

**Figure 12.20** Alpha-expansion graph setup. Each pixel node (a,b,c,d,e) is connected to the source and the sink by edges with costs  $U_{\bullet}(\bar{\alpha})$  and  $U_{\bullet}(\alpha)$ , respectively. In the minimum cut, exactly one of these links will be cut. The nodes and vertices describing the relationship between neighboring pixels depend on their current labels, which may be  $\alpha-\alpha$  as for pixels  $a$  and  $b$ ,  $\alpha-\beta$  as for pixels  $b$  and  $c$ ,  $\beta-\beta$  as for pixels  $c$  and  $d$  or  $\beta-\gamma$  as for pixels  $d$  and  $e$ . For the last case, an auxiliary node  $k$  must be added to the graph.

$\alpha$ . Accordingly, we associate the unary costs for each edge being set to  $\alpha$  or its original label with the two links from each pixel. If the pixel already has label  $\alpha$ , then we set the cost of being set to  $\bar{\alpha}$  to  $\infty$ .

The remaining structure of the graph is dynamic: it changes at each iteration depending on the choice of  $\alpha$  and the current labels. There are four possible relationships between adjacent pixels:

- Pixel  $i$  has label  $\alpha$  and the pixel  $j$  has label  $\alpha$ . Here, the final configuration is inevitably  $\alpha-\alpha$ , and so the pairwise cost is zero and there is no need to add further edges connecting nodes  $i$  and  $j$  in the graph. Pixels  $a$  and  $b$  in figure 12.20 have this relationship.
- The first pixel has label  $\alpha$  but the second pixel has a different label  $\beta$ . Here the final solution may be  $\alpha-\alpha$  with zero pairwise cost or  $\alpha-\beta$  with pairwise cost  $P_{ij}(\alpha, \beta)$ . Here we add a single edge connecting pixel  $j$  to pixel  $i$  with pairwise cost  $P_{ij}(\alpha, \beta)$ . Pixels  $b$  and  $c$  in figure 12.20 have this relationship.
- Both pixels  $i$  and  $j$  have the same label  $\beta$ . Here the final solution may be  $\alpha-\alpha$  with zero pairwise cost,  $\beta-\beta$  with zero pairwise cost,  $\alpha-\beta$  with pairwise cost  $P_{ij}(\alpha, \beta)$  or  $\beta-\alpha$  with pairwise cost  $P_{ij}(\beta, \alpha)$ . We add two edges between the pixels representing the two non-zero pairwise costs. Pixels  $c$  and  $d$  in figure 12.20 have this relationship.
- Pixel  $i$  has label  $\beta$  and pixel  $j$  has a second label  $\gamma$ . Here the final solution may be  $\alpha-\alpha$  with zero pairwise cost,  $\beta-\gamma$  with pairwise cost  $P_{ij}(\beta, \gamma)$ ,  $\beta-\alpha$

with pairwise cost  $P_{ij}(\beta, \alpha)$ , or  $\alpha - \gamma$  with pairwise cost  $P_{ij}(\alpha, \gamma)$ . We add a new vertex  $k$  between vertices  $i$  and  $j$ , and add the three non-zero pairwise costs to edges  $k - \alpha$ ,  $i - k$ , and  $j - k$ , respectively. Pixels  $d$  and  $e$  in figure 12.20 have this relationship.

Three example cuts are shown in figure 12.21.

Note that this construction critically relies on the triangle inequality (equation 12.22). For example, consider pixels  $d$  and  $e$  in figure 12.21a. If the triangle inequality does not hold so that  $P_{de}(\beta, \gamma) > P_{de}(\beta, \alpha) + P_{de}(\alpha, \gamma)$ , then the wrong costs will be assigned; rather than the link  $k - \alpha$ , the two links  $d - k$  and  $e - k$  will both be cut, and the wrong cost will be assigned. In practice, it is sometimes possible to ignore this constraint by *truncating* the offending cost  $P_{ij}(\beta, \gamma)$  and running the algorithm as normal. After the cut is done, the true objective function (sum of the unary and pairwise costs) can be computed for the new label map and the answer accepted if the cost has decreased.

**Problem 12.9**

It should be emphasized that although each step optimally updates the objective function with respect to expanding  $\alpha$ , this algorithm is not guaranteed to converge to the overall global minimum. However, it can be proven that the result is within a factor of two of the minimum and often it behaves much better.

Figure 12.22 shows an example of multi-label denoising using the alpha-expansion algorithm. On each iteration, one of the labels is chosen and expands, and the appropriate region is denoised. Sometimes the label is not supported at all by the unary costs and nothing happens. The algorithm terminates when no choice of  $\alpha$  causes any further change.

## 12.5 Conditional random fields

In the models presented in this chapter, the Markov random fields have described the prior  $Pr(\mathbf{w})$  in a generative model of the image data. We could alternatively describe the joint probability distribution  $Pr(\mathbf{w}, \mathbf{x})$  with the undirected model

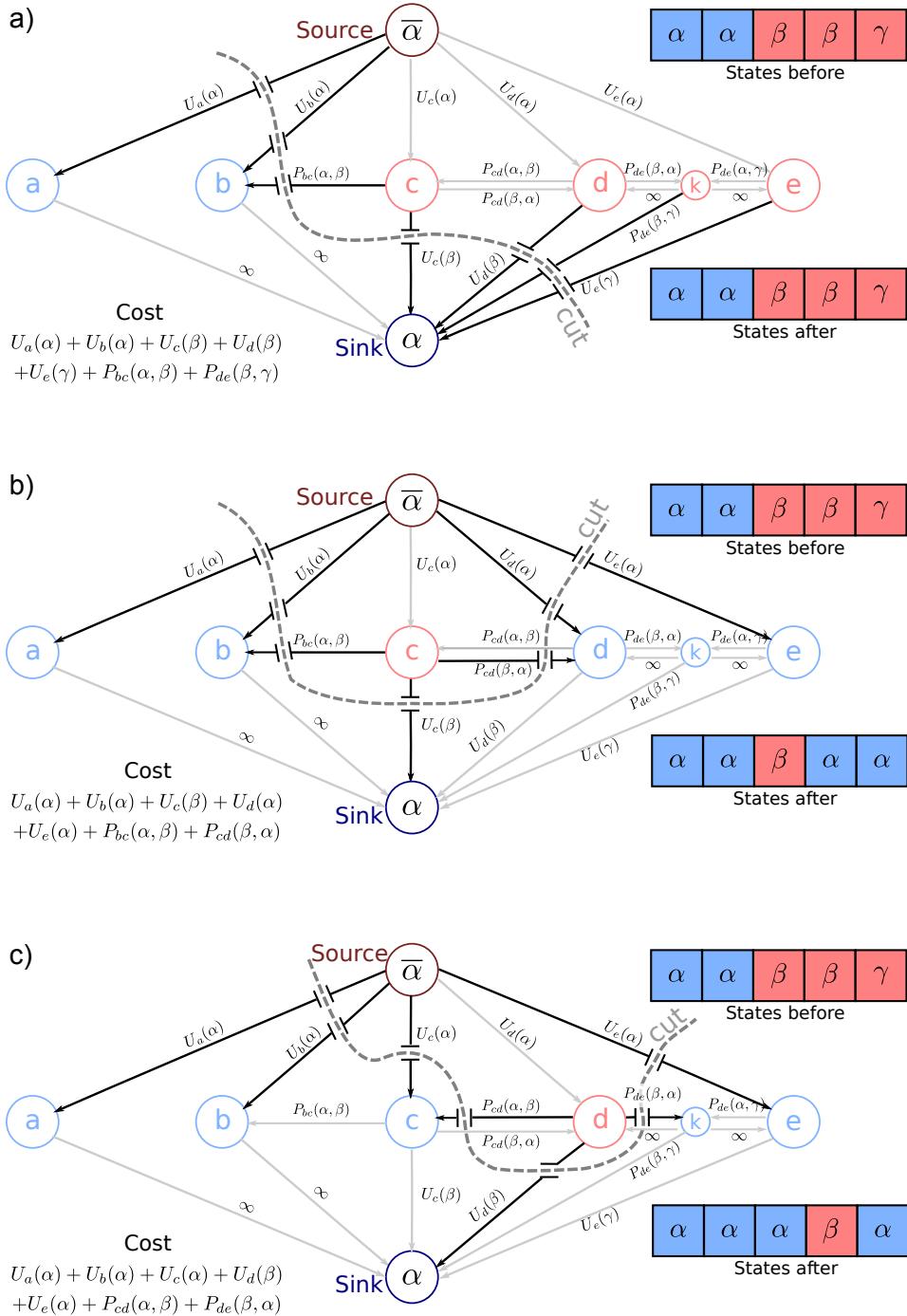
$$Pr(\mathbf{w}, \mathbf{x}) = \frac{1}{Z} \exp \left[ - \sum_c \psi_C[\mathbf{w}] - \sum_d \zeta[\mathbf{w}, \mathbf{x}] \right], \quad (12.23)$$

where the functions  $\psi[\bullet]$  encourage certain configurations of the label field, and the functions  $\zeta[\bullet, \bullet]$  encourage agreement between the data and the label field. If we now condition on the data (i.e., assume that it is fixed), then we can use the relation  $Pr(\mathbf{w}|\mathbf{x}) \propto Pr(\mathbf{w}, \mathbf{x})$  to write

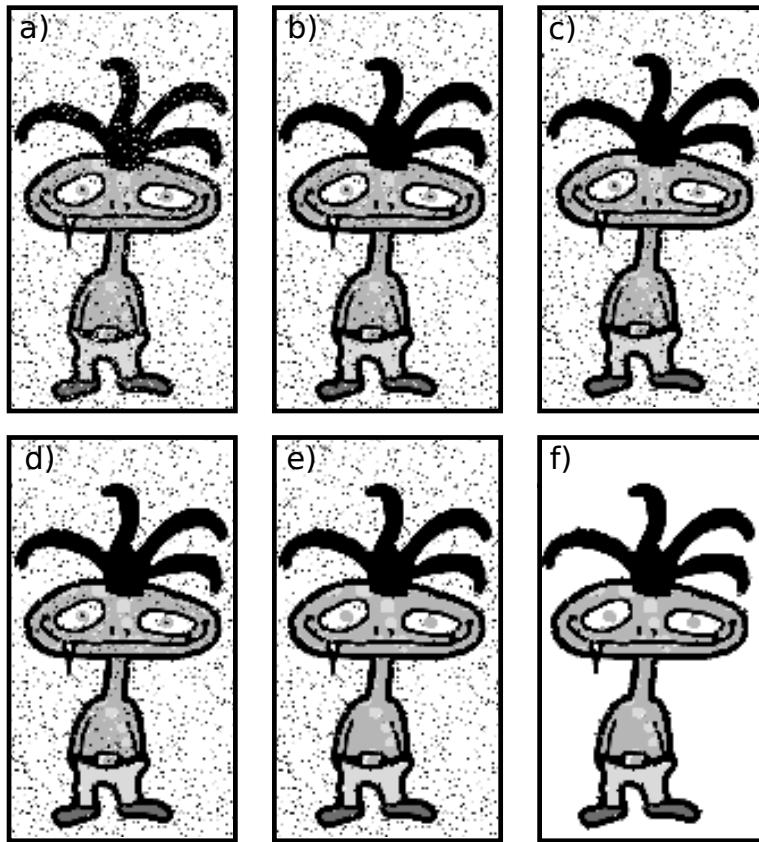
$$Pr(\mathbf{w}|\mathbf{x}) = \frac{1}{Z_2} \exp \left[ - \sum_c \psi_C[\mathbf{w}] - \sum_d \zeta[\mathbf{w}, \mathbf{x}] \right], \quad (12.24)$$

where  $Z_2 = Z \cdot Pr(\mathbf{x})$ . This discriminative model is known as a *conditional random field* or *CRF*.

We can choose the functions  $\zeta[\bullet, \bullet]$  so that they each determine the compatibility of one label  $w_n$  to its associated measurement  $x_n$ . If the functions  $\psi[\bullet]$  are used to



**Figure 12.21** Alpha-expansion algorithm. a-c) Example cuts on this graph illustrate that the appropriate unary and pairwise costs are always paid.

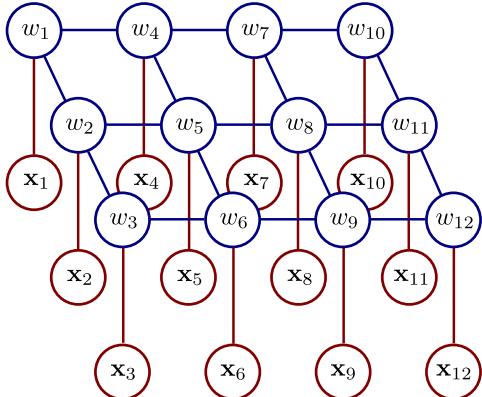


**Figure 12.22** Alpha-expansion algorithm for denoising task. a) Observed noisy image. b) Label 1 (black) is expanded, removing noise from the hair. c-f) Subsequent iterations in which the labels corresponding to the boots, trousers, skin and background are expanded respectively.

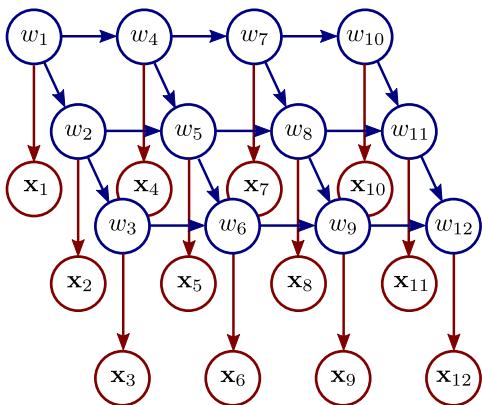
encourage smoothness between neighboring labels, then the negative log posterior probability will again be the sum of unary and pairwise terms. The MAP labels  $\hat{\mathbf{w}}$  can hence be found by minimizing a cost function of the form

$$\hat{\mathbf{w}} = \operatorname{argmin}_{w_1 \dots w_N} \left[ \sum_{n=1}^N U_n(w_n) + \sum_{(m,n) \in \mathcal{C}} P_{mn}(w_m, w_n) \right], \quad (12.25)$$

and the graphical model will be as in figure 12.23. This cost function can be minimized using the graph cuts techniques described throughout this chapter.



**Figure 12.23** Graphical model for conditional random field (compare to figure 12.4). The posterior probability of the labels  $\mathbf{w}$  is a Markov random field for fixed data  $\mathbf{x}$ . In this model, the two sets of cliques relate (i) neighbouring labels and (ii) each label to its associated measurement. Since this model only includes unary and pairwise interactions between the labels, the unknown labels  $\{w_n\}_{n=1}^N$  can be optimized using graph cut techniques.



**Figure 12.24** Directed graphical model for grid. Although this model appears similar to the pairwise Markov random field model, it represents a different factorization of the joint probability. In particular the factorization contains terms involving three variables such as  $Pr(w_5|w_2, w_4)$ . This means that the resulting cost function for MAP inference is no longer amenable to exact solution using graph-cut methods. In this case an attractive alternative is to use sampling based methods as it is easy to generate samples from this directed model.

## 12.6 Higher order models

The models that we have discussed so far have only connected immediate neighbors. However, these only allow us to model relatively simple statistical properties of the label field. One way to improve this situation is to consider each variable  $w_n \in \{1 \dots K\}$  as representing the index of a square patch of labels from a predefined library. The pairwise MRF now encodes the affinity of neighboring patches for each other. Unfortunately, the resulting costs are less likely to be submodular, or even obey the triangle inequality, and the number  $K$  of patches in the library is usually very large, making graph cut algorithms inefficient.

A second approach to modeling more complex statistical properties of the label field is to increase the number of the connections. For the undirected models (CRF, MRF) this would mean introducing larger cliques. For example, to model local texture, we might connect all of the variables in every  $5 \times 5$  region of the image. Unfortunately, inference is hard in these models; optimizing the resulting complex cost functions is still an open research topic.

## 12.7 Directed models for grids

The Markov random field and conditional random field models are attractive because we can use graph-cuts approaches to search for the MAP solution. However, they have the drawback that it is very hard to learn the parameters of the model because they are based on undirected models. An obvious alternative is to use a similar directed model (figure 12.24). Here, learning is relatively easy, but it turns out that MAP inference using graph cuts is not generally possible.

To see this, consider the cost function for MAP inference in this model,

$$\begin{aligned}\hat{w}_{1\dots N} &= \operatorname{argmax}_{w_{1\dots N}} [\log[Pr(\mathbf{x}_{1\dots N}|w_{1\dots N})] + \log[Pr(w_{1\dots N})]] \\ &= \operatorname{argmax}_{w_{1\dots N}} \left[ \sum_{n=1}^N \log[Pr(\mathbf{x}_n|w_n)] + \sum_{n=1}^N \log[Pr(w_n|w_{pa[n]})] \right] \\ &= \operatorname{argmin}_{w_{1\dots N}} \left[ \sum_{n=1}^N -\log[Pr(\mathbf{x}_n|w_n)] - \sum_{n=1}^N \log[Pr(w_n|w_{pa[n]})] \right],\end{aligned}\quad (12.26)$$

where we have multiplied the objective function by minus one and now seek the minimum. This minimization problem now has the general form

$$\hat{w}_{1\dots N} = \operatorname{argmin}_{w_{1\dots N}} \left[ \sum_{n=1}^N U_n(w_n) + \sum_{n=1}^N T_n(w_n, w_{pa_1[n]}, w_{pa_2[n]}) \right], \quad (12.27)$$

where  $U_n(w_n)$  is called a *unary term* reflecting the fact that it only depends on a single element  $w_n$  of the label field and  $T_n(w_n, w_{pa_1[n]}, w_{pa_2[n]})$  is called a *three-wise term* reflecting the fact in general the label at a pixel is conditioned on the two parents  $pa_1[n]$  and  $pa_2[n]$  above and to the left of the current position.

Notice that this cost function is fundamentally different from the cost function for MAP inference in a pairwise MRF (equation 12.11): it includes three-wise terms and there is no known polynomial algorithm to optimize this criterion. However, since this model is a directed graphical model, it is easy to generate samples from this model, and this can be exploited for approximate inference methods such as computing the empirical max marginals.

## 12.8 Applications

The models and algorithms in this chapter are used in a large number of computer vision applications, including stereo vision, motion estimation, background subtraction, interactive segmentation, semantic segmentation, image editing, image denoising, image super-resolution, and building 3D models. Here, we review a few key examples. We consider background subtraction which is a simple application with binary labels and interactive segmentation which uses binary labels in a system that simultaneously estimates the parameters in the likelihood terms. Then we

consider stereo, motion estimation, and image editing, all of which are multi-label graph-cut problems. We consider super-resolution which is a multi-label problem where the units are patches rather than pixels and which there are so many labels that the alpha-expansion algorithm is not suitable. Finally, we consider drawing samples from directed grid models to generate novel images.

### 12.8.1 Background subtraction

First, let us revisit the background subtraction algorithm that we first encountered in section 6.6.2. In background subtraction, the goal is to associate a binary label  $\{w_n\}_{n=1}^N$  with each of the  $N$  pixels in the image, indicating whether this pixel belongs to the foreground or background, based on the observed RGB data  $\{\mathbf{x}_n\}_{n=1}^N$  at each pixel. When the pixel is background ( $w_n = 0$ ), the data are assumed to be generated from a normal distribution with known mean  $\boldsymbol{\mu}_{n0}$  and covariance  $\boldsymbol{\Sigma}_{n0}$ . When the pixel is foreground ( $w_n = 1$ ), a uniform distribution over the data is assumed, so that

$$\begin{aligned} Pr(\mathbf{x}_n | w = 0) &= \text{Norm}_{\mathbf{x}_n}[\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n] \\ Pr(\mathbf{x}_n | w = 1) &= \kappa, \end{aligned} \quad (12.28)$$

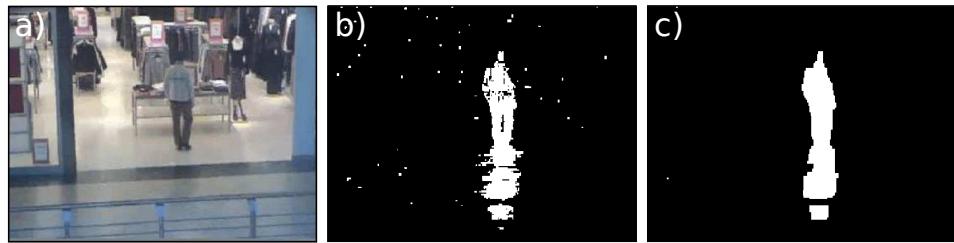
where  $\kappa$  is a constant.

In the original description, we assumed that the models at each pixel were independent, and when we inferred the labels, the results were noisy (figure 12.25b). We now place a Markov random field prior over the binary labels where the pairwise cliques are organized as a grid (as in most of the models in this chapter) and where the potential functions encourage smoothness. Figure 12.25 illustrates the results of performing inference in this model using the graph cuts algorithm. There are now far fewer isolated foreground regions and fewer holes in the foreground object. The model has still erroneously discovered the shadow; a more sophisticated model would be required to deal with this problem.

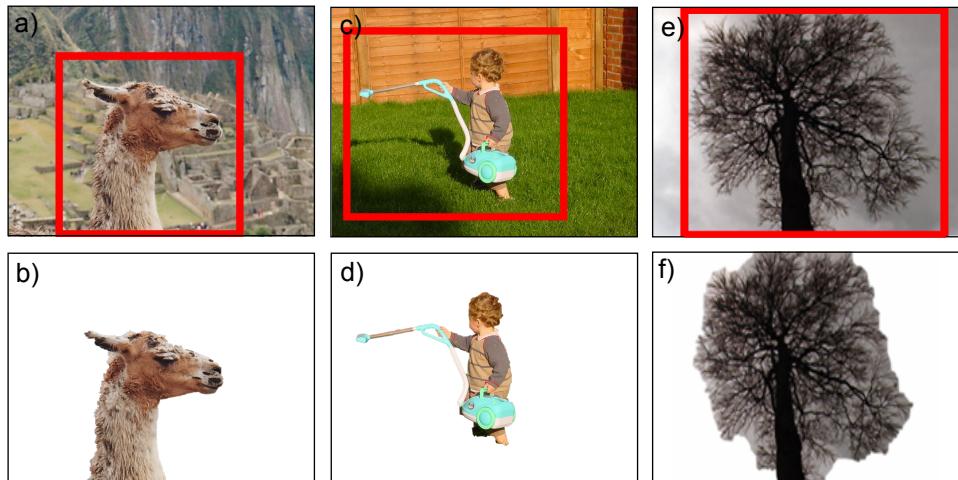
### 12.8.2 Interactive segmentation (GrabCut)

The goal of interactive segmentation is to cut out the foreground object in a photo, based on some input from the user (figure 12.26). More precisely, we aim to associate a binary label  $\{w_n\}_{n=1}^N$  to each of the  $N$  pixels in the image, indicating whether this pixel belongs to the foreground or background, based on the observed RGB data  $\{\mathbf{x}_n\}_{n=1}^N$  at each pixel. However, unlike background subtraction, we do not have any prior knowledge of either the foreground or the background.

In the GrabCut system of Rother *et al.* (2005) the likelihoods of observing the background ( $w = 0$ ) and foreground ( $w = 1$ ) are each modeled as a mixture of  $K$  Gaussians so that



**Figure 12.25** Background subtraction revisited. a) Original image. b) MAP solution of background subtraction model with independent pixels. The solution contains noise. c) MAP solution of background subtraction model with Markov random field prior. This smoothed solution has eliminated most of the noise.



**Figure 12.26** Grab Cut. a) The user draws a bounding box around the object of interest. b) The algorithm segments the foreground from the background by alternating between building color models and segmenting the image. c-d) A second example. e-f) Failure mode. This algorithm does not segment ‘wiry’ objects well as the pairwise costs for tracing around all the boundaries are prohibitive. Adapted from Rother *et al.* (2005). ©2005 ACM.

$$Pr(\mathbf{x}_n | w = j) = \sum_{k=1}^K \lambda_{jk} \text{Norm}_{\mathbf{x}_n}[\boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}], \quad (12.29)$$

and the prior over the labels is modeled as a pairwise connected Markov random field with the potentials chosen to encourage smoothness.

In this application, the image may have a wide variety of content, and so there is no suitable training data from which to learn the parameters  $\{\lambda_{jk}, \mu_{jk}, \Sigma_{jk}\}_{j=1,k=1}^{2,K}$  of the foreground and background color models. However, we note that (i) if we knew the color models, we could perform the segmentation via MAP inference with the graph cuts algorithm and (ii) if we knew the segmentation, then we could compute the foreground and background color models based on the pixels assigned to each category. This observation leads to an alternating approach to inference in this model, in which the segmentation and parameters are computed in turn until the system converges.

In the *Grabcut* algorithm, the user draws a bounding box around the desired object to be segmented. This effectively defines a rough segmentation (pixels within the box are foreground, and pixels outside are background) from which the system is initialized. If the segmentation is not correct after the alternating optimization algorithm converges, the user may ‘paint’ regions of the image with a foreground or background brush, indicating that these *must* belong to the appropriate class in the final solution. In practice, this means that the unary costs are set to ensure that these take the appropriate values, and the alternating solution is run again from this point until convergence. Example results are shown in figure 12.26.

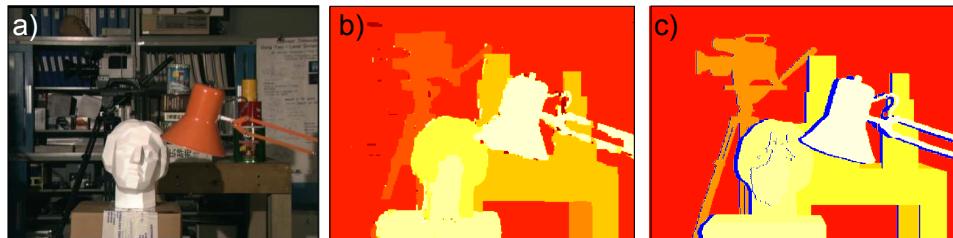
To improve the performance of this algorithm, it is possible to modify the MRF so that the pairwise cost for changing from foreground to background label is less where there is an edge in the image. This is referred to as using *geodesic distance*. From a pure probabilistic viewpoint, this is somewhat dubious as the MRF prior should embody what we know about the task before seeing the data, and hence cannot depend on the image. However, this is largely a philosophical objection, and the method works well in practice for a wide variety of objects. A notable failure mode is in segmenting ‘wiry’ objects such as trees. Here, the model is not prepared to pay the extensive pairwise costs to cut exactly around the many edges of the object, and so the segmentation is poor.

### 12.8.3 Stereo vision

In stereo vision, the goal is to infer a discrete multivalued label  $\{w_n\}_{n=1}^N$  representing the disparity (horizontal shift) at each pixel in the image, given the observed image data  $\{\mathbf{x}_n\}_{n=1}^N$ . More details about the likelihood terms in this problem can be found in section 11.8.2, where we described tree-based priors for the unknown disparities. A more suitable approach is to use an MRF prior.

As for the denoising example, it is undesirable to use an MRF prior where the costs are a convex function of the difference in neighboring labels. This results in a MAP solution where the edges of objects are smoothed. Hence, it is usual to use a non-convex prior such as the Potts function, which embodies the idea that the scene consists of smooth surfaces, with sudden jumps in depth between them where the size of the jump is unimportant.

Boykov *et al.* (1999) used the alpha-expansion algorithm to perform approximate inference in a model of this sort (figure 12.27). The performance of this algorithm is good, but errors are found where there is no true match in the other image (i.e., where the corresponding point is occluded by another object). Kol-



**Figure 12.27** Stereo vision. a) One image of the original stereo pair. b) Disparity estimated using the method of Boykov *et al.* (1999). c) Ground truth disparity. Blue pixels indicate regions which are occluded in the second image, and so do not have a valid match or disparity. The algorithm does not take account of this fact, and produces noisy estimates in these regions. Adapted from Boykov *et al.* (1999).

mogorov & Zabih (2001) subsequently developed a bespoke graph for dealing with occlusions in stereo vision, and an alpha-expansion algorithm for optimizing the associated cost function. These methods can also be applied to *optical flow* in which we attempt to identify pixel correspondences between adjacent frames in a video sequence. Unlike in stereo vision, there is no guarantee that the corresponding matches will be on the same scanline, but other than this, the problem is very similar.

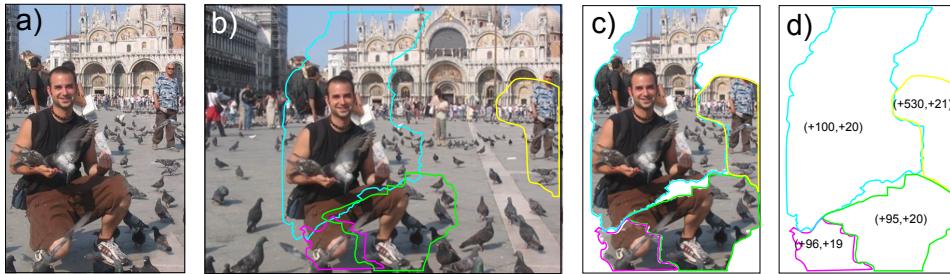
#### 12.8.4 Rearranging Images

Markov random field models can also be used for rearranging images; we are given an original image  $I^{(1)}$  and wish to create a new image  $I^{(2)}$  by rearranging the pixels from  $I^{(1)}$  in some way. Depending on the application, we may wish to change the dimensions of the original image (termed *image re-targeting*), remove an object or move an object from one place to another.

Pritch *et al.* (2009) constructed a model with variables  $\mathbf{w} = \{w_1 \dots w_N\}$  at each of the  $N$  pixels of  $I^{(2)}$ . Each possible value of  $w_n \in \{1 \dots K\}$  represents a 2D relative offset to image  $I^{(1)}$  that tells us which pixel from image  $I^{(1)}$  will appear at the  $n^{\text{th}}$  pixel of the new image. The label map  $\mathbf{w}$  is hence termed a shift-map, as it represents 2D shifts to the original image. Each possible shift-map defines a different output image  $I^{(2)}$  (figure 12.28).

Pritch *et al.* (2009) model the shift-map  $\mathbf{w}$  as an MRF with pairwise costs that encourage smoothness. The result of this is that only shift-maps that are piecewise constant have high probability: in other words, new images which consist of large chunks of the original image that have been copied verbatim, are favored. They modify the pairwise costs, so that they are lower when adjacent labels encode offsets with similar surrounding regions. This means that where the label does change, it does so in such a way that there is no visible seam in the output image.

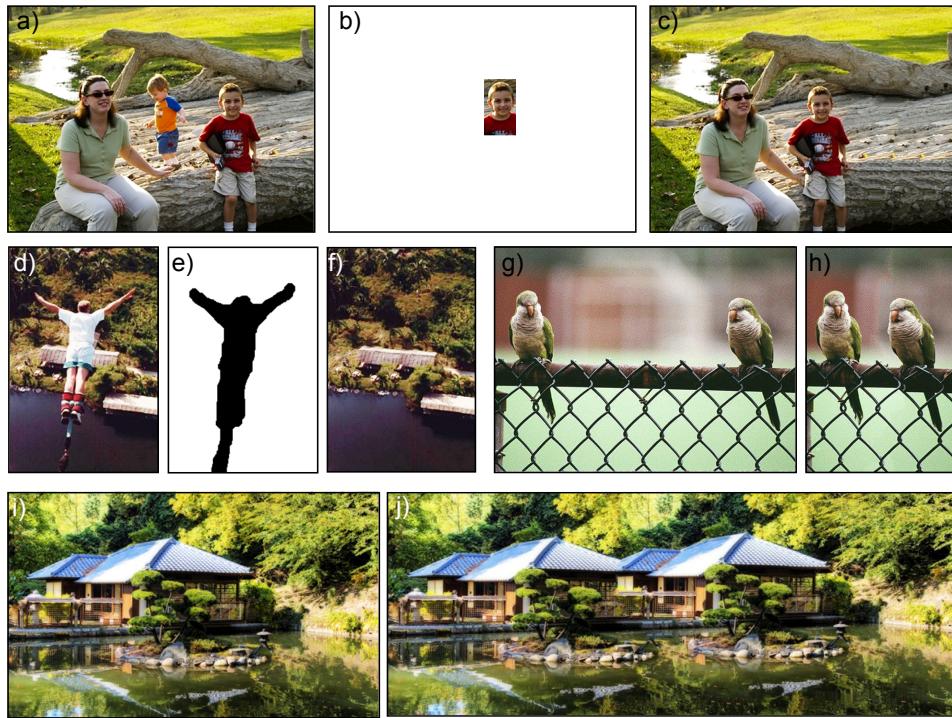
The remainder of the model depends on the application (figure 12.29):



**Figure 12.28** Shift maps for image re-targeting to reduce width. a) New image  $I^{(2)}$  is created from b) the original image  $I^{(1)}$  by copying piecewise regions (five regions shown). c) These regions are carefully chosen to produce a seamless result. d) The underlying representation is a shiftmap – a label at each pixel of the new image that specifies the 2D offset to the position in the original image that will be copied from. An MRF encourages the labels to be piecewise constant, and hence the result tends to consist of large chunks copied verbatim. Figure shows method of Pritch *et al.* (2009).

- To *move* an object, we specify unary costs in the new region that ensure that we copy the desired object here. The remainder of the shifts are left free to vary, but favor small offsets so that parts of the scene that are far from the change tend to be unperturbed.
- To *replace* an area of the image, we specify unary costs so that the remainder of the image must have a shift of zero (verbatim copying), and the shift in the missing region must be such that it copies from outside the region.
- To *retarget* an image to larger width, we set the unary costs so that the left and right edges of the new image are forced to have shifts that correspond to the left and right of the original image. We also use the unary costs to specify that vertical shifts must be small.
- To *retarget* an image to a smaller width (figure 12.28), we additionally specify that the horizontal offset can only increase as we move from left to right. This ensures that the new image does not contain replicated objects and that their horizontal order remains constant.

In each case the best solution can be found using the alpha-expansion algorithm. Since the pairwise terms do not form a metric here, it is necessary to truncate the relevant costs (see section 12.4.1). In practice, there are many labels and so Pritch *et al.* (2009) introduce a *coarse-to-fine* scheme in which a low resolution version of the image is initially synthesized and the result of this is used to guide further refinements at higher resolutions.



**Figure 12.29** Applications of shift maps. Shift maps can be used to a) take an object from the original image b) move it to a new position and c) then fill in the remaining pixels to produce a new picture. d) They can also be used to remove an undesirable object e) specified by a mask from an image by f) filling in the missing area. g-h) Finally they can be used to retarget an original image to a smaller size, or i-j) to re-target an original image to a larger size. Results from method of Pritch *et al.* (2009).

### 12.8.5 Super-resolution

Image *super-resolution* can also be framed as inference within a Markov random field model. Here, the basic unit of currency is an image patch rather than a pixel. For example, consider dividing the original image into a regular grid of  $N$  low resolution  $3 \times 3$  patches  $\{\mathbf{x}_n\}_{n=1}^N$ . The goal is to infer a set of corresponding labels  $\{w_n\}_{n=1}^N$  at each position in the grid. Each label can take one of  $K$  values, each of which corresponds to a different possible high resolution  $7 \times 7$  patch. These patches are extracted from training images.

The pairwise cost for placing high-resolution patches together is determined by the agreement at the abutting edge. The unary cost for choosing a patch at a given position depends on the agreement between the proposed high-resolution patch and the observed low-resolution patch. This can be computed by downsampling the high-resolution patch to  $3 \times 3$  pixels, and then using a normal noise model.



**Figure 12.30** Super resolution. a) The observed image, which is broken down into a regular grid of low resolution patches. b) We infer a regular grid of labels, each of which corresponds to a high resolution patch, and ‘quilt’ these together to form the super-resolved image. c) Ground truth. Adapted from Freeman *et al.* (2000). ©2000 Springer.

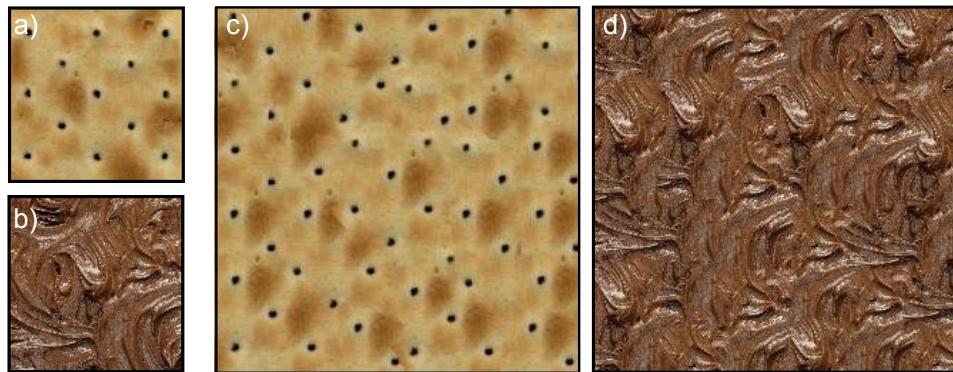
In principle, we could perform inference in this model with a graph cut formulation, but there are two problems. First, the resulting cost function is not submodular. Second, the number of possible high-resolution patches must be very large, and so the alpha-expansion algorithm (which chooses these in turn) would be extremely inefficient.

Freeman *et al.* (2000) used loopy belief propagation to perform approximate inference in a model similar to this. To make this relatively fast, they used only a subset of  $J \ll K$  possible patches at each position, where these were chosen so that they were the  $J$  patches which agreed best with the observed data (and so had the lowest unary costs). Although the results (figure 12.30) are quite convincing, they are sadly far from the imagined feats of TV crime drama.

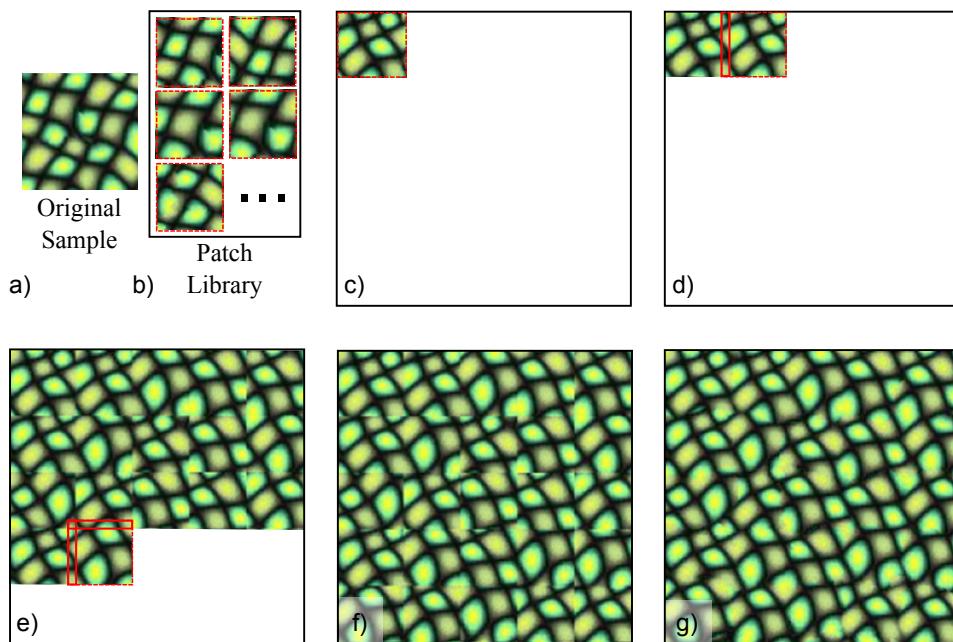
## 12.8.6 Texture synthesis

The applications so far have all been based on performing inference in the undirected Markov random field model. We now consider the directed model. Inference is difficult in this model due to the presence of ‘three-wise’ terms in the associated cost function (see section 12.7). However, generation from this model is relatively easy; since this is a directed model, we can use an ancestral sampling technique to generate examples. One possible application of this technique is for *texture synthesis*.

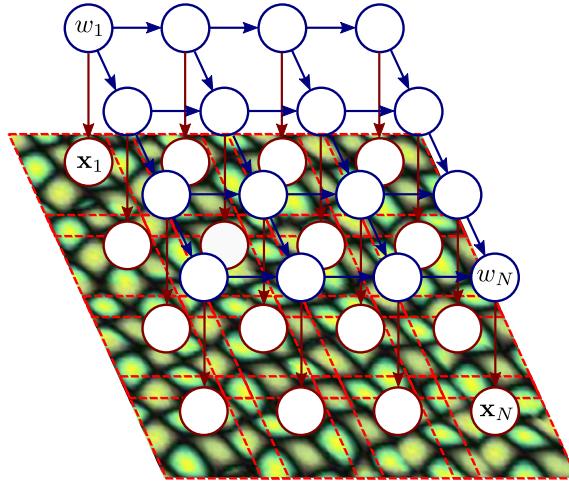
The goal of texture synthesis is to learn a generative model from a small patch of texture such that when we draw samples from the model, they look like extended examples of the same texture (figure 12.31). The particular technique that we describe here is known as *image quilting* and was originally described by Efros & Freeman (2001). We will first describe the algorithm as it was initially conceived, and then relate it to the directed model for grids.



**Figure 12.31** Texture synthesis. a,b ) Original texture samples. c,d) Synthesized textures using *image quilting*. Adapted from Efros & Freeman (2001).



**Figure 12.32** Image quilting. a) Original texture sample. b) Library of all overlapping patches from the original texture sample. c) The first patch is chosen randomly from the library. d) The second patch is chosen randomly from the  $k$  library patches that are most similar in the overlapping region. e) In subsequent rows, patches are chosen so that the overlapping region agrees with the previously placed patches to the left and above. f) This continues until we reach the bottom-right of the image. g) The patches are then blended together to give the final results.

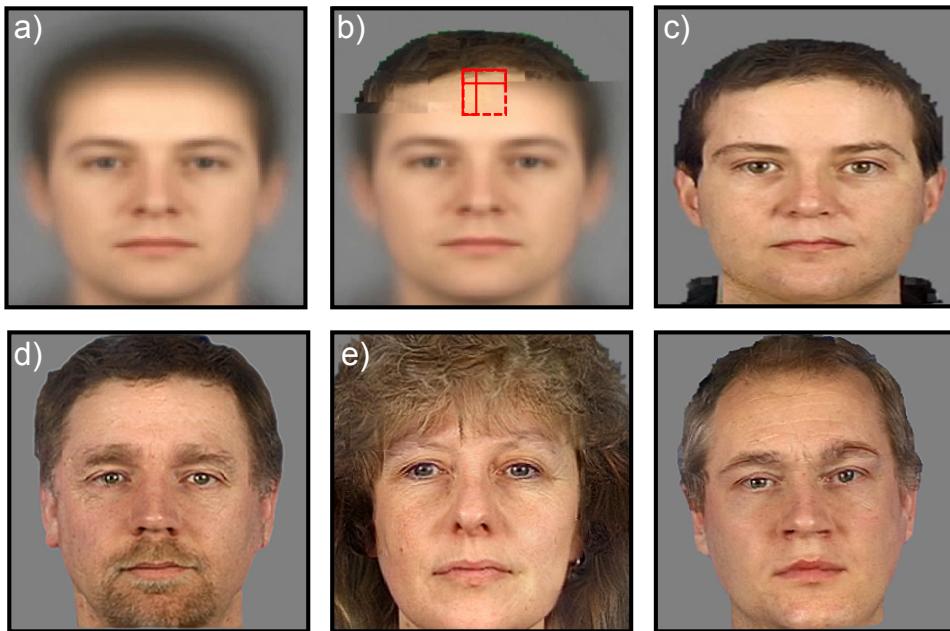


**Figure 12.33** Image quilting as ancestral sampling from a graphical model. When we synthesize images, we are effectively ancestral sampling from a directed grid model where each hidden node represents a patch index and each observed variable represents the patch data.

The first step (see figure 12.32) is to extract all possible patches of a given size from the input texture to form a *patch library*. The synthesized image will consist of a regular grid of these library patches such that each overlaps its neighbors by a few pixels. A new texture is synthesized starting in the top-left of this grid and proceeding to the bottom-right. At each position, a library patch is chosen such that it is visually consistent with the patches that have previously been placed above and to the left.

For the top-left position, we randomly choose a patch from the library. We then consider placing a second patch to the right of the first patch, such that they overlap by roughly 1/6 of their width. We search through the library for the  $J$  patches where the squared RGB intensity difference in the overlapping region is smallest. We choose one of these  $J$  patches randomly and place it into the image at the second position. We continue in this way, synthesizing the top row of patches in the image. When we reach the second row, we must consider the overlap with the patches to the left and above in deciding whether a candidate library patch is suitable: we choose the  $J$  patches where the total RGB difference between the overlapping portions of the candidate patch and the previously chosen patches is minimal. This process continues until we reach the bottom-right of the image.

In this way, we synthesize a new example of the texture (figure 12.32a-f). By forcing the overlapping regions to be similar, we enforce visual consistency between adjacent patches. By choosing randomly from the  $J$  best patches, we ensure that the result is stochastic: if we always chose the most visually consistent patch, we would replicate the original texture verbatim. At the end of this process, it is common to blend the resulting patches together to remove remaining artifacts in



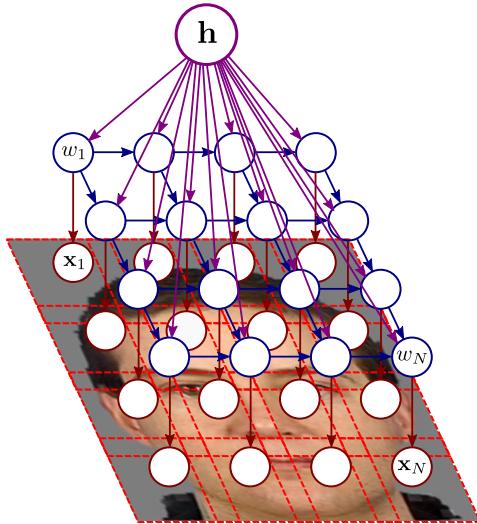
**Figure 12.34** Synthesizing novel faces. a) A sample is drawn from a subspace model (see chapter 7) that has been trained on facial images. b) Texture synthesis now proceeds but with two differences from before. First, the choice of patch must now agree with the sample from the subspace model as well as the previously placed patches. Second, the library patches are now different at each position: in this way we ensure that a nose patch is always chosen in the center and so on. c) After completing the synthesis and blending together the patches. d-f) Three more examples of synthesized faces. Adapted from Mohammed *et al.* (2009). ©2009 ACM.

the overlapping region (figure 12.32g).

Image quilting can be thought of as ancestral sampling from the directed model for images (figure 12.33). The observed data  $\{\mathbf{x}_n\}_{n=1}^N$  are the output patches, and the hidden labels  $\{w_n\}_{n=1}^N$  represent the patch index. The labels are conditioned on their parents with a probability distribution that allows a constant probability if the overlapping region is one of the  $J$  closest and zero otherwise. The only real change is that the relationship between label and observed data is now deterministic: a given label always produces exactly the same output patch.

### 12.8.7 Synthesizing novel faces

Mohammed *et al.* (2009) presented a related technique to synthesize more complex objects, such as frontal faces (figure 12.35), based on a large database of weakly aligned training examples. Faces have a distinct spatial structure, and we must



**Figure 12.35** Graphical model for synthesizing novel faces. When we generate a new image we are ancestral sampling from a directed image model, where each label  $w$  is conditioned on the hidden variable  $\mathbf{h}$  of the subspace model. Adapted from Mohammed *et al.* (2009). ©2009 ACM.

ensure that our model enforces these constraints. To this end, we build a separate library of patches for each position in the image. This ensures that the features have roughly the correct spatial relations: the nose always appears in the center and the chin at the bottom.

