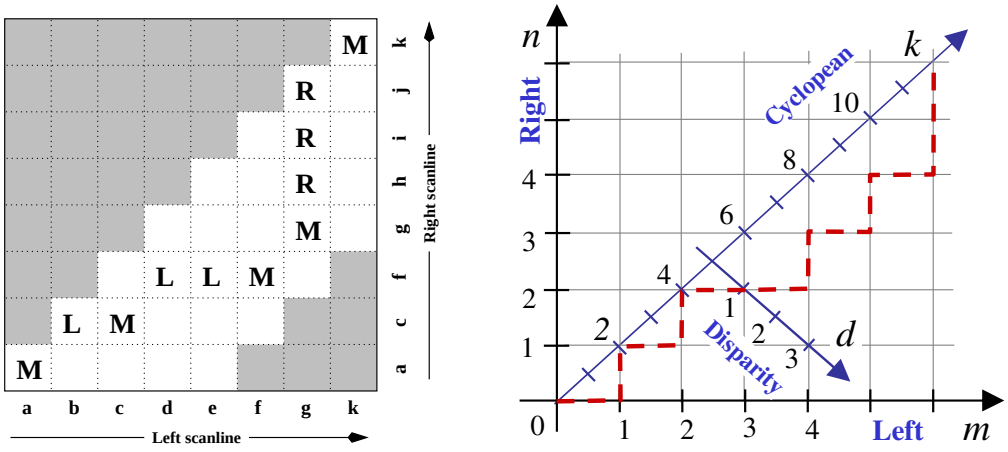


Figure 10.10: *Stereo matching using dynamic programming, as illustrated by (a) Scharstein and Szeliski (2002) and (b) Kolmogorov* et al. *(2006). For each pair of corresponding scanlines, a minimizing path through the matrix of all pairwise matching costs (DSI) is selected. Lowercase letters (a–k) symbolize 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 and R, corresponding to points only visible in the left and right image, respectively. Usually, only a limited disparity range is considered, which is 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, show an "unskewed" x-d slice through the DSI.*

$D(m, n)$ is computed by combining its DSI value

$$C'(m, n) = C(m + n, m - n, y) \tag{10.11}$$

with one of its predecessor cost values. Using the representation shown in Figure 10.10a, which allows for "diagonal" moves, the aggregated match costs can be recursively computed as

$$\begin{aligned} D(m, n, M) &= \min(D(m-1, n-1, M), D(m-1, n, L), D(m-1, n-1, R)) + C'(m, n) \\ D(m, n, L) &= \min(D(m-1, n-1, M), D(m-1, n, L)) + O \\ D(m, n, R) &= \min(D(m, n-1, M), D(m, n-1, R)) + O, \end{aligned} \tag{10.12}$$

where O is a per-pixel occlusion cost. *[Note: Daniel: can you check that this is right?]* The aggregation rules corresponding to Figure 10.10b are given in (Kolmogorov *et al.* 2006), which also uses a two-state foreground-background model for bi-layer segmentation. *[Note: Antonio: if you can provide me with the 1-layer update equations, I'll include them here. I just couldn't easily figure out from your paper which states in Figure 10.10b are M, L, or R.]*

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 methods propose ways of addressing the latter (Ohta and Kanade 1985, Belhumeur 1996, Cox *et al.* 1996, Bobick and Intille 1999, Birchfield and Tomasi 1999, Kolmogorov *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 in (Scharstein and Szeliski 2002), is to neglect the vertical smoothness constraints in (10.9) and to simply optimize independent scanlines in the global energy function (10.7), which can easily be done using a recursive algorithm,

$$D(x, y, d) = C(x, y, d) + \min_{d'} \{C(x-1, y, d') + \rho_d(d(x, y) - d'(x-1, y))\} . \tag{10.13}$$

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 (10.7). Unfortunately, it still suffers from the same streaking artifacts as dynamic programming.

A much better approach is to evaluate the cumulative cost function (10.13) from multiple directions, e.g, from the 8 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.

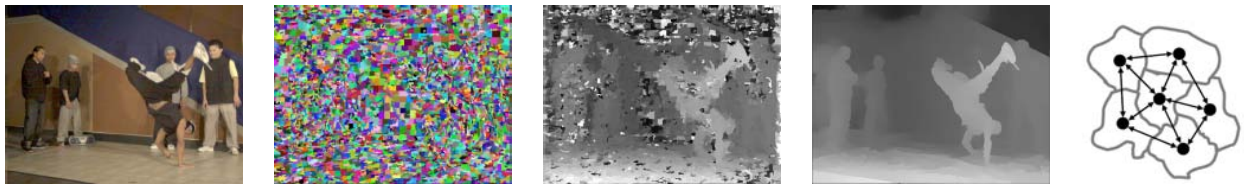
Even though dynamic programming and scanline optimization algorithms do not generally produce *the* most accurate stereo reconstructions, when combined with sophisticated aggregation strategies, they can produce very fast and high-quality results.