In principle, we could now apply a standard image quilting approach by synthesizing patches starting in the top-left, and moving to the bottom-right. Unfortunately, the resulting faces can drift in appearance (e.g., from male to female) as we move through the image. To prevent this from happening, we condition the patch synthesis on a draw from a factor analysis model (section 7.6) which has been trained with frontal faces. A sample from this model looks like a blurry, but globally coherent, face. Now when we choose potential patches, they must agree with both the previously placed patches to the left and above, but also be similar to the appropriate part of the blurry sample from the subspace model. The generated images from this model look like highly realistic human faces.

In terms of probability, the labels  $\{w_n\}_{n=1}^N$  in this model are conditioned not only on their ancestors  $w_{pa}$ , but also on the hidden variable in the subspace model  $\mathbf{h}$ . This connects to every patch label  $\{w_n\}_{n=1}^N$  and gives the resulting image a greater visual coherence than the Markov connections of the patches alone.

## Discussion

Models for grids are ubiquitous in vision: they occur in almost all applications that attempt to associate a label with each position in the image. Depending on the application, this label may indicate the depth, object type, segmentation mask, or motion at that pixel. Unfortunately, most problems of this type are NP hard and so we must resort to efficient approximate inference techniques such as the alpha-expansion algorithm.

## Notes

**MRFs and CRFs:** Markov random fields were first investigated in computer vision by Geman & Geman (1984), although much of the early work dealt with continuous variables rather than the discrete case as discussed in this chapter. A good review can be found in Li (2010). Conditional random fields were first used in computer vision by Kumar & Hebert (2003). An overview can be found in Sutton & McCallum (2011).

**Applications:** Grid-based models and graph cuts are used extensively in vision and graphics. A partial list of applications includes stereo vision (Kolmogorov & Zabih 2001; Woodford *et al.* 2009), optical flow (Kolmogorov & Zabih 2001), texture synthesis (Kwatra *et al.* 2003), photo-montage (Agarwala *et al.* 2004), summarizing photo-collections with collages (Rother *et al.* 2005; Rother *et al.* 2006), bi-layer segmentation (Kolmogorov *et al.* 2006), interactive segmentation (Rother *et al.* 2004; Boykov *et al.* 2001), super-resolution (Freeman *et al.* 2000), image re-targeting (Pritch *et al.* 2009), denoising (Greig *et al.* 1989), over-segmentation (Moore *et al.* 2010; Veksler *et al.* 2010), image colorization (Levin *et al.* 2004), semantic segmentation (Shotton *et al.* 2009), multi-view reconstruction (Kolmogorov & Zabih 2002; Vogiatzis *et al.* 2007), and matching image points (Isack & Boykov 2012).

**Graph cuts:** The first application of graph cuts to inference in an MRF is due to Greig *et al.* (1989) who investigated binary denoising. However, it was not until the work of Boykov *et al.* (2001) that this result was rediscovered and graph cuts became widely used. Ishikawa (2003) presented the exact solution for multi-label graph cuts with convex potentials, and this was generalized by Schlesinger & Flach (2006). The presentation in this chapter is a hybrid of these two methods. Boykov *et al.* (2001) introduced the idea of optimizing non-convex multi-label energies via a series of binary problems. They proposed two algorithms of this kind: the alpha-beta swap in which pairs of labels are exchanged for one another and the alpha-expansion algorithm. They also proved that the alpha-expansion solution is guaranteed to be within a factor of two of the true solution. In the same spirit, Lempitsky *et al.* (2010) and Kumar *et al.* (2011) have proposed more complex ‘moves’. Tarlow *et al.* (2011) elucidates the connection between graph cut methods and max-product belief propagation. For more detailed overviews of graph cut methods, consult Boykov & Veksler (2006), Felzenszwalb & Zabih (2011) and Blake *et al.* (2011).

**Max-flow:** Graph-cut methods rely on algorithms for computing maximum flow. The most common of these are the augmenting paths method of Ford & Fulkerson (1962) and the push-relabel method of Goldberg & Tarjan (1988). Details of these and other approaches to the same problem can be found in any standard textbook on algorithms such as Cormen *et al.* (2001). The most common technique in computer vision is a modified version of the augmented paths algorithm due to Boykov & Kolmogorov (2004) that has been demonstrated to have very good performance for vision problems. Kohli & Torr (2005), Juan & Boykov (2006) and Alahari *et al.* (2008) have all investigated methods for improving the efficiency of graph cuts by reusing solutions to similar graph cut problems (e.g., based on the solution to the previous frames in a time-sequence).

**Cost functions and optimization:** Kolmogorov & Zabih (2004) provide a summary of the cost functions that can be optimized using the basic graph cuts max flow formulation with binary variables. Kolmogorov & Rother (2007) summarize graph cut approaches to non-submodular energies. Rother *et al.* (2007) and Komodakis *et al.* (2008) present algorithms that can approximately optimize more general cost functions.

**Constraint edges:** Recent work has investigated bespoke graph constructions that make heavy use of constraint edges (edges of infinite strength) to ensure that the solution

conforms to a certain structure. For example, Delong & Boykov (2009) devised a method that forced certain labels to surround others, and Moore *et al.* (2010) describe a method that forces the label field to conform to a lattice. See also Felzenszwalb & Veksler (2010) for a related scheme based on dynamic programming.

**Higher-order cliques:** All of the methods discussed in this chapter assume pairwise connections; the cliques include only two discrete variables. However, to model more complex statistics of the label field, it is necessary to include more than two variables in the cliques, and these are known as *higher order* models. Roth & Black (2009) demonstrated good denoising and inpainting results with a continuous MRF model of this kind, and Domke *et al.* (2008) demonstrated the efficacy of a directed model in which each variable was conditioned on a number of variables above and to the right in the image. There has recently been considerable interest in developing algorithms for MAP estimation in models with discrete variables and higher-order cliques (Ishikawa 2009; Kohli *et al.* 2009a; Kohli *et al.* 2009b; Rother *et al.* 2009).

**Other approaches to MAP estimation:** There are many other contemporary approaches to MAP estimation in MRFs and CRFs. These include loopy belief propagation (Weiss & Freeman 2001), quadratic pseudo-boolean optimization which is used in non-submodular cost functions (Kolmogorov & Rother 2007), random walks (Grady 2006), and linear programming (LP) relaxations (Weiss *et al.* 2011) and various approaches to maximize the LP lower bound such as tree reweighted message passing (Wainright *et al.* 2005; Kolmogorov 2006). An experimental comparison between different energy minimization methods for MRFs can be found in Szeliski *et al.* (2008).

**Texture synthesis:** Texture synthesis was originally investigated as a continuous problem and the focus was on modeling the joint statistics of the RGB values in a small patch (Heeger & Bergen 1995; Portilla & Simoncelli 2000). Although texture synthesis as a continuous problem is still an active research area (e.g., Heess *et al.* 2009), these early methods were displaced by methods that represented the texture in terms of discrete variables (either by quantizing the RGB values, indexing patches or using a shift-map representation). The resulting algorithms (e.g., Efros & Leung 1999; Wei & Levoy 2000; Efros & Freeman 2001; Kwatra *et al.* 2003) were originally described as heuristic approaches to generating textures, but can also be interpreted as exact or approximate ways to draw samples from directed or undirected grid models.

**Interactive segmentation:** The use of graph cuts for interactive segmentation algorithms was pioneered by Boykov & Jolly (2001). In early works (Boykov & Jolly 2001; Boykov & Funka Lea 2006; Li *et al.* 2004) the user interacted with the image by placing marks indicating foreground and background regions. Grab-cut (Rother *et al.* 2004) allowed the user to draw a box around the object in question. More recent systems (Liu *et al.* 2009) are fast enough to allow the user to interactively ‘paint’ the selection onto the images. Current interest in graph cut based segmentation is mainly focused on developing novel priors over the shape that improve performance (e.g., Malcolm *et al.* 2007; Veksler 2008; Chittajallu *et al.* 2010; Freiman *et al.* 2010). To this end, Kumar *et al.* (2005) introduced a method for imposing high-level knowledge about the articulation of the object, Vicente *et al.* (2008) developed an algorithm that is suited for cutting out elongated objects, and Lempitsky *et al.* (2008) used a prior based on a bounding box around the object.

**Stereo vision:** Most state-of-the-art stereo vision algorithms rely on MRFs or CRFs and are solved using either graph cuts (e.g., Kolmogorov & Zabih 2001) or belief propagation (e.g., Sun *et al.* 2003). Comparisons of these approaches can be found in Tappen & Freeman (2003) and Szeliski *et al.* (2008). An active area of research in dense stereo vision is the formulation of the compatibility of the two images given a certain disparity

offset (e.g., Bleyer & Chambon 2010; Hirschmüller & Scharstein 2009) which is rarely based on single pixels in practice (see Yoon & Kweon 2006; Tombari *et al.* 2008).

For more information about stereo vision see the reviews by Scharstein & Szeliski (2002) and Brown *et al.* (2003) or consult Szeliski (2010), which contains a good modern summary. Chapter 11 of this book summarizes dynamic programming approaches. Notable stereo implementations include the region growing approach of Lhuillier & Quan (2002); the systems of Zitnick & Kanade (2000) and Hirschmüller (2005), both of which are available online; and the extremely efficient GPU based system of Sizintsev & Wildes (2010). For an up-to-date quantitative comparison of the latest stereo vision algorithms consult the Middlebury stereo vision website (<http://vision.middlebury.edu/stereo/>).

## Problems

**Problem 12.1** Consider a Markov random field with the structure

$$Pr(x_1, x_2, x_3, x_4) = \frac{1}{Z} \phi[x_1, x_2] \phi[x_2, x_3] \phi[x_3, x_4] \phi[x_4, x_1]$$

but where the variables  $x_1, x_2, x_3$ , and  $x_4$  are continuous and the potentials are defined as

$$\phi[a, b] = \exp [-(a - b)^2].$$

This is known as a *Gaussian Markov random field*. Show that the joint probability is a normal distribution, and find the information matrix (inverse covariance matrix).

**Problem 12.2** Compute the MAP solution to the three-pixel graph cut problem in figure 12.36 by (i) computing the cost of all eight possible solutions explicitly and finding the one with the minimum cost (ii) running the augmenting paths algorithm on this graph by hand and interpreting the minimum cut.

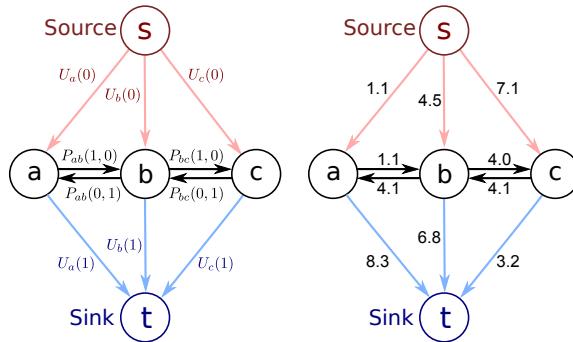
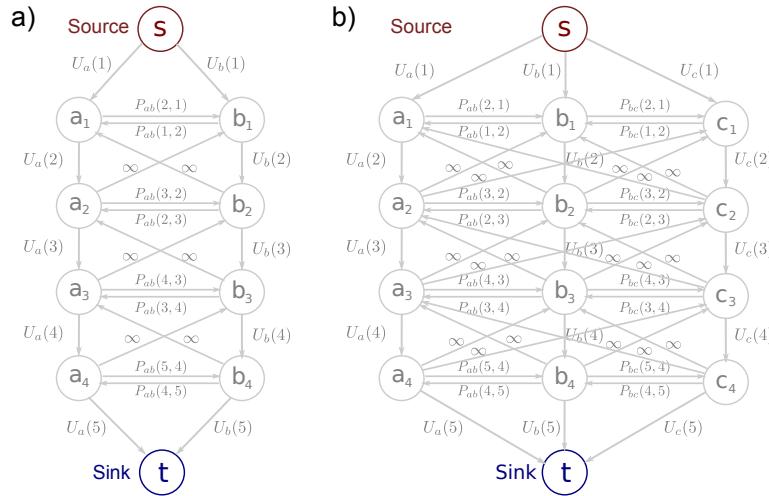


Figure 12.36 Graph for problem 12.2.

**Problem 12.3** Explicitly compute the costs associated with the four possible minimum cuts of the graph in figure 12.10.

**Problem 12.4** Compute the cost for each the four possible cuts of the graph in figure 12.11c.

**Problem 12.5** Consider the graph construction in figure 12.37a, which contains a number of constraint edges of infinite cost (capacity). There are 25 possible minimum cuts on this graph, each of which corresponds to one possible labeling of the two pixels. Write out the cost for each labeling. Which solutions have finite cost for this graph construction?



**Figure 12.37** Alternative multi-label graph constructions. Each of these graphs has extra *constraint links* with infinite weight. These have the effect of giving an infinite cost to a subset of the possible solutions.

**Problem 12.6** Which of the possible minimum cuts of the graph in figure 12.37b have a finite cost?

**Problem 12.7** Confirm that the costs of the cuts in figure 12.14 are as claimed by explicitly performing the summation over the relevant terms  $C_{ij}$ .

**Problem 12.8** Show that the Potts model (figure 12.17c) is not submodular by providing a counter-example to the required criterion:

$$P_{ab}(\beta, \gamma) + P_{ab}(\alpha, \delta) - P_{ab}(\beta, \delta) - P_{ab}(\alpha, \gamma) \geq 0.$$

**Problem 12.9** An alternative to the alpha-expansion algorithm is the alpha-beta swap. Here, a multi-label MRF with non-convex potentials is optimized by repeatedly choosing pairs of labels  $\alpha, \beta$  and performing a binary graph cut that allows them to swap in such a way that the overall cost function decreases. Devise a graph structure that can be used to perform this operation. Hint: consider separate cases for neighboring labels  $(\alpha, \alpha)$ ,  $(\beta, \beta)$ ,  $(\beta, \gamma)$ ,  $(\alpha, \gamma)$  and  $(\gamma, \gamma)$  where  $\gamma$  is a label that is neither  $\alpha$  nor  $\beta$ .



# **Part IV**

# **Preprocessing**



# Part IV: Preprocessing

The main focus of this book is on statistical models for computer vision; the previous chapters concern models that relate visual measurements  $\mathbf{x}$  to the world  $\mathbf{w}$ . However, there has been little discussion of how the measurement vector  $\mathbf{x}$  was created, and it has often been implied that it contains concatenated RGB pixel values. In state-of-the-art vision systems, the image pixel data are almost always *preprocessed* to form the measurement vector.

We define preprocessing to be any transformation of the pixel data *prior to* building the model that relates the data to the world. Such transformations are often ad-hoc heuristics: their parameters are not learned from training data, but they are chosen based on experience of what works well. The philosophy behind image preprocessing is easy to understand; the image data *may* be contingent on many aspects of the real world that do not pertain to the task at hand. For example, in an object detection task, the RGB values will change depending on the camera gain, illumination, object pose and particular instance of the object. The goal of image preprocessing is to remove as much of this unwanted variation as possible, while retaining the aspects of the image that are critical to the final decision.

In a sense, the need for preprocessing represents a failure; we are admitting that we cannot directly model the relationship between the RGB values and the world state. Inevitably, we must pay a price for this. Although the variation due to extraneous factors is jettisoned, it is very probable that some of the task-related information is also discarded. Fortunately, in these nascent years of computer vision, this rarely seems to be the limiting factor that governs the overall performance.

We devote the single chapter in this section to discussing a variety of preprocessing techniques. Although the treatment here is not extensive, it should be emphasized that preprocessing is very important; in practice, the choice of preprocessing technique can influence the performance of vision systems at least as much as the choice of model.



# Chapter 13

## Image preprocessing and feature extraction

This chapter provides a brief overview of modern preprocessing methods for computer vision. In section 13.1, we introduce methods in which we replace each pixel in the image with a new value. Section 13.2 considers the problem of finding and characterizing edges, corners and interest points in images. In section 13.3 we discuss visual descriptors; these are low dimensional vectors that attempt to characterize the interesting aspects of an image region in a compact way. Finally, in section 13.4, we discuss methods for dimensionality reduction.

### 13.1 Per-pixel transformations

We start our discussion of preprocessing with *per-pixel operations*: these methods return a single value corresponding to each pixel of the input image. We denote the original 2D array of pixel data as  $\mathbf{P}$ , where  $p_{ij}$  is the element at the  $i^{th}$  of  $I$  rows and the  $j^{th}$  of  $J$  columns. The element  $p_{ij}$  is a scalar representing the grayscale intensity. Per-pixel operations return a new 2D array  $\mathbf{X}$  of the same size as  $\mathbf{P}$ , containing elements  $x_{ij}$ .

#### 13.1.1 Whitening

The goal of whitening (figure 13.1) is to provide invariance to fluctuations in the mean intensity level and contrast of the image. Such variation may arise because of a change in ambient lighting intensity, the object reflectance, or the camera gain. To compensate for these factors, the image is transformed so that the resulting pixel values have zero mean and unit variance. To this end, we compute the mean  $\mu$  and variance  $\sigma^2$  of the original grayscale image  $\mathbf{P}$ :



**Figure 13.1** Whitening and histogram equalization. a) A number of faces which have been captured with widely varying contrasts and mean levels b) After whitening the images have the same mean and variance. c) After histogram equalization the distribution of gray values is approximately uniform. Both of these transformations reduce the amount of variation due to contrast and intensity changes.

$$\begin{aligned}\mu &= \frac{\sum_{i=1}^I \sum_{j=1}^J p_{ij}}{IJ} \\ \sigma^2 &= \frac{\sum_{i=1}^I \sum_{j=1}^J (p_{ij} - \mu)^2}{IJ}.\end{aligned}\quad (13.1)$$

These statistics are used to transform each pixel value separately so that,

$$x_{ij} = \frac{p_{ij} - \mu}{\sigma}. \quad (13.2)$$

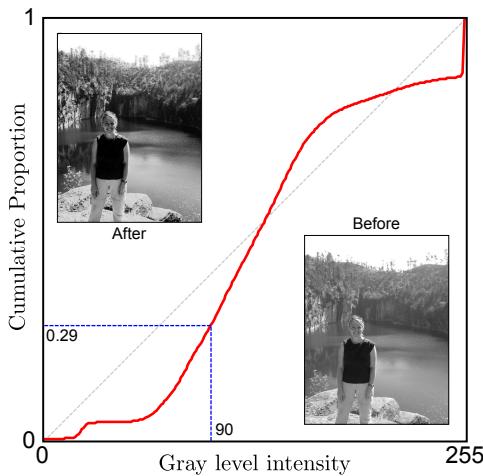
For color images, this operation may be carried out by computing the statistics  $\mu$  and  $\sigma^2$  from all three channels or by separately transforming each of the RGB channels based on their own statistics.

Note that even this simple transform has the potential to hamper subsequent inference about the scene: depending on the task, the absolute intensities may or may not contain critical information. Even the simplest preprocessing methods must be applied with care.

### 13.1.2 Histogram equalization

The goal of histogram equalization (figure 13.1c) is to modify the statistics of the intensity values so that *all* of their moments take predefined values. To this end, a nonlinear transformation is applied that forces the distribution of pixel intensities to be flat.

We first compute the histogram of the original intensities  $\mathbf{h}$  where the  $k^{th}$  of  $K$  entries is given by



**Figure 13.2** Histogram equalization. The abscissa indicates the pixel intensity. The ordinate indicates the proportion of intensities that were less than or equal to this value. This plot can be used as a look up table for histogram equalizing the intensities. For a given intensity value on the abscissa, we choose the new intensity to be the maximum output intensity  $K$  times the value on the ordinate. After this transformation, the intensities are equally distributed. In the example image, many of the pixels are bright. Histogram equalization spreads these bright values out over a larger intensity range, and so has the effect of increasing the contrast in the brighter regions.

$$h_k = \sum_{i=1}^I \sum_{j=1}^J \delta[p_{ij} - k], \quad (13.3)$$

where the operation  $\delta[\bullet]$  returns one if the argument is zero and zero otherwise. We then cumulatively sum this histogram and normalize by the total number of pixels to compute the cumulative proportion  $c$  of pixels that are less than or equal to each intensity level:

$$c_k = \frac{\sum_{l=1}^k h_l}{IJ}. \quad (13.4)$$

Finally, we use the cumulative histogram as a look up table to compute the transformed value so that

$$x_{ij} = K c_{p_{ij}}. \quad (13.5)$$

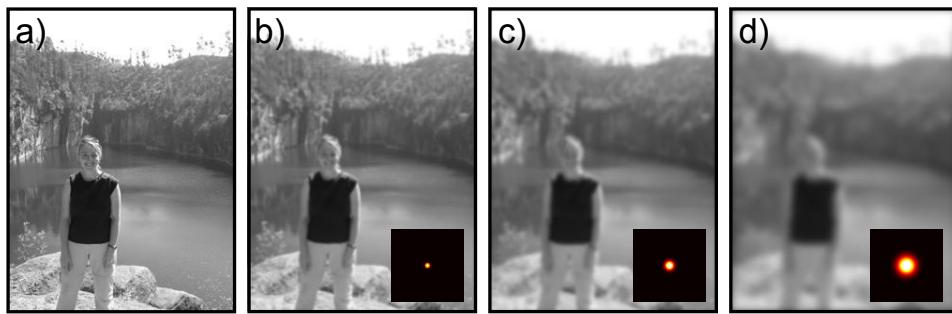
For example, in figure 13.2 the value 90 will be mapped to  $K \times 0.29$  where  $K$  is the maximum intensity (usually 255). The result is a continuous number rather than a discretized pixel intensity, but is in the same range as the original data. The result can be rounded to the nearest integer if subsequent processing demands.

Problem 13.1

### 13.1.3 Linear filtering

After filtering an image, the new pixel value  $x_{ij}$  consists of a weighted sum of the intensities of pixels in the surrounding area of the original image  $\mathbf{P}$ . The weights are stored in a filter kernel  $\mathbf{F}$ , which has entries  $f_{m,n}$ , where  $m \in \{-M \dots M\}$  and  $n \in \{-N \dots N\}$ .

More formally, when we apply a filter, we *convolve* the  $\mathbf{P}$  with the filter  $\mathbf{F}$ , where two-dimensional convolution is defined as



**Figure 13.3** Image blurring. a) Original image. b) Result of convolving with a Gaussian filter (filter shown in bottom right of image). Each pixel in this image is a weighted sum of the surrounding pixels in the original image, where the weights are given by the filter. The result is that the image is slightly blurred. c-e) Convolving with a filter of increasing standard deviation causes the resulting image to be increasingly blurred.

$$x_{ij} = \sum_{m=-M}^M \sum_{n=-N}^N p_{i-m,j-n} f_{m,n}. \quad (13.6)$$

Problem 13.2

Notice that by convention, the filter is flipped in both directions so the top left of the filter  $f_{-M,-N}$  weights the pixel  $p_{i+M,j+N}$  to the right and below the current point in  $\mathbf{P}$ . Many filters used in vision are symmetric in such a way that this flipping makes no practical difference.

Without further modification, this formulation will run into problems near the borders of the image: it needs to access points that are outside the image. One way to deal with this is to use *zero padding* in which it is assumed that the value of  $P$  is 0 outside the defined image region.

We now consider a number of common types of filter.

### Gaussian (blurring) filter

To blur an image, we convolve it with a 2D Gaussian,

$$f(m, n) = \frac{1}{2\pi\sigma^2} \exp \left[ -\frac{m^2 + n^2}{2\sigma^2} \right]. \quad (13.7)$$

Problem 13.4

Each pixel in the resulting image is a weighted sum of the surrounding pixels, where the weights depend on the Gaussian profile: nearer pixels contribute relatively more to the final output. This process blurs the image, where the degree of blurring is dependent on the standard deviation  $\sigma$  of the Gaussian filter (figure 13.3). This is a simple method to reduce noise in images taken at very low light levels.

### First derivative filters and edge filters

A second use for image filtering is to locate places in the image where the intensity changes abruptly. Consider taking the first derivative of the image along the rows. We could approximate this operation by simply computing the difference between two offset pixels along the row. This operation can be accomplished by filtering with the operator  $\mathbf{F} = [-1 \ 0 \ 1]$ . This filter gives zero response when the image is flat in the horizontal direction: it is hence invariant to constant additive luminance changes. It gives a negative response when the image pixel values are increasing as we move in the horizontal direction and a positive response when they are decreasing (recall that convolution flips the filter by  $180^\circ$ ). As such, it is selective for edges in the image.

Problem 13.5  
Problem 13.6

The response to the filter  $\mathbf{F} = [-1 \ 0 \ 1]$  is noisy because of its limited spatial extent. Consequently, slightly more sophisticated filters are used to find edges in practice. Examples include the Prewitt operators (figures 13.4a-b)

$$\mathbf{F}_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}, \quad \mathbf{F}_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, \quad (13.8)$$

and the Sobel operators

$$\mathbf{F}_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \quad \mathbf{F}_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \quad (13.9)$$

where, in each case, the filter  $\mathbf{F}_x$  is a filter selective for edges in the horizontal direction and  $\mathbf{F}_y$  is a filter selective for edges in the vertical direction.

### Laplacian filters

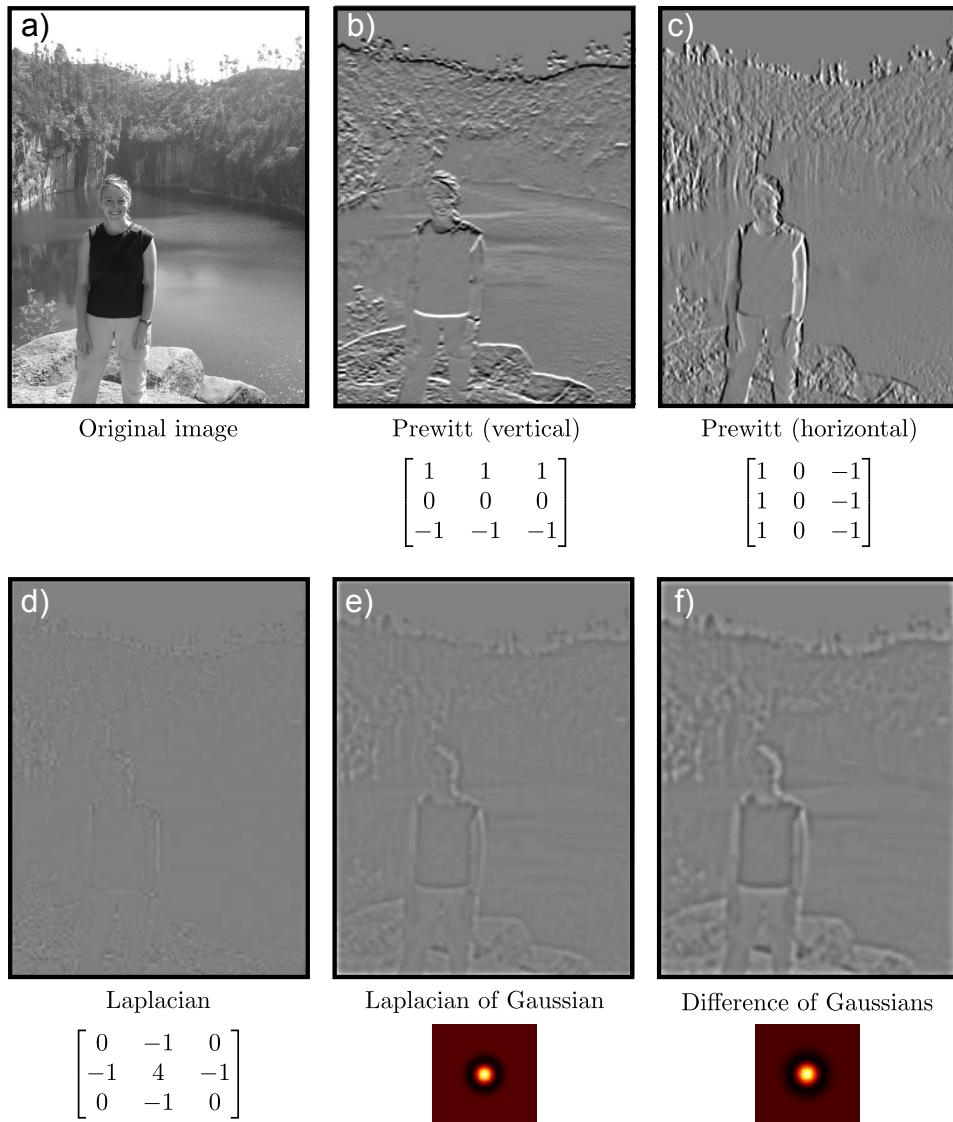
The Laplacian filter is the discrete two dimensional approximation to the Laplacian operator  $\nabla^2$ , and is given by

$$\mathbf{F} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}. \quad (13.10)$$

Applying the discretized filter  $\mathbf{F}$  to an image results in a response of high magnitude where the image is changing, regardless of the direction of that change (figure 13.4c): the response is zero in regions that are flat and significant where edges occur in the image. It is hence invariant to constant additive changes in luminance, and useful for identifying interesting regions of the image.

### Laplacian of Gaussian filters

In practice, the Laplacian operator produces noisy results. A superior approach is to first smooth the image with a Gaussian filter and then apply the Laplacian. Due to the associative property of convolution, we can equivalently convolve the Laplacian filter by a Gaussian and apply the resulting *Laplacian of Gaussian* filter to the image (figure 13.4d). This Laplacian of Gaussian has the advantage that it



**Figure 13.4** Image filtering with first and second derivative operators. The original image is shown in figure 13.3a. a) Convolving with the vertical Prewitt filter produces a response that is proportional to the size and polarity of edges in the vertical direction. b) The horizontal Prewitt filter produces a response to edges in the horizontal direction. c) The Laplacian filter gives a significant response where the image changes rapidly regardless of direction. d) The Laplacian of Gaussian filter produces similar results but the output is smoothed and hence less noisy. e) The difference of Gaussians filter is a common approximation to the Laplacian of Gaussian.

can be tuned to be selective for changes at different scales, depending on the scale of the Gaussian component.

### Difference of Gaussians

The Laplacian of Gaussian filter is very well approximated by the difference of Gaussians filter (compare figures 13.4d and 13.4e). As the name implies, this filter is created by taking the difference of two Gaussians at nearby scales. The same result can be achieved by filtering the image with the two Gaussians separately and taking the difference between the results. Again, this filter responds strongly in regions of the image that are changing at a predetermined scale.

### Gabor filters

Gabor filters are selective for both scale and orientation. The 2D Gabor function is the product of a 2D Gaussian with a 2D sinusoid. It is parameterized by the covariance of the Gaussian and the phase  $\phi$ , orientation  $\omega$ , and wavelength  $\lambda$  of the sine wave. If the Gaussian component is spherical, it is defined by

$$f_{mn} = \frac{1}{2\pi\sigma^2} \exp\left[-\frac{m^2 + n^2}{2\sigma^2}\right] \sin\left[\frac{2\pi(\cos[\omega]m + \sin[\omega]n)}{\lambda} + \phi\right], \quad (13.11)$$

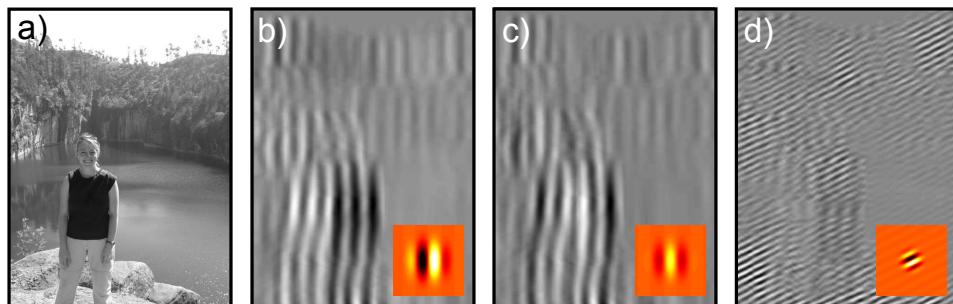
where  $\sigma$  controls the scale of the spherical Gaussian. It is typical to make the wavelength proportional to the scale  $\sigma$  of the Gaussian so a constant number of cycles is visible.

The Gabor filter is selective for elements within the image at a certain frequency and orientation band, and with a certain phase (figure 13.5). It is invariant to constant additive changes in luminance when the sinusoidal component is asymmetric. This is also nearly true for symmetric Gabor functions, as long as several cycles of the sinusoid are visible. A response that is independent of phase can easily be generated by squaring and summing the responses of two Gabor features with the same frequency, orientation, and scale, but with phases that are  $\pi/2$  radians apart. The resulting quantity is termed the *Gabor energy* and is somewhat invariant to small displacements of the image.

Filtering with Gabor functions is motivated by mammalian visual perception: this is one of the first processing operations applied to visual data in the brain. Moreover, it is known from psychological studies that certain tasks (e.g., face detection) are predominantly dependent on information at intermediate frequencies. This may be because high-frequency filters see only a small image region and are hence noisy and relatively uninformative, and low-frequency filters act over a large region and respond disproportionately to slow changes due to lighting.

### Haar-like filters

Haar-like filters consist of adjacent rectangular regions that are balanced so that the average filter value is zero, and they are invariant to constant luminance changes. Depending on the configuration of these regions, they may be similar to derivative or Gabor filters (figure 13.6).



**Figure 13.5** Filtering with Gabor functions. a) Original image. b) After filtering with horizontal asymmetric Gabor function at a large scale (filter shown bottom right). c) Result of filtering with horizontal symmetric Gabor function at a large scale. d) Filtering with vertical Gabor filter (responds to vertical changes). e) Response to diagonal Gabor function.

However, Haar-like filters are noisier than the filters they approximate: they have sharp edges between positive and negative regions, and so moving by a single pixel near an edge may change the response significantly. This drawback is compensated for by the relative speed with which Haar functions can be computed.

To compute Haar functions rapidly, we first form the *integral image* (figure 13.6g). This is an intermediate representation in which each pixel contains the sum of all of the intensity values above and to the left of the current position. So the value in the top-left corner is the original pixel value at that position and the value in the bottom right corner is the sum of all of the pixel values in the image. The values in the other parts of the integral image are between these extremes.

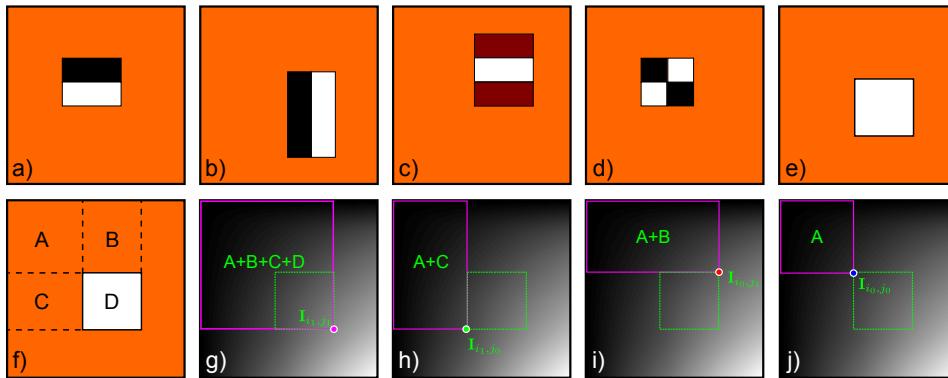
Given the integral image  $\mathbf{I}$ , it is possible to compute the sum of the intensities in any rectangular region with just four operations, regardless of how large this region is. Consider the region is defined by the range  $[i_0, i_1]$  down the columns and  $[j_0, j_1]$  along the rows. The sum  $S$  of the internal pixel intensities is

$$S = \mathbf{I}_{i_1, j_1} + \mathbf{I}_{i_0, j_0} - \mathbf{I}_{i_1, j_0} - \mathbf{I}_{i_0, j_1}. \quad (13.12)$$

The logic behind this calculation is illustrated in figure 13.6f-i.

#### Problem 13.3

Since Haar-like filters are composed of rectangular regions, they can be computed using a similar trick. For a filter with two adjacent rectangular regions, six operations are required. With three adjacent rectangular regions, eight operations are required. When the filter dimensions  $M$  and  $N$  are large, this approach compares very favorably to a naïve implementation of conventional filtering which requires  $O(MN)$  operations to compute the filter response due to a  $M \times N$  kernel. Haar-like filters are often used in real-time applications such as face detection because of the speed with which they can be computed.



**Figure 13.6** Haar-like filters. a-d) Haar-like filters consist of rectangular regions. Convolution with Haar-like filters can be done in constant time. e) To see why, consider the problem of filtering with this single rectangular region. f) We denote the sum of the pixel values in these four regions as  $A, B, C$  and  $D$ . Our goal is to compute  $D$ . g) The integral image has a value that is the sum of the intensities of the pixels above and to the left of the current position. The integral image at position  $(i_1, j_1)$  hence has value  $A + B + C + D$ . h) The integral image at  $(i_1, j_0)$  has value  $A + C$ . i) The integral image at  $(i_0, j_1)$  has value  $A + B$ . j) The integral image at  $(i_0, j_0)$  has value  $A$ . The sum of the pixels in region  $D$  can now be computed as  $\mathbf{I}_{i_1, j_1} + \mathbf{I}_{i_0, j_0} - \mathbf{I}_{i_1, j_0} - \mathbf{I}_{i_0, j_1} = (A + B + C + D) + A - (A + C) - (A + B) = D$ . This requires just four operations regardless of the size of the original square region.

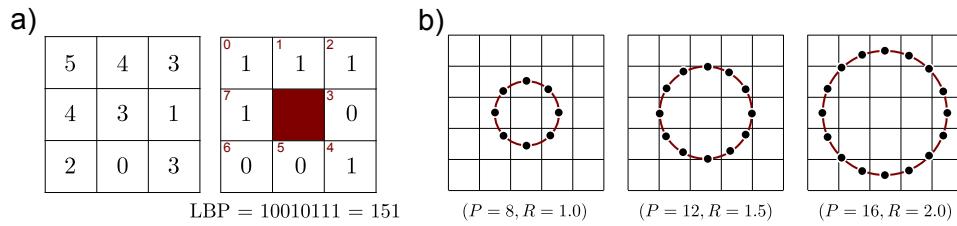
### 13.1.4 Local binary patterns

The local binary patterns (LBP) operator returns a discrete value at each pixel that characterizes the local texture in a way that is partially invariant to luminance changes. For this reason, features based on local binary patterns are commonly used as a substrate for face recognition algorithms.

The basic LBP operator compares the eight neighboring pixel intensities to the center pixel intensity, assigning a 0 or a 1 to each neighbor depending on whether they are less than or greater than the center value. These binary values are then concatenated in a predetermined order and converted to a single decimal number that represents the ‘type’ of local image structure (figure 13.7).

With further processing, the LBP operator can be made orientation invariant: the binary representation is repeatedly subjected to bit-wise shifts to create eight new binary values, and the minimum of these values is chosen. This reduces the number of possible LBP values to 36. In practice, it has been found that the distribution over these 36 LBP values is dominated by those that are relatively *uniform*. In other words, binary strings where transitions are absent (e.g., 00000000, 11111111) or infrequent (e.g., 00001111, 00111111) occur most frequently. The number of texture classes can be further reduced by aggregating all of the non-uniform LPBs into a single class. Now the local image structure is categorized into

Problem 13.7



**Figure 13.7** Local binary patterns. a) The local binary pattern (LBP) is computed by comparing the central pixel to each of its eight neighbors. The binary value associated with each position is set to one if that neighbor is greater than or equal to the central pixel. The eight binary values can be read out and combined to make a single 8-bit number. b) Local binary patterns can be computed over larger areas by comparing the current pixels to the (interpolated) image at positions on a circle. This type of LBP is characterized by the number of samples  $P$  and the radius of the circle  $R$ .

nine LBP types (eight rotationally invariant uniform patterns and one non uniform class).

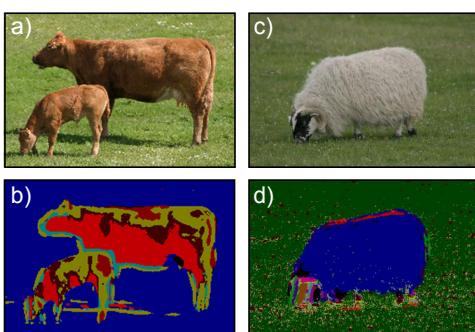
The LBP operator can be extended to use neighborhoods of different sizes: the central pixel is compared to positions in a circular pattern (figure 13.7b). In general, these positions do not exactly coincide with the pixel grid, and the intensity at these positions must be estimated using bilinear interpolation. This extended LBP operator can capture texture at different scales in the image.

### 13.1.5 Texton maps

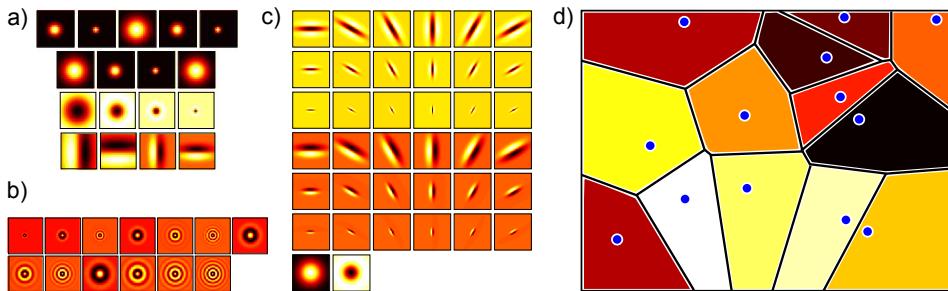
The term ‘texton’ stems from the study of human perception and refers to a primitive perceptual element of texture. In other words, it roughly occupies the role that a phoneme takes in speech recognition. In a machine vision context a texton is a discrete variable that designates which one of a finite number of possible texture classes is present in a region surrounding the current pixel. A texton map is an image in which the texton is computed at every pixel (figure 13.8).

Texton assignment depends on training data. A bank of  $N$  filters is convolved with a set of training images. The responses are concatenated to form one  $N \times 1$  vector for each pixel position in each training image. These vectors are then clustered into  $K$  classes using the K-means algorithm (section 13.4.4). Textons are computed for a new image by convolving it with the same filter bank. For each pixel, the texton is assigned by noting which cluster mean is closest to the  $N \times 1$  filter output vector associated with the current position.

The choice of filter bank seems to be relatively unimportant. One approach has been to use Gaussians at scales  $\sigma$ ,  $2\sigma$ , and  $4\sigma$  to filter all three color channels, and derivatives of Gaussians at scales  $2\sigma$  and  $4\sigma$  and Laplacians of Gaussians at scales  $\sigma$ ,  $2\sigma$ ,  $4\sigma$ , and  $8\sigma$  to filter the luminance (figure 13.9a). In this way, both color and



**Figure 13.8** Texton maps. In a texton map each pixel is replaced by the texton index. This index characterizes the texture in the surrounding region. a) Original image. b) Associated texton map. Note how similar regions are assigned the same texton index (indicated by color). c) Original image. d) Associated texton map (using different filter bank from (b)). Texton maps are often used in semantic image segmentation. Adapted from Shotton *et al.* (2009). ©2009 Springer.



**Figure 13.9** Textons. The image is convolved with a filter bank to yield an  $N \times 1$  vector of filter responses at each position. Possible choices for the filter bank include a) a combination of Gaussians, derivatives of Gaussians, and Laplacians of Gaussians; b) rotationally invariant filters; and c) the maximum response (MR8) database. d) In training, the  $N \times 1$  filter response vectors are clustered using K-means. For new data, the texton index is assigned based on the nearest of these clusters. Thus, the filter space is effectively partitioned into Voronoi regions.

texture information is captured.

It may be desirable to compute textons that are invariant to orientation. One way of achieving this is to choose rotationally invariant filters to form the filter bank (figure 13.9b). However, these have the undesirable property of not responding at all to oriented structures in the image. The Maximum Response (MR8) filter bank is designed to provide a rotationally invariant measure of local texture which does not discard this information. The MR8 filter bank (figure 13.9c) consists of a Gaussian and a Laplacian of Gaussian filter, an edge filter at three scales and a bar filter (a symmetric oriented filter) at the same three scales. The edge and bar filter are replicated at 6 orientations at each scale, giving a total of 38 filters. To induce rotational invariance only the *maximum* filter response over orientation is used. Hence the final vector of filter responses consists of eight numbers, corresponding to the Gaussian and Laplacian filters (already invariant) and the maximum responses over orientation of the edge and bar filters at each of the three scales.



**Figure 13.10** Reconstruction from edges. a) Original image. b) Edge map. Each edge pixel has associated scale and orientation information as well as a record of the luminance levels at either side. c) The image can be reconstructed almost perfectly from the edges and their associated information. Adapted from Elder (1999). ©1999 Springer.

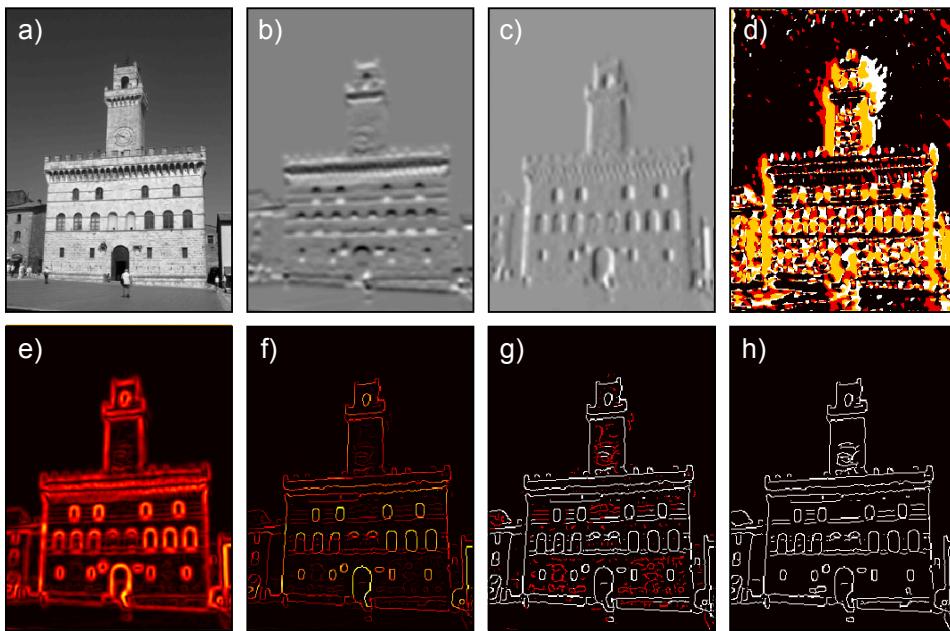
## 13.2 Edges, corners, and interest points

In this section, we consider methods that aim to identify informative parts of the image. In *edge detection* the goal is to return a binary image where a non-zero value denotes the presence of an edge in the image. Edge detectors optionally also return other information such as the orientation and scale associated with the edge. Edge maps are a highly compact representation of an image, and it has been shown that it is possible to reconstruct an image very accurately with just information about the edges in the scene (figure 13.10).

*Corners* are positions in the image that contain rich visual information and can be found reproducibly in different images of the same object (figure 13.12). There are many schemes to find corners, but they all aim to identify points that are locally unique. Corner detection algorithms were originally developed for geometric computer vision problems such as wide baseline image matching; here we see the same scene from two different angles and wish to identify which points correspond to which. In recent years, corners have also been used in object recognition algorithms (where they are usually referred to as *interest points*). The idea here is that the regions surrounding interest points contain information about which object class is present.

### 13.2.1 Canny edge detector

To compute edges with the Canny edge detector (figure 13.11), the image  $\mathbf{P}$  is first blurred and then convolved with a pair of orthogonal derivative filters such as Pre-witt filters to create images  $\mathbf{H}$  and  $\mathbf{V}$  containing derivatives in the horizontal and vertical directions, respectively. For pixel  $(i, j)$ , the orientation  $\theta_{ij}$  and magnitude  $a_{ij}$  of the gradient is computed using

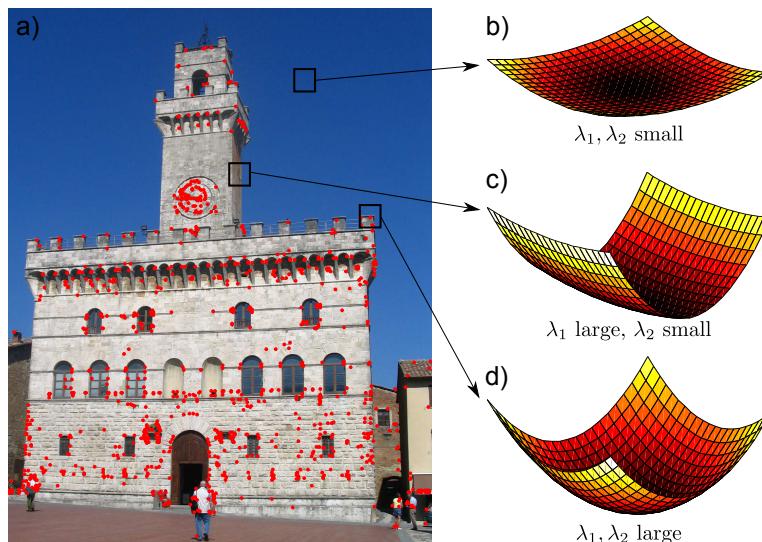


**Figure 13.11** Canny edge detection. a) Original image. b) Result of vertical Prewitt filter. c) Results of horizontal Prewitt filter. d) Quantized orientation map. e) Gradient amplitude map. f) Amplitudes after non-maximal suppression. g) Thresholding at two levels: the white pixels are above the higher threshold. The red pixels are above the lower threshold but below the higher one. h) Final edge map after hysteresis thresholding contains all of the white pixels from (g) and those red pixels that connect to them.

$$\begin{aligned}\theta_{ij} &= \arctan[v_{ij}/h_{ij}] \\ a_{ij} &= \sqrt{h_{ij}^2 + v_{ij}^2}.\end{aligned}\quad (13.13)$$

A simple approach would be to assign an edge to position  $(i, j)$  if the amplitude there exceeds a critical value. This is termed *thresholding*. Unfortunately, it produces poor results: the amplitude map takes high values on the edge, but also at adjacent positions. The Canny edge detector eliminates these unwanted responses using a method known as *non-maximum suppression*.

In non-maximum suppression the gradient orientation is quantized into one of four angles  $\{0^\circ, 45^\circ, 90^\circ, 135^\circ\}$ , where angles  $180^\circ$  apart are treated as equivalent. The pixels associated with each angle are now treated separately. For each pixel the amplitude is set to zero if either of the neighboring two pixels *perpendicular* to the gradient have higher values. For example, for a pixel where the gradient orientation is vertical (the image is changing in the horizontal direction), the pixels to the left and right are examined and the amplitude is set to zero if either of these



**Figure 13.12** Harris corner detector. a) Image with detected corners. The corner detection algorithm is based on the image structure tensor, which captures information about the distribution of gradients around the point. b) In flat regions, both singular values of the image structure tensor are small. c) On edges, one is small and the other large. d) At corners, both are large indicating that the image is changing quickly in both directions.

are greater than the current value. In this way, the gradients at the maximum of the edge amplitude profile are retained, and those away from this maximum are suppressed.

A binary edge map can now be computed by comparing the remaining non-zero amplitudes to a fixed threshold. However, for any given threshold there will be misses (places where there are real edges, but their amplitude falls below the threshold) and false positives (pixels labeled as edges where none exist in the original image). To decrease these undesirable phenomena, knowledge about the continuity of real-world edges is exploited. Two thresholds are defined. All of the pixels whose amplitude is above the higher threshold are labeled as edges, and this threshold is chosen so that there are few false positives. To try to decrease the number of misses, pixels that are above the lower amplitude threshold *and* are connected to an existing edge pixel are also labeled as edges. By iterating this last step, it is possible to trace along weaker parts of strong contours. This technique is known as *hysteresis thresholding*.

### 13.2.2 Harris corner detector

The Harris corner detector (figure 13.12) considers the local gradients in the horizontal and vertical directions around each point. The goal is to find points in the image where the image intensity is varying in both directions (a corner) rather than in one direction (an edge), or neither (a flat region). The Harris corner detector bases this decision on the *image structure tensor*

$$\mathbf{S}_{ij} = \sum_{m=i-D}^{i+D} \sum_{n=j-D}^{j+D} w_{mn} \begin{bmatrix} h_{mn}^2 & h_{mn}v_{mn} \\ h_{mn}v_{mn} & v_{mn}^2 \end{bmatrix}, \quad (13.14)$$

where  $\mathbf{S}_{ij}$  is the image structure tensor at position  $(i, j)$ , which is computed over a square region of size  $(2D+1) \times (2D+1)$  around the current position. The term  $h_{mn}$  denotes the response of a horizontal derivative filter (such as the Sobel) at position  $(m, n)$  and the term  $v_{mn}$  denotes the response of a vertical derivative filter. The term  $w_{mn}$  is a weight that diminishes the contribution of positions that are far from the central pixel  $(i, j)$ .

To identify whether a corner is present, the Harris corner detector considers the singular values  $\lambda_1, \lambda_2$  of the image structure tensor. If both singular values are small, then the region around the point is smooth and this position is not chosen. If one singular value is large but the other small, then the image is changing in one direction but not the other, and point lies on or near an edge. However, if both singular values are large, then this image is changing rapidly in both directions in this region and the position is deemed to be a corner.

In fact the Harris detector does not directly compute the singular values, but evaluates a criterion which accomplishes the same thing more efficiently:

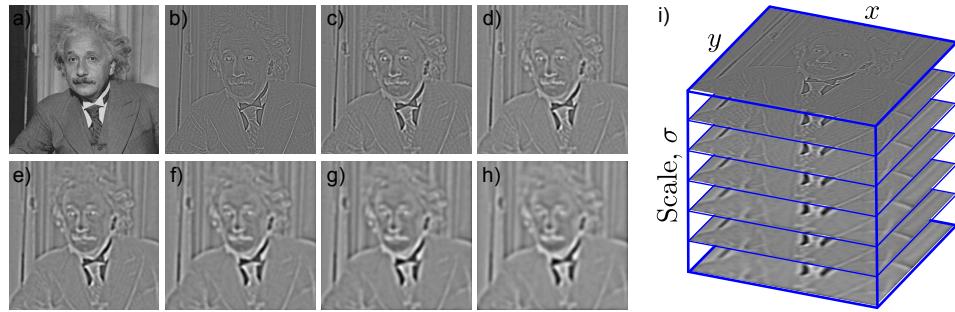
$$c_{ij} = \lambda_1 \lambda_2 - \kappa(\lambda_1^2 + \lambda_2^2) = \det[\mathbf{S}_{ij}] - \kappa \cdot \text{trace}[\mathbf{S}_{ij}], \quad (13.15)$$

where  $\kappa$  is a constant (values between 0.04 and 0.15 are sensible). If the value of  $c_{ij}$  is greater than a predetermined threshold, then a corner may be assigned. There is usually an additional non-maximal suppression stage similar to that in the Canny edge detector to ensure that only peaks in the function  $c_{ij}$  are retained.

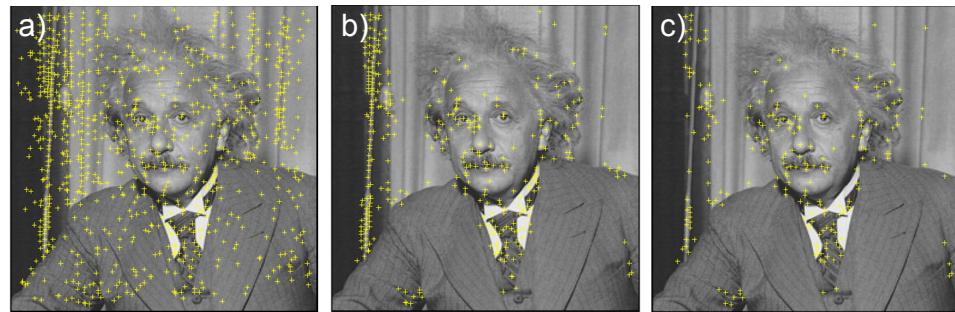
### 13.2.3 SIFT detector

The scale invariant feature transform (SIFT) detector is a second method for identifying interest points. Unlike the Harris corner detector, it associates a scale and orientation to each of the resulting interest points. To find the interest points a number of operations are performed in turn.

The intensity image is filtered with a difference of Gaussian kernel at a series of  $K$  increasingly coarse scales (figure 13.13). Then the filtered images are stacked to make a 3D volume of size  $I \times J \times K$  where  $I$  and  $J$  are the vertical and horizontal size of the image. *Extrema* are identified within this volume: these are positions where the 26 3D voxel neighbors (from a  $3 \times 3 \times 3$  block) are either all greater than or all less than the current value.



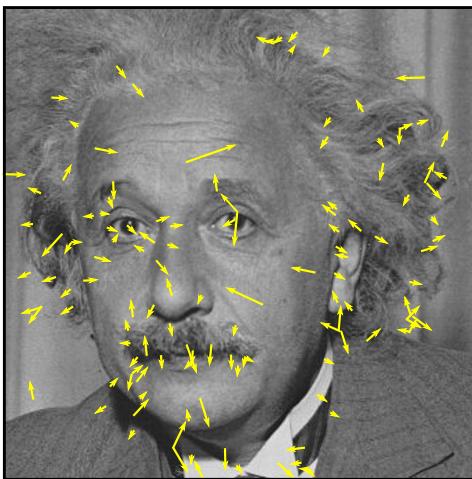
**Figure 13.13** The SIFT detector. a) Original image. b-h) The image is filtered with difference of Gaussian kernels at a range of increasing scales. i) The resulting images are stacked to create a 3D volume. Points that are local extrema in the filtered image volume (i.e., are either greater than or less than all 26 3D neighbors) are considered to be candidates for interest points.



**Figure 13.14** Refinement of SIFT detector candidates. a) Positions of extrema in the filtered image volume (figure 13.13i). Note that the scale is not shown. These are considered candidates to be interest points. b) Remaining candidates after eliminating those in smooth regions. c) Remaining candidate points after removing those on edges using the image structure tensor.

These extrema are localized to sub-voxel accuracy, by applying a local quadratic approximation and returning the position of the peak or trough. The quadratic approximation is made by taking a Taylor expansion about the current point. This provides a position estimate that has sub-pixel resolution and an estimate of the scale that is more accurate than the resolution of the scale sampling. Finally, the image structure tensor  $\mathbf{S}_{ij}$  (equation 13.14) is computed at the location and scale of each point. Candidate points in smooth regions and on edges are removed by considering the singular values of  $\mathbf{S}_{ij}$  as in the Harris corner detector (figure 13.14).

This procedure returns a set of interest points that are localized to sub-pixel



**Figure 13.15** Results of SIFT detector. Each final interest point is indicated using an arrow. The length of the arrow indicates the scale with which the interest point is identified and the angle of the arrow indicates the associated orientation. Notice that there are some positions in the image where the orientation was not unique and here two interest points are used, one associated with each orientation. An example of this is on the right shirt collar. Subsequent descriptors that characterize the structure of the image around the interest points are computed relative to this scale and orientation and hence inherit some invariance to these factors.

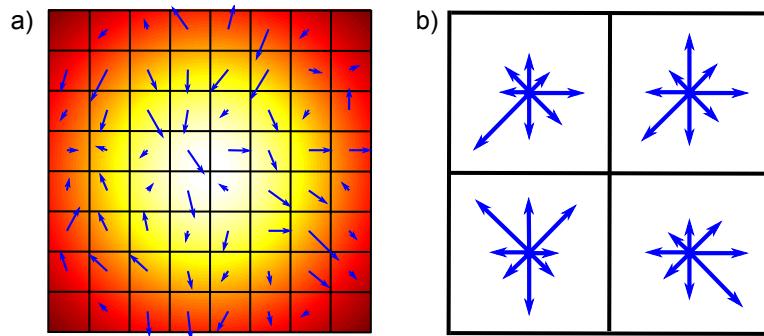
accuracy and associated accurately with a particular scale. Finally, a unique orientation is also assigned to each interest point. To this end, the amplitude and orientation of the local gradients are computed (equations 13.13) in a region surrounding the interest point whose size is proportional to the identified scale. An orientation histogram is then computed over this region with 36 bins covering all  $360^\circ$  of orientation. The contribution to the histogram depends on the gradient amplitude and is weighted by a Gaussian profile centered at the location of the interest point, so that nearby regions contribute more. The orientation of the interest point is assigned to be the peak of this histogram. If there is a second peak within 80% of the maximum, we may choose to compute descriptors at two orientations at this point. The final detected points are hence associated with a particular orientation and scale (figure 13.15).

## 13.3 Descriptors

In this section, we consider *descriptors*. These are compact representations that summarize the contents of an image region.

### 13.3.1 Histograms

The simplest approach to aggregating information over a large image region is to compute a histogram of the responses in this area. For example, we might collate RGB pixel intensities, filter responses, local binary patterns, or textons into a histogram depending on the application. The histogram entries can be treated as discrete and modeled with a categorical distribution, or treated as a continuous vector quantity.



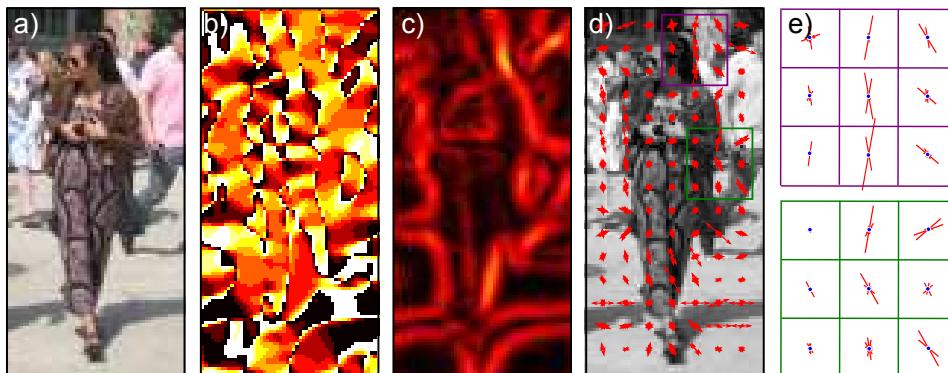
**Figure 13.16** SIFT descriptor a) Gradients are computed for every pixel within a region around the interest point. b) This region is subdivided into cells. Information is pooled within these cells to form an 8D histogram. These histograms are concatenated to provide a final descriptor that pools locally to provide invariance to small deformations but also retains some spatial information about the image gradients. In this figure, information from an  $8 \times 8$  pixel patch has been divided to make a  $2 \times 2$  grid of cells. In the original implementation of the SIFT detector, a  $16 \times 16$  patch was divided into a  $4 \times 4$  grid of cells.

For continuous quantities such as filter responses, the level of quantization is critical. Quantizing the responses into many bins potentially allows fine discrimination between responses. However, if data are scarce then many of these bins will be empty, and it is harder to reliably determine the statistics of the descriptor. One approach is to use an adaptive clustering method such as K-means (section 13.4.4) to automatically determine the bin sizes and shapes.

Histogramming is a useful approach for tasks where spatial resolution is not paramount: for example, to classify a large region of texture, it makes sense to pool information. However, this approach is largely unsuitable for characterizing structured objects: the spatial layout of the object is important for identification. We now introduce two representations for image regions that retain some spatial information but also pool information locally and thus provide invariance to small displacements and warps of the image. Both the SIFT descriptor (section 13.3.2) and the HOG descriptor (section 13.3.3) concatenate several histograms that were computed over spatially distinct blocks.

### 13.3.2 SIFT descriptors

The scale invariant feature transform (SIFT) descriptor (figure 13.16) characterizes the image region around a given point. It is usually used in conjunction with interest points that were found using the SIFT detector. These interest points are associated with a particular scale and rotation, and the SIFT descriptor would typically be computed over a square region that is transformed by these values.



**Figure 13.17** HOG descriptor. a) Original image. b) Gradient orientation, quantized into nine bins from 0 to  $180^\circ$ . c) Gradient magnitude. d) Cell descriptors are 9D orientation histograms that are computed within  $6 \times 6$  pixel regions. e) Block descriptors are computed by concatenating  $3 \times 3$  blocks of cell descriptors. The block descriptors are normalized. The final HOG descriptor consists of the concatenated block descriptors.

The goal is to characterize the image region in a way that is partially invariant to intensity and contrast changes and small geometric deformations.

To compute the SIFT descriptor, we first compute gradient orientation and amplitude maps (equation 13.13) as for the Canny edge detector over a  $16 \times 16$  pixel region around the interest point. The resulting orientation is quantized into eight bins spread over the range  $0 - 360^\circ$ . Then the  $16 \times 16$  detector region is divided into a regular grid of non-overlapping  $4 \times 4$  cells. Within each of these cells an eight dimensional histogram of the image orientations is computed. Each contribution to the histogram is weighted by the associated gradient amplitude and by distance so that positions further from the interest point contribute less. The  $4 \times 4 = 16$  histograms are concatenated to make a single  $128 \times 1$  vector, which is then normalized.

The descriptor is invariant to constant intensity changes as it is based on gradients. The final normalization provides some invariance to contrast. Small deformations do not affect the descriptor too much as it pools information within each cell. However, by keeping the information from each cell separate, some spatial information is retained.

### 13.3.3 Histogram of oriented gradients

The Histogram of Oriented Gradients (HOG) descriptor attempts to construct a more detailed characterization of the spatial structure with a small image window. It is a useful preprocessing step for algorithms that detect objects with quasi-regular structure such as pedestrians. Like the SIFT descriptor, the HOG

descriptor consists of a collection of normalized histograms computed over spatially offset patches; the result is a descriptor that captures coarse spatial structure, but is invariant to small local deformations.

The process of computing a HOG descriptor suitable for pedestrian detection consists of the following stages. First, the orientation and amplitude of the image gradients are computed at every pixel in a  $64 \times 128$  window using equation 13.13. The orientation is quantized into nine bins spread over the range  $0 - 180^\circ$ . The  $64 \times 128$  detector region is divided into a regular grid of overlapping  $6 \times 6$  cells. A 9D orientation histogram is computed within each cell, where the contribution to the histogram is weighted by the gradient amplitude and the distance from the center of the cell so that more central pixels contribute more. For each  $3 \times 3$  block of cells, the descriptors are concatenated and normalized to form a block descriptor. All of the block descriptors are concatenated to form the final HOG descriptor.

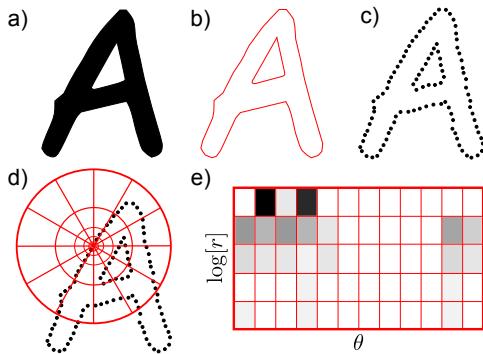
The final descriptor contains spatially pooled information about local gradients (within each cell), but maintains some spatial resolution (as there are many cells). It creates invariance to contrast polarity by only using the gradient magnitudes. It creates invariance to local contrast strength by normalizing relative to each block. The HOG descriptor is similar in spirit to the SIFT descriptor but is distinguished by being invariant to contrast polarity, having a higher spatial resolution of computed histograms and performing normalization more locally.

### 13.3.4 Bag of words descriptor

The descriptors discussed thus far have been intended to characterize small regions of images. Often these regions have been connected to interest points. The bag of words representation attempts to characterize a larger region or an entire image by summarizing the statistics of the descriptors (e.g., SIFT) associated with all of the interest points in a region.

Each observed descriptor is considered to be one of a finite vocabulary of possible descriptors (termed *visual words*). Collectively, this vocabulary is known as a *dictionary*. The bag of words descriptor is simply a histogram describing the frequency of observing these words giving no regard to their position. To compute the dictionary, interest points are found in a large number of images and the associated descriptor is computed. These descriptors are clustered using K-means (section 13.4.4). To compute the bag of words representation, each descriptor is assigned to the nearest word in this dictionary.

The bag of words representation is a remarkably good substrate for object recognition. This is somewhat surprising given that it surrenders any knowledge about the spatial configuration about the object. Of course, the drawback of approaches based on the bag of words is that it is very hard to localize the object after we have identified its presence or to decide how many instances are present using the same model.



**Figure 13.18** Shape context descriptor. a) Object silhouette. b) Contour of silhouette. c) Points are placed at equally spaced intervals around the silhouette. d) A log polar sampling array is centered at each point. e) The shape of the object relative to this point is captured by the histogram over the bins of the log polar array. The final descriptor would consist of a concatenation of the values from histograms from multiple points around the edge of the object.

### 13.3.5 Shape context descriptor

For certain vision tasks, the silhouette of the object contains much more information than the RGB values themselves. Consider, for example, the problem of body pose estimation: given an image of a human being, the goal is to estimate the 3D joint angles of the body. Unfortunately, the RGB values of the image depend on the person's clothing and are relatively uninformative. In such situations, it is wiser to attempt to characterize the shape of the object.

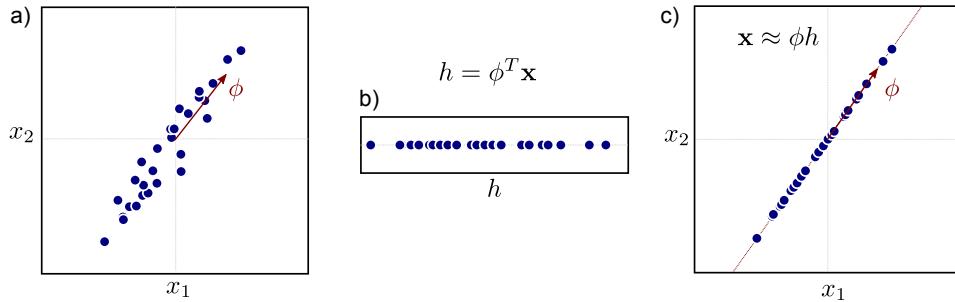
The shape context descriptor is a fixed length vector that characterizes the object contour. Essentially, it encodes the relative position of points on the contour. In common with the SIFT and HOG descriptors, it pools information locally over space to provide a representation that can capture the overall structure of the object but is not affected too much by small spatial variations.

To compute the shape context descriptor (figure 13.18), a discrete set of points is sampled along the contour of the object. A fixed length vector is associated with each point that characterizes the relative position of the other points. To this end, a log polar sampling array is centered on the current point. A histogram is then computed where each bin contains the number of the other points on the silhouette that fell into each bin of the log-polar array. The choice of the log-polar scheme means that the descriptor is very sensitive to local changes in the shape, but only captures the approximate configuration of distant parts.

The collection of histograms for all of the points on this image captures the shape. However, to directly match to another shape, the point correspondence must be established. It is possible to make this descriptor invariant to orientation by evaluating the orientation of the contour at each point and rotating the log polar sampling scheme so that it is aligned with this orientation.

## 13.4 Dimensionality reduction

It is often desirable to reduce the dimensionality of either the original or pre-processed image data. If we can do this without losing too much information, then the resulting models will require fewer parameters and be faster to learn and to



**Figure 13.19** Reduction to a single dimension. a) Original data and direction  $\phi$  of maximum variance. b) The data are projected onto  $\phi$  to produce a one dimensional representation. c) To reconstruct the data, we re-multiply by  $\phi$ . Most of the original variation is retained. PCA extends this model to project high dimensional data onto the  $K$  orthogonal dimensions with the most variance, to produce a  $K$  dimensional representation.

use for inference.

Dimensionality reduction is possible because a given type of image data (e.g., RGB values from face images) usually lie in a tiny subset of the possible data space; not all sets of RGB values look like real images, and not all real images look like faces. We refer to the subset of the space occupied by a given dataset as a *manifold*. Dimensionality reduction can hence be thought of as a change of variables: we move from the original coordinate system to the (reduced) coordinate system within the manifold.

Our goal is hence to find a low-dimensional (or hidden) representation  $\mathbf{h}$  which can approximately explain the data  $\mathbf{x}$ , so that

$$\mathbf{x} \approx f[\mathbf{h}, \boldsymbol{\theta}], \quad (13.16)$$

where  $f[\bullet, \bullet]$  is a function that takes the hidden variable and a set of parameters  $\boldsymbol{\theta}$ . We would like the lower-dimensional representation to capture all of the relevant variation in the original data. Hence, one possible criterion for choosing the parameters is to minimize the least squares *reconstruction error*, so that

$$\hat{\boldsymbol{\theta}}, \hat{\mathbf{h}}_{1\dots I} = \underset{\boldsymbol{\theta}, \mathbf{h}_1\dots I}{\operatorname{argmin}} \left[ \sum_{i=1}^I (\mathbf{x}_i - f[\mathbf{h}_i, \boldsymbol{\theta}])^T (\mathbf{x}_i - f[\mathbf{h}_i, \boldsymbol{\theta}]) \right], \quad (13.17)$$

where  $\mathbf{x}_i$  is the  $i^{th}$  of  $I$  training examples. In other words, we aim to find a set of low dimensional variables  $\{\mathbf{h}_i\}_{i=1}^I$  and a mapping from  $\mathbf{h}$  to  $\mathbf{x}$  so that it reconstructs the original data as closely as possible in a least squares sense.

### 13.4.1 Approximation with a single number

Let us first consider a very simple model in which we attempt to represent each observed datum with a single number (figure 13.19), so that

$$\mathbf{x}_i \approx \phi h_i + \mu, \quad (13.18)$$

where the parameter  $\mu$  is the mean of the data set and the parameter  $\phi$  is a basis vector mapping the low dimensional representation  $\mathbf{h}$  back to the original data space  $\mathbf{x}$ . For simplicity, we will assume from now on that the mean of the dataset is zero and  $\mathbf{x}_i \approx \phi h_i$ . This can be achieved by computing the empirical mean  $\mu$  and subtracting it from every example  $\mathbf{x}_i$ .

The learning algorithm optimizes the criterion

$$\hat{\phi}, \hat{h}_{1\dots I} = \underset{\phi, h_{1\dots I}}{\operatorname{argmin}} [E] = \underset{\phi, h_{1\dots I}}{\operatorname{argmin}} \left[ \sum_{i=1}^I (\mathbf{x}_i - \phi h_i)^T (\mathbf{x}_i - \phi h_i) \right]. \quad (13.19)$$

Careful consideration of the cost function (equation 13.19) reveals an immediate problem: the solution is ambiguous as we can multiply the basis function  $\phi$  by any constant  $k$  and divide each of the hidden variables  $\{h_i\}_{i=1}^I$  by the same number to yield exactly the same cost. To resolve this problem, we force the vector  $\phi$  to have unit length. This is accomplished by adding in a Lagrange multiplier  $\lambda$  so that the cost function becomes

$$\begin{aligned} E &= \sum_{i=1}^I (\mathbf{x}_i - \phi h_i)^T (\mathbf{x}_i - \phi h_i) + \lambda(\phi^T \phi - 1) \\ &= \sum_{i=1}^I \mathbf{x}_i^T \mathbf{x}_i - 2h_i \phi^T \mathbf{x}_i + h_i^2 + \lambda(\phi^T \phi - 1). \end{aligned} \quad (13.20)$$

To minimize the function, we first take the derivative with respect to  $h_i$ , and then equate the resulting expression to zero to yield

$$\hat{h}_i = \hat{\phi}^T \mathbf{x}_i. \quad (13.21)$$

In other words, to find the reduced dimension representation  $h_i$  we simply project the observed data onto the vector  $\phi$ .

We now take the derivative of equation 13.20 with respect to  $\phi$ , substitute in the solution for  $h_i$ , equate the result to zero, and re-arrange to get

$$\sum_{i=1}^I \mathbf{x}_i \mathbf{x}_i^T \hat{\phi} = \lambda \hat{\phi}, \quad (13.22)$$

or in matrix form

$$\mathbf{X} \mathbf{X}^T \hat{\phi} = \lambda \hat{\phi}, \quad (13.23)$$

where the matrix  $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_I]$  contains the data examples in its columns. This is an eigenvalue problem. To find the optimal vector, we compute the SVD  $\mathbf{ULV}^T = \mathbf{XX}^T$  and choose the first column of  $\mathbf{U}$ .

The scatter matrix  $\mathbf{XX}^T$  is a constant multiple of the covariance matrix, and so this has a simple geometric interpretation. The optimal vector  $\phi$  to project onto corresponds to the principal direction of the covariance ellipse. This makes intuitive sense; we retain information from the direction in space where the data vary most.

### 13.4.2 Principal component analysis

Principal component analysis (PCA) generalizes the above model. Instead of finding a scalar variable  $h_i$  that represents the  $i^{th}$  data example  $\mathbf{x}_i$ , we now seek a  $K$  dimensional vector  $\mathbf{h}_i$ . The relation between the hidden and observed spaces is

$$\mathbf{x}_i \approx \Phi \mathbf{h}_i, \quad (13.24)$$

where the matrix  $\Phi = [\phi_1, \phi_2, \dots, \phi_K]$  contains  $K$  basis functions or *principal components*; the observed data are modeled as a weighted sum of the principal components, where the  $k^{th}$  dimension of  $\mathbf{h}_i$  weights the  $k^{th}$  component.

The solution for the unknowns  $\Phi$  and  $\mathbf{h}_{1\dots I}$  can now be written as

$$\hat{\Phi}, \hat{\mathbf{h}}_{1\dots I} = \underset{\Phi, \mathbf{h}_{1\dots I}}{\operatorname{argmin}} [E] = \underset{\Phi, \mathbf{h}_{1\dots I}}{\operatorname{argmin}} \left[ \sum_{i=1}^I (\mathbf{x}_i - \Phi \mathbf{h}_i)^T (\mathbf{x}_i - \Phi \mathbf{h}_i) \right]. \quad (13.25)$$

Once more, the solution to this is non-unique as we can post-multiply  $\Phi$  by any matrix  $\mathbf{A}$  and pre-multiply each hidden variable  $\mathbf{h}_i$  by the inverse  $\mathbf{A}^{-1}$  and still get the same cost. To (partially) resolve this problem, we add the extra constraint that  $\Phi^T \Phi = \mathbf{I}$ . In other words, we force the principal components to be orthogonal and length one. This gives a modified cost function of

$$E = \sum_{i=1}^I (\mathbf{x}_i - \Phi \mathbf{h}_i)^T (\mathbf{x}_i - \Phi \mathbf{h}_i) + \lambda (\Phi^T \Phi - \mathbf{I}), \quad (13.26)$$

where  $\lambda$  is a Lagrange multiplier. We now minimize this expression with respect to  $\Phi, \mathbf{h}_{1\dots I}$  and  $\lambda$ . The expression for the hidden variables becomes

$$\mathbf{h}_i = \Phi^T \mathbf{x}_i. \quad (13.27)$$

The  $K$  principal components  $\Phi = [\phi_1, \phi_2, \dots, \phi_K]$  are now found by computing the singular value decomposition  $\mathbf{ULV}^T = \mathbf{XX}^T$  and taking the first  $K$  columns of  $\mathbf{U}$ . In other words, to reduce the dimensionality, we project the data  $\mathbf{x}_i$  onto a hyperplane defined by the  $K$  largest axes of the covariance ellipsoid.

This algorithm is very closely related to probabilistic principal component analysis (section 17.5.1). Probabilistic PCA additionally models the noise that accounts for the inexact approximation in equation 13.24. Factor analysis (section 7.6) is also very similar, but constructs a more sophisticated model of this noise.

Algorithm 13.1

### 13.4.3 Dual principal component analysis

The preceding method requires us to compute the SVD of the scatter matrix  $\mathbf{XX}^T$ . Unfortunately, if the data had dimension  $D$ , then this is a  $D \times D$  matrix, which may be very large. We can sidestep this problem by using dual variables. We define  $\Phi$  as a weighted sum of the original data points so that

$$\Phi = \mathbf{X}\Psi, \quad (13.28)$$

where  $\Psi = [\psi_1, \psi_2, \dots, \psi_K]$  is a  $I \times K$  matrix representing these weights. The associated cost function now becomes

$$E = \sum_{i=1}^I (\mathbf{x}_i - \mathbf{X}\Psi\mathbf{h}_i)^T (\mathbf{x}_i - \mathbf{X}\Psi\mathbf{h}_i) + \lambda(\Psi^T \mathbf{X}^T \mathbf{X}\Psi - \mathbf{I}). \quad (13.29)$$

The solution for the hidden variables becomes

$$\mathbf{h}_i = \Psi^T \mathbf{X}^T \mathbf{x}_i = \Phi^T \mathbf{x}_i, \quad (13.30)$$

and the  $K$  dual principal components  $\Psi = [\psi_1, \psi_2, \dots, \psi_K]$  are extracted from the matrix  $\mathbf{U}$  in the SVD  $\mathbf{ULV}^T = \mathbf{X}^T \mathbf{X}$ . This is a smaller problem of size  $I \times I$ , and is more efficient when the number of data examples  $I$  is less than the dimensionality of the observed space  $D$ .

Notice that this algorithm does not require the original data points: it only requires the inner products between them and so it is amenable to kernelization. This resulting method is known as kernel PCA.

### 13.4.4 The K-means algorithm

A second common approach to dimensionality reduction is to abandon a continuous representation altogether and represent each data point using one of a limited set of prototype vectors. In this model, the data are approximated as

$$\mathbf{x}_i \approx \boldsymbol{\mu}_{h_i}, \quad (13.31)$$

Algorithm 13.2

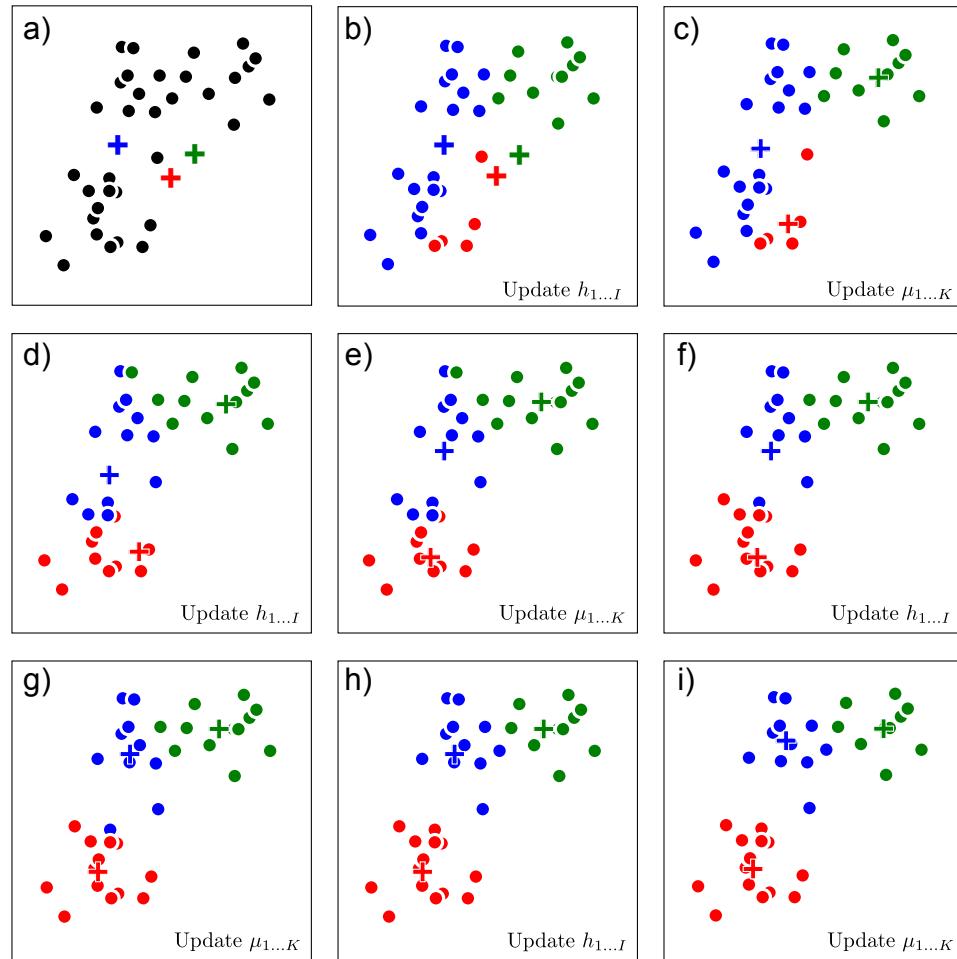
where  $h_i \in \{1, 2, \dots, K\}$  is an index that identifies which of the  $K$  prototype vectors  $\{\boldsymbol{\mu}_k\}_{k=1}^K$  approximates the  $i^{th}$  example.

To find the assignment indices and the prototype vectors (figure 13.20) we optimize

$$\hat{\boldsymbol{\mu}}_{1\dots K}, \hat{h}_{1\dots I} = \operatorname{argmin}_{\boldsymbol{\mu}, h} \left[ \sum_{i=1}^I (\mathbf{x}_i - \boldsymbol{\mu}_{h_i})^T (\mathbf{x}_i - \boldsymbol{\mu}_{h_i}) \right]. \quad (13.32)$$

In the K-means algorithm this cost function is minimized using an alternating strategy in which we first assign each datapoint to the nearest prototype

$$\hat{h}_i = \operatorname{argmin}_{h_i} \left[ (\mathbf{x}_i - \boldsymbol{\mu}_{h_i})^T (\mathbf{x}_i - \boldsymbol{\mu}_{h_i}) \right], \quad (13.33)$$



**Figure 13.20** K-means algorithm for  $K = 3$  clusters. a) We initialize the three prototype vectors (crosses) to random positions. We alternately b) assign the data to the nearest prototype vector and c) update the prototype vectors to be equal to the mean of the points assigned to them. d-i) We repeat these steps until there is no further change.

and then update the prototypes

$$\begin{aligned}\hat{\boldsymbol{\mu}}_k &= \underset{\boldsymbol{\mu}_k}{\operatorname{argmin}} \left[ \sum_{i=1}^I \left[ (\mathbf{x}_i - \boldsymbol{\mu}_{h_i})^T (\mathbf{x}_i - \boldsymbol{\mu}_{h_i}) \right] \right] \\ &= \frac{\sum_{i=1}^I \mathbf{x}_i \delta[h_i - k]}{\sum_{i=1}^I \delta[h_i - k]},\end{aligned}\quad (13.34)$$

where  $\delta[\bullet]$  is a function that returns one when its argument is zero and zero otherwise.

wise. In other words, the new prototype  $\hat{\mu}_k$  is simply the average of the data points that are assigned to this cluster. This algorithm is not guaranteed to converge to the global minimum and so it requires sensible starting conditions.

The K-means algorithm is very closely related to the mixtures of Gaussians model (section 7.4). The main differences are that the mixtures of Gaussians model is probabilistic and defines a density over the data space. It also assigns weights to the clusters and describes their covariance.

Problem 13.8  
Problem 13.9  
Problem 13.10

## Conclusion

Careful reading of the information in this chapter should convince you that there are certain recurring ideas in image preprocessing. To make a descriptor invariant to intensity changes we *filter* the image and *normalize* the filter responses over the region. A unique descriptor orientation and scale can be computed by *maximizing* over responses at different orientations and scales. To create invariance to small spatial changes, local responses are *pooled*. Despite the simplicity of these ideas, it is remarkable how much impact they have on the performance of real systems.

## Notes

**Image Processing:** There are numerous texts on image processing which contain far more information than I could include in this chapter. I would particularly recommend the books by O' Gorman *et al.* (2008), Gonzalez & Woods (2002), Pratt (2007), and Nixon & Aguado (2008). A comprehensive recent summary of local image features can be found in Li & Allinson (2008).

**Edge and corner detection:** The Canny edge detector was first described in Canny (1986). Elder (1999) investigated whether it was possible to reconstruct an image based on edge information alone. Nowadays, it is common to use machine learning methods to identify object boundaries in images (e.g., Dollár *et al.* 2006)

Early work in corner detection (interest point detection) includes that of Moravec (1983), Förstner (1986) and the Harris corner detector (Harris & Stephens 1988) which we described in this chapter. Other more recent efforts to identify stable points and regions include the SUSAN corner detector (Smith & Brady 1997), a saliency-based descriptor (Kadir & Brady 2001), maximally stable extremal regions (Matas *et al.* 2002), the SIFT detector (Lowe 2004) and the FAST detector (Rosten & Drummond 2006). There has been considerable recent interest in *affine invariant* interest point detection, which aims to find features that are stable under affine transforms of the image (e.g., Schaffalitzky & Zisserman 2002; Mikolajczyk & Schmid 2002; Mikolajczyk & Schmid 2004). Mikolajczyk *et al.* (2005) present a quantitative comparison of different affine region detectors. A recent review of this area can be found in Tuytelaars & Mikolajczyk (2007).

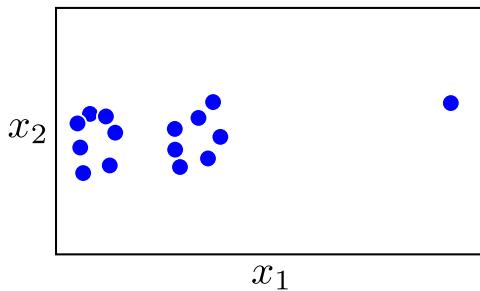
**Image descriptors:** For robust object recognition and image matching it is crucial to characterize the region around the detected interest point in a way that is compact and stable to changes in the image. To this end, Lowe (2004) developed the SIFT descriptor, Dalal & Triggs (2005) developed the HOG descriptor, and Forssén & Lowe (2007) developed a descriptor for use with maximally stable extremal regions. Bay *et al.* (2008) developed a very efficient version of SIFT features known as SURF. Mikolajczyk & Schmid (2005) present a quantitative comparison of region descriptors. Recent work on image descriptors has applied machine learning techniques to optimize their performance (Brown *et al.* 2011; Philbin *et al.* 2010).

More information about local binary patterns can be found in Ojala *et al.* (2002). More information about the shape context descriptor can be found in Belongie *et al.* (2002).

**Dimensionality reduction:** Principal components analysis is a linear dimensionality reduction method. However, there are also many nonlinear approaches that describe a manifold of images in high dimensions with fewer parameters. Notable methods include kernel PCA (Schölkopf *et al.* 1997), ISOMAP (Tenenbaum *et al.* 2000), local linear embedding Roweis & Saul (2000), charting (Brand 2002), the Gaussian process latent variable model (Lawrence 2004), and Laplacian eigenmaps (Belkin & Niyogi 2001). Recent reviews of dimensionality reduction can be found in Burgess (2010) and De La Torre (2011).

## Problems

**Problem 13.1** Consider an 8-bit image in which the pixel values are evenly distributed in the range 0 to 127, with no pixels taking a value of 128 or larger. Draw the cumulative histogram for this image (see figure 13.2). What will the histogram of pixel intensities



**Figure 13.21** Clustering with the K-means algorithm in the presence of outliers (problem 13.9). This data set contains two clusters and a single outlier (the point on the right hand side). The outlier causes problems for the K-means algorithm when  $K = 2$  clusters are used due to the implicit assumption that the clusters can be modeled as normal distributions with spherical covariance.

look like after applying histogram equalization?

**Problem 13.2** Consider a continuous image  $p[i, j]$  and a continuous filter  $f[n, m]$ . In the continuous domain, the operation  $f \otimes p$  of convolving an image with the filter is defined as:

$$f \otimes p = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p[i - m, j - n] f[m, n] dm dn.$$

Now consider two filters  $f$  and  $g$ . Prove that convolving the image first with  $f$  and then with  $g$  has the same effect as convolving  $f$  with  $g$  and then convolving the image with the result. In other words:

$$g \otimes (f \otimes p) = (g \otimes f) \otimes p.$$

Does this result extend to discrete images?

**Problem 13.3** Describe the series of operations that would be required to compute the Haar-like filters in figures 13.6a-d from an integral image. How many points from the integral image are needed to compute each?

**Problem 13.4** Consider a blurring filter where each pixel in an image is replaced by a weighted average of local intensity values, but the weights decrease if these intensity values differ markedly from the central pixel. What effect would this *bilateral filter* have when applied to an image?

**Problem 13.5** Define a  $3 \times 3$  filter that is specialized to detecting luminance changes at a  $45^\circ$  angle and gives a positive response where the image intensity increases from the bottom left to the bottom right of the image.

**Problem 13.6** Define a  $3 \times 3$  filter that responds to the second derivative in the horizontal direction, but is invariant to the gradient and absolute intensity in the horizontal direction and invariant to all changes in the vertical direction.

**Problem 13.7** Why are most local binary patterns in a natural image typically uniform or near-uniform?

**Problem 13.8** Give one example of a 2D data set where the mixtures of Gaussians model will succeed in clustering the data, but the K-means algorithm will fail.

**Problem 13.9** Consider the data in figure 13.21. What do you expect to happen if we run the K-means algorithm with two clusters on this data set? Suggest a way to resolve this problem.

**Problem 13.10** An alternative approach to clustering the data would be to find modes (peaks) in the density of the points. This potentially has the advantage of also automatically selecting the number of clusters. Propose an algorithm to find these modes.

# **Part V**

# **Models for geometry**



# Part V: Models for geometry

In part V, we finally acknowledge the process by which real-world images are formed. Light is emitted from one or more sources and travels through the scene interacting with the materials via physical processes such as reflection, refraction, and scattering. Some of this light enters the camera and is measured. We have a very good understanding of this forward model. Given known geometry, light sources, and material properties, computer graphics techniques can simulate what will be seen by the camera very accurately.

The ultimate goal for a vision algorithm would be a complete *reconstruction*, in which we aim to invert this forward model, and estimate the light sources, materials, and geometry from the image. Here, we aim to capture a structural description of the world: we seek an understanding of where things are, and to measure their optical properties, rather than a semantic understanding. Such a structural description can be exploited to navigate around the environment or build 3D models for computer graphics.

Unfortunately, full visual reconstruction is very challenging. For one thing, the solution is non-unique. For example, if the light source intensity increases, but the object reflectance decreases commensurately, the image will remain unchanged. Of course, we could make the problem unique by imposing prior knowledge, but even then reconstruction remains difficult; it is hard to effectively parameterize the scene, and the problem is highly non-convex.

In this part of the book, we consider a family of models that approximate both the 3D scene and the observed image with sparse sets of visual primitives (points). The forward model that maps the proxy representation of the world (3D points) to the proxy representation of the image (2D points) is much simpler than the full light transport model, and is called the *projective pinhole camera*. We investigate the properties of this model in chapter 14.

In chapter 15, we consider the situation where the pinhole camera views a plane in the world; there is now a one-to-one mapping between points on the plane and points in the image, and we characterize this mapping with a family of 2D transformations. In chapter 16, we will further exploit the pinhole camera model to recover a sparse geometric model of the scene.