[Note: Do I need to further rationalize the discussion of MRFs with the Image Processing §3.6.2 and Appendix §B.5 sections?]

10.5.1 Segmentation-based techniques

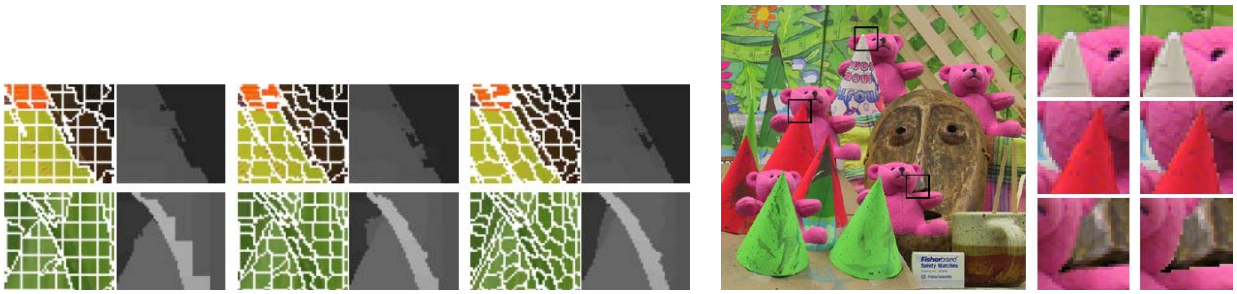
While most stereo matching algorithms perform their computations on a per-pixel basis, some of the more recent techniques first segment the images into regions and then try to label each region with a disparity.

For example, Tao *et al.* (2001) segment the reference image, estimate per-pixel disparities using a local technique, and then do local plane fits inside each segment before applying smoothness constraints between neighborhing segments. Zitnick *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) (Figure 10.11e), iterative relaxation (or loopy belief propagation, in the more recent work) is used to adjust the disparity



(a) (b) (c) (d) (e)

Figure 10.11: *Segmentation-based stereo matching (Zitnick et al. 2004): (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).*



(a) (b)

Figure 10.12: *Stereo matching with adaptive over-segmentation and matting (Taguchi et al. 2008): (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).*

estimates for each segment, as shown in Figure 10.11. In their most recent paper, Taguchi *et al.* (2008) refine the segment shapes as part of the optimization process, which leads to much improved results, as shown in Figure 10.12.

Even more accurate results are obtained by Klaus *et al.* (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 Middlebury evaluation site, and even now (in early 2009), it is still the second highest ranked algorithm.

As it turns out, 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 neighborhing plane fit parameters. Another highly ranked algorithm by Yang *et al.* (2009) uses the color correlation approach of Yoon and Kweon (2006) and hierarchical belief

Figure 10.13: *Background replacement using z-keying with a bi-layer segmentation algorithm (Kolmogorov* et al. *2006).*

propagation to obtain an initial set of disparity estimates. After left-right consistency checking to detect occluded pixel, the data terms for low-confidence and occluded pixels are recomputed using segmentation-based planes fits, and one or more rounds of hierarchical belief propagation are used to obtain the final disparity estimates.

Another important ability of segment-based stereo algorithms, which they share with algorithms that use explicit layers (Baker *et al.* 1998, Szeliski and Golland 1999) or boundary extraction (Hasinoff *et al.* 2006), is the ability to extract fractional pixel alpha mattes at depth discontinuities. This ability is crucial when attempting to create virtual view interpolation without clinging boundary artifacts (Zitnick *et al.* 2004) and also to seamlessly insert virtual objects (Taguchi *et al.* 2008), as shown in Figure 10.12b.