# Chapter 14

## The pinhole camera

This chapter introduces the pinhole or projective camera. This is a purely geometric model that describes the process whereby points in the world are projected into the image. Clearly, the position in the image depends on the position in the world, and the pinhole camera model captures this relationship.

To motivate this model, we will consider the problem of *sparse stereo reconstruction* (figure 14.1). We are given two images of a rigid object taken from different positions. Let us assume that we can identify corresponding 2D features between the two images – points that are projected versions of the same position in the 3D world. Now the goal is to establish this 3D position using the observed 2D feature points. The resulting 3D information could be used by a robot to help it navigate through the scene, or to facilitate object recognition.

### 14.1 The pinhole camera

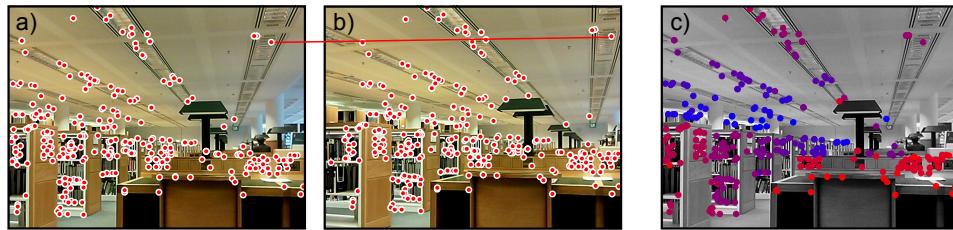
In real life, a pinhole camera consists of a closed chamber<sup>1</sup> with a small hole (the pinhole) in the front (figure 14.2). Rays from an object in the world pass through this hole to form an inverted image on the back face of the box or *image plane*. Our goal is to build a mathematical model of this process.

It is slightly inconvenient that the image from the pinhole camera is upside-down. Hence, we instead consider the *virtual image* that would result from placing the image plane *in front of* the pinhole. Of course, it is not physically possible to build a camera this way, but it is mathematically equivalent to the true pinhole model (except that the image is the right way up) and it is easier to think about. From now on, we will always draw the image plane in front of the pinhole.

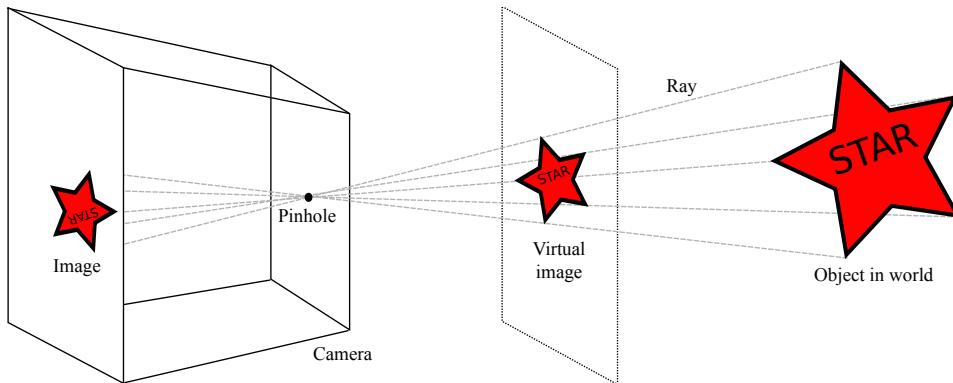
Figure 14.3 illustrates the pinhole camera model and defines some terminology. The pinhole itself (the point at which the rays converge) is called the *optical center*. We will assume for now that the optical center is at the origin of the 3D world coordinate system in which points are represented as  $\mathbf{w} = [u, v, w]^T$ . The virtual

---

<sup>1</sup>This is not an accidental choice of word. The term *camera* is derived from the Latin word for ‘chamber.’



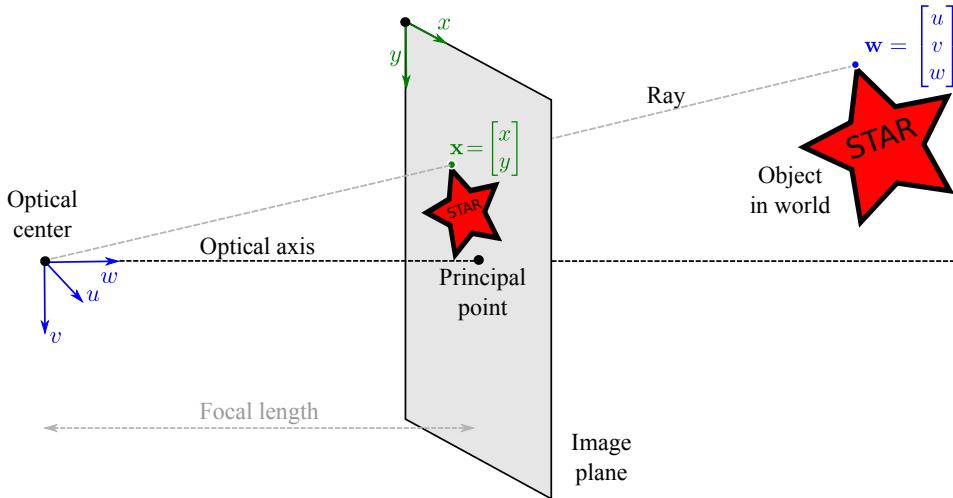
**Figure 14.1** Sparse stereo reconstruction. a,b) We are given two images of the same scene, taken from different positions, and a set of  $I$  pairs of points in these images that are known to correspond to the same points in the world (e.g., the points connected by the red-line are a corresponding pair). c) Our goal is to establish the 3D position of each of the world points. Here, the depth is encoded by color so that closer points are red and more distant points are blue.



**Figure 14.2** The pinhole camera model. Rays from an object in the world pass through the pinhole in the front of the camera and form an image on the back plane (the image plane). This image is upside-down, so we can alternatively consider the virtual image that would have been created if the image plane was in front of the pinhole. This is not physically possible, but it is more convenient to work with.

image is created on the *image plane*, which is displaced from the optical center along the  $w$ -axis or *optical axis*. The point where the optical axis strikes the image plane is known as the *principal point*. The distance between the principal point and the optical center (i.e., the distance between the image plane and the pinhole) is known as the *focal length*.

The pinhole camera model is a generative model that describes the likelihood  $Pr(\mathbf{x}|\mathbf{w})$  of observing a feature at position  $\mathbf{x} = [x, y]^T$  in the image, given that it is the projection of a 3D point  $\mathbf{w} = [u, v, w]^T$  in the world. Although light transport



**Figure 14.3** Pin-hole camera model terminology. The optical center (pinhole) is placed at the origin of the 3D world coordinate system  $(u, v, w)$ , and the image plane (where the virtual image is formed) is displaced along the  $w$ -axis, which is also known as the optical axis. The position where the optical axis strikes the image plane is called the principal point. The distance between the image plane and the optical center is called the focal length.

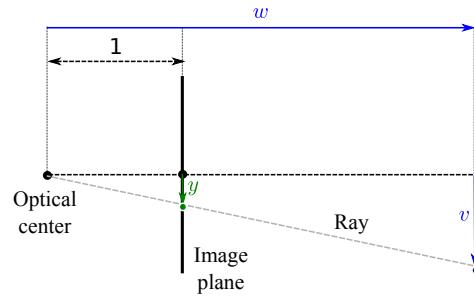
is essentially deterministic, we will nonetheless build a probability model; there is noise in the sensor, and unmodeled factors in the feature detection process can also affect the measured image position. However, for pedagogical reasons we will defer a discussion of this uncertainty until later, and temporarily treat the imaging process as if it were deterministic.

Our task then is to establish the position  $\mathbf{x} = [x, y]^T$  where the 3D point  $\mathbf{w} = [u, v, w]^T$  is imaged. Considering figure 14.3, it is clear how to do this. We connect a ray between  $\mathbf{w}$  and the optical center. The image position  $\mathbf{x}$  can be found by observing where this ray strikes the image plane. This process is called *perspective projection*. In the next few sections, we will build a more precise mathematical model of this process. We will start with a very simple camera model (the normalized camera) and build up to a full camera parameterization.

### 14.1.1 The normalized camera

In the *normalized camera*, the focal length is one, and it is assumed that the origin of the 2D coordinate system  $(x, y)$  on the image plane is centered at the principal point. Figure 14.4 shows a 2D slice of the geometry of this system (the  $u$ - and  $x$ -axes now point upward out of the page and cannot be seen). By similar triangles, it can easily be seen that the  $y$ -position in the image of the world point

**Figure 14.4** Normalized camera. The focal length is one, and the 2D image coordinate system  $(x, y)$  is centered on the principal point (only  $y$ -axis shown). By similar triangles, the  $y$  position in the image of a point at  $(u, v, w)$  is given by  $v/w$ . This corresponds to our intuition: as an object gets more distant, its projection becomes closer to the center of the image.



at  $\mathbf{w} = [u, v, w]^T$  is given by  $v/w$ . More generally, in the normalized camera, a 3D point  $\mathbf{w} = [u, v, w]^T$  is projected into the image at  $\mathbf{x} = [x, y]^T$  using the relations

$$\begin{aligned} x &= \frac{u}{w} \\ y &= \frac{v}{w}, \end{aligned} \tag{14.1}$$

where  $x, y, u, v$ , and  $w$  are measured in the same real-world units (e.g., mm).

### 14.1.2 Focal length parameters

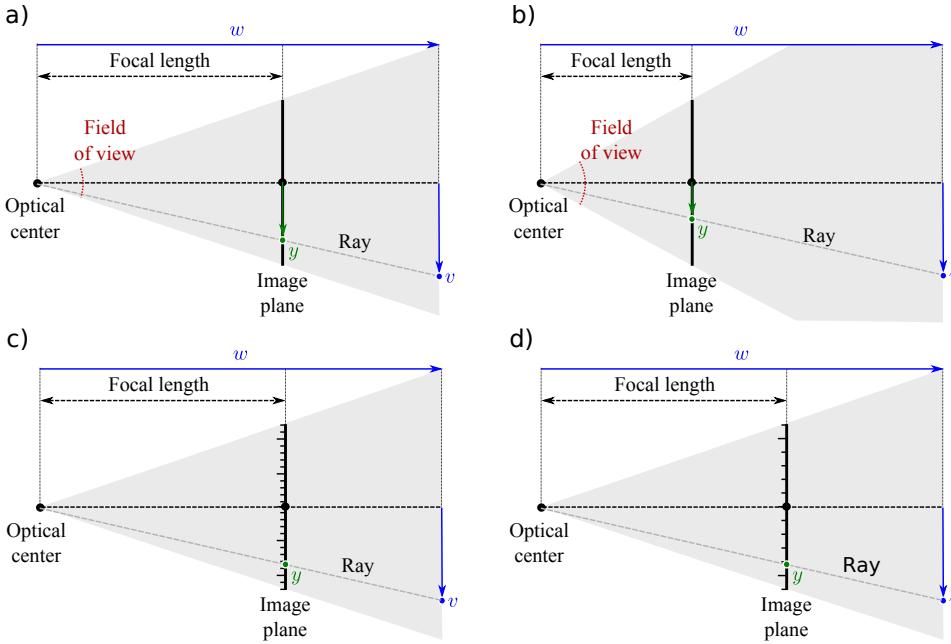
The normalized camera is unrealistic; for one thing, in a real camera, there is no particular reason why the focal length should be one. Moreover, the final position in the image is measured in pixels, not physical distance, so the model must take into account the photoreceptor spacing. Both of these factors have the effect of changing the mapping between points  $\mathbf{w} = [u, v, w]^T$  in the 3D world and their 2D positions  $\mathbf{x} = [x, y]^T$  in the image plane by a constant scaling factor  $\phi$  (figure 14.5), so that

$$\begin{aligned} x &= \frac{\phi u}{w} \\ y &= \frac{\phi v}{w}. \end{aligned} \tag{14.2}$$

To add a further complication, the spacing of the photoreceptors may differ in the  $x$ - and  $y$ -directions, and so the scaling may be different in each direction, giving the relations

$$\begin{aligned} x &= \frac{\phi_x u}{w} \\ y &= \frac{\phi_y v}{w}, \end{aligned} \tag{14.3}$$

where  $\phi_x$  and  $\phi_y$  are separate scaling factors for the  $x$ - and  $y$ -directions. These parameters are known as the *focal length parameters* in the  $x$ - and  $y$ -directions,



**Figure 14.5** Focal length and photoreceptor spacing. a-b) Changing the distance between the optical center and the image plane (the focal length) changes the relationship between the 3D world point  $w = [u, v, w]^T$  and the 2D image point  $x = [x, y]^T$ . In particular, if we take the original focal length (a) and halve it (b), the 2D image coordinate is also halved. The *field of view* of the camera is the total angular range that is imaged (usually different in the  $x$ - and  $y$ -directions). When the focal length decreases, the field of view increases. c-d) The position in the image  $x = [x, y]^T$  is usually measured in pixels. Hence, the position  $x$  depends on the density of the receptors on the image plane. If we take the original photoreceptor density (c) and halve it (d), then the 2D image coordinate is also halved. Hence, the photoreceptor spacing and focal length both change the mapping from rays to pixels in the same way.

but this name is somewhat misleading - they account for not just the distance between the optical center and the principal point (the true focal length) but also the photoreceptor spacing.

### 14.1.3 Offset and skew parameters

The model so far is still incomplete in that pixel position  $x = [0, 0]^T$  is at the principal point (where the  $w$ -axis intersects the image plane). In most imaging systems, the pixel position  $x = [0, 0]^T$  is at the top-left of the image rather than the center. To cope with this, we add *offset parameters*  $\delta_x$  and  $\delta_y$  so that

$$\begin{aligned} x &= \frac{\phi_x u}{w} + \delta_x \\ y &= \frac{\phi_y v}{w} + \delta_y, \end{aligned} \quad (14.4)$$

where  $\delta_x$  and  $\delta_y$  are the offsets in pixels from the top-left corner of the image to the position where the  $w$  axis strikes the image plane. Another way to think about this is that the vector  $[\delta_x, \delta_y]^T$  is the position of the principal point in pixels.

If the image plane is exactly centered on the  $w$ -axis, these offset parameters should be half the image size: for a  $640 \times 480$  VGA image  $\delta_x$  and  $\delta_y$  would be 320 and 240, respectively. However, in practice it is difficult and superfluous to manufacture cameras with the imaging sensor perfectly centered, and so we treat the offset parameters as variable quantities.

We also introduce a *skew* term  $\gamma$  which moderates the projected position  $x$  as a function of the height  $v$  in the world. This parameter has no clear physical interpretation, but can help explain the projection of points into the image in practice. The resulting camera model is

$$\begin{aligned} x &= \frac{\phi_x u + \gamma v}{w} + \delta_x \\ y &= \frac{\phi_y v}{w} + \delta_y. \end{aligned} \quad (14.5)$$

#### 14.1.4 Position and orientation of camera

Finally, we must account for the fact that the camera is not always conveniently centered at the origin of the world coordinate system with the optical axis exactly aligned with the  $w$ -axis. In general, we may want to define an arbitrary world coordinate system that may be common to more than one camera. To this end, we express the world points  $\mathbf{w}$  in the coordinate system of the camera before they are passed through the projection model, using the coordinate transformation:

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{21} & \omega_{22} & \omega_{23} \\ \omega_{31} & \omega_{32} & \omega_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} + \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}, \quad (14.6)$$

or

$$\mathbf{w}' = \boldsymbol{\Omega} \mathbf{w} + \boldsymbol{\tau}, \quad (14.7)$$

where  $\mathbf{w}'$  is the transformed point,  $\boldsymbol{\Omega}$  is a  $3 \times 3$  rotation matrix, and  $\boldsymbol{\tau}$  is a  $3 \times 1$  translation vector.

#### 14.1.5 Full pinhole camera model

We are now in a position to describe the full camera model, by combining equations

14.5 and 14.6. A 3D point  $\mathbf{w} = [u, v, w]^T$  is projected to a 2D point  $\mathbf{x} = [x, y]^T$  by the relations

$$\begin{aligned} x &= \frac{\phi_x(\omega_{11}u + \omega_{12}v + \omega_{13}w + \tau_x) + \gamma(\omega_{21}u + \omega_{22}v + \omega_{23}w + \tau_y)}{\omega_{31}u + \omega_{32}v + \omega_{33}w + \tau_z} + \delta_x \\ y &= \frac{\phi_y(\omega_{21}u + \omega_{22}v + \omega_{23}w + \tau_y)}{\omega_{31}u + \omega_{32}v + \omega_{33}w + \tau_z} + \delta_y. \end{aligned} \quad (14.8)$$

There are two sets of parameters in this model. The *intrinsic* or *camera parameters*  $\{\phi_x, \phi_y, \gamma, \delta_x, \delta_y\}$  describe the camera itself, and the *extrinsic parameters*  $\{\boldsymbol{\Omega}, \boldsymbol{\tau}\}$  describe the position and orientation of the camera in the world. For reasons that will become clear in section 14.3.1, we will store the intrinsic parameters in the *intrinsic matrix*  $\boldsymbol{\Lambda}$  where

$$\boldsymbol{\Lambda} = \begin{bmatrix} \phi_x & \gamma & \delta_x \\ 0 & \phi_y & \delta_y \\ 0 & 0 & 1 \end{bmatrix}. \quad (14.9)$$

We can now abbreviate the full projection model (equations 14.8) by just writing

$$\mathbf{x} = \text{pinhole}[\mathbf{w}, \boldsymbol{\Lambda}, \boldsymbol{\Omega}, \boldsymbol{\tau}]. \quad (14.10)$$

Finally, we must account for the fact that the estimated position of a feature in the image may differ from our predictions. There are a number of reasons for this, including noise in the sensor, sampling issues, and the fact that the detected position in the image may change at different viewpoints. We model these factors with additive noise that is normally distributed with a spherical covariance to give the final relation

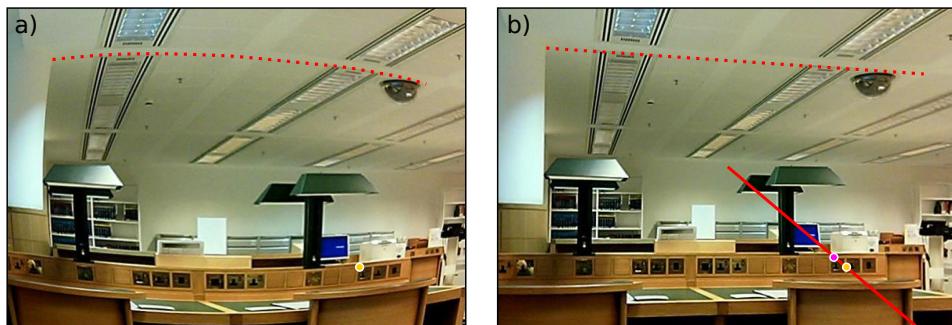
$$Pr(\mathbf{x}|\mathbf{w}, \boldsymbol{\Lambda}, \boldsymbol{\Omega}, \boldsymbol{\tau}) = \text{Norm}_{\mathbf{x}} [\text{pinhole}[\mathbf{w}, \boldsymbol{\Lambda}, \boldsymbol{\Omega}, \boldsymbol{\tau}], \sigma^2 \mathbf{I}], \quad (14.11)$$

where  $\sigma^2$  is the variance of the noise.

Note that the pinhole camera is a *generative model*. We are describing the likelihood  $Pr(\mathbf{x}|\mathbf{w}, \boldsymbol{\Lambda}, \boldsymbol{\Omega}, \boldsymbol{\tau})$  of observing a 2D image point  $\mathbf{x}$  given a 3D world point  $\mathbf{w}$  and the parameters  $\{\boldsymbol{\Lambda}, \boldsymbol{\Omega}, \boldsymbol{\tau}\}$ .

### 14.1.6 Radial distortion

In the previous section, we introduced the pinhole camera model. However, it has probably not escaped your attention that real-world cameras are rarely based on the pinhole: they have a lens (or possibly a system of several lenses) that collects light from a larger area and re-focuses it on the image plane. In practice, this leads to a number of deviations from the pinhole model. For example, some parts of the image may be out of focus, which essentially means that the assumption that a point in the world  $\mathbf{w}$  maps to a single point in the image  $\mathbf{x}$  is no longer valid. There are more complex mathematical models for cameras that deal effectively with this situation, but they are not discussed here.



**Figure 14.6** Radial distortion. The pinhole model is only an approximation of the true imaging process. One important deviation from this model is a 2D warping in which points deviate from their expected positions by moving along radial lines from the center of the image by an amount that depends on the distance from the center. This is known as radial distortion. a) An image that suffers from radial distortion is easily spotted because lines that were straight in the world are mapped to curves in the image (e.g., red dotted line). b) After applying the inverse radial distortion model, straight lines in the world now correctly map to straight lines in the image. The distortion caused the magenta point to move along the red radial line to the position of the yellow point.

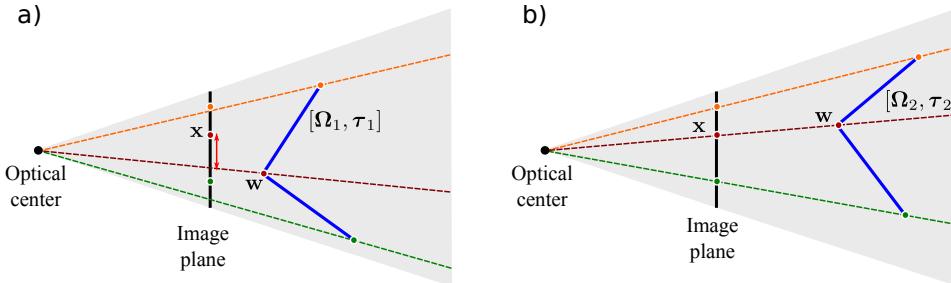
However, there is one deviation from the pinhole model that must be addressed. *Radial distortion* is a nonlinear warping of the image that depends on the distance from the center of the image. In practice, this occurs when the field of view of the lens system is large. It can easily be detected in an image, because straight lines in the world no longer project to straight lines in the image (figure 14.6).

Radial distortion is commonly modeled as a polynomial function of the distance  $r$  from the center of the image. In the normalized camera, the final image positions  $(x', y')$  are expressed as functions of the original positions  $(x, y)$  by

$$\begin{aligned} x' &= x(1 + \beta_1 r^2 + \beta_2 r^4) \\ y' &= y(1 + \beta_1 r^2 + \beta_2 r^4), \end{aligned} \quad (14.12)$$

where the parameters  $\beta_1$  and  $\beta_2$  control the degree of distortion. These relations describe a family of possible distortions that approximate the true distortion closely for most common lenses.

This distortion is implemented after perspective projection (division by  $w$ ) but before the effect of the intrinsic parameters (focal length, offset, etc.), so the warping is relative to the optical axis and not the origin of the pixel coordinate system. We will not discuss radial distortion further in this volume. However, it is important to realize that, for accurate results, all of the algorithms in this and chapters 15 and 16 should account for radial distortion. When the field of view is large, it is particularly critical to incorporate this into the pinhole camera model.



**Figure 14.7** Problem 1 - Learning extrinsic parameters (exterior orientation problem). Given points  $\{\mathbf{w}_i\}_{i=1}^I$  on a known object (blue lines), their positions  $\{\mathbf{x}_i\}_{i=1}^I$  in the image (circles on image plane), and known intrinsic parameters  $\Lambda$ , find the rotation  $\Omega$  and translation  $\tau$  relating the camera and the object. a) When the rotation or translation are wrong, the image points predicted by the model (where the rays strike the image plane) do not agree well with the observed points  $\mathbf{x}_i$ . b) When the rotation and translation are correct, they agree well and the likelihood  $Pr(\mathbf{x}_i|\mathbf{w}, \Lambda, \Omega, \tau)$  will be high.

## 14.2 Three geometric problems

Now that we have described the pinhole camera model, we will consider three important geometric problems. Each is an instance of learning or inference within this model. We will first describe the problems themselves, and then tackle them one by one later in the chapter.

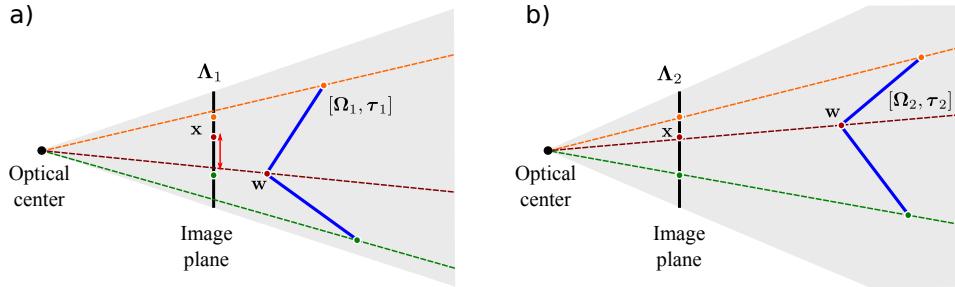
### 14.2.1 Problem 1: Learning extrinsic parameters

We aim to recover the position and orientation of the camera relative to a known scene. This is sometimes known as the *perspective-n-point* (*PnP*) problem or the *exterior orientation* problem. One common application is augmented reality, where we need to know this relationship to render virtual objects that appear to be stable parts of the real scene.

The problem can be stated more formally as follows: we are given a known object, with  $I$  distinct 3D points  $\{\mathbf{w}_i\}_{i=1}^I$ , their corresponding projections in the image  $\{\mathbf{x}_i\}_{i=1}^I$ , and known intrinsic parameters  $\Lambda$ . Our goal is to estimate the rotation  $\Omega$  and translation  $\tau$  that map points in the coordinate system of the object to points in the coordinate system of the camera so that

$$\hat{\Omega}, \hat{\tau} = \underset{\Omega, \tau}{\operatorname{argmax}} \left[ \sum_{i=1}^I \log [Pr(\mathbf{x}_i|\mathbf{w}_i, \Lambda, \Omega, \tau)] \right]. \quad (14.13)$$

This is a maximum likelihood learning problem, in which we aim to find parameters  $\Omega, \tau$  that make the predictions  $\text{pinhole}[\mathbf{w}_i, \Lambda, \Omega, \tau]$  of the model agree with the observed 2D points  $\mathbf{x}_i$  (figure 14.7).



**Figure 14.8** Problem 2 - Learning intrinsic parameters. Given a set of points  $\{\mathbf{w}_i\}_{i=1}^I$  on a known object in the world (blue lines) and the 2D positions  $\{\mathbf{x}_i\}_{i=1}^I$  of these points in an image, find the intrinsic parameters  $\Lambda$ . To do this, we must also simultaneously estimate the extrinsic parameters  $\Omega, \tau$ . a) When the intrinsic or extrinsic parameters are wrong, the prediction of the pinhole camera (where rays strike the image plane) will deviate significantly from the observed 2D points. b) When the intrinsic and extrinsic parameters are correct, the prediction of the model will agree with the observed image.

### 14.2.2 Problem 2: Learning intrinsic parameters

We aim to estimate the intrinsic parameters  $\Lambda$  that relate the direction of rays through the optical center to coordinates on the image plane. This estimation process is known as *calibration*. Knowledge of the intrinsic parameters is critical if we want to use the camera to build 3D models of the world.

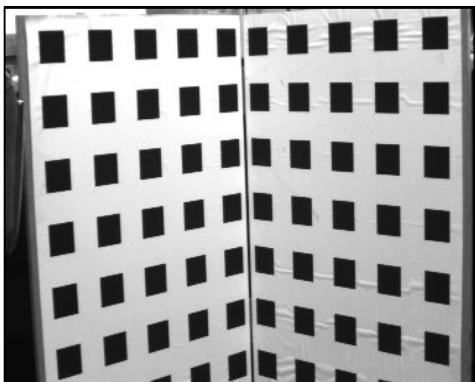
The calibration problem can be stated more formally as follows: given a known 3D object, with  $I$  distinct 3D points  $\{\mathbf{w}_i\}_{i=1}^I$  and their corresponding projections in the image  $\{\mathbf{x}_i\}_{i=1}^I$ , estimate the intrinsic parameters:

$$\hat{\Lambda} = \underset{\Lambda}{\operatorname{argmax}} \left[ \max_{\Omega, \tau} \left[ \sum_{i=1}^I \log [Pr(\mathbf{x}_i | \mathbf{w}_i, \Lambda, \Omega, \tau)] \right] \right]. \quad (14.14)$$

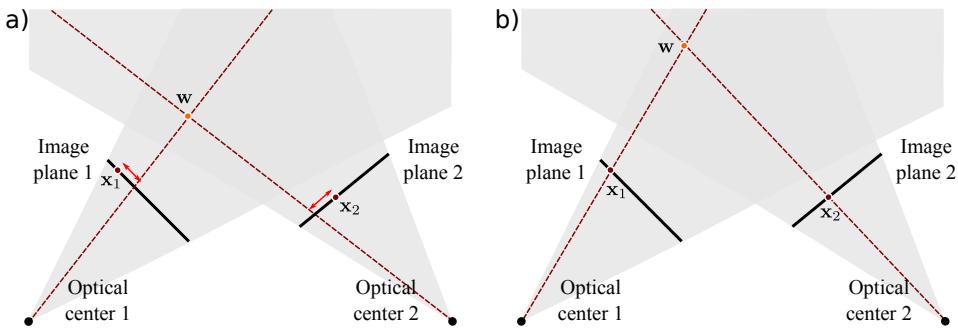
Once more, this is a maximum likelihood learning problem in which we aim to find parameters  $\Lambda, \Omega, \tau$  that make the predictions of the model **pinhole**[ $\mathbf{w}_i, \Lambda, \Omega, \tau$ ] agree with the observed 2D points  $\mathbf{x}_i$  (figure 14.8). We do not particularly care about the extrinsic parameters  $\Omega, \tau$ ; finding these is just a means to the end of estimating the intrinsic parameters  $\Lambda$ .

The calibration process requires a known 3D object, on which distinct points can be identified, and their projections in the image found. A common approach is to construct a bespoke 3D *calibration target*<sup>2</sup> that achieves these goals (figure 14.9).

<sup>2</sup>It should be noted that, in practice, calibration is more usually based on a number of views of a known 2D planar object (see section 15.4.2).



**Figure 14.9** Camera calibration target. One way to calibrate the camera (estimate its intrinsic parameters) is to view a 3D object (a camera calibration target) for which the geometry is known. The marks on the surface are at known 3D positions in the frame of reference of the object, and are easy to locate in the image using basic image-processing techniques. It is now possible to find the intrinsic and extrinsic parameters that optimally map the known 3D positions to their 2D projections in the image. Image from Hartley & Zisserman (2004).



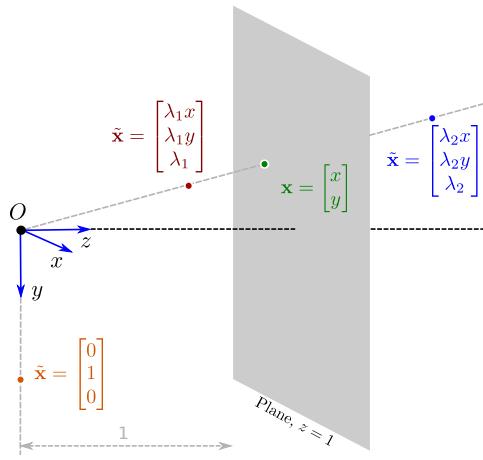
**Figure 14.10** Problem 3 - Inferring 3D world points. Given two cameras with known position and orientation, and the projections  $\mathbf{x}_1$  and  $\mathbf{x}_2$  of the same 3D point in each image, the goal of calibrated stereo reconstruction is to infer the 3D position  $\mathbf{w}$  of the world point. a) When the estimate of the world point (red circle) is wrong, the predictions of the pinhole camera models (where rays strike the image plane) will deviate from the observed data (brown circles on image plane). b) When the estimate of  $\mathbf{w}$  is correct, the predictions of the model agree with the observed data.

### 14.2.3 Problem 3: Inferring 3D world points

We aim to estimate the 3D position of a point  $\mathbf{w}$  in the scene, given its projections  $\{\mathbf{x}_j\}_{j=1}^J$  in  $J \geq 2$  calibrated cameras. When  $J = 2$ , this is known as *calibrated stereo reconstruction*. With  $J > 2$  calibrated cameras, it is known to as *multi-view reconstruction*. If we repeat this process for many points, the result is a sparse 3D point cloud. This could be used to help an autonomous vehicle navigate through the environment, or to generate an image of the scene from a new viewpoint.

More formally, the multi-view reconstruction problem can be stated as follows: given  $J$  calibrated cameras in known positions (i.e., cameras with known  $\mathbf{A}, \mathbf{\Omega}, \boldsymbol{\tau}$ ) viewing the same 3D point  $\mathbf{w}$  and knowing the corresponding 2D projections  $\{\mathbf{x}_j\}_{j=1}^J$  in the  $J$  images , establish the 3D position  $\mathbf{w}$  of the point in the

**Figure 14.11** Geometric interpretation of homogeneous coordinates. The different scalar multiples  $\lambda$  of the homogeneous 3-vector  $\tilde{\mathbf{x}}$  define a ray through the origin of a coordinate space. The corresponding 2D image point  $\mathbf{x}$  can be found by considering the 2D point that this ray strikes on the plane at  $z = 1$ . An interesting side-effect of this representation is that it is possible to represent points at infinity (known as *ideal points*). For example, the homogeneous coordinate  $[0, 1, 0]^T$  defines a ray that is parallel to  $z = 1$  and so never intersects the plane. It represents the point at infinity in direction  $[0, 1]^T$ .



world:

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}} \left[ \sum_{j=1}^J \log[Pr(\mathbf{x}_j | \mathbf{w}, \Lambda_j, \Omega_j, \tau_j)] \right]. \quad (14.15)$$

The form of this inference problem is similar to that of the preceding learning problems: we perform an optimization in which we manipulate the variable of interest  $\mathbf{w}$  until the predictions  $\text{pinhole}[\mathbf{w}, \Lambda_j, \Omega_j, \tau_j]$  of the pinhole camera models agree with the data  $\mathbf{x}_j$  (figure 14.10). For obvious reasons, the principle behind reconstruction is known as *triangulation*.

#### 14.2.4 Solving the problems

We have introduced three geometric problems, each of which took the form of a learning or inference problem using the pinhole camera model. We formulated each in terms of maximum likelihood estimation, and in each case this results in an optimization problem.

Unfortunately, none of the resulting objective functions can be optimized in closed form; each solution requires the use of nonlinear optimization. In each case, it is critical to have a good initial estimate of the unknown quantities to ensure that the optimization process converges to the global maximum. In the remaining part of this chapter we develop algorithms that provide these initial estimates. The general approach is to choose new objective functions that can be optimized in closed form, and where the solution is close to the solution of the true problem.

## 14.3 Homogeneous coordinates

To get good initial estimates of the geometric quantities in the preceding optimization problems, we play a simple trick: we change the representation of both the 2D image points and 3D world points so that the projection equations become linear. After this change, it is possible to find solutions for the unknown quantities in closed form. However, it should be emphasized that these solutions *do not* directly address the original optimization criteria: they minimize more abstract objective functions based on *algebraic error* whose solutions are not guaranteed to be the same as those for the original problem. However, they are generally close enough to provide a good starting point for a nonlinear optimization of the true cost function.

We convert the original Cartesian representation of the 2D image points  $\mathbf{x}$  to a 3D *homogeneous* coordinate  $\tilde{\mathbf{x}}$  so that

$$\tilde{\mathbf{x}} = \lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad (14.16)$$

Problem 14.4  
Problem 14.5  
Problem 14.6

where  $\lambda$  is an arbitrary scaling factor. This is a redundant representation in that any scalar multiple  $\lambda$  represents the same 2D point. For example, the homogeneous vectors  $\tilde{\mathbf{x}} = [2, 4, 2]^T$  and  $\tilde{\mathbf{x}} = [3, 6, 3]^T$  both represent the Cartesian 2D point  $\mathbf{x} = [1, 2]^T$ , where scaling factors  $\lambda = 2$  and  $\lambda = 3$  have been used, respectively.

Converting between homogeneous and Cartesian coordinates is easy. To move to homogeneous coordinates, we choose  $\lambda = 1$  and simply append a 1 to the original 2D Cartesian coordinate. To recover the Cartesian coordinates, we divide the first two entries of the homogeneous 3-vector by the third, so that if we observe the homogeneous vector  $\tilde{\mathbf{x}} = [\tilde{x}, \tilde{y}, \tilde{z}]^T$ , then we can recover the Cartesian coordinate  $\mathbf{x} = [x, y]^T$  as

$$\begin{aligned} x &= \frac{\tilde{x}}{\tilde{z}} \\ y &= \frac{\tilde{y}}{\tilde{z}}. \end{aligned} \quad (14.17)$$

Further insight into the relationship between the two representations is given in figure 14.11.

It is similarly possible to represent the 3D world point  $\mathbf{w}$  as a homogenous 4D vector  $\tilde{\mathbf{w}}$  so that

$$\tilde{\mathbf{w}} = \lambda \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}, \quad (14.18)$$

where  $\lambda$  is again an arbitrary scaling factor. Once more, the conversion from Cartesian to homogeneous coordinates can be achieved by appending a 1 to the original 3D vector  $\mathbf{w}$ . The conversion from homogeneous to Cartesian coordinates is achieved by dividing each of the first three entries by the last.

### 14.3.1 Camera model in homogeneous coordinates

It is hard to see the point of converting the 2D image points to homogeneous 3-vectors and converting the 3D world point to homogeneous 4-vectors until we re-examine the pinhole projection equations,

$$\begin{aligned} x &= \frac{\phi_x u + \gamma v}{w} + \delta_x \\ y &= \frac{\phi_y v}{w} + \delta_y, \end{aligned} \quad (14.19)$$

where we have temporarily assumed that the world point  $\mathbf{w} = [u, v, w]^T$  is in the same coordinate system as the camera.

In homogeneous coordinates, these relationships can be expressed as a set of linear equations

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_x & \gamma & \delta_x & 0 \\ 0 & \phi_y & \delta_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}. \quad (14.20)$$

To convince ourselves of this, let us write these relations explicitly:

$$\begin{aligned} \lambda x &= \phi_x u + \gamma v + \delta_x w \\ \lambda y &= \phi_y v + \delta_y w \\ \lambda &= w. \end{aligned} \quad (14.21)$$

We solve for  $x$  and  $y$  by converting back to Cartesian coordinates: we divide the first two relations by the third to yield the original pinhole model (equation 14.19).

Let us summarize what has happened: the original mapping from 3D Cartesian world points to 2D Cartesian image points is nonlinear (due to the division by  $w$ ). However, the mapping from 4D homogeneous world points to 3D homogeneous image points is linear. In the homogeneous representation, the nonlinear component of the projection process (division by  $w$ ) has been side-stepped: this operation still occurs, but it is in the final conversion back to 2D Cartesian coordinates, and thus does not trouble the homogeneous camera equations.

To complete the model, we add the extrinsic parameters  $\{\boldsymbol{\Omega}, \boldsymbol{\tau}\}$  that relate the world coordinate system and the camera coordinate system, so that

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_x & \gamma & \delta_x & 0 \\ 0 & \phi_y & \delta_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} & \tau_x \\ \omega_{21} & \omega_{22} & \omega_{23} & \tau_y \\ \omega_{31} & \omega_{32} & \omega_{33} & \tau_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}, \quad (14.22)$$

or in matrix form

$$\lambda \tilde{\mathbf{x}} = [\mathbf{\Lambda} \quad \mathbf{0}] \begin{bmatrix} \boldsymbol{\Omega} & \boldsymbol{\tau} \\ \mathbf{0}^T & 1 \end{bmatrix} \tilde{\mathbf{w}}, \quad (14.23)$$

where  $\mathbf{0} = [0, 0, 0]^T$ . The same relations can be simplified to

$$\lambda \tilde{\mathbf{x}} = \boldsymbol{\Lambda} [\boldsymbol{\Omega} \quad \boldsymbol{\tau}] \tilde{\mathbf{w}}. \quad (14.24)$$

In the next three sections, we revisit the three geometric problems introduced in section 14.2. In each case, we will use algorithms based on homogeneous coordinates to compute good initial estimates of the variable of interest. These estimates can then be improved using nonlinear optimization.

## 14.4 Learning extrinsic parameters

Given a known object, with  $I$  distinct 3D points  $\{\mathbf{w}_i\}_{i=1}^I$ , their corresponding projections in the image  $\{\mathbf{x}_i\}_{i=1}^I$  and known intrinsic parameters  $\boldsymbol{\Lambda}$ , estimate the geometric relationship between the camera and the object determined by the rotation  $\boldsymbol{\Omega}$  and the translation  $\boldsymbol{\tau}$ ,

$$\hat{\boldsymbol{\Omega}}, \hat{\boldsymbol{\tau}} = \underset{\boldsymbol{\Omega}, \boldsymbol{\tau}}{\operatorname{argmax}} \left[ \sum_{i=1}^I \log [Pr(\mathbf{x}_i | \mathbf{w}_i, \boldsymbol{\Lambda}, \boldsymbol{\Omega}, \boldsymbol{\tau})] \right]. \quad (14.25)$$

Algorithm 14.1

This is a non-convex problem, so we make progress by expressing it in homogeneous coordinates. The relationship between the  $i^{th}$  homogeneous world point  $\tilde{\mathbf{w}}_i$  and the  $i^{th}$  corresponding homogeneous image point  $\tilde{\mathbf{x}}_i$  is

$$\lambda_i \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_x & \gamma & \delta_x \\ 0 & \phi_y & \delta_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} & \tau_x \\ \omega_{21} & \omega_{22} & \omega_{23} & \tau_y \\ \omega_{31} & \omega_{32} & \omega_{33} & \tau_z \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ w_i \\ 1 \end{bmatrix}. \quad (14.26)$$

We would like to discard the effect of the (known) intrinsic parameters  $\boldsymbol{\Lambda}$ . To this end, we pre-multiply both sides of the equation by the inverse of the intrinsic matrix  $\boldsymbol{\Lambda}$  to yield

$$\lambda_i \begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} = \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} & \tau_x \\ \omega_{21} & \omega_{22} & \omega_{23} & \tau_y \\ \omega_{31} & \omega_{32} & \omega_{33} & \tau_z \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ w_i \\ 1 \end{bmatrix}. \quad (14.27)$$

The transformed coordinates  $\tilde{\mathbf{x}}' = \boldsymbol{\Lambda}^{-1} \tilde{\mathbf{x}}$  are known as *normalized image coordinates*: they are the coordinates that would have resulted if we had used a normalized camera. In effect, pre-multiplying by  $\boldsymbol{\Lambda}^{-1}$  compensates for the idiosyncrasies of this particular camera.

We now note that the last of these three equations allows us to solve for the constant  $\lambda_i$ , so that

$$\lambda_i = \omega_{31}u_i + \omega_{32}v_i + \omega_{33}w_i + \tau_z, \quad (14.28)$$

and we can now substitute this back into the first two equations to get the relations

$$\begin{bmatrix} (\omega_{31}u_i + \omega_{32}v_i + \omega_{33}w_i + \tau_z)x'_i \\ (\omega_{31}u_i + \omega_{32}v_i + \omega_{33}w_i + \tau_z)y'_i \end{bmatrix} = \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} & \tau_x \\ \omega_{21} & \omega_{22} & \omega_{23} & \tau_y \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ w_i \\ 1 \end{bmatrix}. \quad (14.29)$$

These are two linear equations with respect to the unknown quantities  $\Omega$  and  $\tau$ . We can take the two equations provided by each of the  $I$  pairs of points in the world  $\mathbf{w}$  and the image  $\mathbf{x}$  to form the system of equations

$$\begin{bmatrix} u_1 & v_1 & w_1 & 1 & 0 & 0 & 0 & -u_1x'_1 & -v_1x'_1 & -w_1x'_1 & -x'_1 \\ 0 & 0 & 0 & 0 & u_1 & v_1 & w_1 & 1 & -u_1y'_1 & -v_1y'_1 & -w_1y'_1 & -y'_1 \\ u_2 & v_2 & w_2 & 1 & 0 & 0 & 0 & -u_2x'_2 & -v_2x'_2 & -w_2x'_2 & -x'_2 \\ 0 & 0 & 0 & 0 & u_2 & v_2 & w_2 & 1 & -u_2y'_2 & -v_2y'_2 & -w_2y'_2 & -y'_2 \\ \vdots & \vdots \\ u_I & v_I & w_I & 1 & 0 & 0 & 0 & -u_Ix'_I & -v_Ix'_I & -w_Ix'_I & -x'_I \\ 0 & 0 & 0 & 0 & u_I & v_I & w_I & 1 & -u_Iy'_I & -v_Iy'_I & -w_Iy'_I & -y'_I \end{bmatrix} = \begin{bmatrix} \omega_{11} \\ \omega_{12} \\ \omega_{13} \\ \tau_x \\ \omega_{21} \\ \omega_{22} \\ \omega_{23} \\ \tau_y \\ \omega_{31} \\ \omega_{32} \\ \omega_{33} \\ \tau_z \end{bmatrix} = \mathbf{0}. \quad (14.30)$$

This problem is now in the standard form  $\mathbf{Ab} = \mathbf{0}$  of a *minimum direction problem*. We seek the value of  $\mathbf{b}$  that minimizes  $|\mathbf{Ab}|^2$  subject to the constraint  $|\mathbf{b}| = 1$  (to avoid the uninteresting solution  $\mathbf{b} = \mathbf{0}$ ). The solution can be found by computing the singular value decomposition  $\mathbf{A} = \mathbf{ULV}^T$  and setting  $\hat{\mathbf{b}}$  to be the last column of  $\mathbf{V}$  (see appendix C.7.2).

The estimates of  $\Omega$  and  $\tau$  that we extract from  $\mathbf{b}$  have had an arbitrary scale imposed on them, and we must find the correct scaling factor. This is possible because the rotation  $\Omega$  has a pre-defined scale (its rows and columns must all have norm one). In practice, we first find the closest true rotation matrix to  $\Omega$  which also forces our estimate to be a valid orthogonal matrix. This is an instance of the *orthogonal Procrustes problem* (appendix C.7.3). The solution is found by computing the singular value decomposition  $\Omega = \mathbf{ULV}^T$  and setting  $\hat{\Omega} = \mathbf{UV}^T$ . Now, we re-scale the translation  $\tau$ . The scaling factor can be estimated by taking the average ratio of the nine entries of our initial estimate of  $\Omega$  to the final one  $\hat{\Omega}$  so that

$$\hat{\tau} = \sum_{m=1}^3 \sum_{n=1}^3 \frac{\hat{\Omega}_{mn}}{\Omega_{mn}} \tau. \quad (14.31)$$

Finally, we must check that the sign of  $\tau_z$  is positive, indicating that the object is in front of the camera. If this is not the case then we multiply both  $\hat{\tau}$  and  $\hat{\omega}$  by minus one.

This scrappy algorithm is typical of methods that use homogeneous coordinates. The resulting estimates  $\hat{\boldsymbol{\tau}}$  and  $\hat{\boldsymbol{\Omega}}$  can be quite inaccurate in the presence of noise in the measured image positions. However, they usually suffice as a reasonable starting point for the subsequent nonlinear optimization of the true objective function (equation 14.25) for this problem. That optimization must be carried out while ensuring that  $\boldsymbol{\Omega}$  remains a valid rotation matrix (see appendix B.4).

Note that this algorithm requires a minimum of 11 equations to solve the minimum direction problem. Since each point contributes two equations, this means we require  $I = 6$  points for a unique solution. However, there are only really six unknowns (the rotation and translation in 3D) and so a minimal solution would require only  $I = 3$  points. Minimal solutions for this problem have been developed and are discussed in the notes at the end of the chapter.

## 14.5 Learning intrinsic parameters

We now address the second problem. In *camera calibration* we attempt to learn the intrinsic parameters based on viewing a known object or *calibration target*. More precisely, we are given a known object, with  $I$  distinct 3D points  $\{\mathbf{w}_i\}_{i=1}^I$  and their corresponding 2D projections in the image  $\{\mathbf{x}_i\}_{i=1}^I$ , and aim to form maximum likelihood estimates of the intrinsic parameters  $\boldsymbol{\Lambda}$ ,

$$\hat{\boldsymbol{\Lambda}} = \underset{\boldsymbol{\Lambda}}{\operatorname{argmax}} \left[ \underset{\boldsymbol{\Omega}, \boldsymbol{\tau}}{\max} \left[ \sum_{i=1}^I \log [Pr(\mathbf{x}_i | \mathbf{w}_i, \boldsymbol{\Lambda}, \boldsymbol{\Omega}, \boldsymbol{\tau})] \right] \right]. \quad (14.32)$$

A simple (but inefficient) approach to this problem is to use a coordinate ascent method in which we alternately

- estimate the extrinsic parameters for fixed intrinsic parameters (problem 1),

$$\hat{\boldsymbol{\Omega}}, \hat{\boldsymbol{\tau}} = \underset{\boldsymbol{\Omega}, \boldsymbol{\tau}}{\operatorname{argmax}} \left[ \sum_{i=1}^I \log [Pr(\mathbf{x}_i | \mathbf{w}_i, \boldsymbol{\Lambda}, \boldsymbol{\Omega}, \boldsymbol{\tau})] \right], \quad (14.33)$$

using the procedure described in the section 14.4 and then

- estimate the intrinsic parameters for fixed extrinsic parameters,

$$\hat{\boldsymbol{\Lambda}} = \underset{\boldsymbol{\Lambda}}{\operatorname{argmax}} \left[ \sum_{i=1}^I \log [Pr(\mathbf{x}_i | \mathbf{w}_i, \boldsymbol{\Lambda}, \boldsymbol{\Omega}, \boldsymbol{\tau})] \right]. \quad (14.34)$$

By iterating these two steps, we will get closer and closer to the correct solution. Since we already know how to solve the first of these two subproblems, we now concentrate on solving the second. Happily there is a closed form solution that does not even require homogeneous coordinates.

Given known world points  $\{\mathbf{w}_i\}_{i=1}^I$ , their projections  $\{\mathbf{x}_i\}_{i=1}^I$ , and known extrinsic parameters  $\{\boldsymbol{\Omega}, \boldsymbol{\tau}\}$ , our goal is now to compute the intrinsic matrix  $\boldsymbol{\Lambda}$ , which

Algorithm 14.2

contains the intrinsic parameters  $\{\phi_x, \phi_y, \gamma, \delta_x, \delta_y\}$ . We will apply the maximum likelihood method

$$\begin{aligned}\hat{\Lambda} &= \underset{\Lambda}{\operatorname{argmax}} \left[ \sum_{i=1}^I \log [\operatorname{Norm}_{\mathbf{x}_i} [\operatorname{pinhole}[\mathbf{w}_i, \Lambda, \Omega, \tau], \sigma^2 \mathbf{I}]] \right] \\ &= \underset{\Lambda}{\operatorname{argmin}} \left[ \sum_{i=1}^I (\mathbf{x}_i - \operatorname{pinhole}[\mathbf{w}_i, \Lambda, \Omega, \tau])^T (\mathbf{x}_i - \operatorname{pinhole}[\mathbf{w}_i, \Lambda, \Omega, \tau]) \right],\end{aligned}\quad (14.35)$$

which results in a least squares problem (see section 4.4.1).

Now we note that the projection function  $\operatorname{pinhole}[\bullet, \bullet, \bullet, \bullet]$  (equation 14.8) is linear with respect to the intrinsic parameters, and can be written as  $\mathbf{A}_i \mathbf{h}$  where

$$\mathbf{A}_i = \begin{bmatrix} \frac{\omega_{11} u_i + \omega_{12} v_i + \omega_{13} w_i + \tau_x}{\omega_{31} u_i + \omega_{32} v_i + \omega_{33} w_i + \tau_z} & \frac{\omega_{21} u_i + \omega_{22} v_i + \omega_{23} w_i + \tau_x}{\omega_{31} u_i + \omega_{32} v_i + \omega_{33} w_i + \tau_z} & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{\omega_{21} u_i + \omega_{22} v_i + \omega_{23} w_i + \tau_y}{\omega_{31} u_i + \omega_{32} v_i + \omega_{33} w_i + \tau_z} & 1 \end{bmatrix} \quad (14.36)$$

and  $\mathbf{h} = [\phi_x, \gamma, \delta_x, \phi_y, \delta_y]^T$ . Consequently, the problem has the form

$$\hat{\mathbf{h}} = \underset{\mathbf{h}}{\operatorname{argmin}} \left[ \sum_{i=1}^I (\mathbf{A}_i \mathbf{h} - \mathbf{x}_i)^T (\mathbf{A}_i \mathbf{h} - \mathbf{x}_i) \right], \quad (14.37)$$

which we recognize as a least squares problem that can be solved in closed form (appendix C.7.1).

We have described this alternating approach for pedagogical reasons; it is simple to understand and implement. However, we emphasize that this is not really a practical method as the convergence will be very slow. A better approach would be to perform a couple of iterations of this method and then optimize both the intrinsic and extrinsic parameters simultaneously using a nonlinear optimization technique such as the Gauss Newton method (appendix B.2.3) with the original criterion (equation 14.32). This optimization must be done while ensuring that the extrinsic parameter  $\Omega$  remains a valid rotation matrix (see appendix B.4).

## 14.6 Inferring 3D world points

Algorithm 14.3

Finally, we consider the multi-view reconstruction problem. Given  $J$  calibrated cameras in known positions (i.e., cameras with known  $\Lambda, \Omega, \tau$ ), viewing the same 3D point  $\mathbf{w}$  and knowing the corresponding projections in the images  $\{\mathbf{x}_j\}_{j=1}^J$ , establish the position of the point in the world.

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}} \left[ \sum_{j=1}^J \log [Pr(\mathbf{x}_j | \mathbf{w}, \Lambda_j, \Omega_j, \tau_j)] \right]. \quad (14.38)$$

This cannot be solved in closed form and so we move to homogeneous coordinates where we can solve for a good initial estimate in closed form. The relationship

between the homogeneous world point  $\tilde{\mathbf{w}}$  and the  $j^{th}$  corresponding homogeneous image point  $\tilde{\mathbf{x}}_j$  is

$$\lambda_j \begin{bmatrix} x_j \\ y_j \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_{xj} & \gamma_j & \delta_{xj} \\ 0 & \phi_{yj} & \delta_{yj} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_{11j} & \omega_{12j} & \omega_{13j} & \tau_{xj} \\ \omega_{21j} & \omega_{22j} & \omega_{23j} & \tau_{yj} \\ \omega_{31j} & \omega_{32j} & \omega_{33j} & \tau_{zj} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}, \quad (14.39)$$

where we have appended the index  $j$  to the intrinsic and extrinsic parameters to denote the fact that they belong to the  $j^{th}$  camera. Pre-multiplying both sides by the intrinsic matrix  $\Lambda_j^{-1}$  to convert to normalized image coordinates gives

$$\lambda_j \begin{bmatrix} x'_j \\ y'_j \\ 1 \end{bmatrix} = \begin{bmatrix} \omega_{11j} & \omega_{12j} & \omega_{13j} & \tau_{xj} \\ \omega_{21j} & \omega_{22j} & \omega_{23j} & \tau_{yj} \\ \omega_{31j} & \omega_{32j} & \omega_{33j} & \tau_{zj} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}, \quad (14.40)$$

where  $x'_j$  and  $y'_j$  denote the normalized image coordinates in the  $j^{th}$  camera.

We use the third equation to establish that  $\lambda_j = \omega_{31j}u + \omega_{32j}v + \omega_{33j}w + \tau_{zj}$ . Substituting into the first two equations, we get

$$\begin{bmatrix} (\omega_{31j}u + \omega_{32j}v + \omega_{33j}w + \tau_{zj})x'_j \\ (\omega_{31j}u + \omega_{32j}v + \omega_{33j}w + \tau_{zj})y'_j \end{bmatrix} = \begin{bmatrix} \omega_{11j} & \omega_{12j} & \omega_{13j} & \tau_{xj} \\ \omega_{21j} & \omega_{22j} & \omega_{23j} & \tau_{yj} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}. \quad (14.41)$$

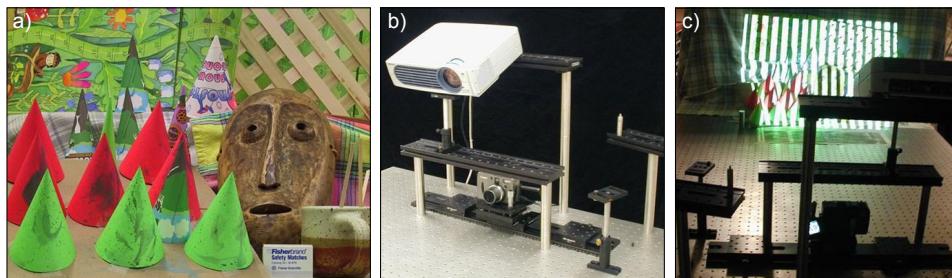
These equations can be re-arranged to provide two linear constraints on the three unknown quantities in  $\mathbf{w} = [u, v, w]^T$ :

$$\begin{bmatrix} \omega_{31j}x'_j - \omega_{11j} & \omega_{32j}x'_j - \omega_{12j} & \omega_{33j}x'_j - \omega_{13j} \\ \omega_{31j}y'_j - \omega_{21j} & \omega_{32j}y'_j - \omega_{22j} & \omega_{33j}y'_j - \omega_{23j} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \tau_{xj} - \tau_{zj}x'_j \\ \tau_{yj} - \tau_{zj}y'_j \end{bmatrix}. \quad (14.42)$$

With multiple cameras, we can build a larger system of equations and solve for  $\mathbf{w}$  in a least squares sense (appendix C.7.1). This typically provides a good starting point for the subsequent nonlinear optimization of the criterion in equation 14.38.

This calibrated reconstruction algorithm is the basis for methods that construct 3D models. However, there are several parts missing from the argument.

- The method requires us to have found the points  $\{\mathbf{x}_j\}_{j=1}^J$  that correspond to the same world point  $\mathbf{w}$  in each of the  $J$  images. This process is called *correspondence* and is discussed in chapters 15 and 16.
- The method requires the intrinsic and extrinsic parameters. Of course, these could be computed from a calibration target using the method of section 14.5. However, it is still possible to perform reconstruction when the system is uncalibrated; this is known as projective reconstruction as the result is ambiguous up to a 3D projective transformation. Furthermore, if a single camera



**Figure 14.12** Depth maps from structured light. a) A three-dimensional scene that we wish to capture. b) The capture hardware consists of a projector and a camera, which both view the scene from different positions. c) The projector is used to illuminate the scene and the camera records the pattern of illumination from its viewpoint. The resulting images contain information that can be used to compute a 3D reconstruction. Adapted from Scharstein & Szeliski (2003). ©2003 IEEE.

was used to take all of the images, it is possible to estimate the single intrinsic matrix and extrinsic parameters from a sequence, and reconstruct points in a scene up to a constant scaling factor. Chapter 16 presents an extended discussion of this method.

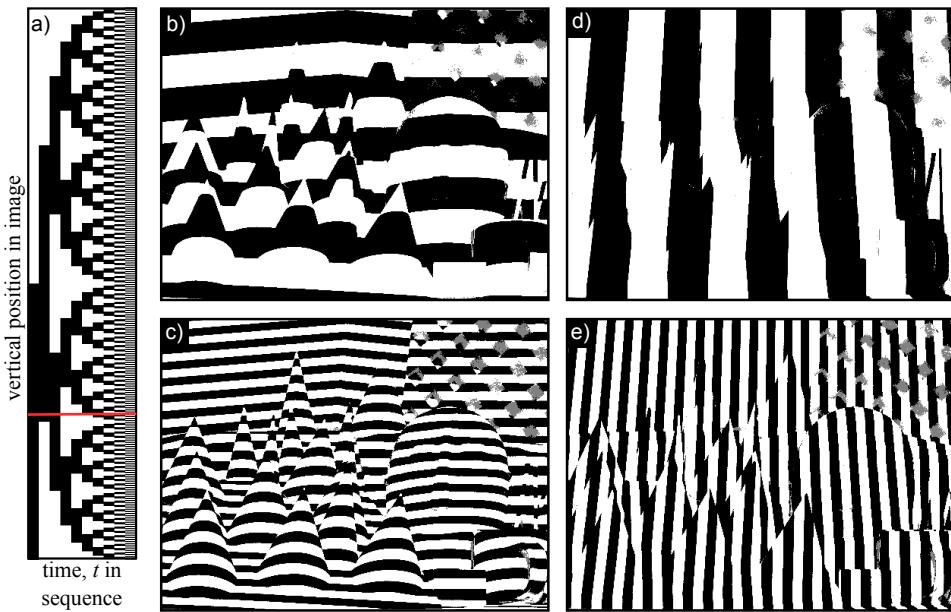
## 14.7 Applications

We discuss two applications for the techniques in this chapter. We consider a method to construct 3D models based on projecting structured light onto the object, and a method for generating novel views of an object based on an approximate model built from the silhouettes of the object.

### 14.7.1 Depth from structured light

In section 14.6 we showed how to compute the depth of a point given its position in two or more calibrated cameras. However, we did not discuss how to find matching points in the two images. We defer a full answer to this question to chapter 16, but here we will develop a method that circumvents this problem. The method will be based on a projector and a camera, rather than two cameras.

It is crucial to understand that the geometry of a projector is exactly the same as that of a camera: the projector has an optical center and has a regular pixel array that is analogous to the sensor in the camera. Each pixel in the projector corresponds to a direction in space (a ray) through the optical center, and this relationship can be captured by a set of intrinsic parameters. The major difference is



**Figure 14.13** Projector to camera correspondence with structured light patterns. a) To establish the vertical position in the projector image, we present a sequence of horizontally striped patterns. Each height in the projected image receives a unique sequence of black and white values, so we can determine the height (e.g. red line) by measuring this sequence. b-c) Two examples of these horizontally striped patterns. d-e) Two examples of vertically striped patterns that are part of a sequence designed to estimate the horizontal position in the projector pattern. Adapted from Scharstein & Szeliski (2003). ©2003 IEEE.

that a projector sends outgoing light along these rays, whereas the camera captures incoming light along them.

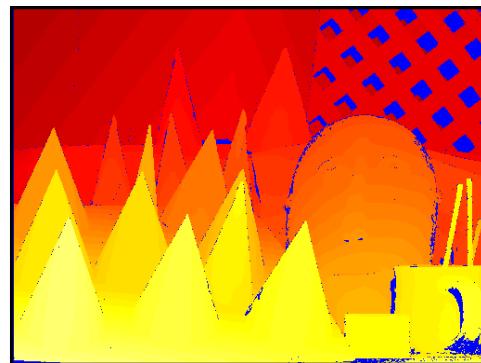
Consider a system that comprises a single camera and a projector that are displaced relative to one another but that point at the same object (figure 14.12). For simplicity, we will assume that the system is calibrated (i.e., the intrinsic matrices and the relative positions of the camera and projector are known). It is now easy to estimate the depth of the scene: we illuminate the scene using the projector *one pixel at a time*, and find the corresponding pixel in the camera by observing which part of the image gets brighter. We now have two corresponding points and can compute the depth using the method of section 14.6. In practice, this technique is very time consuming as a separate image must be captured for each pixel in the projector. Scharstein & Szeliski (2003) used a more practical technique using *structured light* in which a series of horizontal and vertical stripe patterns is projected onto the scene that allow the mapping between pixels in the projector and those in the camera to be computed.

To understand how this works, consider a projector image in which the top

Problem 14.7

Problem 14.8

**Figure 14.14** Recovered depth map for scene in figure 14.12 using the structured light method. Pixels marked as blue are places where the depth is uncertain: these include positions in the image that were occluded with respect to the projector, so no light was cast onto them. Scharstein & Szeliski (2003) also captured the scene with two cameras under normal illumination; they subsequently used the depth map from the structured light as ground truth data for assessing stereo vision algorithms. Adapted from Scharstein & Szeliski (2003). ©2003 IEEE.



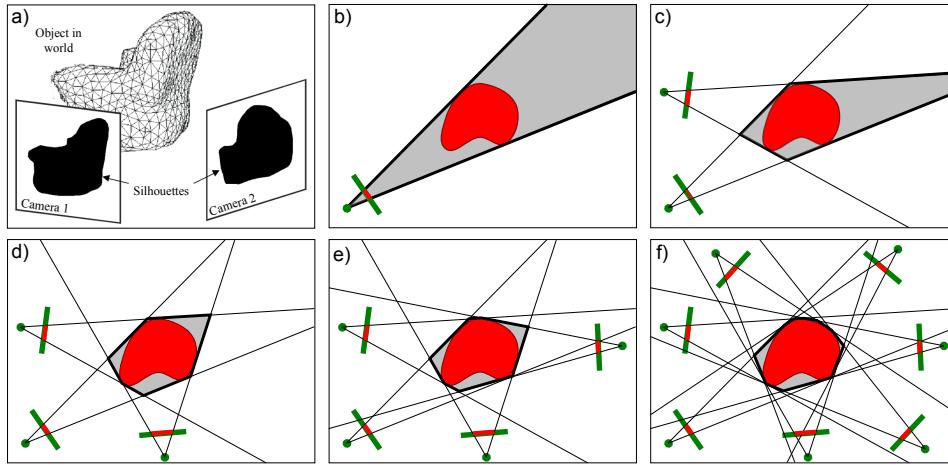
half is light and the bottom half is dark. We capture two images  $I_1$  and  $I_2$  of the scene, corresponding to when the projector shows this pattern and when it shows its inverse. We then take the difference  $I_1 - I_2$  between the images. Pixels in the camera image where the difference is positive must now belong to the top half of the projector image and pixels where the difference is negative must belong to the bottom half of the image. We now illuminate the image with a second pattern, which divides the image into four horizontally oriented regions of black and white stripes. By capturing the image illuminated by this second pattern and its inverse, we can hence deduce whether each pixel is in the top or bottom half of the region determined by the first pattern. We continue in this way with successively finer patterns, refining the estimated position at each pixel until we know it accurately. The whole procedure is repeated with vertical striped patterns to estimate the horizontal position.

In practice, more sophisticated coding schemes are used as the preceding sequence can induce artifacts; for example, this sequence means that there is always a boundary between black and white in the center of the projector image. It may be hard to establish the correspondence for a camera pixel, which always straddles this boundary. One solution is to base the sequences on *Gray codes* which have a more complex structure and avoid this problem (figure 14.13). The estimated depth map of the scene in figure 14.12 is shown in figure 14.14.

### 14.7.2 Shape from silhouette

The preceding system computed a 3D model of a scene based on explicit correspondences between a projector and a camera. We now consider an alternative method for computing 3D models that does not require explicit correspondence. As the name suggests, *shape from silhouette* estimates the shape of an object based on its silhouette in a number of images.

The principle is illustrated in figure 14.15. Given a single camera, we know that an object must lie somewhere within the bundle of rays that fall within its

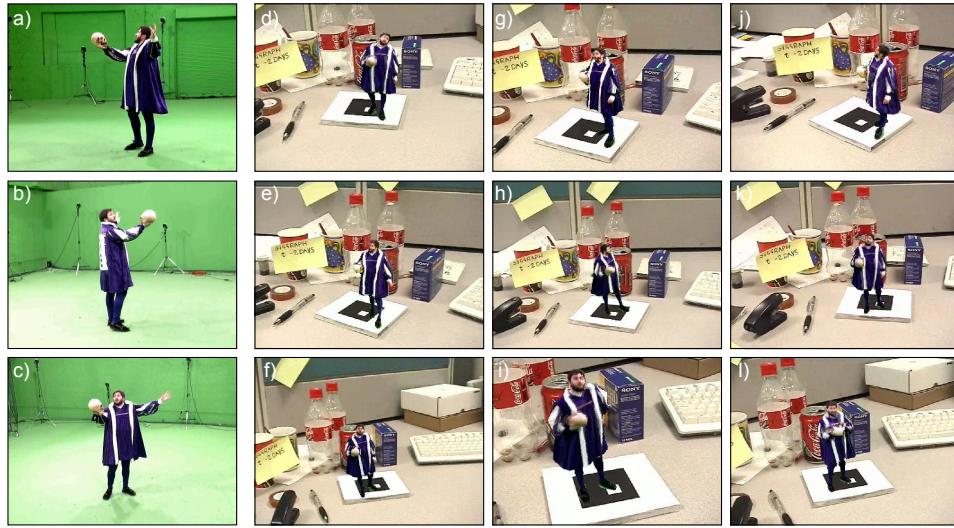


**Figure 14.15** Shape from silhouette a) The goal is to recover information about the shape of the object based on the silhouettes in a number of cameras. b) Consider a single camera viewing an object (2D slice shown). We know that the object must lie somewhere within the shaded region defined by the silhouette. c) When we add a second camera, we know that the object must lie within the intersection of the regions determined by the silhouettes (gray region). d-f) As we add more cameras, the approximation to the true shape becomes closer and closer. Unfortunately, however many cameras we add, we can never capture the concave region.

silhouette. Now consider adding a second camera. We also know that the object must lie somewhere within the bundle of rays corresponding to the silhouette in this image. Hence, we can refine our estimate of the shape to the 3D intersection of these two ray bundles. As we add more cameras, the possible region of space that the object can lie in is reduced.

This procedure is attractive because the silhouettes can be computed robustly and quickly using a background subtraction approach. However, there is also a downside; even if we have an infinite number of cameras viewing the object, some aspects of the shape will not be present in the final 3D region. For example, the concave region at the back of the seat of the chair in figure 14.15a cannot be recovered as it is not represented in the silhouettes. In general, the ‘best possible’ shape estimate is known as the *visual hull*.

We will now develop an algorithm based on shape from silhouette that can be used to generate images of an object from novel poses. One application of this is for augmented reality systems, in which we wish to superimpose an object over a real image in such a way that it looks like it is a stable part of the scene. This can be accomplished by establishing the position and pose of the camera relative to the scene (section 14.4), and then generating a novel image of the object from the same viewpoint. Figure 14.16 depicts an example application of this kind, in which the performance of an actor is captured and re-broadcast as if he is standing



**Figure 14.16** Generating novel views. a-c) An actor is captured from 15 cameras in a green screen studio. d-l) Novel views of the actor are now generated and superimposed on the scene. The novel view is carefully chosen so that it matches the direction that the camera views the desktop, giving the impression that the actor is a stable part of the scene. Adapted from Prince *et al.* (2002). ©2002 IEEE.

on the desk.

Prince *et al.* (2002) described a method to generate a novel image from a virtual camera with intrinsic matrix  $\Lambda$  and extrinsic parameters  $\{\Omega, \tau\}$ . They considered each pixel in the virtual camera in turn, and computed the direction of the ray  $\mathbf{r}$  passing through this point. With respect to the virtual camera itself, this ray has a direction

$$\mathbf{r} = \Lambda^{-1}\tilde{\mathbf{x}}, \quad (14.43)$$

where  $\tilde{\mathbf{x}} = [x, y, 1]^T$  is the position of the point in the image, expressed in homogeneous coordinates. With respect to the global coordinate system, a point  $\mathbf{w}$  that is  $\kappa$  units along the ray can be expressed as

$$\mathbf{w} = \tau + \kappa\Omega\mathbf{r}. \quad (14.44)$$

The depth of the object at this pixel is then computed by exploring along this direction; it is determined by an explicit search starting at the virtual camera projection center and proceeding outward along the ray corresponding to the pixel center (figure 14.17). Each candidate 3D point along this ray is evaluated for potential occupancy. A candidate point is unoccupied if its projection into any of the real images is marked as background. When the first point is found for which

all of the projected positions are marked as foreground, this is considered the depth with respect to the virtual object and the search stops.

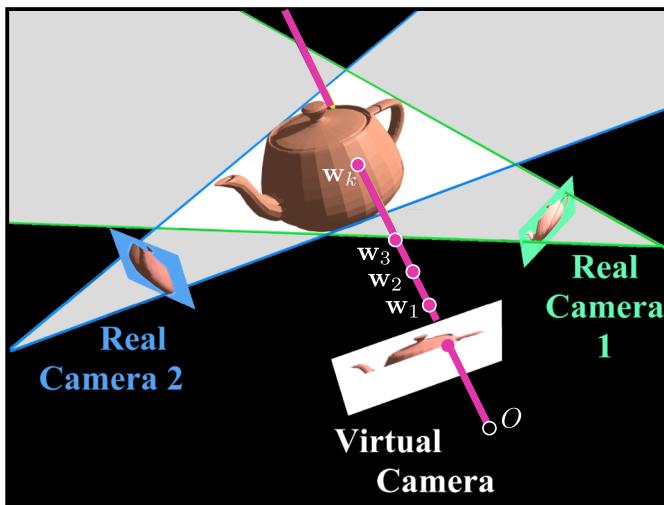
The 3D point where the ray through the current virtual pixel meets the visual hull is now known. To establish the color of this pixel in the virtual image, the point is projected into a nearby real image, and the color is copied. In general, the projection will not be exactly centered on a pixel, and to remedy this, the color value is estimated using bilinear or bicubic interpolation.

This procedure can be accomplished very quickly; the system of Prince *et al.* (2002) depicted in figure 14.16 ran at interactive speeds, but the quality is limited to the extent that the visual hull is a reasonable approximation of the true shape. If this approximation is bad, the wrong depth will be estimated, the projections into the real images will be inaccurate, and the wrong color will be sampled.

Problem 14.9  
Problem 14.10

## Discussion

In this chapter, we have introduced a model for a pinhole camera and discussed learning and inference algorithms for this model. In the next chapter, we consider what happens when this camera views a planar scene; here there is a one-to-one mapping between points in the scene and points in the image. In chapter 16, we return to the pinhole camera model and consider reconstruction of a scene from multiple cameras in unknown positions.



**Figure 14.17** Novel view generation. A new image is generated, one pixel at a time, by testing a sequence of points  $w_1, w_2 \dots$  along the ray through the pixel. Each point is tested to see if it is within the visual hull by projecting it into the real images. If it lies within the silhouette in every real image, then it is within the visual hull and the search stops. In this case, point  $w_k$  is the first point on the surface. To establish the color for the pixel, this point is projected into a nearby real image and the color is copied. Adapted from Prince *et al.* (2002). ©2002 IEEE.

## Notes

**Camera geometry:** There are detailed treatments of camera geometry in Hartley & Zisserman (2004), Ma *et al.* (2004) and Faugeras *et al.* (2001). Aloimonos (1990) and Mundy & Zisserman (1992) developed a hierarchy of camera models (see problem 14.3). Tsai (1987) and Faugeras (1993) both present algorithms for camera calibration from a 3D object. However, it is now more usual to calibrate cameras from multiple images of a planar object (see section 15.4) due to the difficulties associated with accurately machining a 3D object. For a recent summary of camera models and geometric computer vision, consult Sturm *et al.* (2011).

The projective pinhole camera discussed in this chapter is by no means the only camera model used in computer vision; there are specialized models for the pushbroom camera (Hartley & Gupta 1994), fish-eye lenses (Devernay & Faugeras 2001; Claus & Fitzgibbon 2005), catadioptric sensors (Geyer & Daniilidis 2001; Míkušík & Pajdla 2003; Claus & Fitzgibbon 2005) and perspective cameras imaging through an interface into a medium (Treibitz *et al.* 2008).

**Estimating extrinsic parameters:** A large body of work addresses the PnP problem of estimating the geometric relation between the camera and a rigid object. Lepetit *et al.* (2009) present a recent approach that has low complexity with respect to the number of points used and provides a quantitative comparison with other approaches. Quan & Lan (1999) and Gao *et al.* (2003) present minimal solutions based on three points.

**Structured light:** The structured light method discussed in this chapter is due to Scharstein & Szeliski (2003), although the main goal of their paper was to generate ground truth for stereo vision applications. The use of structured light has a long history in computer vision (e.g., Vuylsteke & Oosterlinck 1990), with the main research issue being the choice of pattern to project (Salvi *et al.* 2004; Batlle *et al.* 1998; Horn & Kiryati 1999).

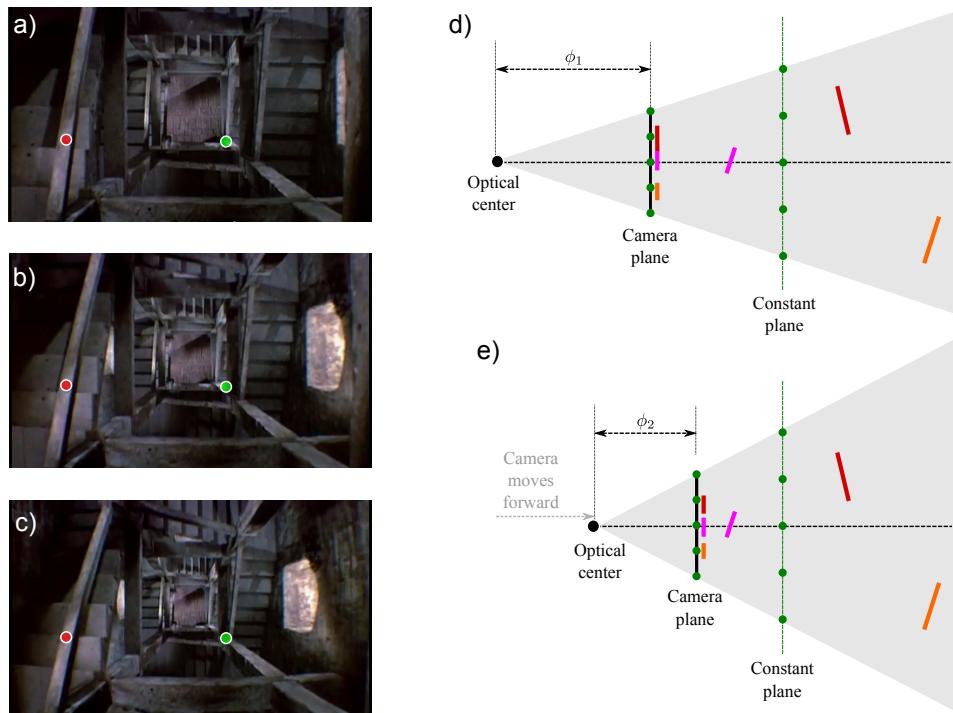
**Shape from silhouette:** The recovery of shape from multiple silhouettes of an object dates back to at least Baumgart (1974). Laurentini (1994) introduced the concept of the visual hull and described its properties. The shape from silhouette algorithm discussed in this chapter was by Prince *et al.* (2002) and is closely related to earlier work by Matusik *et al.* (2000) but rather simpler to explain. Recent work in this area has considered a probabilistic approach to pixel occupancy (Franco & Boyer 2005), the application of human silhouette priors (Grauman *et al.* 2003), the use of temporal sequences of silhouettes (Cheung *et al.* 2004), and approaches that characterize the intrinsic projective features of the visual hull.

**Human performance capture:** Modern interest in human performance capture was stimulated by Kanade *et al.* (1997). More recent work in this area includes Starck *et al.* (2009), Theobalt *et al.* (2007), Vlasic *et al.* (2008), de Aguiar *et al.* (2008), and Ballan & Cortelazzo (2008).

## Problems

**Problem 14.1** A pinhole camera has a sensor that is  $1\text{cm} \times 1\text{cm}$  and a horizontal field of view of  $60^\circ$ . What is the distance between the optical center and the sensor? The same camera has a resolution of 100 pixels in the horizontal direction and 200 pixels in the vertical direction (i.e., the pixels are not square). What are the focal length parameters  $f_x$  and  $f_y$  from the intrinsic matrix?

**Problem 14.2** We can use the pinhole camera model to understand a famous movie effect. *Dolly zoom* was first used in Alfred Hitchcock's *Vertigo*. As the protagonist looks down a



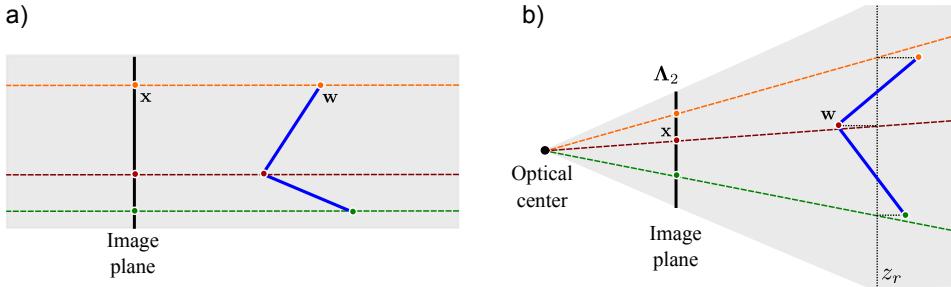
**Figure 14.18** Dolly Zoom. (a-c) Three frames from ‘Vertigo’ sequence in which the stairwell appears to distort. Nearby objects remain in roughly the same place whereas object further away systematically move through the sequence. To see this, consider the red and green circles that are at the same  $(x, y)$  position in each frame. The red circle remains on the near bannister, but the green circle is on the floor of the stairwell in the first image but halfway up the stairs in the last image. d) To understand this effect consider a camera viewing a scene which consists of several green points at the same depth and some other surfaces (colored lines). e) We move the camera along the  $w$ -axis but simultaneously change the focal length so that the green points are imaged at the same position. Under these changes, objects in the plane of the green points are static, but other parts of the scene move and may even occlude one another.

stairwell, it appears to deform (figure 14.18) in a strange way. The background seems to move away from the camera, while the foreground remains at a constant position.

In terms of the camera model, two things occur simultaneously during the dolly zoom sequence: the camera moves along the  $w$ -axis, and the focal distance of the camera changes. The distance moved and the change of focal length are carefully chosen so that objects in a pre-defined plane remain at the same position. However, objects out of this plane move relative to one another (figures 14.18d-e).

I want to capture two pictures of a scene at either end of a ‘dolly zoom’. Before the zoom, the camera is at  $w = 0$ , the distance between the optical center and the image plane is 1cm, and the image plane is  $1\text{cm} \times 1\text{cm}$ . After the zoom, the camera is at  $w = 100\text{cm}$ . I

want the plane at  $w = 500\text{cm}$  to be stable after the camera movement. What should the new distance between the optical center and the image plane be?



**Figure 14.19** Alternative camera models. a) Orthographic camera. Rays are parallel and orthogonal to image plane. b) Weak perspective model. Points are projected orthogonally onto a reference plane at distance  $z_r$  from the camera and then pass to the image plane by perspective projection.

**Problem 14.3** Figure 14.19 shows two different camera models: the orthographic and weak perspective cameras. For each camera, devise the relationship between the homogeneous world points and homogeneous image points. You may assume that the world coordinate system and the camera coordinate system coincide, so there is no need to introduce the extrinsic matrix.

**Problem 14.4** A 2D line can be expressed as  $ax + by + c = 0$  or in homogeneous terms

$$\mathbf{l}\tilde{\mathbf{x}} = 0,$$

where  $\mathbf{l} = [a, b, c]$ . Find the point where the homogeneous lines  $\mathbf{l}_1$  and  $\mathbf{l}_2$  join where:

1.  $\mathbf{l}_1 = [3, 1, 1]$ , and  $\mathbf{l}_2 = [-1, 0, 1]$
2.  $\mathbf{l}_1 = [1, 0, 1]$ , and  $\mathbf{l}_2 = [3, 0, 1]$

*Hint:* the  $3 \times 1$  homogeneous point vector  $\tilde{\mathbf{x}}$  must satisfy both  $\mathbf{l}_1\tilde{\mathbf{x}} = 0$  and  $\mathbf{l}_2\tilde{\mathbf{x}} = 0$ . In other words it should be orthogonal to both  $\mathbf{l}_1$  and  $\mathbf{l}_2$ .

**Problem 14.5** Find the line joining the homogeneous points  $\tilde{\mathbf{x}}_1$  and  $\tilde{\mathbf{x}}_2$  where

$$\tilde{\mathbf{x}}_1 = [2, 2, 1]^T, \tilde{\mathbf{x}}_2 = [-2, -2, 1]^T.$$

**Problem 14.6** A conic  $\mathbf{C}$  is a geometric structure that can represent ellipses and circles in the 2D image. The condition for a point to lie on a conic is given by

$$[x \ y \ 1] \begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0,$$

or

$$\tilde{\mathbf{x}}^T \mathbf{C} \tilde{\mathbf{x}} = 0.$$

Describe an algorithm to estimate the parameters  $a, b, c, d, e, f$  given several points  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$  that are known to lie on the conic. What is the minimum number of points that your algorithm requires to be successful?

**Problem 14.7** Devise a method to find the intrinsic matrix of a projector using a camera and known calibration object.

**Problem 14.8** What is the minimum number of binary striped light patterns of the type illustrated in figure 14.13 required to estimate the camera-projector correspondences for a projector image of size  $H \times W$ ?

**Problem 14.9** There is a potential problem with the shape from silhouette algorithm as described; the point that we have found on the surface of the object may be occluded by another part of the object with respect to the nearest camera. Consequently, when we copy the color, we will get the wrong value. Propose a method to circumvent this problem.

**Problem 14.10** In the augmented reality application (figure 14.16), the realism might be enhanced if the object had a shadow. Propose an algorithm that could establish whether a point on the desktop (assumed planar) is shadowed by the object with respect to a point light source at a known position.

# Chapter 15

## Models for transformations

In this chapter, we consider a pinhole camera viewing a plane in the world. In these circumstances, the camera equations simplify to reflect the fact that there is a one-to-one mapping between points on this plane and points in the image.

Mappings between the plane and the image can be described using a family of 2D geometric transformations. In this chapter, we characterize these transformations and show how to estimate their parameters from data. We revisit the three geometric problems from chapter 14 for the special case of a planar scene.

To motivate the ideas of this chapter, consider an augmented reality application in which we wish to superimpose 3D content onto a planar marker (figure 15.1). To do this, we must establish the rotation and translation of the plane relative to the camera. We will do this in two stages. First, we will estimate the 2D transformation between points on the marker and points in the image. Second, we will extract the rotation and translation from the transformation parameters.

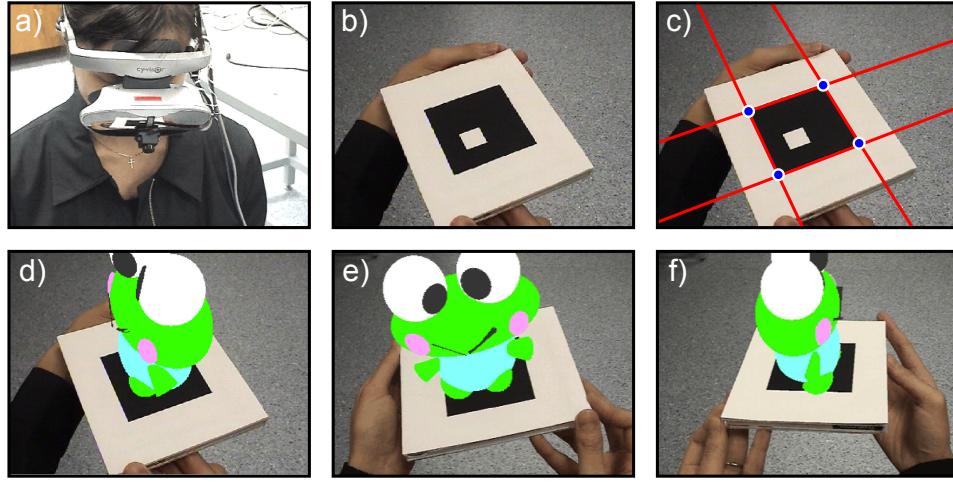
### 15.1 2D transformation models

In this section, we consider a family of 2D transformations, starting with the simplest and working toward the most general. We will motivate each by considering viewing a planar scene under different viewing conditions.

#### 15.1.1 Euclidean transformation model

Consider a calibrated camera viewing a fronto-parallel plane at known distance,  $D$  (i.e., a plane whose normal corresponds to the  $w$ -axis of the camera). This may seem like a contrived situation, but it is exactly what happens in machine inspection applications: an overhead camera views a conveyor belt and examines objects that contain little or no depth variation.

We assume that a position on the plane can be described by a 3D position  $\mathbf{w} = [u, v, 0]^T$ , measured in real-world units such as millimeters. The  $w$ -coordinate



**Figure 15.1** Video see-through augmented reality with a planar scene. a) The user views the world through a head mounted display with a camera attached to the front. The images from the camera are analyzed and augmented in near-real time and displayed to the user. b) Here, the world consists of a planar 2D marker. c) The marker corners are found by fitting edges to its sides and finding their intersections. d) The geometric transformation between the 2D positions of the corners on the marker surface and the corresponding positions in the image is computed. This transformation is analyzed to find the rotation and translation of the camera relative to the marker. This allows us to superimpose a 3D object as if it were rigidly attached to the surface of the image. e-f) As the marker is manipulated the superimposed object changes pose appropriately.

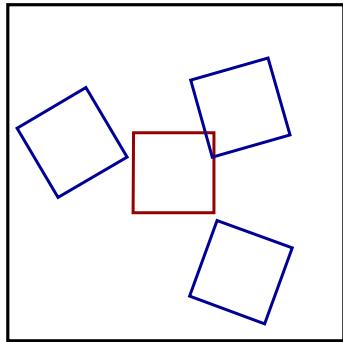
represents directions perpendicular to the plane, and is hence always zero. Consequently, we will sometimes treat  $\mathbf{w} = [u, v]^T$  as a 2D coordinate.

Applying the pinhole camera model to this situation gives

$$\lambda \tilde{\mathbf{x}} = \mathbf{\Lambda}[\boldsymbol{\Omega}, \boldsymbol{\tau}] \tilde{\mathbf{w}}, \quad (15.1)$$

where  $\tilde{\mathbf{x}}$  is the 2D observed image position represented as a homogeneous 3-vector and  $\tilde{\mathbf{w}}$  is the 3D point in the world represented as a homogeneous 4-vector. Writing this out explicitly, we have

$$\begin{aligned} \lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} &= \begin{bmatrix} \phi_x & \gamma & \delta_x \\ 0 & \phi_y & \delta_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_{11} & \omega_{12} & 0 & \tau_x \\ \omega_{21} & \omega_{22} & 0 & \tau_y \\ 0 & 0 & 1 & D \end{bmatrix} \begin{bmatrix} u \\ v \\ 0 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \phi_x & \gamma & \delta_x \\ 0 & \phi_y & \delta_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_{11} & \omega_{12} & \tau_x \\ \omega_{21} & \omega_{22} & \tau_y \\ 0 & 0 & D \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \end{aligned} \quad (15.2)$$



**Figure 15.2** The 2D Euclidean transformation describes 2D rigid rotations and translations. The blue squares are all Euclidean transformations of the original red square. The transformation has three parameters: the rotation angle and the translations in the  $x$ -and  $y$ -directions. When a camera views a fronto-parallel plane at a known distance, the relation between the normalized camera coordinates and the 2D positions on the plane is a Euclidean transformation.

where the 3D rotation matrix  $\Omega$  takes a special form with only four unknowns, reflecting the fact that the plane is known to be fronto-parallel.

We can move the distance parameter  $D$  into the intrinsic matrix without changing the last of these three equations and equivalently write

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_x & \gamma & \delta_x \\ 0 & \phi_y & \delta_y \\ 0 & 0 & D \end{bmatrix} \begin{bmatrix} \omega_{11} & \omega_{12} & \tau_x \\ \omega_{21} & \omega_{22} & \tau_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}. \quad (15.3)$$

If we now eliminate the effect of this modified intrinsic matrix, by pre-multiplying both left and right by its inverse, we get

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \omega_{11} & \omega_{12} & \tau_x \\ \omega_{21} & \omega_{22} & \tau_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \quad (15.4)$$

where  $x'$  and  $y'$  are camera coordinates that are normalized with respect to this modified intrinsic matrix.

The mapping in equation 15.4 is known as a *Euclidean transformation*. It can be equivalently written in Cartesian coordinates as

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \omega_{11} & \omega_{12} \\ \omega_{21} & \omega_{22} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} \tau_x \\ \tau_y \end{bmatrix}, \quad (15.5)$$

or for short, we may write

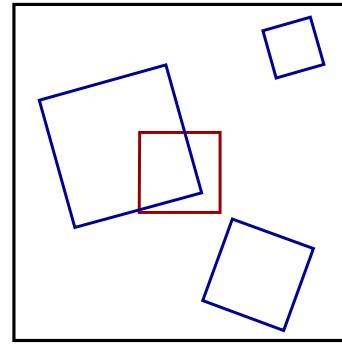
$$\mathbf{x}' = \mathbf{euc}[\mathbf{w}, \boldsymbol{\Omega}, \boldsymbol{\tau}], \quad (15.6)$$

where  $\mathbf{x}' = [x', y']^T$  contains the normalized camera coordinates and  $\mathbf{w} = [u, v]^T$  is the real-world position on the plane.

The Euclidean transformation describes rigid rotations and translations in the plane (figure 15.2). Although this transformation appears to take six separate

Problem 15.1

**Figure 15.3** The similarity transformation describes rotations, translations and isotropic scalings. The blue quadrilaterals are all similarity transformations of the original red square. The transformation has four parameters: the rotation angle, the scaling and the translations in the  $x$ - and  $y$ -directions. When a camera views a fronto-parallel plane at unknown distance, the relation between the normalized camera co-ordinates and positions on the plane is a similarity.



parameters, the rotation matrix  $\Omega$  can be re-expressed in terms of the rotation angle  $\theta$ ,

$$\begin{bmatrix} \omega_{11} & \omega_{12} \\ \omega_{21} & \omega_{22} \end{bmatrix} = \begin{bmatrix} \cos[\theta] & \sin[\theta] \\ -\sin[\theta] & \cos[\theta] \end{bmatrix}, \quad (15.7)$$

and hence the actual number of parameters is three (the two offsets  $\tau_x$  and  $\tau_y$ , and the rotation,  $\theta$ ).

### 15.1.2 Similarity transformation model

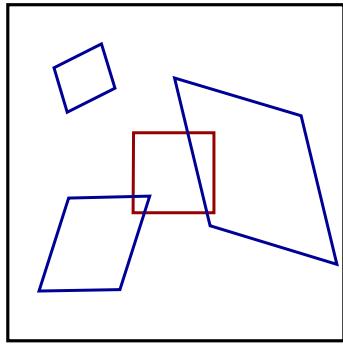
Now consider a calibrated camera viewing a fronto-parallel plane at *unknown* distance  $D$ . The relationship between image points  $\mathbf{x} = [x, y]^T$  and points  $\mathbf{w} = [u, v, 0]^T$  on the plane is once more given by equation 15.2. Converting to normalized image co-ordinates by pre-multiplying both sides by the inverse of the intrinsic matrix gives

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \omega_{11} & \omega_{12} & 0 & \tau_x \\ \omega_{21} & \omega_{22} & 0 & \tau_y \\ 0 & 0 & 1 & D \end{bmatrix} \begin{bmatrix} u \\ v \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \omega_{11} & \omega_{12} & \tau_x \\ \omega_{21} & \omega_{22} & \tau_y \\ 0 & 0 & D \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}. \quad (15.8)$$

We now multiply each of these three equations by  $\rho = 1/D$  to get

$$\rho \lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \rho \omega_{11} & \rho \omega_{12} & \rho \tau_x \\ \rho \omega_{21} & \rho \omega_{22} & \rho \tau_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}. \quad (15.9)$$

This is the homogeneous representation of the *similarity transformation*. However, it is usual to incorporate  $\rho$  into the constant  $\lambda$  on the left hand side so that  $\lambda \leftarrow \rho \lambda$  and into the translation parameters so that  $\tau_x \leftarrow \rho \tau_x$  and  $\tau_y \leftarrow \rho \tau_y$  on the right hand side to yield



**Figure 15.4** The affine transformation describes rotations, translations, scalings and shears. The blue quadrilaterals are all affine transformations of the original red square. The affine transformation has six parameters: the translations in the  $x$ - and  $y$ -directions and four parameters that determine the other effects. Notice that lines that were originally parallel remain parallel after the affine transformation is applied, so in each case the square becomes a parallelogram.