[Note: Say something about the Middlebury evaluation Web site `http://vision.middlebury.edu/stereo` *(Scharstein and Szeliski 2002), just as you did in §7.4, Figure 7.14. Perhaps show some of the datasets (color images) and ground truth and/or best performing disparity maps.]*

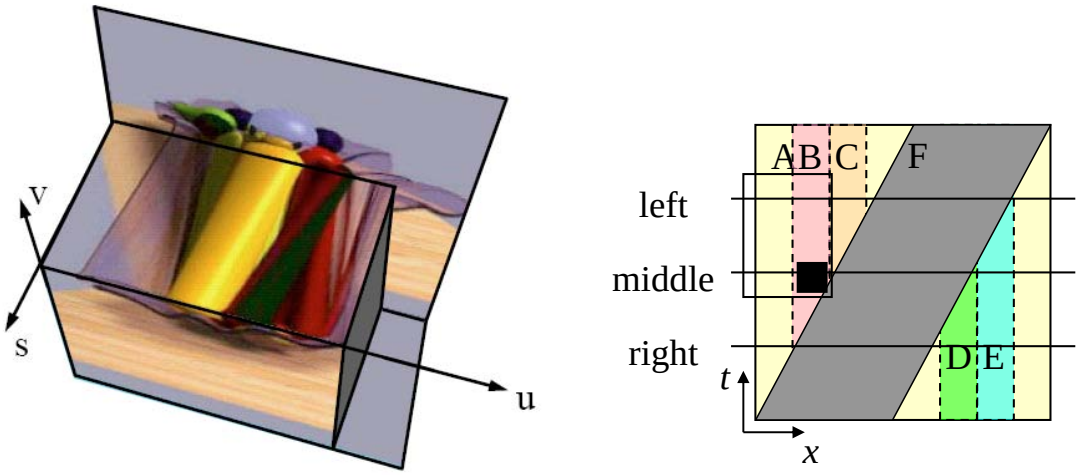
10.5.2 *Application*: Z-keying and background replacement

[Note: Still need to write this section: Figure 10.12b shows one synthetic (non-real-time) example.]

Show some z-keying results (Figure 10.2g), and also virtual view generation (Figure 10.2h–j) (Kanade *et al.* 1996).

Kolmogorov *et al.* (2006)'s newest background replacement results using DP and color segmentation (Figure 10.13)

Forward reference to Virtual Viewpoint Video §13.5



(a) (b)

Figure 10.14: *EPI image from Lumigraph paper (Gortler et al. 1996) and a schematic EPI from (Kang et al. 2001). (a) The Lumigraph (Lightfield) §12.3 is the 4-D space of all light rays passing through a volume of space. Taking a 2D slide results in all light rays embedded in a plane, and is equivalent to a scanline taken from a stacked* epipolar plane image *volume. Objects at different depth 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 10.15 showing the three images (middle, left, right) as slices through the EPI volume. The spatially and temporally shifted window around the black pixel is indicated by the rectangle, showing the the right image is not being used in matching.*

10.6 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 §10.1.2, it is possible to resample all neighboring k images at each disparity hypothesis d into a generalized disparity space 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 (10.4),

$$C(x, y, d) = \sum_k \rho(\tilde{I}(x, y, d, k) - I_r(x, y)). \tag{10.14}$$

This is the basis of the well-known sum of summed-squared-difference (SSSD) and SSAD approaches (Okutomi and Kanade 1993, Kang *et al.* 1995), which can be extended to reason about

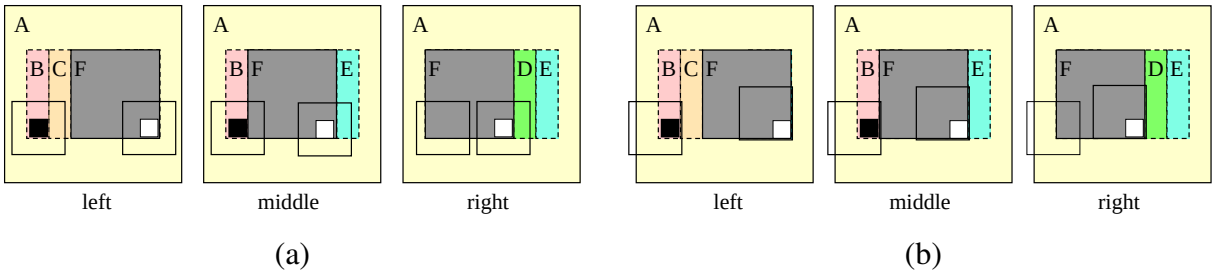
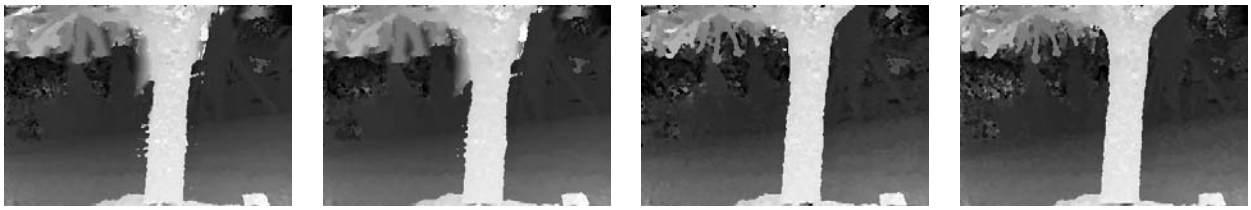