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \rho\omega_{11} & \rho\omega_{12} & \tau_x \\ \rho\omega_{21} & \rho\omega_{22} & \tau_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}. \quad (15.10)$$

Converting to Cartesian coordinates, we have

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \rho\omega_{11} & \rho\omega_{12} \\ \rho\omega_{21} & \rho\omega_{22} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} \tau_x \\ \tau_y \end{bmatrix}, \quad (15.11)$$

or for short,

$$\mathbf{x}' = \mathbf{sim}[\mathbf{w}, \boldsymbol{\Omega}, \boldsymbol{\tau}, \rho]. \quad (15.12)$$

The similarity transformation is a Euclidean transformation with a scaling (figure 15.3) and has four parameters: the rotation, the scaling, and two translations.

### 15.1.3 Affine transformation model

We motivated each of the previous transformations by considering a camera viewing a fronto-parallel plane. Ultimately, we wish to describe the relationship between image points and points on a plane in general position. As an intermediate step, let us generalize the transformations in equations 15.4 and 15.10 to

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} & \tau_x \\ \phi_{21} & \phi_{22} & \tau_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \quad (15.13)$$

where  $\phi_{11}, \phi_{12}, \phi_{21}$ , and  $\phi_{22}$  are now unconstrained, so can take arbitrary values. This is known as an *affine transformation*. In Cartesian coordinates, we have

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} \tau_x \\ \tau_y \end{bmatrix}, \quad (15.14)$$



**Figure 15.5** Approximating projection of a plane. a) A planar object viewed with a camera with a narrow field of view (long focal length) from a large distance. The depth variation within the object is small compared to the distance from the camera to the plane. Here, perspective distortion is small, and the relationship between points in the image and points on the surface is well approximated by an affine transform. b) The same planar object viewed with a wide field of view (short focal length) from a short distance. The depth variation within the object is comparable to the average distance from the camera to the plane. An affine transformation cannot describe this situation well. c) However, a projective transformation (homography) captures the relationship between points on the surface and points in this image.

or for short, we might write

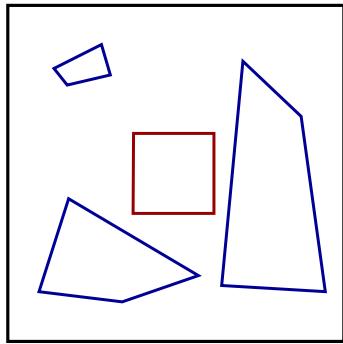
$$\mathbf{x}' = \text{aff}[\mathbf{w}, \Phi, \boldsymbol{\tau}]. \quad (15.15)$$

Note that the camera calibration matrix  $\Lambda$  also has the form of an affine transformation (i.e., a  $3 \times 3$  matrix with two zeros in the bottom row). The product of two affine transformations is a third affine transformation, so if equation 15.15 is true, then there is also an affine transformation between points on the plane and the original (un-normalized) pixel positions.

#### Problem 15.7

The affine transformation encompasses both Euclidean and similarity transformations, but also includes *shears* (figure 15.4). However, it is far from general, and a notable restriction is that parallel lines are always mapped to other parallel lines. It has six unknown parameters, each of which can take any value.

The question remains as to whether the affine transformation really does provide a good mapping between points on a plane and their positions in the image. This is indeed the case when the depth variation of the plane as seen by the camera is small relative to the mean distance from the camera. In practice, this occurs when the viewing angle is not too oblique, the camera is distant, and the field of view is small (figure 15.5a). In more general situations, the affine transformation is not a good approximation. A simple counter-example is the convergence of parallel train tracks in an image as they become more distant. The affine transformation cannot describe this situation, as it can only map parallel lines on the object to parallel lines in the image.



**Figure 15.6** The projective transformation (also known as a collinearity or homography) can map any four points in the plane to any other four points. Rotations, translations, scalings, and shears are all special cases. The blue quadrilaterals are all projective transformations of the original red square. The projective transformation has eight parameters. Lines that were parallel are not constrained to remain parallel after the projective transformation is applied.

### 15.1.4 Projective transformation model

Finally, we investigate what really happens when a pinhole camera views a plane from an arbitrary viewpoint. The relationship between a point  $\mathbf{w} = [u, v, 0]^T$  on the plane and the position  $\mathbf{x} = [x, y]^T$  to which it is projected is

$$\begin{aligned}\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} &= \begin{bmatrix} \phi_x & \gamma & \delta_x \\ 0 & \phi_y & \delta_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} & \tau_x \\ \omega_{21} & \omega_{22} & \omega_{23} & \tau_y \\ \omega_{31} & \omega_{32} & \omega_{33} & \tau_z \end{bmatrix} \begin{bmatrix} u \\ v \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} \phi_x & \gamma & \delta_x \\ 0 & \phi_y & \delta_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_{11} & \omega_{12} & \tau_x \\ \omega_{21} & \omega_{22} & \tau_y \\ \omega_{31} & \omega_{32} & \tau_z \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}. \quad (15.16)\end{aligned}$$

Combining the two  $3 \times 3$  matrices by multiplying them together, the result is a transformation with the general form

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} \\ \phi_{21} & \phi_{22} & \phi_{23} \\ \phi_{31} & \phi_{32} & \phi_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \quad (15.17)$$

and is variously known as a *projective transformation*, a *collinearity*, or a *homography*. In Cartesian coordinates the homography is written as

$$\begin{aligned}x &= \frac{\phi_{11}u + \phi_{12}v + \phi_{13}}{\phi_{31}u + \phi_{32}v + \phi_{33}} \\ y &= \frac{\phi_{21}u + \phi_{22}v + \phi_{23}}{\phi_{31}u + \phi_{32}v + \phi_{33}}, \quad (15.18)\end{aligned}$$

or for short,

$$\mathbf{x} = \text{hom}[\mathbf{w}, \Phi]. \quad (15.19)$$

The homography can map any four points in the plane to any other four

Problem 15.8

points (figure 15.6). It is a linear transformation in homogenous coordinates (equation 15.17) but is nonlinear in Cartesian coordinates (equation 15.18). It subsumes the Euclidean, similarity, and affine transformations as special cases. It exactly describes the mapping between the 2D coordinates of points on a plane in the real world and their positions in an image of that plane (figure 15.5c).

Although there are nine entries in the matrix  $\Phi$ , the homography only contains eight degrees of freedom; the entries are redundant with respect to scale. It is easy to see that a constant re-scaling of all nine values produces the same transformation, as the scaling factor cancels out of the numerator and denominator in equation 15.18. The properties of the homography are discussed further in section 15.5.1.

### 15.1.5 Adding uncertainty

The four geometric models presented in the preceding sections are deterministic. However, in a real system, the measured positions of features in the image are subject to noise, and we need to incorporate this uncertainty into our models. In particular, we will assume that the positions  $\mathbf{x}_i$  in the image are corrupted by normally distributed noise with spherical covariance so that, for example, the likelihood for the homography becomes

$$Pr(\mathbf{x}|\mathbf{w}) = \text{Norm}_{\mathbf{x}} [\text{hom}[\mathbf{w}, \Phi], \sigma^2 \mathbf{I}] . \quad (15.20)$$

This is a generative model for the 2D image data  $\mathbf{x}$ . It can be thought of as a simplified version of the pinhole camera model that is specialized for viewing planar scenes; it is a recipe that tells us how to find the position in the image  $\mathbf{x}$  corresponding to a point  $\mathbf{w}$  on the surface of the planar object in the world.

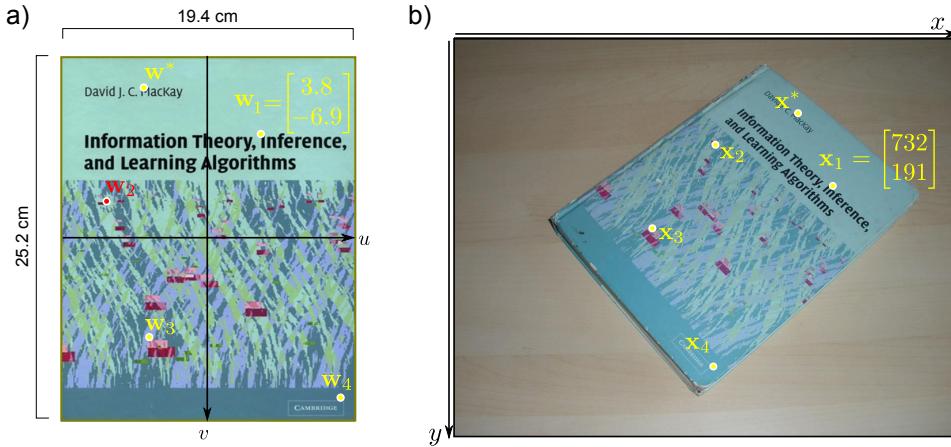
The learning and inference problems in this model (figure 15.7) are:

- **Learning:** we are given pairs of points  $\{\mathbf{x}_i, \mathbf{w}_i\}_{i=1}^I$  where  $\mathbf{x}_i$  is a position in the image and  $\mathbf{w}_i$  is the corresponding position on the plane in the world. The goal is to use these to establish the parameters  $\theta$  of the transformation. For example, in the case of the homography, the parameters  $\theta$  would comprise the nine entries of the matrix  $\Phi$ .
- **Inference:** we are given a new point in the image  $\mathbf{x}^*$ , and our goal is to find the position on the plane  $\mathbf{w}^*$  that projected to it.

We consider these two problems in the following sections.

## 15.2 Learning in transformation models

We are given a set of  $I$  2D positions  $\mathbf{w}_i = [u_i, v_i]^T$  on the surface of the plane and the  $I$  corresponding 2D image positions  $\mathbf{x}_i = [x_i, y_i]^T$  in the image. We select a



**Figure 15.7** Learning and inference for transformation models. a) Planar object surface (position measured in cm). b) Image (position measured in pixels). In learning, we estimate the parameters of the mapping from points  $\mathbf{w}$  on the object surface to image positions  $\mathbf{x}_i$  based on pairs  $\{\mathbf{x}_i, \mathbf{w}_i\}_{i=1}^I$  of known correspondences. We can use this mapping to find the position  $\mathbf{x}^*$  in the image to which a point  $\mathbf{w}^*$  on the object surface will project. In inference, we reverse this process: given a position  $\mathbf{x}^*$  in the image, the goal is to establish the corresponding position  $\mathbf{w}^*$  on the object surface.

transformation class of the form  $\text{trans}[\mathbf{w}_i, \boldsymbol{\theta}]$ . The goal of the learning algorithm is then to estimate the parameters  $\boldsymbol{\theta}$  that best map the points  $\mathbf{w}_i$  to the image positions  $\mathbf{x}_i$ .

Adopting the maximum likelihood approach, we have

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[ \prod_{i=1}^I \operatorname{Norm}_{\mathbf{x}_i} [\text{trans}[\mathbf{w}_i, \boldsymbol{\theta}], \sigma^2 \mathbf{I}] \right] \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[ \sum_{i=1}^I \log [\operatorname{Norm}_{\mathbf{x}_i} [\text{trans}[\mathbf{w}_i, \boldsymbol{\theta}], \sigma^2 \mathbf{I}]] \right],\end{aligned}\quad (15.21)$$

where, as usual, we have taken the logarithm which is a monotonic transformation and hence does not affect the position of the maximum. Substituting in the expression for the normal distribution and simplifying, we get the least squares problem

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left[ \sum_{i=1}^I (\mathbf{x}_i - \text{trans}[\mathbf{w}_i, \boldsymbol{\theta}])^T (\mathbf{x}_i - \text{trans}[\mathbf{w}_i, \boldsymbol{\theta}]) \right].\quad (15.22)$$

The solutions to this least squares problem for each of the four transformation types are presented in sections 15.2.1–15.2.4. The details differ in each case, but

they have the common approach of reducing the problem into a standard form for which the solution is known. The algorithms are somewhat involved, and these sections can be skipped on first reading.

### 15.2.1 Learning Euclidean parameters

Algorithm 15.1

The Euclidean transformation is determined by a  $2 \times 2$  rotation matrix  $\Omega$  and a  $2 \times 1$  translation vector  $\tau = [\tau_x, \tau_y]^T$  (equations 15.4 and 15.5). Each pair of matching points  $\{\mathbf{x}_i, \mathbf{w}_i\}$  contributes two constraints to the solution (deriving from the  $x$ - and  $y$ -coordinates). Since there are three underlying degrees of freedom, we will require at least  $I=2$  pairs of points to get a unique estimate.

Our goal is to solve the problem

$$\begin{aligned}\hat{\Omega}, \hat{\tau} &= \underset{\Omega, \tau}{\operatorname{argmin}} \left[ \sum_{i=1}^I (\mathbf{x}_i - \mathbf{euc}[\mathbf{w}_i, \Omega, \tau])^T (\mathbf{x}_i - \mathbf{euc}[\mathbf{w}_i, \Omega, \tau]) \right] \\ &= \underset{\Omega, \tau}{\operatorname{argmin}} \left[ \sum_{i=1}^I (\mathbf{x}_i - \Omega \mathbf{w}_i - \tau)^T (\mathbf{x}_i - \Omega \mathbf{w}_i - \tau) \right],\end{aligned}\quad (15.23)$$

with the constraint that  $\Omega$  is a rotation matrix so that  $\Omega \Omega^T = \mathbf{I}$  and  $|\Omega| = 1$ .

An expression for the translation vector can be found by taking the derivative of the objective function with respect to  $\tau$ , setting the result to zero and simplifying. The result is the mean difference vector between the two sets of points after the rotation has been applied

$$\hat{\tau} = \frac{\sum_{i=1}^I \mathbf{x}_i - \Omega \mathbf{w}_i}{I} = \boldsymbol{\mu}_x - \Omega \boldsymbol{\mu}_w,\quad (15.24)$$

where  $\boldsymbol{\mu}_x$  is the mean of the points  $\{\mathbf{x}_i\}$  and  $\boldsymbol{\mu}_w$  is the mean of the points  $\{\mathbf{w}_i\}$ . Substituting this result into the original criterion, we get

$$\hat{\Omega} = \underset{\Omega}{\operatorname{argmin}} \left[ \sum_{i=1}^I ((\mathbf{x}_i - \boldsymbol{\mu}_x) - \Omega(\mathbf{w}_i - \boldsymbol{\mu}_w))^T ((\mathbf{x}_i - \boldsymbol{\mu}_x) - \Omega(\mathbf{w}_i - \boldsymbol{\mu}_w)) \right].\quad (15.25)$$

Defining matrices  $\mathbf{B} = [\mathbf{x}_1 - \boldsymbol{\mu}_x, \mathbf{x}_2 - \boldsymbol{\mu}_x, \dots, \mathbf{x}_I - \boldsymbol{\mu}_x]$  and  $\mathbf{A} = [\mathbf{w}_1 - \boldsymbol{\mu}_w, \mathbf{w}_2 - \boldsymbol{\mu}_w, \dots, \mathbf{w}_I - \boldsymbol{\mu}_w]$ , we can re-write the objective function for the best rotation  $\Omega$  as

$$\hat{\Omega} = \underset{\Omega}{\operatorname{argmin}} [|\mathbf{B} - \Omega \mathbf{A}|_F] \quad \text{subject to } \Omega \Omega^T = \mathbf{I}, |\Omega| = 1,\quad (15.26)$$

where  $|\bullet|_F$  denotes the Frobenius norm. This is an example of an *orthogonal Procrustes problem*. A closed form solution can be found by computing the SVD  $\mathbf{ULV}^T = \mathbf{BA}^T$ , and then choosing  $\hat{\Omega} = \mathbf{VU}^T$  (see appendix C.7.3).

### 15.2.2 Learning similarity parameters

The similarity transformation is determined by a  $2 \times 2$  rotation matrix  $\Omega$ , a  $2 \times 1$  translation vector  $\tau$ , and a scaling factor  $\rho$  (equations 15.10 and 15.11). There are four underlying degrees of freedom, so we will require at least  $I = 2$  pairs of matching points  $\{\mathbf{x}_i, \mathbf{w}_i\}$  to guarantee a unique solution.

Algorithm 15.2

The objective function for maximum likelihood fitting of the parameters is

$$\begin{aligned}\hat{\Omega}, \hat{\tau}, \hat{\rho} &= \underset{\Omega, \tau, \rho}{\operatorname{argmin}} \left[ \sum_{i=1}^I (\mathbf{x}_i - \text{sim}[\mathbf{w}_i, \Omega, \tau, \rho])^T (\mathbf{x}_i - \text{sim}[\mathbf{w}_i, \Omega, \tau, \rho]) \right] \\ &= \underset{\Omega, \tau, \rho}{\operatorname{argmin}} \left[ \sum_{i=1}^I (\mathbf{x}_i - \rho \Omega \mathbf{w}_i - \tau)^T (\mathbf{x}_i - \rho \Omega \mathbf{w}_i - \tau) \right],\end{aligned}\quad (15.27)$$

with the constraint that  $\Omega$  is a rotation matrix so that  $\Omega \Omega^T = \mathbf{I}$  and  $|\Omega| = 1$ .

To optimize this criterion, we compute  $\Omega$  exactly as for the Euclidean transform. The maximum likelihood solution for the scaling factor is given by

$$\hat{\rho} = \frac{\sum_{i=1}^I (\mathbf{x}_i - \mu_x)^T \hat{\Omega} (\mathbf{w}_i - \mu_w)}{\sum_{i=1}^I (\mathbf{w}_i - \mu_w)^T (\mathbf{w}_i - \mu_w)},\quad (15.28)$$

and the translation can be found using

$$\hat{\tau} = \frac{\sum_{i=1}^I (\mathbf{x}_i - \hat{\rho} \hat{\Omega} \mathbf{w}_i)}{I}.\quad (15.29)$$

### 15.2.3 Learning affine parameters

The affine transformation is parameterized by an unconstrained  $2 \times 2$  matrix  $\Phi$  and a  $2 \times 1$  translation vector  $\tau$  (equations 15.13 and 15.14). There are six unknowns, and so we need a minimum of  $I = 3$  pairs of matching points  $\{\mathbf{x}_i, \mathbf{w}_i\}$  to guarantee a unique solution. The learning problem can be stated as

$$\begin{aligned}\hat{\Phi}, \hat{\tau} &= \underset{\Phi, \tau}{\operatorname{argmin}} \left[ \sum_{i=1}^I (\mathbf{x}_i - \text{aff}[\mathbf{w}_i, \Phi, \tau])^T (\mathbf{x}_i - \text{aff}[\mathbf{w}_i, \Phi, \tau]) \right] \\ &= \underset{\Phi, \tau}{\operatorname{argmin}} \left[ \sum_{i=1}^I (\mathbf{x}_i - \Phi \mathbf{w}_i - \tau)^T (\mathbf{x}_i - \Phi \mathbf{w}_i - \tau) \right].\end{aligned}\quad (15.30)$$

Algorithm 15.3

To solve this problem, observe that we can re-express  $\Phi \mathbf{w}_i + \tau$  as a linear function of the unknown elements of  $\Phi$  and  $\tau$

$$\Phi \mathbf{w}_i + \boldsymbol{\tau} = \begin{bmatrix} u_i & v_i & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_i & v_i & 1 \end{bmatrix} \begin{bmatrix} \phi_{11} \\ \phi_{12} \\ \tau_x \\ \phi_{21} \\ \phi_{22} \\ \tau_y \end{bmatrix} = \mathbf{A}_i \mathbf{b}, \quad (15.31)$$

where  $\mathbf{A}_i$  is a  $2 \times 6$  matrix based on the point  $\mathbf{w}_i$ , and  $\mathbf{b}$  contains the unknown parameters. The problem can now be written as

$$\hat{\mathbf{b}} = \underset{\mathbf{b}}{\operatorname{argmin}} \left[ \sum_{i=1}^I (\mathbf{x}_i - \mathbf{A}_i \mathbf{b})^T (\mathbf{x}_i - \mathbf{A}_i \mathbf{b}) \right], \quad (15.32)$$

which is a linear least squares problem and can be solved easily (appendix C.7.1).

#### 15.2.4 Learning projective parameters

The projective transformation or homography is parameterized by a  $3 \times 3$  matrix  $\Phi$  (equations 15.17 and 15.18) which is ambiguous up to scale, giving a total of eight degrees of freedom. Consequently, we need a minimum of  $I = 4$  pairs of corresponding points for a unique solution. This neatly matches our expectations: a homography can map any four points in the plane to any other four points, and so it is reasonable that we should need at least four pairs of points to determine it.

The learning problem can be stated as

$$\begin{aligned} \hat{\Phi} &= \underset{\Phi}{\operatorname{argmin}} \left[ \sum_{i=1}^I (\mathbf{x}_i - \text{hom}[\mathbf{w}_i, \Phi])^T (\mathbf{x}_i - \text{hom}[\mathbf{w}_i, \Phi]) \right] \\ &= \underset{\Phi}{\operatorname{argmin}} \left[ \sum_{i=1}^I \left( x_i - \frac{\phi_{11}u_i + \phi_{12}v_i + \phi_{13}}{\phi_{31}u_i + \phi_{32}v_i + \phi_{33}} \right)^2 + \left( y_i - \frac{\phi_{21}u_i + \phi_{22}v_i + \phi_{23}}{\phi_{31}u_i + \phi_{32}v_i + \phi_{33}} \right)^2 \right]. \end{aligned} \quad (15.33)$$

Unfortunately, there is no closed form solution to this nonlinear problem and to find the answer, we must rely on gradient-based optimization techniques. Since there is a scale ambiguity, this optimization would normally be carried out under the constraint that the sum of the squares of the elements of  $\Phi$  is one.

A successful optimization procedure depends on a good initial starting point, and for this we use the *direct linear transformation* or *DLT* algorithm. The DLT algorithm uses homogeneous coordinates where the homography is a linear transformation and finds a closed form solution for the algebraic error. This is not the same as optimizing the true objective function (equation 15.33), but provides a result that is usually very close to the true answer and can be used as a starting point for the nonlinear optimization of the true criterion. In homogeneous coordinates we have

$$\lambda \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} \\ \phi_{21} & \phi_{22} & \phi_{23} \\ \phi_{31} & \phi_{32} & \phi_{33} \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix}. \quad (15.34)$$

Each homogeneous coordinate can be considered as a direction in 3D space (figure 14.11). So, this equation states that the left-hand side  $\tilde{\mathbf{x}}_i$  represents the same direction in space as the right-hand side,  $\Phi \tilde{\mathbf{w}}_i$ . If this is the case, their cross product must be zero, so that

$$\tilde{\mathbf{x}} \times \Phi \tilde{\mathbf{w}} = \mathbf{0}. \quad (15.35)$$

Writing this constraint in full gives the relations

Problem 15.9

$$\begin{bmatrix} y(\phi_{31}u + \phi_{32}v + \phi_{33}) - (\phi_{21}u + \phi_{22}v + \phi_{23}) \\ (\phi_{11}u + \phi_{12}v + \phi_{13}) - x(\phi_{31}u + \phi_{32}v + \phi_{33}) \\ x(\phi_{21}u + \phi_{22}v + \phi_{23}) - y(\phi_{11}u + \phi_{12}v + \phi_{13}) \end{bmatrix} = \mathbf{0}. \quad (15.36)$$

This appears to provide three linear constraints on the elements of  $\Phi$ . However, only two of these three equations are independent, so we discard the third. We now stack the first two constraints from each of the  $I$  pairs of points  $\{\mathbf{x}_i, \mathbf{w}_i\}$  to form the system

$$\begin{bmatrix} 0 & 0 & 0 & -u_1 & -v_1 & -1 & y_1u_1 & y_1v_1 & y_1 \\ u_1 & v_1 & 1 & 0 & 0 & 0 & -x_1u_1 & -x_1v_1 & -x_1 \\ 0 & 0 & 0 & -u_2 & -v_2 & -1 & y_2u_2 & y_2v_2 & y_2 \\ u_2 & v_2 & 1 & 0 & 0 & 0 & -x_2u_2 & -x_2v_2 & -x_2 \\ \vdots & \vdots \\ 0 & 0 & 0 & -u_I & -v_I & -1 & y_Iu_I & y_Iv_I & y_I \\ u_I & v_I & 1 & 0 & 0 & 0 & -x_Iu_I & -x_Iv_I & -x_I \end{bmatrix} \begin{bmatrix} \phi_{11} \\ \phi_{12} \\ \phi_{13} \\ \phi_{21} \\ \phi_{22} \\ \phi_{23} \\ \phi_{31} \\ \phi_{32} \\ \phi_{33} \end{bmatrix} = \mathbf{0}, \quad (15.37)$$

which has the form  $\mathbf{A}\phi = \mathbf{0}$ .

We solve this system of equations in a least squares sense with the constraint  $\phi^T \phi = 1$  to prevent the trivial solution  $\phi = \mathbf{0}$ . This is a standard problem (see appendix C.7.2). To find the solution, we compute the SVD  $\mathbf{A} = \mathbf{U}\mathbf{L}\mathbf{V}^T$  and choose  $\phi$  to be the last column of  $\mathbf{V}$ . This is reshaped into a  $3 \times 3$  matrix  $\Phi$  and used as a starting point for the nonlinear optimization of the true criterion (equation 15.33).

## 15.3 Inference in transformation models

We have introduced four transformations (Euclidean, similarity, affine, projective) that relate positions  $\mathbf{w}$  on a real-world plane to their projected positions  $\mathbf{x}$  in the

Algorithm 15.5

image, and discussed how to learn their parameters. In each case, the transformation took the form of a generative model  $Pr(\mathbf{x}|\mathbf{w})$ . In this section, we consider how to infer the world position  $\mathbf{w}$  from the image position  $\mathbf{x}$ .

For simplicity, we will take a maximum likelihood approach to this problem. For the generic transformation  $\text{trans}[\mathbf{w}_i, \theta]$ , we seek

$$\begin{aligned}\hat{\mathbf{w}} &= \underset{\mathbf{w}}{\operatorname{argmax}} [\log [\text{Norm}_{\mathbf{x}} [\text{trans}[\mathbf{w}, \theta], \sigma^2 \mathbf{I}]]] \\ &= \underset{\mathbf{w}}{\operatorname{argmin}} \left[ (\mathbf{x} - \text{trans}[\mathbf{w}, \theta])^T (\mathbf{x} - \text{trans}[\mathbf{w}, \theta]) \right].\end{aligned}\quad (15.38)$$

It is clear that this will be achieved when the image point and the predicted image point agree exactly so that

$$\mathbf{x} = \text{trans}[\mathbf{w}, \theta]. \quad (15.39)$$

We can find the  $\mathbf{w} = [u, v]^T$  that makes this true by moving to homogeneous coordinates. Each of the four transformations can be written in the form

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} \\ \phi_{21} & \phi_{22} & \phi_{23} \\ \phi_{31} & \phi_{32} & \phi_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \quad (15.40)$$

where the exact expressions are given by equations 15.4, 15.10, 15.13, and 15.17.

To find the position  $\mathbf{w} = [u, v]^T$ , we simply pre-multiply by the inverse of the transformation matrix to yield

$$\lambda' \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} \\ \phi_{21} & \phi_{22} & \phi_{23} \\ \phi_{31} & \phi_{32} & \phi_{33} \end{bmatrix}^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad (15.41)$$

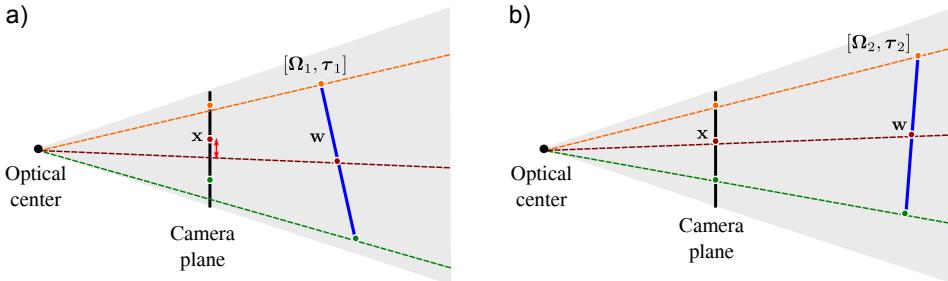
and then recover  $u$  and  $v$  by converting back to Cartesian co-ordinates.

## 15.4 Three geometric problems for planes

We have introduced transformation models that mapped from 2D coordinates on the plane to 2D coordinates in the image, the most general of which was the homography. Now we relate this model back to the full pinhole camera and revisit the three geometric problems from chapter 14 for the special case of a planar scene. We will show how to

- learn the extrinsic parameters (compute the geometric relationship between the plane and the camera),
- learn the intrinsic parameters (calibrate from a plane), and
- infer the 3D coordinates of a point in the plane relative to the camera, given its image position.

These three problems are illustrated in figures 15.8, 15.9 and 15.11, respectively.



**Figure 15.8** Problem 1 - Learning extrinsic parameters. Given points  $\{\mathbf{w}_i\}_{i=1}^I$  on a plane, their positions  $\{\mathbf{x}_i\}_{i=1}^I$  in the image and intrinsic parameters  $\Lambda$ , find the rotation  $\Omega$  and translation  $\tau$  relating the camera and the plane. a) When the rotation and translation are wrong, the image points predicted by the model (where the rays strike the image plane) do not agree well with the observed points  $\mathbf{x}$ . b) When the rotation and translation are correct, they agree well and the likelihood  $Pr(\mathbf{x}|\mathbf{w}, \Lambda, \Omega, \tau)$  will be high.

### 15.4.1 Problem 1: learning extrinsic parameters

Given  $I$  3D points  $\{\mathbf{w}_i\}_{i=1}^I$  that lie on a plane so that  $w_i = 0$ , their corresponding projections in the image  $\{\mathbf{x}_i\}_{i=1}^I$  and known intrinsic matrix  $\Lambda$ , estimate the extrinsic parameters

Algorithm 15.6

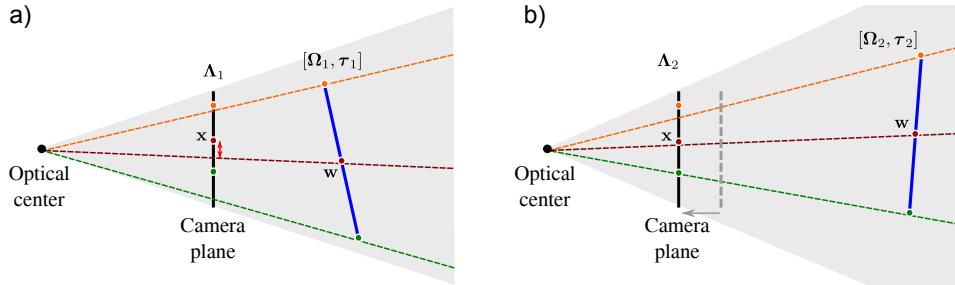
$$\begin{aligned}\hat{\Omega}, \hat{\tau} &= \underset{\Omega, \tau}{\operatorname{argmax}} \left[ \sum_{i=1}^I \log [Pr(\mathbf{x}_i | \mathbf{w}_i, \Lambda, \Omega, \tau)] \right] \\ &= \underset{\Omega, \tau}{\operatorname{argmax}} \left[ \sum_{i=1}^I \log [\text{Norm}_{\mathbf{x}_i}[\text{pinhole}[\mathbf{w}_i, \Lambda, \Omega, \tau], \sigma^2 \mathbf{I}]] \right], \quad (15.42)\end{aligned}$$

where  $\Omega$  is a  $3 \times 3$  rotation matrix and  $\tau$  is a  $3 \times 1$  translation vector (figure 15.8). The extrinsic parameters transform points  $\mathbf{w} = [u, v, 0]$  on the plane into the coordinate system of the camera. Unfortunately, this problem (still) cannot be solved in closed form and requires nonlinear optimization. As usual, it is possible to get a good initial estimate for the parameters using a closed form algebraic solution based on homogeneous coordinates.

From equation 15.16, the relation between a homogeneous point on the plane and its projection in the image is

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \lambda' \begin{bmatrix} \phi_x & \gamma & \delta_x \\ 0 & \phi_y & \delta_y \\ 0 & 0 & D \end{bmatrix} \begin{bmatrix} \omega_{11} & \omega_{12} & \tau_x \\ \omega_{21} & \omega_{22} & \tau_y \\ \omega_{31} & \omega_{32} & \tau_z \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} \\ \phi_{21} & \phi_{22} & \phi_{23} \\ \phi_{31} & \phi_{32} & \phi_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad (15.43)$$

Our approach is to (i) calculate the homography  $\Phi$  between the points  $\mathbf{w} = [u, v]^T$  on the plane and the points  $\mathbf{x} = [x, y]^T$  in the image using the method of



**Figure 15.9** Problem 2 - Learning intrinsic parameters. Given a set of points  $\{\mathbf{w}_i\}_{i=1}^I$  on a plane in the world (blue line) and the 2D positions  $\{\mathbf{x}_i\}_{i=1}^I$  of these points in an image, find the intrinsic parameters  $\Lambda$ . To do this, we must also simultaneously estimate the extrinsic parameters  $\Omega, \tau$ . a) When the intrinsic and extrinsic parameters are wrong, the prediction of the pinhole camera (where rays strike the image plane) deviates significantly from the known 2D points. b) When they are correct, the prediction of the model will agree with the observed image. To make the solution unique, the plane must be seen from several different angles.

section 15.2.4 and then (ii) decompose this homography to recover the rotation matrix  $\Omega$  and translation vector  $\tau$ .

As a first step toward this decomposition, we eliminate the effect of the intrinsic parameters by pre-multiplying the estimated homography by the inverse of the intrinsic matrix  $\Lambda$ . This gives a new homography  $\Phi' = \Lambda^{-1}\Phi$  such that

$$\begin{bmatrix} \phi'_{11} & \phi'_{12} & \phi'_{13} \\ \phi'_{21} & \phi'_{22} & \phi'_{23} \\ \phi'_{31} & \phi'_{32} & \phi'_{33} \end{bmatrix} = \lambda' \begin{bmatrix} \omega_{11} & \omega_{12} & \tau_x \\ \omega_{21} & \omega_{22} & \tau_y \\ \omega_{31} & \omega_{32} & \tau_z \end{bmatrix} \quad (15.44)$$

To estimate the first two columns of the rotation matrix  $\Omega$ , we compute the SVD of the first two columns of  $\Phi'$

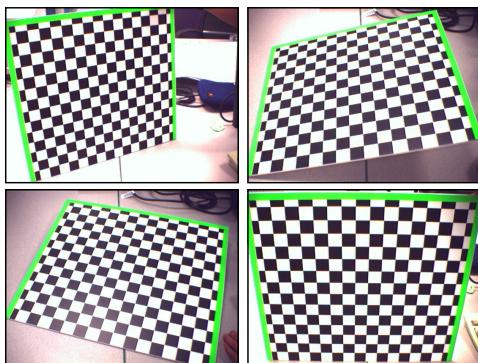
$$\begin{bmatrix} \phi'_{11} & \phi'_{12} \\ \phi'_{21} & \phi'_{22} \\ \phi'_{31} & \phi'_{32} \end{bmatrix} = \mathbf{U} \mathbf{L} \mathbf{V}^T, \quad (15.45)$$

and then set

$$\begin{bmatrix} \omega_{11} & \omega_{12} \\ \omega_{21} & \omega_{22} \\ \omega_{31} & \omega_{32} \end{bmatrix} = \mathbf{U} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{V}^T. \quad (15.46)$$

These operations find the closest valid first two columns of a rotation matrix to the first two columns of  $\Phi'$  in a least squares sense. This approach is very closely related to the solution to the orthogonal Procrustes problem (appendix C.7.3).

We then compute the last column  $[\omega_{13}, \omega_{23}, \omega_{33}]^T$  of the rotation matrix by taking the cross product of the first two columns: this guarantees a vector that is



**Figure 15.10** Calibration from a plane. It is considerably easier to make a 2D calibration target than to machine an accurate 3D object. Consequently, calibration is usually based on viewing planes such as this checkerboard. Unfortunately, a single view of a plane is not sufficient to uniquely determine the intrinsic parameters. Therefore cameras are usually calibrated using several images of the same plane, where the plane has a different pose relative to the camera in each image.

also length one and is perpendicular to the first two columns, but the sign may still be wrong: we test the determinant of the resulting rotation matrix  $\Omega$ , and if it is -1, we multiply the last column by -1.

We can now estimate the scaling factor  $\lambda'$  by taking the average of the scaling factors between these six elements

$$\lambda' = \frac{\sum_{m=1}^3 \sum_{n=1}^2 \phi'_{mn} / \omega_{mn}}{6}, \quad (15.47)$$

and this allows us to estimate the translation vector as  $\tau = [\phi'_{13}, \phi'_{23}, \phi'_{33}]^T / \lambda'$ . The result of this algorithm is a very good initial estimate of the extrinsic matrix  $[\Omega, \tau]$ , which can be improved by optimizing the correct objective function (equation 15.42).

### 15.4.2 Problem 2: learning intrinsic parameters

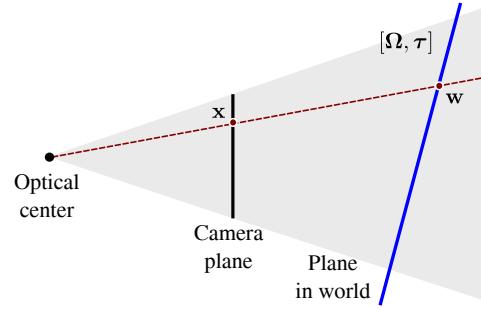
In this section, we revisit the problem of learning the intrinsic parameters of the camera. This is known as *camera calibration* (figure 15.9). In section 14.5, we developed a method based on viewing a special 3D calibration target. In practice, it is hard to manufacture a three-dimensional object with easy-to-find visual features at precisely known locations. However, it is easy to manufacture a 2D object of this kind. For example, it is possible to print out a checkerboard pattern and attach this to a flat surface (figure 15.10). Unfortunately, a single view of a 2D calibration object is not sufficient to uniquely identify the intrinsic parameters. However, observing the same pattern from several different viewpoints does suffice.

Algorithm 15.7

Hence, the calibration problem can be reformulated as follows: given a planar object, with  $I$  distinct 3D points  $\{\mathbf{w}_i\}_{i=1}^I$  on the surface and the corresponding projections in  $J$  images  $\{\mathbf{x}_{ij}\}_{i=1, j=1}^{I, J}$ , establish the intrinsic parameters in the form of the intrinsic matrix  $\Lambda$ :

$$\hat{\Lambda} = \underset{\Lambda}{\operatorname{argmax}} \left[ \underset{\Omega_1 \dots \Omega_J, \tau_1 \dots \tau_J}{\max} \left[ \sum_{i=1}^I \sum_{j=1}^J \log [Pr(\mathbf{x}_{ij} | \mathbf{w}_i, \Lambda, \Omega_j, \tau_j)] \right] \right]. \quad (15.48)$$

**Figure 15.11** Inferring a 3D position relative to the camera. When the object is planar (blue line), there is a one to one mapping between points  $\mathbf{x}$  in the image and points  $\mathbf{w}$  on the plane. If we know the intrinsic matrix, and the rotation and translation of the plane relative to the camera, we can infer the 3D position  $\mathbf{w}$  from the observed 2D image point  $\mathbf{x}$ .



A simple approach to this problem is to use a coordinate ascent technique in which we alternately

- estimate the  $J$  extrinsic matrices relating the object frame of reference to the camera frame of reference in each of the  $J$  images,

$$\hat{\Omega}_j, \hat{\tau}_j = \underset{\Omega_j, \tau_j}{\operatorname{argmax}} \left[ \sum_{i=1}^I \log [Pr(\mathbf{x}_{ij} | \mathbf{w}_i, \Lambda, \Omega_j, \tau_j)] \right], \quad (15.49)$$

using the method of the previous section and then,

- estimate the intrinsic parameters using a minor variation of the method described in section 14.5:

$$\hat{\Lambda} = \underset{\Lambda}{\operatorname{argmax}} \left[ \sum_{i=1}^I \sum_{j=1}^J \log [Pr(\mathbf{x}_{ij} | \mathbf{w}_i, \Lambda, \Omega_j, \tau_j)] \right]. \quad (15.50)$$

As for the original calibration method (section 14.5), a few iterations of this procedure will yield a useful initial estimate of the intrinsic parameters, and these can be improved by directly optimizing the true objective function (equation 15.48). It should be noted that this method is quite inefficient and is described for pedagogical reasons; it is easy both to understand and to implement. A modern implementation would use a more sophisticated technique to find the intrinsic parameters (see Hartley & Zisserman 2004).

### 15.4.3 Problem 3: inferring 3D position relative to camera

Given a calibrated camera (i.e., camera with known  $\Lambda$ ) that is known to be related to a planar scene by extrinsic parameters  $\Omega, \tau$ , find the 3D point  $\mathbf{w}$  that is responsible for a given 2D position  $\mathbf{x}$  in the image.

When the scene is planar, there is normally a one-to-one relationship between points in the 3D world and points in the image. To compute the 3D point corresponding to a point  $\mathbf{x}$ , we initially infer the position  $\mathbf{w} = [u, v, 0]^T$  on the plane. We exploit our knowledge of the intrinsic and extrinsic parameters to compute the homography  $\Phi$  mapping from points in the world to points in the image,

$$\mathbf{T} = \begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} \\ \phi_{21} & \phi_{22} & \phi_{23} \\ \phi_{31} & \phi_{32} & \phi_{33} \end{bmatrix} = \begin{bmatrix} \phi_x & \gamma & \delta_x \\ 0 & \phi_y & \delta_y \\ 0 & 0 & D \end{bmatrix} \begin{bmatrix} \omega_{11} & \omega_{12} & \tau_x \\ \omega_{21} & \omega_{22} & \tau_y \\ \omega_{31} & \omega_{32} & \tau_z \end{bmatrix}. \quad (15.51)$$

We then infer the coordinates  $\mathbf{w} = [u, v, 0]^T$  on the plane by inverting this transformation

$$\tilde{\mathbf{w}} = \mathbf{T}^{-1}\tilde{\mathbf{x}}. \quad (15.52)$$

Finally, we transfer the coordinates back to the frame of reference of the camera to give

$$\mathbf{w}' = \Omega\mathbf{w} + \boldsymbol{\tau}. \quad (15.53)$$

## 15.5 Transformations between images

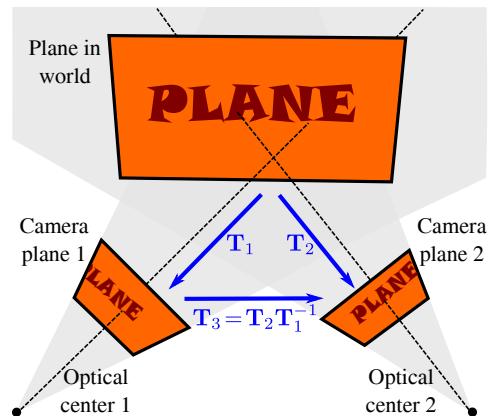
So far, we have considered transformations between a plane in the world and its image in the camera. We now consider two cameras viewing the same planar scene. The one-to-one mapping from positions on the plane to the positions in the first camera can be described by a homography. Similarly, the one-to-one mapping from positions on the plane to the positions in the second camera can be described by a second homography. It follows that there is a one-to-one mapping from the position in the first camera to the position in the second camera. This can also be described by a homography. The same logic follows for the other transformation types. Consequently, it is very common to find pairs of real-world images that are geometrically related by one of these transformations. For example, the image of the photograph in figure 15.5a is related by a homography to that in 15.5b.

Let us denote the  $3 \times 3$  matrix mapping points on the plane to points in the first image by  $\mathbf{T}_1$ . Similarly, we denote the  $3 \times 3$  matrix mapping points on the plane to points in the second image by  $\mathbf{T}_2$ . To map from image 1 to image 2, we first apply the transformation from image 1 to the plane itself. By the argument of the previous section, this is  $\mathbf{T}_1^{-1}$ . Then we apply the transformation from the plane to image 2, which is  $\mathbf{T}_2$ . The mapping  $\mathbf{T}_3$  from image 1 to image 2 is the concatenation of these operations and is hence  $\mathbf{T}_3 = \mathbf{T}_2\mathbf{T}_1^{-1}$  (figure 15.12).

### 15.5.1 Geometric properties of the homography

We've seen that transformations between planes in the world and the image plane are described by homographies, and so are transformations between multiple images of a real-world plane. There is another important family where the images are related to one another by the homography. Recall that the homography mapping point  $\mathbf{x}_1$  to point  $\mathbf{x}_2$  is linear in homogeneous coordinates:

**Figure 15.12** Transformations between images. Two cameras view the same planar scene. The relations between the 2D points on this plane and the two images are captured by the  $3 \times 3$  transformation matrices  $\mathbf{T}_1$  and  $\mathbf{T}_2$ , respectively. It follows that the transformation from the first image to the points on the plane is given by  $\mathbf{T}_1^{-1}$ . We can compute the transformation  $\mathbf{T}_3$  from the first image to the second image by transforming from the first image to the plane and then transforming from the plane to the second image, giving the final result  $\mathbf{T}_3 = \mathbf{T}_2 \mathbf{T}_1^{-1}$ .



$$\lambda \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} \\ \phi_{21} & \phi_{22} & \phi_{23} \\ \phi_{31} & \phi_{32} & \phi_{33} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}. \quad (15.54)$$

The homogeneous coordinates represent 2D points as directions or rays in a 3D space (figure 14.11). When we apply a homography to a set of 2D points, we can think of this as applying a linear transformation (rotation, scaling, and shearing) to a bundle of rays in 3D. The positions where the transformed rays strike the plane at  $w = 1$  determine the final 2D positions.

We could yield the same results by keeping the rays fixed and applying the inverse transformation to the plane so that it cuts the rays in a different way. Since any plane can be mapped to any other plane by a linear transformation, it follows that the images created by cutting a ray bundle with different planes are all related to one another by homographies (figure 15.13). In other words, the images seen by different cameras *with the pinhole in the same place* are related by homographies. So, for example, if a camera zooms (the focal length increases), then the images before and after the zoom are related by a homography.