Figure 10.15: *Spatio-temporally shiftable windows (Kang et al. 2001). These figure show a simple three image sequence (the middle image is the reference image) with a moving frontal gray square (marked F) and a stationary background. Regions B, C, D, and E are partially occluded. (a) A regular SSD algorithm will make mistakes when matching pixels in these regions (e.g. the window centered on the black pixel in region B), and also 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 10.14b shows an epipolar plane image (EPI) corresponding to this sequence and describes in more detail how temporal selection works.*

likely patterns of occlusion (Nakamura *et al.* 1996). (More recent work by Gallup *et al.* (2008) show how to adapt the baseline(s) 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 *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 7.15c and 10.14 (Bolles *et al.* 1987, Baker and Bolles 1989, Baker 1989). As you can see in Figure 10.14, as a camera translates horizontally (in a standard horizontally rectified geometry), objects at different depth move sideways at a rate inversely proportional to their depth (10.1).[7] Foreground objects occlude background objects, which can be seen as *EPI-strips* (Criminisi *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 also to make inferences about translucent object (Tsin *et al.* 2006) and specular reflections (Swaminathan *et al.* 2002, Criminisi *et al.* 2005). Alternatively, we can treat the series of images

[7] The 4-dimensional generalization of the EPI is the *lightfield*, which we will study in §12.3. In principle, there is enough information in a lightfield to recover both the objects shapes and their BRDFs, although relatively little progress has been made to date on this topic (Soatto *et al.* 2003).



(a) (b) (c) (d)

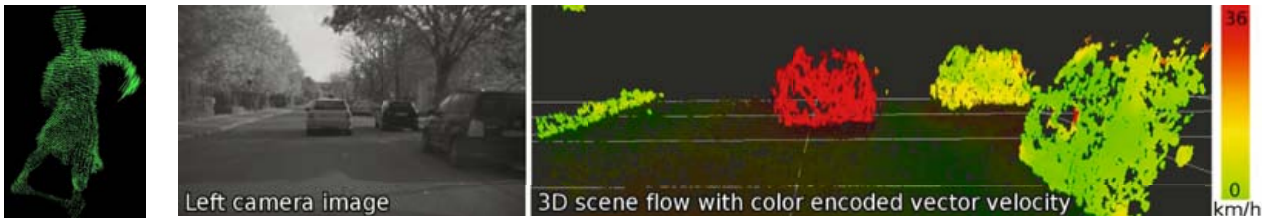
Figure 10.16: *Local (5 × 5 window-based) matching results (Kang* et al. *2001): (a) non-spatially perturbed (centered) window; (b) spatially perturbed window; (c) using best 5 of 10 neighboring frames; (d) using better half sequence. Notice how the results near the tree trunk are improved using temporal selection.*

as a set of sequential observations and merge them together using Kalman filtering (Matthies *et al.* 1989) or maximum likelihood inference (Cox 1994).

When fewer images are available, it becomes necessary to fall back on aggregation techniques such as windows or global optimization. With additional input images, however, the likelihood that some images (or portions of images) will be occluded increases. It is therefore prudent to adjust not only the best window locations using a shiftable window approach, as shown in Figure 10.15a, 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 10.15b (Kang *et al.* 2001). Figure10.14b 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 10.16, 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 in (Kang and Szeliski 2004), 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 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 other ones. As described in §7.4.1 (7.71–7.74), the problem can be formulated as the simultaneous estimation of depth maps at key frames (Figure 7.15c) 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



(a) (b)

Figure 10.17: *Three dimensional scene flow: (a) computed from a multi-camera dome surrounding the dancer shown in Figure 10.2h–j (Vedula et al. 2005b); (b) computed from stereo cameras mounted on a moving vehicle (Wedel et al. 2008).*

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 *et al.* (2008) and Zhang *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 (Vedula *et al.* 2005b, Zhang and Kambhamettu 2003, Pons *et al.* 2007, Huguet and Devernay 2007, Wedel *et al.* 2008). Figure 10.17a shows an image of the 3D scene flow for the tango dancer shown in Figure 10.2h–j, while Figure 10.17b 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 §13.5, as demonstrated in (Vedula *et al.* 2005a).

10.6.1 Volumetric and 3D surface reconstruction

> *The goal of multi-view stereo is to reconstruct a complete 3D object model from a collection of images taken from known camera viewpoints* –Seitz *et al.* (2006)

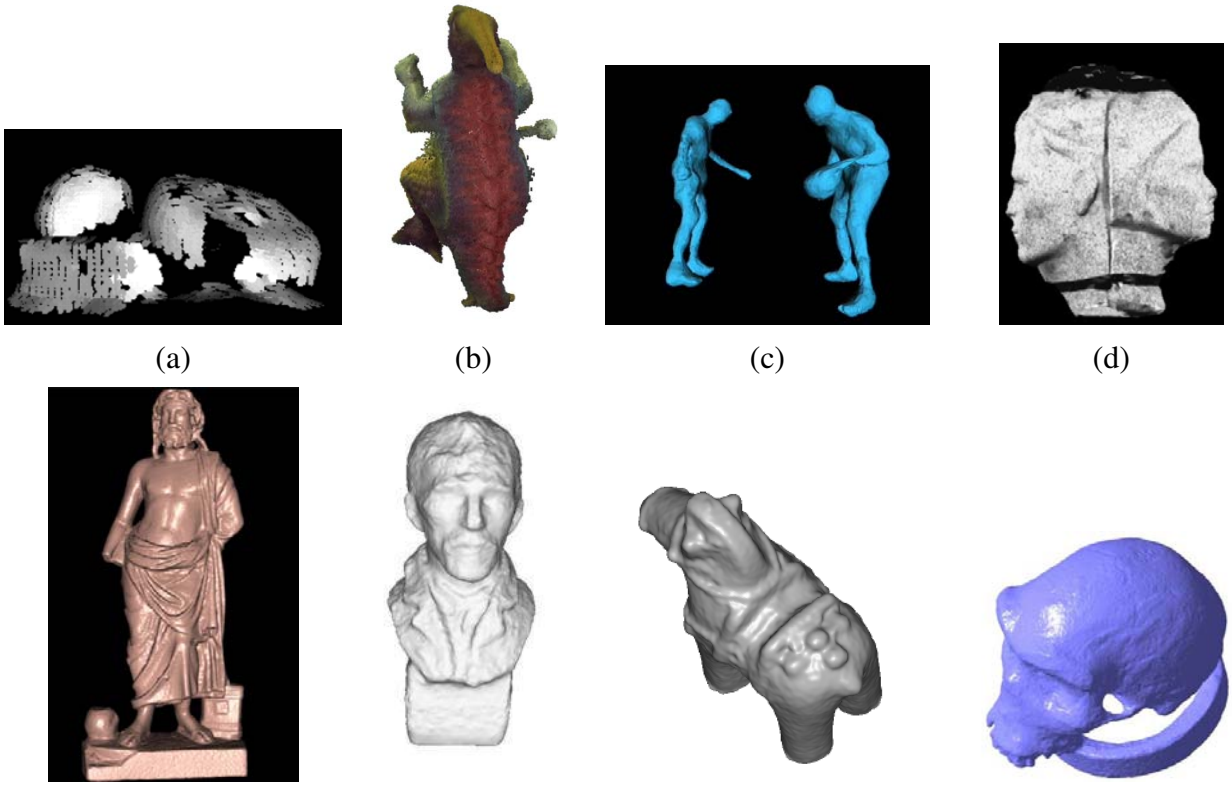
The most challenging, but potentially useful, variant of multi-view stereo reconstruction is to create globally consistent 3D models. This topic has a long history in computer vision, starting with surface mesh reconstruction techniques such as the one of Fua and Leclerc (1995) (Figure 10.18a). A variety 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 *et al.* 2000a, Slabaugh *et al.* 2004, Vogiatzis *et al.* 2007), level sets (Faugeras and Keriven 1998, Pons *et al.* 2007), polygonal meshes (Fua and Leclerc 1995,

Narayanan *et al.* 1998, Hernandez and Schmitt 2004, Furukawa and Ponce 2006), and multiple depth maps (Kolmogorov and Zabih 2002). Figure 10.18 shows representative examples of 3D object models reconstructed using some of these techniques.

In order to organize and compare this large number of techniques, Seitz *et al.* (2006) developed a six point taxonomomy 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 *et al.* 2006) along with the evaluation web site `http://vision.middlebury.edu/mview/`, which contains pointers to even more recent papers and results.