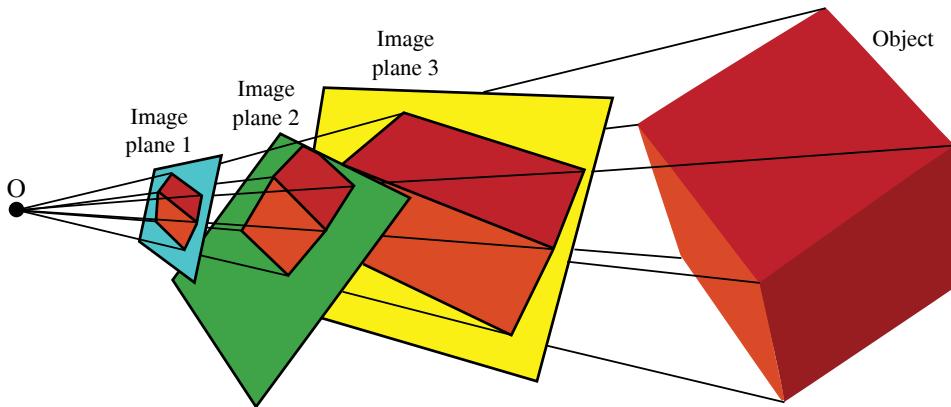
This relationship encompasses an important special case (figure 15.14). If the camera rotates *but does not translate*, then the image plane still intersects the same set of rays. It follows that the projected points  $\mathbf{x}_1$  before the rotation and the projected points  $\mathbf{x}_2$  after the rotation are related by a homography. It can be shown that the homography  $\Phi$  mapping from image 1 to image 2 is given by

$$\Phi = \Lambda \Omega_2 \Lambda^{-1}, \quad (15.55)$$

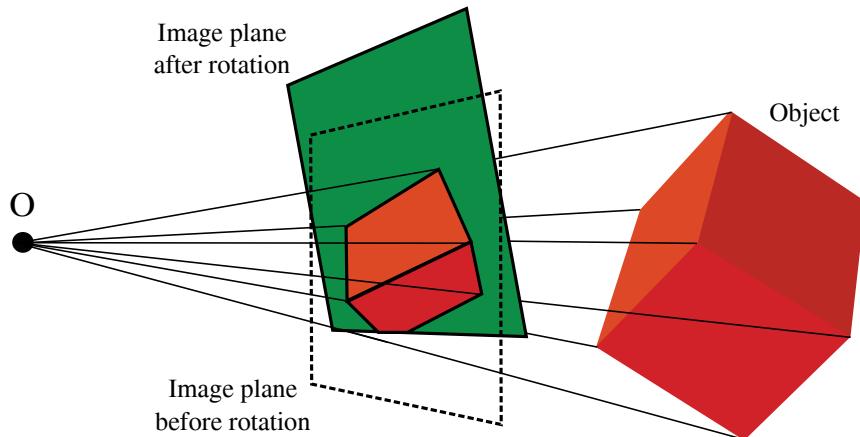
where  $\Lambda$  is the intrinsic matrix and  $\Omega_2$  is the rotation matrix that maps the coordinate system of the second camera to the first. This relationship is exploited when we stitch together images to form panoramas (section 15.7.2).

In conclusion, the homography maps between:

- points on a plane in the world and their positions in an image,



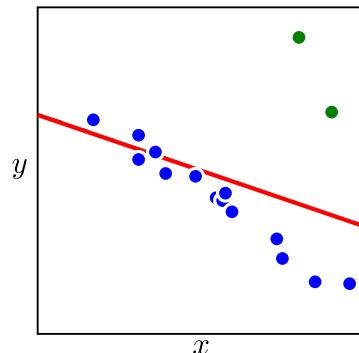
**Figure 15.13** Geometric interpretation of homography. A ray bundle is formed by connecting rays from an optical center to points on a real-world object (the cube). A set of planes cut the ray bundle, forming a series of images of the cube. Each of these images is related to every other by a homography.



**Figure 15.14** Images under pure camera rotation. When the camera rotates but does not translate, the bundle of rays remains the same, but is cut by a different plane. It follows that the two images are related by a homography.

- points in two different images of the same plane, and
- two images of a 3D object where the camera has rotated but not translated.

**Figure 15.15** Motivation for random sample consensus (RANSAC). The majority of this data set (blue points) can be explained well by a linear regression model, but there are two outliers (green points). Unfortunately, if we fit the linear regression model to all of this data, the mean prediction (red line) is dragged toward the outliers and no longer describes the majority of the data well. The RANSAC algorithm circumvents this problem by establishing which data points are outliers and fitting the model to the remaining data.



### 15.5.2 Computing transformations between images

In the previous sections we have argued that it is common for two images to be related to one another by a homography. If we denote the points in image 1 by  $\mathbf{x}_i$  and their corresponding positions in image 2 as  $\mathbf{y}_i$ , then we could for example describe the mapping as a homography

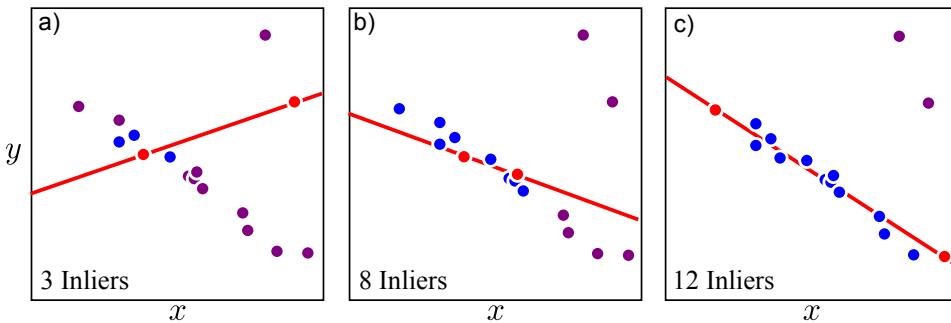
$$Pr(\mathbf{x}_i|\mathbf{y}_i) = \text{Norm}_{\mathbf{x}_i} [\mathbf{hom}[\mathbf{y}_i, \Phi], \sigma^2 \mathbf{I}] . \quad (15.56)$$

This method ascribes all of the noise to the first image and it should be noted that it is not quite correct: we should really build a model that explains both sets of image data with a set of hidden variables representing the original 3D points so that the estimated point positions in each image are subject to noise. Nonetheless, the model in equation 15.56 works well in practice. It is possible to learn the parameters using the technique of section 15.2.

## 15.6 Robust learning of transformations

We have discussed transformation models that can be used either to (i) map the positions of points on a real-world plane to their projections in the image or (ii) map the positions of points in one image to their corresponding positions in another. Until now, we have assumed that we know a set of corresponding points from which to learn the parameters of the transformation model. However, establishing these correspondences automatically is a challenging task in itself.

Let us consider the case where we wish to compute a transformation between two images. A simple method to establish correspondences would be to compute interest points in each image and to characterize the region around each point using a region descriptor such as the SIFT descriptor (section 13.3.2). We could then greedily associate points based on the similarity of the region descriptors (as in figure 15.17c-d). Depending on the scene, this is likely to produce a set of matches that are 70 – 90% correct. Unfortunately the remaining erroneous



**Figure 15.16** RANSAC procedure. a) We select a random minimal subset of points to fit the line (red points). We fit the line to these points and count how many of the other points agree with this solution (blue points). These are termed inliers. Here there are only three inliers. b,c) This procedure is repeated with different minimal subsets of points. After a number of iterations, we choose the fit that had the most inliers. We refit the line using only the inliers from this fit.

correspondences (which we will call *outliers*) can severely hamper our ability to compute the transformation between the images. To cope with this problem, we need robust learning methods.

### 15.6.1 RANSAC

*Random sample consensus* or *RANSAC* is a general method for fitting models to data where the data are corrupted by outliers. These outliers violate the assumptions of the underlying probability model (usually a normal distribution) and can cause the estimated parameters to deviate significantly from their correct values.

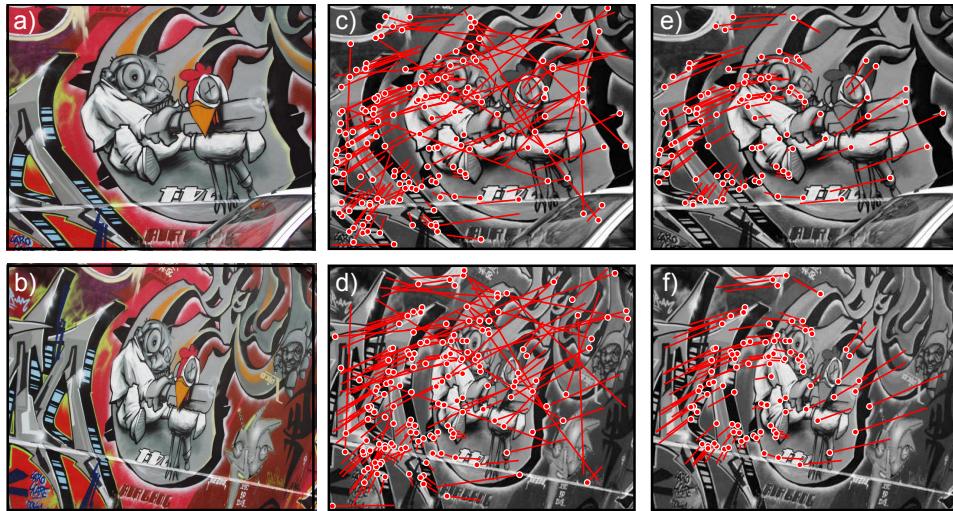
For pedagogical reasons, we will describe RANSAC using the example of linear regression. We will subsequently return to the problem of learning parameters in transformation models. The linear regression model is

$$Pr(y|x) = \text{Norm}_y [ax + b, \sigma^2] \quad (15.57)$$

and was discussed at length in section 8.1. Figure 15.15 shows an example of the predicted mean for  $y$  when we fit the model to data with two outliers. The fit has been unduly influenced by the outliers and no longer describes the data well.

The goal of RANSAC is to identify which points are outliers and to eliminate them from the final fit. This is a chicken and egg problem: if we had the final fit, then it would be easy to identify the outliers (they are not well described by the model), and if we knew which points were outliers, it would be easy to compute the final fit.

RANSAC works by repeatedly fitting models based on random subsets of the data. The hope is that sooner or later, there will be no outliers in the chosen subset,



**Figure 15.17** Fitting a homography with RANSAC. a,b) Original images of a textured plane. c,d) 162 strongest matches selected by greedy method. In each case the associated line travels from the interest point to the position of its match in the other image. These matches are clearly polluted by outliers. e,f) Model fitted from 102 inliers identified by applying 100 iterations of RANSAC. These matches form a coherent pattern as they are all described by a homography.

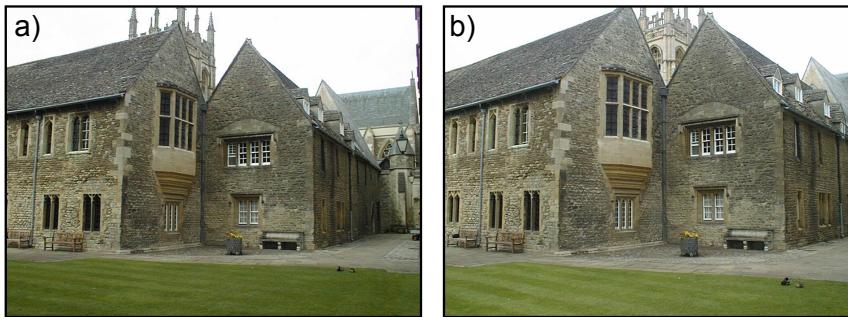
and so we will fit a good model. To enhance the probability of this happening, RANSAC chooses subsets of the minimal size required to uniquely fit the model. For example, in the case of the line, it would choose subsets of size two.

Having chosen a minimal subset of data points and fitted the model, RANSAC assesses its quality. It does this by classifying points as inliers or outliers. This requires some knowledge about the expected amount of variation around the true model. For linear regression this means we need some prior knowledge of the variance parameter  $\sigma^2$ . If a data point exceeds the expected variation (perhaps two standard deviations from the mean), then it is classified as an outlier. Otherwise, it is an inlier. For each minimal subset of data we count the number of inliers.

We repeat this procedure a number of times: on each iteration we choose a random minimal subset of points, fit a model, and count the number of data points that agree (the inliers). After a predetermined number of iterations, we then choose the model that had the most inliers and re-fit the model from these alone.

The complete RANSAC algorithm hence proceeds as follows (figure 15.16)

1. Randomly choose a minimal subset of data.
2. Use this subset to estimate the parameters.
3. Compute the number of inliers for this model.
4. Repeat steps 1-3 a fixed number of times.



**Figure 15.18** Piecewise planarity. a) Although this scene is clearly not well described by a plane, it is well described by a set of planes. b) Consequently, the mapping to this second image of the same scene can be described by a set of homographies. Images from Oxford Colleges dataset.

### 5. Re-estimate model using inliers from the best fit.

If we know the degree to which our data are polluted with outliers, it is possible to estimate the number of iterations that provide any given chance of finding the correct answer.

Now let us return to the question of how to apply the model to fitting geometric transformations. We will use the example of a homography (equation 15.56). Here we repeatedly choose subsets of hypothesized matches between the two images. The size of the subset is chosen to be four as this is the minimum number of pairs of points that are needed to uniquely identify the eight degrees of freedom of the homography.

For each subset, we count the number of inliers by evaluating equation 15.56 and selecting those where the likelihood exceeds some pre-determined value. In practice, this means measuring the distance between the points  $\mathbf{x}_i$  in the first image and the mapped position  $\text{hom}[\mathbf{y}, \Phi]$  of the points from the second image. After repeating this many times, we identify the trial with the most inliers (figure 15.17) and recompute the model from these inliers alone.

Problem 15.12

Algorithm 15.8

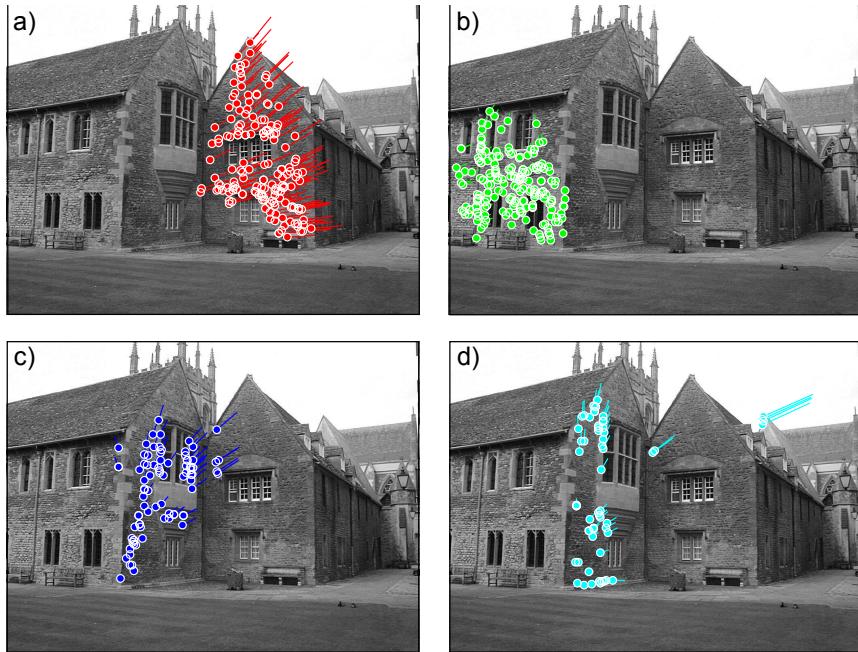
Problem 15.13

## 15.6.2 Sequential RANSAC

Now let us look at a more challenging task. So far, we assumed that one set of points could be mapped to another using a single transformation model. However, many scenes containing man-made objects are piecewise planar (figure 15.18). It follows that images of such scenes are related by piecewise homographies. In this section, we will consider methods for fitting this type of model.

Algorithm 15.9

The first approach is *sequential RANSAC*. The idea is simple: we fit a single homography to the scene using the RANSAC method. In principle this will correctly fit a model to one of the planes in the image and reject all the other points



**Figure 15.19** Sequential fitting of homographies mapping between images in figure 15.18 using RANSAC. a) Running the RANSAC fitting procedure once identifies a set of points that all lie on the same plane in the image. b) Running the RANSAC procedure again on the points that were not explained by the first plane identifies a second set of points that lie on a different plane. c) Third iteration. the plane discovered does not correspond to a real plane in the scene. d) Fourth iteration. The model erroneously associates parts of the image which are quite separate. This sequential approach fails for two reasons. First, it is greedy and cannot recover from earlier errors. Second, it does not encourage spatial smoothness (nearby points should belong to the same model).

as outliers. We then remove the points that belong to this plane (i.e., the inliers to the final homography) and repeat the procedure on the remaining points. Ideally, each iteration will identify a new plane in the image.

Unfortunately, this method does not work well in practice (figure 15.19) for two reasons. First, the algorithm is greedy: any matches that are erroneously incorporated into, or missed by one of the earlier models cannot be correctly assigned later. Second, the model has no notion of spatial coherence, ignoring the intuition that nearby points are more likely to belong to the same plane.

### 15.6.3 PEaRL

The *Propose, Expand and Re-Learn* or *PEaRL* algorithm solves both of these problems. In the ‘propose’ stage,  $K$  hypothetical models are generated where  $K$  is usually of the order of several thousand. This can be achieved by using a RANSAC type procedure in which minimal subsets of points are chosen, a model is fitted, and the inliers are counted. This is done repeatedly until we have several thousand models  $\{\Phi_k\}_{k=1}^K$  that each have a reasonable degree of support.

Algorithm 15.10

In the ‘expand’ stage, we model the assignment of matched pairs to the proposed models as a multi-label Markov random field. We associate a label  $l_i \in [1, 2, \dots, K]$  to each match  $\{\mathbf{x}_i, \mathbf{y}_i\}$  where the value of the label determines which one of the  $K$  models is present at this point. The likelihood of each data pair  $\{\mathbf{x}_i, \mathbf{y}_i\}$  under the  $k^{th}$  model is given by

$$Pr(\mathbf{x}_i | \mathbf{y}_i, l_i = k) = \text{Norm}_{\mathbf{x}_i} [\mathbf{hom}[\mathbf{y}_i, \Phi_k], \sigma^2 \mathbf{I}] . \quad (15.58)$$

To incorporate spatial coherence, we choose a prior over the labels that encourages neighbors to take similar values. In particular, we choose an MRF with a Potts model potential

$$Pr(\mathbf{l}) = \frac{1}{Z} \exp \left[ - \sum_{i,j \in \mathcal{N}_p} w_{ij} \delta[l_i - l_j] \right] , \quad (15.59)$$

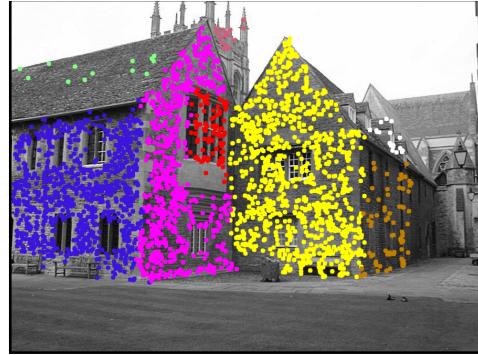
where  $Z$  is the constant partition function,  $w_{ij}$  is a weight associated with the pair of matches  $i$  and  $j$ , and  $\delta[\bullet]$  is a function that returns 1 when the argument is zero and returns 0 otherwise. The neighborhood  $\mathcal{N}_i$  of each point must be chosen in advance. One simple technique is to compute the  $K$ -nearest neighbors for each point in the first image and declare two points to be neighbors if either is in the other’s set of closest neighbors. The weights  $w_{ij}$  are chosen so that points that are close in the image are more tightly coupled than points that are distant.

The goal of the ‘expand’ stage is then to infer the labels  $l_i$  and hence the association between models and data points. This can be accomplished using the alpha expansion algorithm (section 12.4.1). Finally, having associated each data point with one of the models, we move to the ‘relearn’ stage. Here, we re-estimate the parameters of each model based on the data that was associated with it. The ‘expand’ and ‘relearn’ stages are iterated until no further progress is made. At the end of this process, we throw away any models that do not have sufficient support in the final solution (figure 15.20).

## 15.7 Applications

In this section we present two examples of the techniques in this chapter. First, we discuss augmented reality in which we attempt to render an object onto a plane in the scene. This application exploits the fact that there is a homography between the image of the plane and the original object surface, and uses the method of

**Figure 15.20** Results of the PEaRL algorithm. It formulates the problem as inference in a multi-label MRF. The MRF label associated with each matching pair denotes the index of one of a set of possible models. Inferring these labels is alternated with refining the parameters of the proposed models. The colored points denote different labels in the final solution. The algorithm has successfully identified many of the surfaces in the scene. Adapted from Isack & Boykov (2012). ©2012 Springer.



section 15.4.1 to decompose this homography to find the relative position of the camera and the plane. Second, we discuss creating visual panoramas. The method exploits the fact that multiple images taken from a camera that has rotated but not translated are all related by homographies.

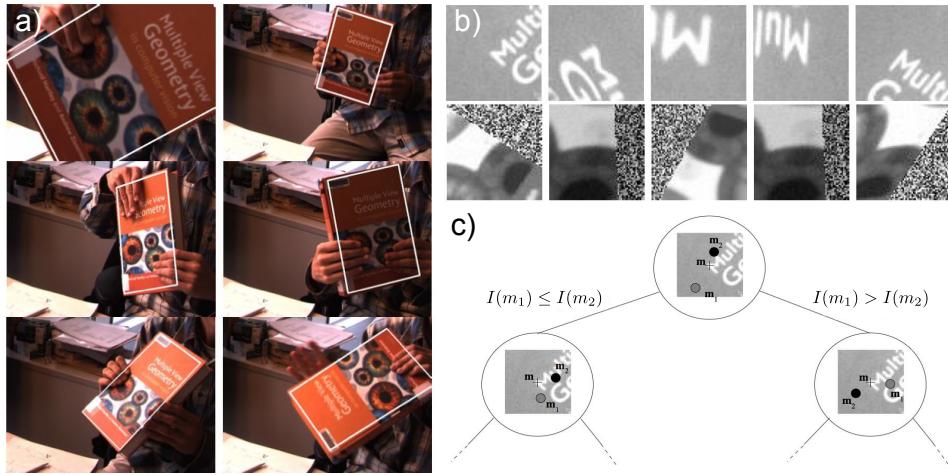
### 15.7.1 Augmented reality tracking

Figure 15.1 shows an example of augmented reality tracking. To accomplish this, the following steps were taken. First, the four corners of the marker are found as in figure 15.1c; the marker was designed so that the corners can be easily identified using a set of image processing operations. In brief, the image is thresholded, and then connected dark regions are found. The pixels around the edge of each region are identified. Then four 2D lines are fit to the edge pixels using sequential RANSAC. Regions that are not well explained by four lines are discarded. The only remaining region in this case is the square marker.

The positions where the fitted lines intersect provides four points  $\{\mathbf{x}_i\}_{i=1}^4$  in the image, and we know the corresponding positions  $\{\mathbf{w}_i\}_{i=1}^4$  in cm that occur on the surface of the planar marker. It is assumed that the camera is calibrated, and so we know the intrinsic matrix  $\Lambda$ . We can now compute the extrinsic parameters  $\Omega, \tau$  using the algorithm from section 15.4.1.

To render the graphical object, we first set up a viewing frustum (the graphics equivalent of the ‘camera’) so that it has the same field of view as specified by the intrinsic parameters. Then we render the model from the appropriate perspective using the extrinsic parameters as the *modelview matrix*. The result is that the object appears to be attached rigidly to the scene.

In this system, the points on the marker were found using a sequence of image-processing operations. However, this method is rather outdated. It is now possible to reliably identify natural features on an object so there is no need to use a marker with any special characteristics. One way to do this is to match SIFT features between a reference image of the object and the current scene. These interest point descriptors are invariant to image scaling and rotation, and for textured objects it is usually possible to generate a high percentage of correct matches. RANSAC is



**Figure 15.21** Robust tracking using keypoints. a) Lepetit *et al.* (2005) presented a system that automatically tracked objects such as this book. b) In the learning stage, the regions around the keypoints were subjected to a number of random affine transformations. c) Keypoints in the image were classified as belonging to a known keypoint on the object, using a tree-based classifier that compared the intensity at nearby points. Adapted from Lepetit *et al.* (2005). ©2005 IEEE.

used to eliminate any mis-matched pairs.

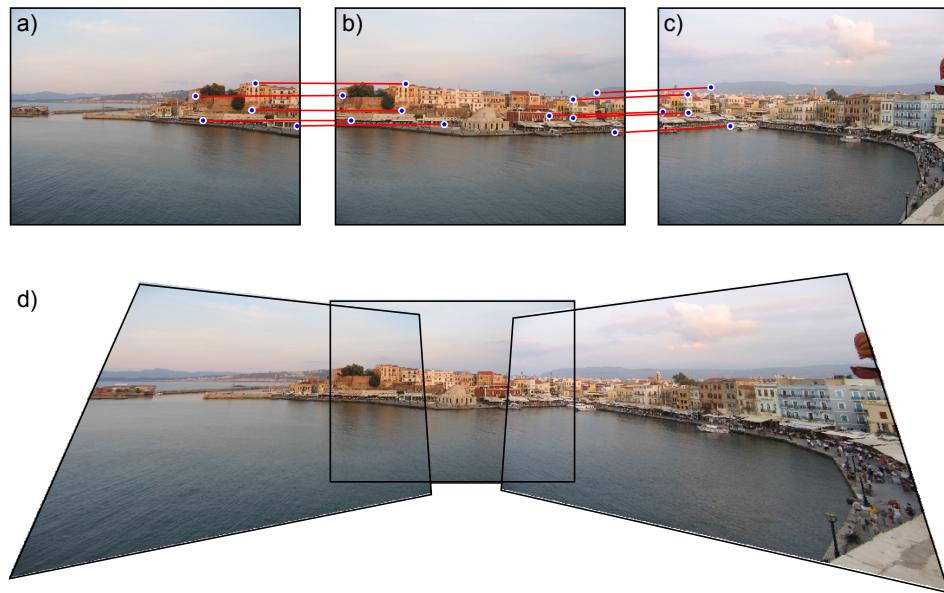
However, SIFT features are themselves relatively slow to compute. Lepetit *et al.* (2005) described a system that applies machine learning techniques to identify objects at interactive speeds (figure 15.21). They first identify  $K$  stable keypoints (e.g., Harris corners) on the object to be tracked. They then create a training set for each keypoint by subjecting the region around it to a large number of random affine transformations. Finally, they train a multi-class classification tree that takes a new keypoint and assigns it to match one of the  $K$  original points. At each branch of the tree, the decision is made by a pairwise intensity comparison. This system works very quickly and reliably matches most of the feature points. Once more, RANSAC can be used to eliminate any erroneous matches.

### 15.7.2 Visual panoramas

A second application of the ideas of this chapter is the computation of visual panoramas. Recall that a set of pictures that are taken by rotating a camera about the optical center are all related by homographies. Hence, if we take pictures of this kind that are partially overlapping, it is possible to map them all into one large image. This process is known as *image mosaicing*.

Problem 15.14

An example is shown in figure 15.22. This was constructed by placing the second



**Figure 15.22** Computing visual panoramas. a-c) Three images of the same scene where the camera has rotated but not translated. Five matching points have been identified by hand between each pair. d) A panorama can be created by mapping the first and third images into the frame of reference of the second image.

(middle) photo into the center of a much larger empty image. Then 5 matches were identified by hand between this expanded second image and the first image and the homography from the expanded second image to the first image was computed. For each empty pixel in the expanded second image we compute the position in the first image using this homography. If this falls within the image boundaries, we copy the pixel value.

This process can be completely automated by finding features and fitting a homography to each pair of images using a robust technique such as RANSAC. In a real system, the final result is typically projected onto a cylinder and unrolled to give a more visually pleasing result.

## Discussion

This chapter has presented a number of important ideas. First, we discussed a family of transformations and how each can be related to a camera viewing the scene under special conditions. These transformations are used widely within machine vision and we will see them exploited in chapters 17, 18 and 19. Second, we

discussed a more practical method for camera calibration based on viewing a plane at a number of different orientations. Finally, we have also presented RANSAC, which is a robust method to fit models, even in the presence of noisy observations.

## Notes

**Transformations:** For more information concerning the hierarchy of 2D transformations, consult Hartley & Zisserman (2004). The closed form solutions for the rotation and similarity transformations are special cases of *Procrustes problems*. Many more details about this type of problem can be found in Gower & Dijksterhuis (2004). The direct linear transformation algorithm for estimating the homography dates back to at least Sutherland (1963). Hartley & Zisserman (2004) present a detailed discussion of different objective functions for estimating homographies. In this chapter, we have discussed the estimation of transformations from point matches. However, it is also possible to compute transformations from other geometric primitives. For example, methods exist to compute the homography from matched lines (see problem 15.3), a combination of points and lines (Murino *et al.* 2002), or conics (Sugimoto 2000).

**Robust estimation:** The RANSAC algorithm is due to Fischler & Bolles (1981). It has spawned many variants, the most notable of which is MLESAC (Torr & Zisserman 2000) which puts this fitting method on a sound probabilistic footing (see problem 15.13 for a related model). Chum *et al.* (2005) and Frahm & Pollefeys (2006) present variants of RANSAC that can cope with degenerate data (where the model is non-unique). Torr (1998) and Vincent & Laganiere (2001) used RANSAC to sequentially estimate multiple geometric entities. Both Raguram *et al.* (2008) and Choi *et al.* (2009) present recent quantitative comparisons of variations of the RANSAC algorithm. The PEaRL algorithm is due to Isack & Boykov (2012). In the original paper, they also include an extra cost which encourages parsimony (to describe the data with as few models as possible) Other approaches to robust estimation include (i) the use of long-tailed distributions such as the t-distribution (section 7.5), (ii) M-estimators (Huber 2009) which replace the least-squares criterion with another function that penalizes large deviations less stringently, and (iii) the self explanatory least median of squares regression (Rousseeuw 1984).

**Augmented reality:** Pose estimation methods for augmented reality initially relied on detecting special patterns in the scene known as fiducial markers. Early examples used circular patterns (e.g. Cho *et al.* 1998; State *et al.* 1996) but these were largely supplanted by square markers (e.g., Rekimoto 1998; Kato *et al.* 2000; Kato & Billinghurst 1999; Koller *et al.* 1997). The system described in the text is ARToolkit (Kato & Billinghurst 1999; Kato *et al.* 2000) and can be downloaded from <http://www.hitl.washington.edu/artoolkit/>.

Other systems have used ‘natural image features.’ For example, Harris (1992) estimated the pose of an object using line segments. Simon *et al.* (2000) and Simon & Berger (2002) estimated the pose of planes in the image using the results of a corner detector. More information about computing the pose of a plane can be found in Sturm (2000).

More recent systems have used interest point detectors such as SIFT features which are robust to changes in illumination and pose (e.g., Skrypnyk & Lowe 2004). To increase the speed of systems, features are matched using machine learning techniques (Lepetit & Fua 2006; Özysal *et al.* 2010) and current systems can now operate at interactive speeds on mobile hardware (Wagner *et al.* 2008). A review of methods to estimate and track the pose of rigid objects can be found in Lepetit & Fua (2005).

**Calibration from a plane:** Algorithms for calibration from several views of a plane can be found in Sturm & Maybank (1999) and Zhang (2000). They are now used much more frequently than calibration based on 3D objects, for the simple reason that accurate 3D objects are harder to manufacture.

**Image mosaics:** The presented method for computing a panorama by creating a mosaic of images is naïve in a number of ways. First, it is more sensible to explicitly estimate

the rotation matrix and calibration parameters rather than the homography (Szeliski & Shum 1997; Shum & Szeliski 2000; Brown & Lowe 2007). Second, the method that we describe projects all of the images onto a single plane, but this does not work well when the panorama is too wide as the images become increasingly distorted. A more sensible approach is to project images onto a cylinder (Szeliski 1996; Chen 1995) which is then unrolled and displayed as an image. Third, the method to blend together images is not discussed. This is particularly important if there are moving objects in the image. A good review of these and other issues can be found in Szeliski (2006) and chapter 9 of Szeliski (2010).

## Problems

**Problem 15.1** The 2D point  $\mathbf{x}_2$  is created by a rotating point  $\mathbf{x}_1$  using the rotation matrix  $\Omega_1$  and then translating it by the translation vector  $\tau_1$  so that

$$\mathbf{x}_2 = \Omega_1 \mathbf{x}_1 + \tau_1.$$

Find the parameters  $\Omega_2$  and  $\tau_2$  of the inverse transformation

$$\mathbf{x}_1 = \Omega_2 \mathbf{x}_2 + \tau_2$$

in terms of the original parameters  $\Omega_1$  and  $\tau_1$ .

**Problem 15.2** A 2D line can be expressed as  $ax + by + c = 0$  or in homogeneous terms

$$\mathbf{l}\tilde{\mathbf{x}} = 0,$$

where  $\mathbf{l} = [a, b, c]$ . If points are transformed so that

$$\tilde{\mathbf{x}}' = \mathbf{T}\tilde{\mathbf{x}},$$

what is the equation of the transformed line?

**Problem 15.3** Using your solution from problem 15.2, develop a linear algorithm for estimating a homography based on a number of matched lines between the two images (i.e., the analogue of the DLT algorithm for matched lines).

**Problem 15.4** A conic (see problem 14.6) is defined by

$$\tilde{\mathbf{x}}^T \mathbf{C} \tilde{\mathbf{x}} = 0,$$

where  $\mathbf{C}$  is a  $3 \times 3$  matrix. If the points in the image undergo the transformation

$$\tilde{\mathbf{x}}' = \mathbf{T}\tilde{\mathbf{x}},$$

then what is the equation of the transformed conic?

**Problem 15.5** All of the 2D transformations in this chapter (Euclidean, similarity, affine, projective) have 3D equivalents. For each class write out the  $4 \times 4$  matrix that describes the 3D transformation in homogeneous coordinates. How many independent parameters does each model have?

**Problem 15.6** Devise an algorithm to estimate a 3D affine transformation based on two sets of matching 3D points. What is the minimum number of points required to get a unique estimate of the parameters of this model?

**Problem 15.7** A 1D affine transformation acts on 1D points  $x$  as  $x' = ax + b$ . Show that the ratio of two distances is *invariant* to a 1D affine transformation so that

$$I = \frac{x_1 - x_2}{x_2 - x_3} = \frac{x'_1 - x'_2}{x'_2 - x'_3}.$$

**Problem 15.8** A 1D projective transform acts on 1D points  $x$  as  $x' = (ax + b)/(cx + d)$ . Show that the *cross-ratio* of distances is *invariant* to a 1D projective transformation so that

$$I = \frac{(x_3 - x_1)(x_4 - x_2)}{(x_3 - x_2)(x_4 - x_1)} = \frac{(x'_3 - x'_1)(x'_4 - x'_2)}{(x'_3 - x'_2)(x'_4 - x'_1)}.$$

It was proposed at one point to exploit this type of invariance to recognize planar objects under different transformations (Rothwell *et al.* 1995). However, this is rather impractical as it assumes that we can identify a number of the points on the object in a first place.

**Problem 15.9** Show that equation 15.36 follows from equation 15.35.

**Problem 15.10** A camera with intrinsic matrix  $\Lambda$  and extrinsic parameters  $\Omega = \mathbf{I}, \tau = \mathbf{0}$  takes an image and then rotates to a new position  $\Omega = \Omega_1, \tau = \mathbf{0}$  and takes a second image. Show that the homography relating these two images is given by

$$\Phi = \Lambda \Omega_1 \Lambda^{-1}.$$

**Problem 15.11** Consider two images of the same scene taken from a camera that rotates, but does not translate between taking the images. What is the minimum number of point matches required to recover a 3D rotation between two images taken using a camera where the intrinsic matrix is known?

**Problem 15.12** Consider the problem of computing a homography from point matches that include outliers. If 50% of the initial matches are correct, how many iterations of the RANSAC algorithm would we expect to have to run in order to have a 95% chance of computing the correct homography?

**Problem 15.13** A different approach to fitting transformations in the presence of outliers is to model the uncertainty as a mixture of two Gaussians. The first Gaussian models the image noise, and the second Gaussian, which has a very large variance, accounts for the outliers. For example, for the affine transformation we would have

$$Pr(\mathbf{x}|\mathbf{w}) = \lambda \text{Norm}_{\mathbf{x}} [\mathbf{aff}[\mathbf{w}, \Phi, \tau], \sigma^2 \mathbf{I}] + (1 - \lambda) \text{Norm}_{\mathbf{x}} [\mathbf{aff}[\mathbf{w}, \Phi, \tau], \sigma_0^2 \mathbf{I}] +$$

where  $\lambda$  is the probability of being an inlier,  $\sigma^2$  is the image noise and  $\sigma_0^2$  is the large variance that accounts for the outliers. Sketch an approach to learning the parameters  $\sigma^2, \Phi, \tau$ , and  $\lambda$  of this model. You may assume that  $\sigma_0^2$  is fixed. Identify a possible weakness of this model.

**Problem 15.14** In the description of how to compute the panorama (section 15.7.2), it is suggested that we take each pixel in the central image and transform it into the other images and then copy the color. What is wrong with the alternate strategy of taking each pixel from the other images and transforming them into the central image?

# Chapter 16

## Multiple cameras

This chapter extends the discussion of the pinhole camera model. In chapter 14 we showed how to find the 3D position of a point based on its projections into multiple cameras. However, this approach was contingent on knowing the intrinsic and extrinsic parameters of these cameras, and this information is often unknown. In this chapter, we discuss methods for reconstruction in the absence of such information. Before reading this chapter, readers should ensure that they are familiar with the mathematical formulation of the pinhole camera (section 14.1).

To motivate these methods, consider a single camera moving around a static object. The goal is to build a 3D model from the images taken by the camera. To do this, we will also need to simultaneously establish the properties of the camera and its position in each frame. This problem is widely known as *structure from motion*, although this is something of a misnomer as both ‘structure’ and ‘motion’ are recovered simultaneously.

The structure from motion problem can be stated formally as follows. We are given  $J$  images of a rigid object that is characterized by  $I$  distinct 3D points  $\{\mathbf{w}_i\}_{i=1}^I$ . The images are taken with the same camera at a series of unknown positions. Given the projections  $\{\mathbf{x}_{ij}\}_{i=1,j=1}^{I,J}$  of the  $I$  points in the  $J$  images, establish the 3D positions  $\{\mathbf{w}_i\}_{i=1}^I$  of the points in the world, the fixed intrinsic parameters  $\boldsymbol{\Lambda}$ , and the extrinsic parameters  $\{\boldsymbol{\Omega}_j, \boldsymbol{\tau}_j\}_{j=1}^J$  for each image:

$$\begin{aligned} & \{\hat{\mathbf{w}}_i\}_{i=1}^I, \{\hat{\boldsymbol{\Omega}}_j, \hat{\boldsymbol{\tau}}_j\}_{j=1}^J, \hat{\boldsymbol{\Lambda}} \\ &= \underset{\mathbf{w}, \boldsymbol{\Omega}, \boldsymbol{\tau}, \boldsymbol{\Lambda}}{\operatorname{argmax}} \left[ \sum_{i=1}^I \sum_{j=1}^J \log [Pr(\mathbf{x}_{ij} | \mathbf{w}_i, \boldsymbol{\Lambda}, \boldsymbol{\Omega}_j, \boldsymbol{\tau}_j)] \right] \\ &= \underset{\mathbf{w}, \boldsymbol{\Omega}, \boldsymbol{\tau}, \boldsymbol{\Lambda}}{\operatorname{argmax}} \left[ \sum_{i=1}^I \sum_{j=1}^J \log [\text{Norm}_{\mathbf{x}_{ij}} [\text{pinhole}[\mathbf{w}_i, \boldsymbol{\Lambda}, \boldsymbol{\Omega}_j, \boldsymbol{\tau}_j], \sigma^2 \mathbf{I}]] \right]. \end{aligned} \tag{16.1}$$

Since this objective function is based on the normal distribution, we can reformulate it as a least-squares problem of the form

$$\{\hat{\mathbf{w}}_i\}_{i=1}^I, \{\hat{\boldsymbol{\Omega}}_j, \hat{\boldsymbol{\tau}}_j\}_{j=1}^J, \hat{\boldsymbol{\Lambda}} = \underset{\mathbf{w}, \boldsymbol{\Omega}, \boldsymbol{\tau}, \boldsymbol{\Lambda}}{\operatorname{argmin}} \left[ \sum_{i=1}^I \sum_{j=1}^J (\mathbf{x}_{ij} - \text{pinhole}[\mathbf{w}_i, \boldsymbol{\Lambda}, \boldsymbol{\Omega}_j, \boldsymbol{\tau}_j])^T (\mathbf{x}_{ij} - \text{pinhole}[\mathbf{w}_i, \boldsymbol{\Lambda}, \boldsymbol{\Omega}_j, \boldsymbol{\tau}_j]) \right], \quad (16.2)$$

in which the goal is to minimize the total squared distance between the observed image points and those predicted by the model. This is known as the *squared reprojection error*. Unfortunately, there is no simple closed form solution for this problem, and we must ultimately rely on nonlinear optimization. However, to ensure that the optimization converges, we need good initial estimates of the unknown parameters.

To simplify our discussion, we will concentrate first on the case where we have  $J=2$  views, and the intrinsic matrix  $\boldsymbol{\Lambda}$  is known. We already saw how to estimate 3D points given known camera positions in section 14.6, so the unresolved problem is to get good initial estimates of the extrinsic parameters. Surprisingly, it is possible to do this based on just examining the positions of corresponding points without having to reconstruct their 3D world positions. To understand why, we must first learn more about the geometry of two views.

## 16.1 Two-view geometry

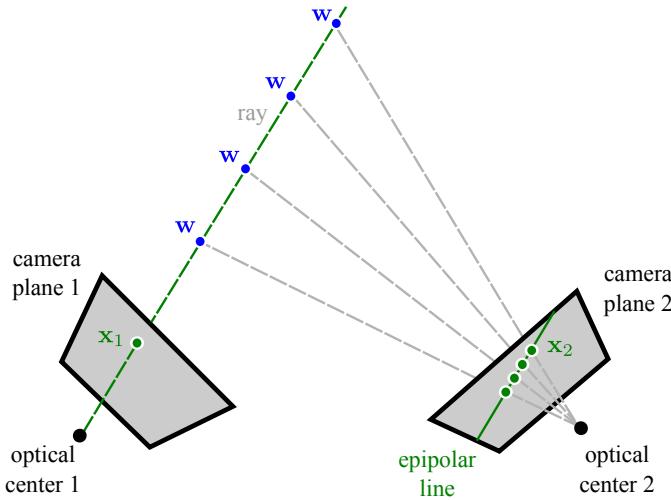
In this section, we show that there is a geometric relationship between corresponding points in two images of the same scene. This relationship depends only on the intrinsic parameters of the two cameras and their relative translation and rotation.

### 16.1.1 The epipolar constraint

Consider a single camera viewing a 3D point  $\mathbf{w}$  in the world. We know that  $\mathbf{w}$  must lie somewhere on the ray that passes through the optical center and position  $\mathbf{x}_1$  on the image plane (figure 16.1). However, from one camera alone, we cannot know how far along this ray the point is.

Now consider a second camera viewing the same 3D world point. We know from the first camera that this point must lie along a particular ray in 3D space. It follows that the projected position  $\mathbf{x}_2$  of this point in the second image must lie somewhere along the projection of this ray in the second image. The ray in 3D projects to a 2D line which is known as an *epipolar line*.

This geometric relationship tells us something important: for any point in the first image, the corresponding point in the second image is constrained to lie on a line. This is known as the *epipolar constraint*. The particular line that it is constrained to lie on depends on the intrinsic parameters of the cameras and the relative translation and rotation of the two cameras (determined by the extrinsic parameters).



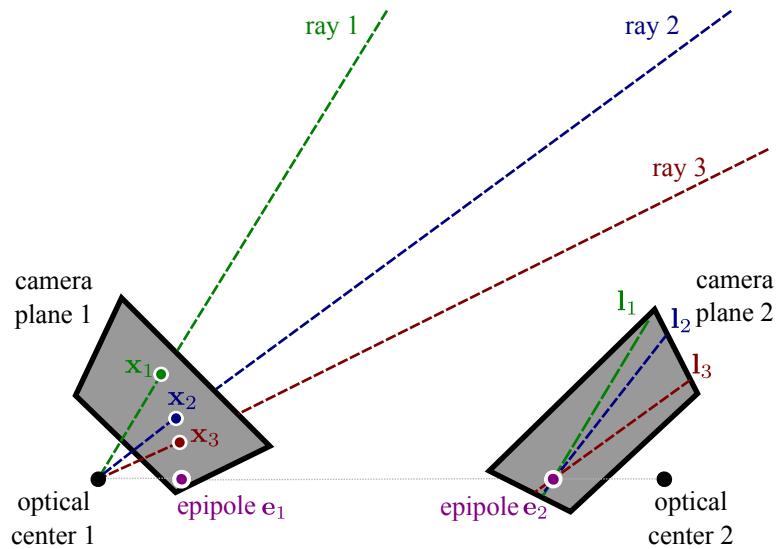
**Figure 16.1** Epipolar line. Consider point  $x_1$  in the first image. The 3D point  $w$  that projected to  $x_1$  must lie somewhere along the ray that passes from the optical center of camera 1 through the position  $x_1$  in the image plane (dashed green line). However, we don't know where along that ray it lies (4 possibilities shown). It follows that  $x_2$ , the projected position in camera 2 must lie somewhere on the projection of this ray. The projection of this ray is a line in image 2 and is referred to as an epipolar line.

The epipolar constraint has two important practical implications.

1. Given the intrinsic and extrinsic parameters, we can find point correspondences relatively easily: for a given point in the first image, we only need to perform a 1D search along the epipolar line in the second image for the corresponding position.
2. The constraint on corresponding points is a function of the intrinsic and extrinsic parameters; given the intrinsic parameters, we can use the observed pattern of point correspondences to determine the extrinsic parameters and hence establish the geometric relationship between the two cameras.

### 16.1.2 Epipoles

Now consider a number of points in the first image. Each is associated with a ray in 3D space. Each ray projects to form an epipolar line in the second image. Since all the rays converge at the optical center of the first camera, the epipolar lines must converge at a single point in the second image plane; this is the image in the second camera of the optical center of the first camera and is known as the *epipole* (figure 16.2). Similarly, points in image 2 induce epipolar lines in image 1, and



**Figure 16.2** Epipoles. Consider several observed points  $\{\mathbf{x}_i\}_{i=1}^I$  in image 1. For each point, the corresponding 3D world position  $\mathbf{w}_i$  lies on a different ray. Each ray projects to an epipolar line  $l_i$  in image 2. Since the rays converge in 3D space at the optical center of camera 1, the epipolar lines must also converge. The point where they converge is known as the epipole  $\mathbf{e}_2$ . It is the projection of the optical center of camera 1 into camera 2. Similarly, the epipole  $\mathbf{e}_1$  is the projection of the optical center of camera 2 into camera 1.

these epipolar lines converge at the epipole in image 1. This is the image in camera 1 of the optical center of camera 2.

The epipoles are not necessarily within the observed images: the epipolar lines may converge to a point outside the visible area. Two common cases are illustrated in figure 16.3. When the cameras are oriented in the same direction (i.e., no relative rotation) and the translation is perpendicular to their optical axes (figure 16.3a) then the epipolar lines are parallel and the epipoles (where they converge) are hence at infinity. When the cameras are oriented in the same direction and the translation is parallel to their optical axes (figure 16.3b), then the epipoles are in the middle of the images and the epipolar lines form a radial pattern. These examples illustrate that the pattern of epipolar lines provides information about the relative position and orientation of the cameras.