Scene representation. As mentioned above, one of the more popular 3D representations is a uniform grid of 3D voxels, which can be reconstructed using a variety of carving (Seitz and Dyer 1999, Kutulakos and Seitz 2000) or optimization (Vogiatzis *et al.* 2007) techniques. Level set techniques §4.4.3 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 *et al.* 2007), which can encode a finer level of detail. Polygonal meshes are another popular representation (Fua and Leclerc 1995, Narayanan *et al.* 1998, Isidoro and Sclaroff 2003, Hernandez and Schmitt 2004, Furukawa and Ponce 2006), which is both the standard representation used in computer graphics, and also readily supports 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 represenatation, e.g., they may start by computing multiple depth maps and then merge them into a 3D object model (Narayanan *et al.* 1998, Furukawa and Ponce 2006, Goesele *et al.* 2006, Goesele *et al.* 2007). *[Note: This paragraph repeats some elements with the previous overview. Thin out the first one, add more references here?]*

Photoconsistency measure. As we discussed in §10.3.1, 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 to project 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 *et al.* 2007).



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 10.18: *Examples of multi-view stereo algorithms: (a) surface-based stereo (Fua and Leclerc 1995); (b) voxel coloring (Seitz and Dyer 1999); (c) depth map merging (Narayanan* et al. *1998); (d) level set evolution (Faugeras and Keriven 1998); (e) silhouette and stereo fusion (Hernandez and Schmitt 2004); (f) multi-view image matching (Pons* et al. *2005); (g) volumetric graph cut (Vogiatzis* et al. *2005); (h) carved visual hulls (Furukawa and Ponce 2006).*
[Note: Download these papers and check that these are all the correct sources for the image (especially (Pons et al. *2005), since they are guessed from Brian's captions in his MView talk. If possible, get better figures, e.g., on white backgrounds.]*

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, e.g., (Kutulakos and Seitz 2000, Faugeras and Keriven 1998, Vogiatzis *et al.* 2007), are classified as using *geometric visibility models* in the taxonomy of Seitz *et al.* (2006). Techniques that select a neighboring subset of image to match are called *quasi-geometric* (Narayanan *et al.* 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, it can be very slow to evaluate, and also 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 in (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, Vogiatzis *et al.* 2007). Approach 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. Finally, multiple depth map approach 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. Graph-cut based approaches use a polynomial-time binary segmentation algorithm to recover the object model defined on the voxel grid (Vogiatzis *et al.* 2007). 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 *et al.* 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 get labelled 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

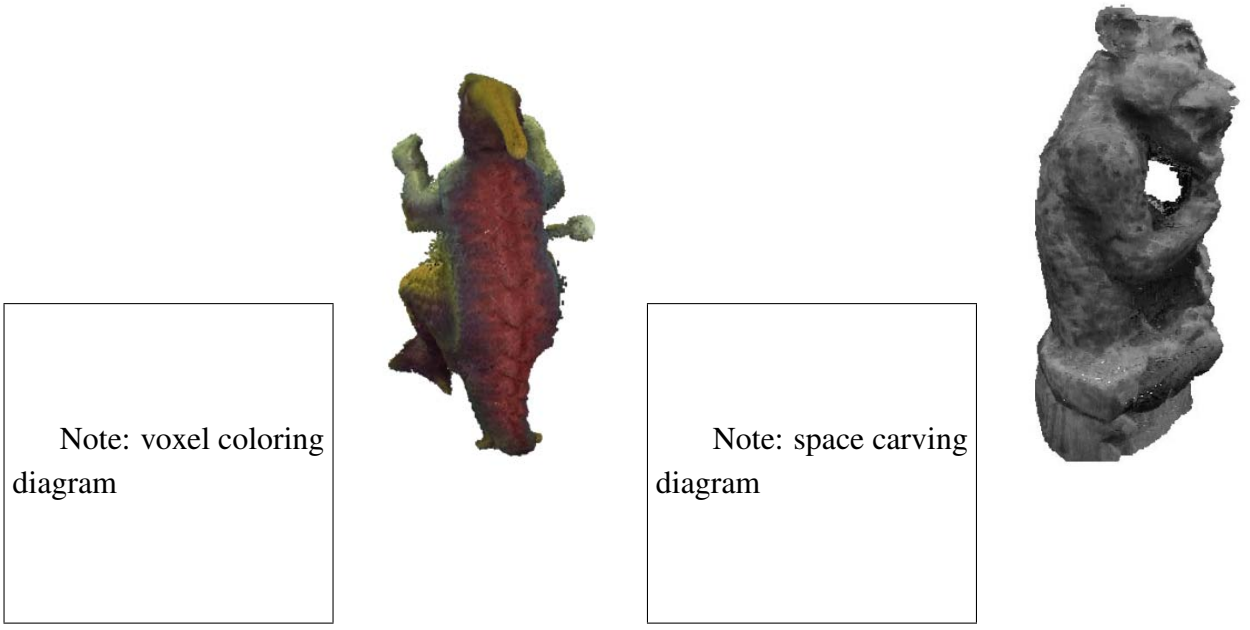


Figure 10.19: *Voxel coloring (Seitz and Dyer 1999) and space carving (Kutulakos and Seitz 2000). [Note: Generate some schematic diagrams for voxel coloring and space carving.]*

to further photoconsistency computations. (A similar approach, albeit without the front to back sweep order, is used in (Szeliski and Golland 1999).) The resulting 3D volume, under noise and resampling-free conditions, is guaranteed to produce both a photoconsistent 3D model and to enclose whatever true 3D object model generated the images (Figure 10.19a–b).

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 satify the voxel coloring constraint are iteratively selected and the 3D voxel grid is alternately carved away along different axes (Figure 10.19c–d).

Another popular approach to multi-view stereo is to first independently compute multiple depth maps and to then merge these (potentially partial) maps into a complete 3D model. Approaches to depth map merging, which are discussed in more detail in §11.4.1, include signed distance functions (Curless and Levoy 1996), used in (Goesele *et al.* 2006), and Poisson surface reconstruction (Kazhdan *et al.* 2006), used by Goesele *et al.* (2007). It is also possible to reconstruct sparser representations such as 3D points and lines, and to then interpolate these to full 3D surfaces §11.5.1 (Taylor 2003).

Figure 10.20: *The six multi-view stereo data sets captured by Seitz* et al. *(2006). Only the first two (temple and dino) are currently being used for the evaluations.*

Initialization requirements. One final element discussed by Seitz *et al.* (2006) is the varying degrees of intialization required by different algorithms. Because some algorithm 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 §11.3.1.

Empirical evaluation. In order to evaluate the large number of design alternatives in multi-view stereo, Seitz *et al.* (2006) collected a dataset of calibrated images using a sphercial gantry. A representative image from each of the six datasets is shown in Figure 10.20, although only the first two datasets have been fully processed and used so far for evaluation. Figure 10.21 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 (Seitz *et al.* 2006), the field of multi-view stereo has continued to advanced at a rapid pace (Strecha *et al.* 2006, Kolev *et al.* 2007, Hernandez *et al.* 2007, Habbecke and Kobbelt 2007, Furukawa and Ponce 2007, Vogiatzis *et al.* 2007, Goesele *et al.* 2007, Sinha *et al.* 2007, Gargallo *et al.* 2007, Merrell *et al.* 2007, Zach *et al.* 2007, Furukawa and Ponce 2008a, Hornung *et al.* 2008, Bradley *et al.* 2008, Campbell *et al.* 2008). The multi-view stereo evaluation web site `http://vision.middlebury.edu/mview/` provides quantitative results for these algorithms along with pointers to where to find these papers.

[Note: If I have more time when editing the book (or if reviewers / readers have some suggestions), I should go back and look at these papers to see if I want to call out any particular papers or ideas individually.]

[Note: Put in more applications after the stereo section?]

10.7 Exercises

Ex 10.1 (Stereo pair rectification) Implement the following simple algorithm §10.1.1:

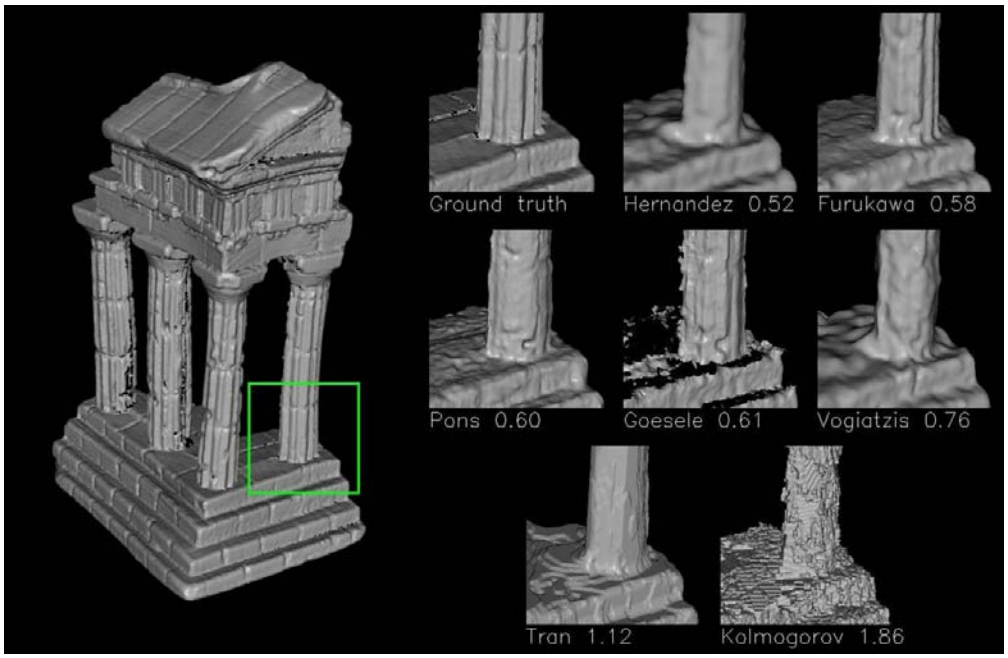


Figure 10.21: *Reconstruction results (details) for 7 algorithms (Hernandez and Schmitt 2004, Furukawa and Ponce 2006, Pons* et al. *2005, Goesele* et al. *2006, Vogiatzis* et al. *2005, Tran and Davis 2002, Kolmogorov and Zabih 2002) evaluated by Seitz* et al. *(2006) on the 47-image Temple Ring dataset. The numbers underneath each detail image are the accuracy of each of these techniques measured in millimeters.*
[Note: No need to get permissions for this figure, since the images are from our talk and are not in the CVPR paper.]

1. Rotate both cameras so that they are looking perpendicular to line joining the two camera centers $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$. 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.)

Now compare your results to the algorithm proposed by Loop and Zhang (1999). Can you think of situations where their approach may be preferable. *[Hint: Possibly when the two cameras are heavily verged, i.e., looking at a nearby object?] [Note: I'd better compare it myself to make sure I haven't re-discovered the same algorithm.]*

Ex 10.2 (Rigid direct alignment) Modify your spline-based or optical flow motion estimator from Exercise 7.4 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 simultaneously estimating a base homography corresponding to a reference plane for the dominant motion and then an epipole for the residual parallax (motion).

Ex 10.3 (Plane sweep) Implement a plane sweep algorithm §10.1.2.

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 §10.3.1 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/warped images). Also, try pre-compensating images for bias/gain variations using one or more of the techniques discussed in §10.3.1.

Ex 10.4 (Aggregation and window-based stereo) Implement one or more of the matching cost aggregation strategies described in §10.4, e.g.,

- 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, Veksler 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:

- compute matches both ways, and pick only the reliable matches (draw the others in another color);
- tag matches that are unsure (whose confidence is too low);
- fill in the matches that are unsure from neighborhing values;
- 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 10.5 (Optimization-based stereo) Compute the disparity space image volume (DSI) using one of the techniques you implemented in Exercise 10.3 and then implement (or more) one of the global optimization techniques described in §10.5 to compute the depth map. Potential choices include:

1. Dynamic programming or scanline optimization (relatively easy).
2. Semi-global optimization (Hirschmüller 2008), which is a simple extension of scanline optimization and performs well.
3. Graph cuts using alpha expansions (Boykov *et al.* 2001), for which you will need to find a max-flow / min-cut algorithm, e.g., `http://vision.middlebury.edu/stereo`.
4. Loopy belief propagation §B.6.3.

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

How well does your algorithm do against local aggregation, e.g., (Yoon and Kweon 2006)?

Can you think of some extensions or modifications to make it even better?

Ex 10.6 (View interpolation, revisited) Compute a dense depth map using one of the techniques you developed above, and use this depth map (or better yet, a depth map per 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 these? *[Hint: §3.5.2 and §12 discuss these problems in more detail.]*

Ex 10.7 (Multi-frame stereo) Extend one of your previous techniques to use multiple input frames §10.6 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 10.8 (Volumetric stereo) Implement voxel coloring (Seitz and Dyer 1999) as a simple extension to the plane sweep algorithm you implemented in Exercise 10.3.

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 *et al.* 2000a, Kutulakos 2000, 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., to only consider pixels that have full alpha or to 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 10.9 (Depth map merging) Use the technique you developed for multi-frame stereo in Exercise 10.7, or a different technique such as the one described in (Goesele *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 *et al.* 2006).