#### Problem 16.1

## 16.2 The essential matrix

Now we will capture these geometric intuitions in the form of a mathematical model. For simplicity, we will assume that the world coordinate system is centered on the first camera so that the extrinsic parameters (rotation and translation) of the first camera are  $\{\mathbf{I}, \mathbf{0}\}$ . The second camera may be in any general position  $\{\boldsymbol{\Omega}, \boldsymbol{\tau}\}$ . We will further assume that the cameras are normalized so that  $\Lambda_1 = \Lambda_2 = \mathbf{I}$ . In homogeneous coordinates, a 3D point  $\mathbf{w}$  is projected into the two cameras as

$$\begin{aligned}\lambda_1 \tilde{\mathbf{x}}_1 &= [\mathbf{I}, \mathbf{0}] \tilde{\mathbf{w}} \\ \lambda_2 \tilde{\mathbf{x}}_2 &= [\boldsymbol{\Omega}, \boldsymbol{\tau}] \tilde{\mathbf{w}}.\end{aligned}\quad (16.3)$$

where  $\tilde{\mathbf{x}}_1$  is the observed position in the first camera,  $\tilde{\mathbf{x}}_2$  is the observed position in the second camera, and both are expressed in homogeneous coordinates.

Expanding the first of these relations we get

$$\lambda_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}, \quad (16.4)$$

This simplifies to

$$\lambda_1 \tilde{\mathbf{x}}_1 = \mathbf{w}. \quad (16.5)$$

By a similar process, the projection in the second camera can be written as

$$\lambda_2 \tilde{\mathbf{x}}_2 = \boldsymbol{\Omega} \mathbf{w} + \boldsymbol{\tau}. \quad (16.6)$$

Finally, substituting equation 16.5 into equation 16.6 yields

$$\lambda_2 \tilde{\mathbf{x}}_2 = \lambda_1 \boldsymbol{\Omega} \tilde{\mathbf{x}}_1 + \boldsymbol{\tau}. \quad (16.7)$$

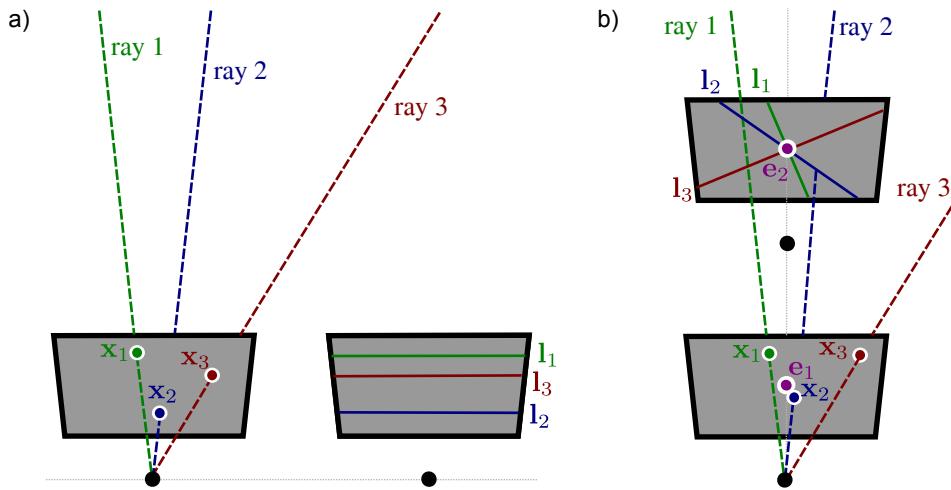
This relationship represents a constraint between the possible positions of corresponding points  $\mathbf{x}_1$  and  $\mathbf{x}_2$  in the two images. The constraint is parameterized by the rotation and translation  $\{\boldsymbol{\Omega}, \boldsymbol{\tau}\}$  of the second camera relative to the first.

We will now manipulate the relationship in equation 16.7 into a form that can be more easily related to the epipolar lines and the epipoles. We first take the cross product of both sides with the translation vector  $\boldsymbol{\tau}$ . This removes the last term as the cross product of any vector with itself is zero. Now we have

$$\lambda_2 \boldsymbol{\tau} \times \tilde{\mathbf{x}}_2 = \lambda_1 \boldsymbol{\tau} \times \boldsymbol{\Omega} \tilde{\mathbf{x}}_1. \quad (16.8)$$

Then we take the inner product of both sides with  $\tilde{\mathbf{x}}_2$ . The left hand side disappears since  $\boldsymbol{\tau} \times \tilde{\mathbf{x}}_2$  must be perpendicular to  $\tilde{\mathbf{x}}_2$ , and so we have

$$\tilde{\mathbf{x}}_2^T \boldsymbol{\tau} \times \boldsymbol{\Omega} \tilde{\mathbf{x}}_1 = 0, \quad (16.9)$$



**Figure 16.3** Epipolar lines and epipoles. a) When the camera movement is a pure translation perpendicular to the optical axis (parallel to the image plane) the epipolar lines are parallel and the epipole is at infinity. b) When the camera movement is a pure translation along the optical axis the epipoles are in the center of the image and the epipolar lines form a radial pattern.

where we have also eliminated the scaling factors  $\lambda_1$  and  $\lambda_2$  by dividing by them. Finally, we note that the cross product operation  $\boldsymbol{\tau} \times$  can be expressed as multiplication by the rank 2 skew-symmetric  $3 \times 3$  matrix  $\boldsymbol{\tau}_\times$ :

$$\boldsymbol{\tau}_\times = \begin{bmatrix} 0 & -\tau_z & \tau_y \\ \tau_z & 0 & -\tau_x \\ -\tau_y & \tau_x & 0 \end{bmatrix}. \quad (16.10)$$

Hence equation 16.9 has the form

$$\tilde{\mathbf{x}}_2^T \mathbf{E} \tilde{\mathbf{x}}_1 = 0, \quad (16.11)$$

Problem 16.3

where  $\mathbf{E} = \boldsymbol{\tau}_\times \boldsymbol{\Omega}$  is known as the *essential matrix*. Equation 16.11 is an elegant formulation of the mathematical constraint between the positions of corresponding points  $\mathbf{x}_1$  and  $\mathbf{x}_2$  in two normalized cameras.

Problem 16.2

### 16.2.1 Properties of the essential matrix

Problem 16.4

The  $3 \times 3$  essential matrix captures the geometric relationship between the two cameras and has rank 2 so that  $\det[\mathbf{E}] = 0$ . The first two singular values of the essential matrix are always identical and the third is zero. It depends only on the rotation and translation between the cameras, each of which has 3 parameters, and so one might think it would have 6 degrees of freedom. However, it operates on

homogeneous variables  $\tilde{\mathbf{x}}_1$  and  $\tilde{\mathbf{x}}_2$  and is hence ambiguous up to scale: multiplying all of the entries of the essential matrix by any constant does not change its properties. For this reason, it is usually considered as having 5 degrees of freedom.

Since there are fewer degrees of freedom than there are unknowns, the nine entries of the matrix must obey a set of algebraic constraints. These can be expressed compactly as

$$2\mathbf{E}\mathbf{E}^T\mathbf{E} - \text{trace}[\mathbf{E}\mathbf{E}^T]\mathbf{E} = \mathbf{0}. \quad (16.12)$$

These constraints are sometimes exploited in the computation of the essential matrix, although in this volume we use a simpler method (section 16.4).

The epipolar lines are easily retrieved from the essential matrix. The condition for a point being on a line is  $ax + by + c = 0$  or

$$\begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0. \quad (16.13)$$

In homogeneous co-ordinates, this can be written as  $\mathbf{l}\tilde{\mathbf{x}} = 0$  where  $\mathbf{l} = [a, b, c]$  is a  $1 \times 3$  vector representing the line.

Now consider the essential matrix relation

$$\tilde{\mathbf{x}}_2^T \mathbf{E} \tilde{\mathbf{x}}_1 = 0. \quad (16.14)$$

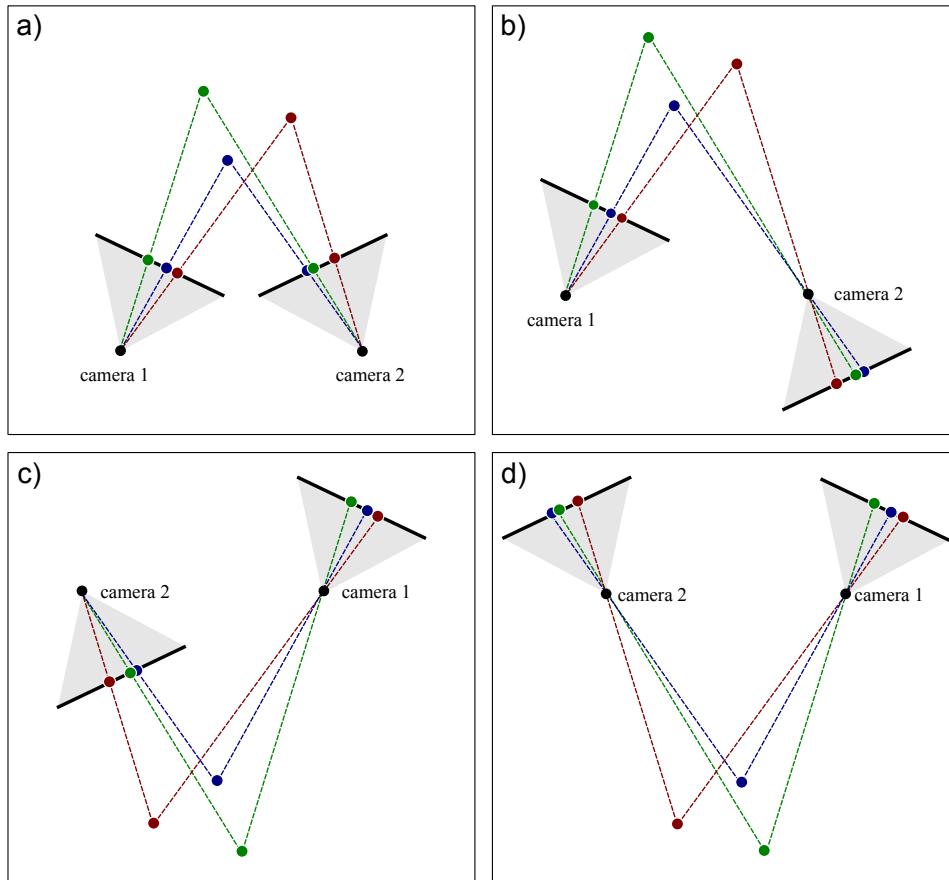
Since  $\tilde{\mathbf{x}}_2^T \mathbf{E}$  is a  $1 \times 3$  vector, this relationship has the form  $\mathbf{l}_1 \tilde{\mathbf{x}}_1 = 0$ . The line  $\mathbf{l}_1 = \tilde{\mathbf{x}}_2^T \mathbf{E}$  is the epipolar line in image 1 due to the point  $\mathbf{x}_2$  in image 2. By a similar argument, we can find the epipolar line  $\mathbf{l}_2$  in the second camera due to the point  $\mathbf{x}_1$  in the first camera. The final relations are

$$\begin{aligned} \mathbf{l}_1 &= \tilde{\mathbf{x}}_2^T \mathbf{E} \\ \mathbf{l}_2 &= \tilde{\mathbf{x}}_1^T \mathbf{E}^T. \end{aligned} \quad (16.15)$$

The epipoles can also be extracted from the essential matrix. Every epipolar line in image 1 passes through the epipole  $\tilde{\mathbf{e}}_1$ , so at the epipole  $\tilde{\mathbf{e}}_1$  we have  $\tilde{\mathbf{x}}_2^T \mathbf{E} \tilde{\mathbf{e}}_1 = 0$  for all  $\tilde{\mathbf{x}}_2$ . This implies that  $\tilde{\mathbf{e}}_1$  must lie in the right null-space of  $\mathbf{E}$  (see appendix C.2.7). By a similar argument, the epipole  $\tilde{\mathbf{e}}_2$  in the second image must lie in the left null space of  $\mathbf{E}$ . Hence, we have the relations

$$\begin{aligned} \tilde{\mathbf{e}}_1 &= \text{null}[\mathbf{E}] \\ \tilde{\mathbf{e}}_2 &= \text{null}[\mathbf{E}^T]. \end{aligned} \quad (16.16)$$

In practice, the epipoles can be retrieved by computing the singular value decomposition  $\mathbf{E} = \mathbf{U}\mathbf{L}\mathbf{V}^T$  of the essential matrix, and setting  $\tilde{\mathbf{e}}_1$  to the last column of  $\mathbf{V}$  and  $\tilde{\mathbf{e}}_2$  to the last row of  $\mathbf{U}$ .



**Figure 16.4** Four-fold ambiguity of reconstruction from two pinhole cameras. The mathematical model for the pinhole camera does not distinguish between points that are in front of and points that are behind the camera. This leads to a four-fold ambiguity when we extract the rotation  $\Omega$  and translation  $\tau$  relating the cameras from the essential matrix. a) Correct solution. Points are in front of both cameras. b) Incorrect solution. The images are identical, but with this interpretation, the points are behind camera 2. c) Incorrect solution with points behind camera 1. d) Incorrect solution with points behind both cameras.

### 16.2.2 Decomposition of essential matrix

We saw previously that the essential matrix is defined as

$$\mathbf{E} = \boldsymbol{\tau}_{\times} \boldsymbol{\Omega}, \quad (16.17)$$

Algorithm 16.1 where  $\boldsymbol{\Omega}$  and  $\boldsymbol{\tau}$  are the rotation matrix and translation vector that map points in the coordinate system of camera 2 to the coordinate system of camera 1, and  $\boldsymbol{\tau}_{\times}$

is a  $3 \times 3$  matrix derived from the translation vector.

We will defer the question of how to compute the essential matrix from a set of corresponding points until section 16.3. For now, we will concentrate on how to decompose a given essential matrix  $\mathbf{E}$  to recover this rotation  $\boldsymbol{\Omega}$  and translation  $\boldsymbol{\tau}$ . This is known as the *relative orientation* problem.

In due course, we shall see that we can compute the rotation exactly, whereas it is only possible to compute the translation up to an unknown scaling factor. This remaining uncertainty reflects the geometric ambiguity of the system; from the images alone, we cannot tell if these cameras are far apart and looking at a large distant object or close together and looking at a small nearby object.

To decompose  $\mathbf{E}$ , we define the matrix

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (16.18)$$

and then take the singular value decomposition  $\mathbf{E} = \mathbf{U}\mathbf{L}\mathbf{V}^T$ . We now choose

$$\begin{aligned} \boldsymbol{\tau}_\times &= \mathbf{U}\mathbf{L}\mathbf{W}\mathbf{U}^T \\ \boldsymbol{\Omega} &= \mathbf{U}\mathbf{W}^{-1}\mathbf{V}^T. \end{aligned} \quad (16.19)$$

It is convention to set the magnitude of the translation vector  $\boldsymbol{\tau}$  that is recovered from the matrix  $\boldsymbol{\tau}_\times$  to unity. The above decomposition is not obvious, but it is easily checked that multiplying the derived expressions for  $\boldsymbol{\tau}_\times$  and  $\boldsymbol{\Omega}$  yields  $\mathbf{E} = \mathbf{U}\mathbf{L}\mathbf{V}^T$ . This method assumes that we started with a valid essential matrix where the first two singular values are identical and the third is zero. If this is not the case (due to noise) we can substitute  $\mathbf{L}' = \text{diag}[1, 1, 0]$  for  $\mathbf{L}$  in the solution for  $\boldsymbol{\tau}_\times$ . For a detailed proof of this decomposition, consult Hartley & Zisserman (2004).

Problem 16.6

This solution is only one of four possible combinations of  $\boldsymbol{\Omega}$  and  $\boldsymbol{\tau}$  that are compatible with  $\mathbf{E}$  (figure 16.4). This four-fold ambiguity is due to the fact that the pinhole model cannot distinguish between objects that are behind the camera (and are not imaged in real cameras) and those that are in front of the camera.

Part of the uncertainty is captured mathematically by our lack of knowledge of the sign of the essential matrix (recall it is ambiguous up to scale) and hence the sign of the recovered translation. Hence, we can generate a second solution by multiplying the translation vector by -1. The other component of the uncertainty results from an ambiguity in the decomposition of the essential matrix; we can equivalently replace  $\mathbf{W}$  for  $\mathbf{W}^{-1}$  in the decomposition procedure, and this leads to two more solutions.

Fortunately, we can resolve this ambiguity using a corresponding pair of points from the two images. For each putative solution, we reconstruct the 3D position associated with this pair (section 14.6). For one of the four possible combinations of  $\boldsymbol{\Omega}, \boldsymbol{\tau}$ , the point will be in front of both cameras, and this is the correct solution. In each of the other three cases, the point will be reconstructed behind one or both of the cameras (figure 16.4). For a robust estimate, we would repeat this procedure with a number of corresponding points and base our decision on the total number of votes for each of the four interpretations.

### 16.3 The fundamental matrix

The derivation of the essential matrix in section 16.2 used normalized cameras (where  $\Lambda_1 = \Lambda_2 = \mathbf{I}$ ). The fundamental matrix plays the role of the essential matrix for cameras with arbitrary intrinsic matrices  $\Lambda_1$  and  $\Lambda_2$ . The general projection equations for the two cameras are

$$\begin{aligned}\lambda_1 \tilde{\mathbf{x}}_1 &= \Lambda_1 [\mathbf{I}, \mathbf{0}] \tilde{\mathbf{w}} \\ \lambda_2 \tilde{\mathbf{x}}_2 &= \Lambda_2 [\Omega, \tau] \tilde{\mathbf{w}},\end{aligned}\tag{16.20}$$

**Problem 16.7** and we can use similar manipulations to those presented in section 16.2 to derive the constraint

$$\tilde{\mathbf{x}}_2^T \Lambda_2^{-T} \mathbf{E} \Lambda_1^{-1} \tilde{\mathbf{x}}_1 = 0,\tag{16.21}$$

or

$$\tilde{\mathbf{x}}_2^T \mathbf{F} \tilde{\mathbf{x}}_1 = 0,\tag{16.22}$$

where the  $3 \times 3$  matrix  $\mathbf{F} = \Lambda_2^{-T} \mathbf{E} \Lambda_1^{-1} = \Lambda_2^{-T} \boldsymbol{\tau}_x \boldsymbol{\Omega} \Lambda_1^{-1}$  is termed the *fundamental matrix*. Like the essential matrix, it also has rank two, but unlike the essential matrix it has seven degrees of freedom.

If we know the fundamental matrix  $\mathbf{F}$  and the intrinsic matrices  $\Lambda_1$  and  $\Lambda_2$ , it is possible to recover the essential matrix  $\mathbf{E}$  using the relation

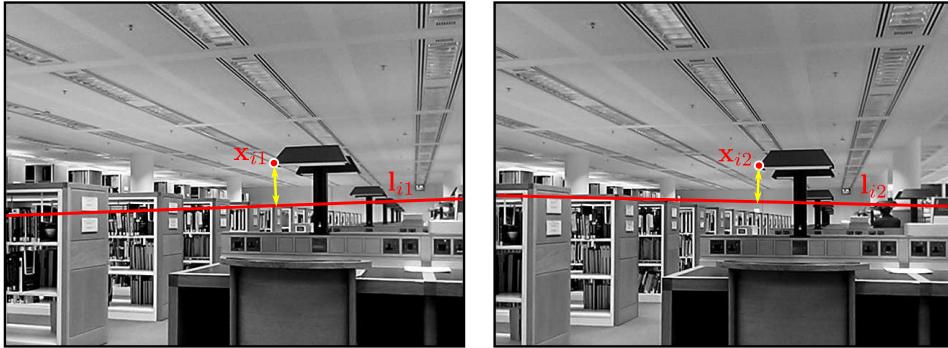
$$\mathbf{E} = \Lambda_2^T \mathbf{F} \Lambda_1,\tag{16.23}$$

and this can further be decomposed to find the rotation and translation between the cameras using the method of section 16.2.2. It follows that for calibrated cameras, if we can estimate the fundamental matrix, then we can find the rotation and translation between the cameras. Hence, we now turn our attention on how to compute the fundamental matrix.

#### 16.3.1 Estimation of the fundamental matrix

The fundamental matrix relation (equation 16.22) is a constraint on the possible positions of corresponding points in the first and second images. This constraint is parameterized by the nine entries of  $\mathbf{F}$ . It follows that if we analyze a set of corresponding points, we can observe *how* they are constrained, and from this we can deduce the entries of the fundamental matrix  $\mathbf{F}$ .

A suitable cost function for the fundamental matrix can be found by considering the epipolar lines. Consider a pair of matching points  $\{\mathbf{x}_{i1}, \mathbf{x}_{i2}\}$  in images 1 and 2, respectively. Each point induces an epipolar line in the other image: the point  $\mathbf{x}_{i1}$  induces line  $\mathbf{l}_{i2}$  in image 2 and the point  $\mathbf{x}_{i2}$  induces the line  $\mathbf{l}_{i1}$  in image 1. When the fundamental matrix is correct, each point should lie exactly on the epipolar line induced by the corresponding point in the other image (figure 16.5). We hence



**Figure 16.5** Cost function for estimating fundamental matrix. The point  $\mathbf{x}_{i1}$  in image 1 induces the epipolar line  $\mathbf{l}_{i2}$  in image 2. When the fundamental matrix is correct, the matching point  $\mathbf{x}_{i2}$  will be on this line. Similarly the point  $\mathbf{x}_{i2}$  in image 2 induces the epipolar line  $\mathbf{l}_{i1}$  in image 1. When the fundamental matrix is correct, the point  $\mathbf{x}_{i1}$  will be on this line. The cost function is the sum of the squares of the distances between these epipolar lines and the points (yellow arrows). This is termed *symmetric epipolar distance*.

minimize the squared distance between every point and the epipolar line predicted by its match in the other image so that

$$\hat{\mathbf{F}} = \underset{\mathbf{F}}{\operatorname{argmin}} \left[ \sum_{i=1}^I \left( (\operatorname{dist}[\mathbf{x}_{i1}, \mathbf{l}_{i1}])^2 + (\operatorname{dist}[\mathbf{x}_{i2}, \mathbf{l}_{i2}])^2 \right) \right], \quad (16.24)$$

where the distance between a 2D point  $\mathbf{x} = [x, y]^T$  and a line  $\mathbf{l} = [a, b, c]$  is

$$\operatorname{dist}[\mathbf{x}, \mathbf{l}] = \frac{ax + by + c}{\sqrt{a^2 + b^2}}. \quad (16.25)$$

Here too, it is not possible to find the minimum of equation 16.24 in closed form, and we must rely on nonlinear optimization methods. It is possible to get a good starting point for this optimization using the *eight-point algorithm*.

### 16.3.2 The eight-point algorithm

The eight-point algorithm converts the corresponding 2D points to homogeneous coordinates and then solves for the fundamental matrix in closed form. It does not directly optimize the cost function in equation 16.24, but instead minimizes an algebraic error. However, the solution to this problem is usually very close to the values that optimize the desired cost function.

In homogeneous coordinates, the relationship between the  $i^{th}$  point  $\mathbf{x}_{i1} = [x_{i1}, y_{i1}]^T$  in image 1 and the  $i^{th}$  point  $\mathbf{x}_{i2} = [x_{i2}, y_{i2}]^T$  in image 2 is

Algorithm 16.2  
Algorithm 16.3

$$\begin{bmatrix} x_{i2} & y_{i2} & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x_{i1} \\ y_{i1} \\ 1 \end{bmatrix} = 0, \quad (16.26)$$

where  $f_{pq}$  represents one of the entries in the fundamental matrix. When we write this constraint out in full, we get

$$x_{i2}x_{i1}f_{11} + x_{i2}y_{i1}f_{12} + x_{i2}f_{13} + y_{i2}x_{i1}f_{21} + y_{i2}y_{i1}f_{22} + y_{i2}f_{23} + x_{i1}f_{31} + y_{i1}f_{32} + f_{33} = 0. \quad (16.27)$$

This can be expressed as an inner product

$$[x_{i2}x_{i1}, x_{i2}y_{i1}, x_{i2}, y_{i2}x_{i1}, y_{i2}y_{i1}, y_{i2}, x_{i1}, y_{i1}, 1]\mathbf{f} = 0, \quad (16.28)$$

where  $\mathbf{f} = [f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33}]^T$  is a vectorized version of the fundamental matrix,  $\mathbf{F}$ .

This provides one linear constraint on the elements of  $\mathbf{F}$ . Consequently, given  $I$  matching points, we can stack these constraints to form the system

$$\mathbf{Af} = \begin{bmatrix} x_{12}x_{11} & x_{12}y_{11} & x_{12} & y_{12}x_{11} & y_{12}y_{11} & y_{12} & x_{11} & y_{11} & 1 \\ x_{22}x_{21} & x_{22}y_{21} & x_{22} & y_{22}x_{21} & y_{22}y_{21} & y_{22} & x_{21} & y_{21} & 1 \\ \vdots & \vdots \\ x_{I2}x_{I1} & x_{I2}y_{I1} & x_{I2} & y_{I2}x_{I1} & y_{I2}y_{I1} & y_{I2} & x_{I1} & y_{I1} & 1 \end{bmatrix} \mathbf{f} = \mathbf{0}. \quad (16.29)$$

Since the elements of  $\mathbf{f}$  are ambiguous up to scale, we solve this system with the constraint that  $|\mathbf{f}| = 1$ . This also avoids the trivial solution  $\mathbf{f} = \mathbf{0}$ . This is a minimum direction problem (see appendix C.7.2). The solution can be found by taking the singular value decomposition,  $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$  and setting  $\mathbf{f}$  to be the last column of  $\mathbf{V}$ . The matrix  $\mathbf{F}$  is then formed by reshaping  $\mathbf{f}$  to form a  $3 \times 3$  matrix.

### Problem 16.8

There are 8 degrees of freedom in the fundamental matrix (it is ambiguous with respect to scale) and so we require a minimum of  $I = 8$  pairs of points. For this reason, this algorithm is called the *eight-point algorithm*.

In practice, there are several further concerns in implementing this algorithm:

- Since the data are noisy, the singularity constraint of the resulting fundamental matrix will not be obeyed in general (i.e., the estimated matrix will be full rank, not rank two). We re-introduce this constraint by taking the singular decomposition of  $\mathbf{F}$ , setting the last singular value to zero, and multiplying the terms back out. This provides the closest singular matrix under a Frobenius norm.
- Equation 16.29 is badly scaled since some terms are on the order of pixels squared ( $\sim 10000$ ) and some are of the order  $\sim 1$ . To improve the quality of the solution, it is wise to pre-normalize the data (see Hartley 1997). We transform the points in image 1 as  $\tilde{\mathbf{x}}'_{i1} = \mathbf{T}_1 \tilde{\mathbf{x}}_{i1}$  and the points in image 2 as  $\tilde{\mathbf{x}}'_{i2} = \mathbf{T}_2 \tilde{\mathbf{x}}_{i2}$ . The transformations  $\mathbf{T}_1$  and  $\mathbf{T}_2$  are chosen to map the mean of the points in their respective image to zero, and to ensure that the variance in the  $x$ - and  $y$ -dimensions is one. We then compute the matrix  $\mathbf{F}'$  from the

transformed data using the eight-point algorithm, and recover the original fundamental matrix as  $\mathbf{F} = \mathbf{T}_2^T \mathbf{F}' \mathbf{T}_1$ .

- The algorithm will only work if the three-dimensional positions  $\mathbf{w}_i$  corresponding to the eight pairs of points  $\mathbf{x}_{i1}, \mathbf{x}_{i2}$  are in general position. For example, if they all fall on a plane, then the equations become degenerate and we cannot get a unique solution; here, the relation between the points in the two images is given by a homography (see chapter 15). Similarly, in the case where there is no translation (i.e.,  $\boldsymbol{\tau} = \mathbf{0}$ ), the relation between the two images is a homography and there is no unique solution for the fundamental matrix.
- In the subsequent nonlinear optimization, we must also ensure that the rank of  $\mathbf{F}$  is two. In order to do this, it is usual to re-parameterize the fundamental matrix to ensure that this will be the case.

## 16.4 Two-view reconstruction pipeline

We now put all of these ideas together and present a rudimentary pipeline for the reconstruction of a static 3D scene based on two images taken from unknown positions, but with cameras where we know the intrinsic parameters. We apply the following steps (figures 16.6 and 16.7).

1. **Compute image features.** We find salient points in each image using an interest point detector such as the SIFT detector (section 13.2.3).
2. **Compute feature descriptors.** We characterize the region around each feature in each image with a low dimensional vector. One possibility would be to use the SIFT descriptor (section 13.3.2).
3. **Find initial matches.** We greedily match features between the two images. For example, we might base this on the squared distance between their region descriptors and stop this procedure when the squared distance exceeds a pre-defined threshold to minimize false matches. We might also reject points where the ratio between the quality of the best and second best match in the other image is too close to one (suggesting that alternative matches are plausible).
4. **Compute fundamental matrix.** We compute the fundamental matrix using the eight-point algorithm. Since some matches are likely to be incorrect, we use a robust estimation procedure such as RANSAC (section 15.6).
5. **Refine matches.** We again greedily match features, but this time we exploit our knowledge of the epipolar geometry: if a putative match is not close to the induced epipolar line, it is rejected. We recompute the fundamental matrix based on all of the remaining point matches.
6. **Estimate essential matrix.** We estimate the essential matrix from the fundamental matrix using equation 16.23.

7. **Decompose essential matrix.** We extract estimates of the rotation and translation between the cameras (i.e., the extrinsic parameters) by decomposing the essential matrix (section 16.2.2). This provides four possible solutions.
8. **Estimate 3D points.** For each solution, we reconstruct the 3D position of the points using the linear solution from section 14.6. We retain the extrinsic parameters where most of the reconstructed points are in front of both cameras.

After this procedure, we have a set of  $I$  points  $\{\mathbf{x}_{i1}\}_{i=1}^I$  in the first image, a set of  $I$  corresponding points  $\{\mathbf{x}_{i2}\}_{i=1}^I$  in the second image, and a good initial estimate of the 3D world positions  $\{\mathbf{w}_i\}_{i=1}^I$  that were responsible for them. We also have initial estimates of the extrinsic parameters  $\{\boldsymbol{\Omega}, \boldsymbol{\tau}\}$ . We now optimize the true cost function

$$\begin{aligned}\hat{\mathbf{w}}_{1\dots I}, \hat{\boldsymbol{\Omega}}, \hat{\boldsymbol{\tau}} &= \underset{\mathbf{w}, \boldsymbol{\Omega}, \boldsymbol{\tau}}{\operatorname{argmax}} \left[ \sum_{i=1}^I \sum_{j=1}^2 \log [Pr(\mathbf{x}_{ij} | \mathbf{w}_i, \boldsymbol{\Lambda}_j, \boldsymbol{\Omega}, \boldsymbol{\tau})] \right] \\ &= \underset{\mathbf{w}, \boldsymbol{\Omega}, \boldsymbol{\tau}}{\operatorname{argmax}} \left[ \sum_{i=1}^I \log [\operatorname{Norm}_{\mathbf{x}_{i1}}[\operatorname{pinhole}[\mathbf{w}_i, \boldsymbol{\Lambda}_1, \mathbf{I}, \mathbf{0}], \sigma^2 \mathbf{I}]] \right. \\ &\quad \left. + \sum_{i=1}^I \log [\operatorname{Norm}_{\mathbf{x}_{i2}}[\operatorname{pinhole}[\mathbf{w}_i, \boldsymbol{\Lambda}_2, \boldsymbol{\Omega}, \boldsymbol{\tau}], \sigma^2 \mathbf{I}]] \right],\end{aligned}\tag{16.30}$$

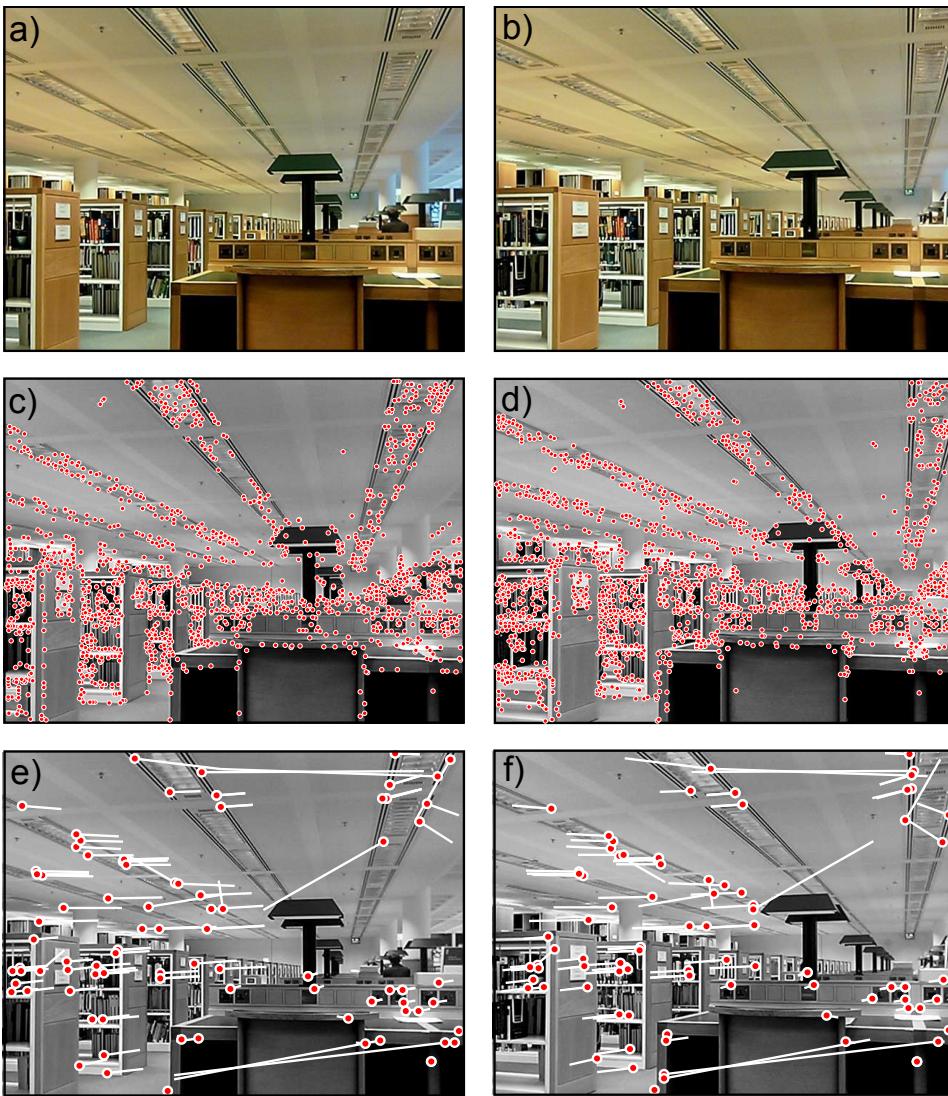
to refine these estimates. In doing so, we must ensure to enforce the constraints that  $|\boldsymbol{\tau}| = 1$  and  $\boldsymbol{\Omega}$  is a valid rotation matrix (see appendix B).

#### 16.4.1 Minimal solutions

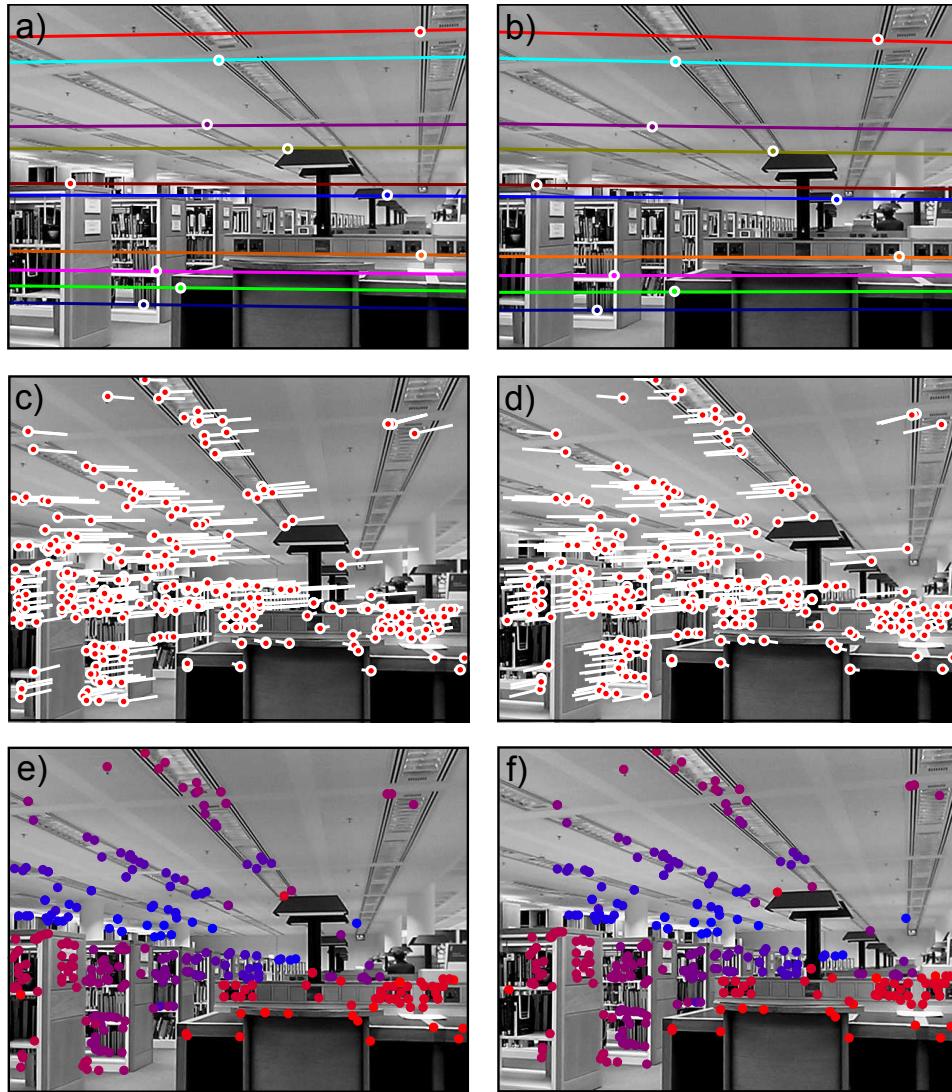
The pipeline described previously is rather naïve; in practice the fundamental and essential matrices can be estimated considerably more efficiently.

For example, the fundamental matrix contains seven degrees of freedom and so it is actually possible to solve for it using only seven pairs of points. Unsurprisingly, this is known as the *seven point algorithm*. It is more complex as it relies on the seven linear constraints and the nonlinear constraint  $\det[\mathbf{F}] = 0$ . However, a robust solution can be computed much more efficiently using RANSAC if only seven points are required.

Even if we use this seven-point algorithm, the method is still inefficient. When we know the intrinsic parameters of the cameras, it is possible to compute the essential matrix (and hence the relative orientation of the cameras) using a minimal five-point solution. This is based on five linear constraints from observed corresponding points and the nonlinear constraints relating the nine parameters of the essential matrix (equation 16.12). This method has the advantages of being much quicker to estimate in the context of a RANSAC algorithm and being robust to non-general configurations of the scene points.



**Figure 16.6** Two view reconstruction pipeline (steps 1-3). a-b) A pair of images of a static scene, captured from two slightly different positions. c-d) SIFT features are computed in each image. A region descriptor is calculated for each feature point that provides a low dimensional characterization of the region around the point. Points in the left and right images are matched using a greedy procedure; the pair of points for which the region descriptors are most similar in a least squares sense are chosen first. Then the pair of remaining points for which the descriptors are most similar are chosen, and so on. This continues until the minimum squared distance between the descriptors exceeds a threshold. e-f) Results of greedy matching procedure: the lines represent the offset to the matching point. Most matches are correct, but there are clearly also some outliers.



**Figure 16.7** Two view reconstruction pipeline (steps 4-8). A fundamental matrix is fitted using a robust estimation procedure such as RANSAC. a-b) Eight matches with maximum agreement from the rest of the data. For each feature, the epipolar line is plotted in the other image. In each case, the matching point lies on or very near the epipolar line. The resulting fundamental matrix can be decomposed to get estimates for the relative rotation and translation between the cameras. c-d) Result of greedily matching original feature points taking into account the epipolar geometry. Matches where the symmetric epipolar distance exceeds a threshold are rejected. e-f) Computed  $w$  coordinate (depth) relative to first camera for each feature. Red features are closer, and blue features are further away. Almost all of the distances agree with our perceived understanding of the scene.

## 16.5 Rectification

The preceding procedure provides a set of sparse matches between the two images. These may suffice for some tasks such as navigation, but if we wish to build an accurate model of the scene, we need to estimate the depth at every point in the image. This is known as *dense reconstruction*.

Dense stereo reconstruction algorithms (see sections 11.8.2 and 12.8.3) generally assume that the corresponding point lies on the same horizontal scanline in the other image. The goal of *rectification* is to preprocess the image pair so that this is true. In other words, we will transform the images so that each epipolar line is horizontal and so the epipolar lines associated with a point fall on the same scanline to that point in the other image. We will describe two different approaches to this problem.

### 16.5.1 Planar rectification

We note that the epipolar lines are naturally horizontal and aligned when the camera motion is purely horizontal and both image planes are perpendicular to the  $w$ -axis (figure 16.3a). The key idea of planar rectification is to manipulate the two images to recreate these viewing conditions. We apply homographies  $\Phi_1$  and  $\Phi_2$  to the two images so that they cut their respective ray bundles in the desired way (figure 16.8).

Algorithm 16.3

In fact there is an entire family of homographies that accomplish this goal. One possible way to select a suitable pair is to first work with image 2. We apply a series of transformations  $\Phi_2 = \mathbf{T}_3 \mathbf{T}_2 \mathbf{T}_1$ , which collectively move the epipole  $\mathbf{e}_2$  to a position at infinity  $[1, 0, 0]^T$ .

We first center the co-ordinate system on the principal point,

$$\mathbf{T}_1 = \begin{bmatrix} 1 & 0 & -\delta_x \\ 0 & 1 & -\delta_y \\ 0 & 0 & 1 \end{bmatrix}. \quad (16.31)$$

Then we rotate the image about this center until the epipole lies on the  $x$ -axis,

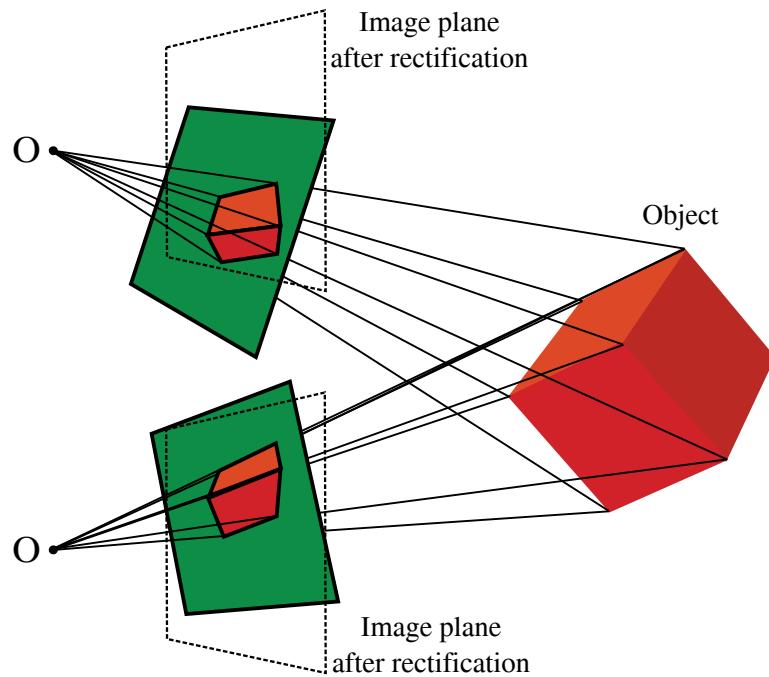
$$\mathbf{T}_2 = \begin{bmatrix} \cos[-\theta] & -\sin[-\theta] & 0 \\ \sin[-\theta] & \cos[-\theta] & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (16.32)$$

where  $\theta = \text{atan2}[e_y, e_x]$  is the angle of the translated epipole  $\mathbf{e} = [e_x, e_y]$ . Finally, we translate the epipole to infinity, using the transformation

$$\mathbf{T}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1/e_x & 0 & 1 \end{bmatrix}, \quad (16.33)$$

where  $e_x$  is the  $x$  coordinate of the epipole after the previous two transformations.

After these transformations, the epipole in the second image is at infinity in the horizontal direction. The epipolar lines in this image must converge at the epipole, and are consequently parallel and horizontal as desired.



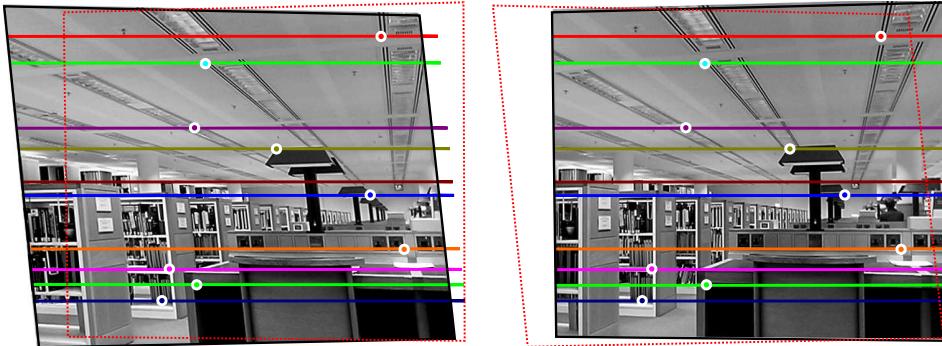
**Figure 16.8** Planar rectification. Green quadrilaterals represent image planes of two cameras viewing a 3D object (cube). The goal of planar rectification is to transform each of these planes so that the final configuration replicates figure 16.3a. After this transformation (dotted lines), the image planes are coplanar, and the translation between the cameras is parallel to this plane. Now the epipolar lines are horizontal and aligned. Since the transformed planes are just different cuts through the respective ray bundles, each transformation can be accomplished using a homography.

Now we consider the first image. We cannot simply apply the same procedure as this will not guarantee that the epipolar lines in the first image will be aligned with those in the second. It transpires however, that there is a family of possible transformations that do make the epipolar lines of this image horizontal and aligned with those in the second image. This family can (not obviously) be parameterized as

$$\Phi_1[\boldsymbol{\alpha}] = (\mathbf{I} + \mathbf{e}_2\boldsymbol{\alpha}^T)\Phi_2\mathbf{M}, \quad (16.34)$$

where  $\mathbf{e}_2 = [1, 0, 0]^T$  is the transformed epipole in the second image, and  $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \alpha_3]^T$  is a 3D vector that selects the particular transformation from the family. The matrix  $\mathbf{M}$  comes from the decomposition of the fundamental matrix into  $\mathbf{F} = \mathbf{S}\mathbf{M}$ , where  $\mathbf{S}$  is skew symmetric (see below). A proof of the relation in equation 16.34 can be found in Hartley & Zisserman (2004).

A sensible criterion is to choose  $\boldsymbol{\alpha}$  so that it minimizes the disparity,



**Figure 16.9** Planar rectification. The images from figures 16.6 and 16.7 have been rectified by applying homographies. After rectification, each point induces an epipolar line in the other image that is horizontal and on the same scanline (compare to figure 16.7a). This means that the match is guaranteed to be on the same scanline. In this figure, the red dotted line is the superimposed outline of the other image.

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \left[ \sum_{i=1}^I (\mathbf{hom}[\mathbf{x}_{i1}, \Phi_1[\boldsymbol{\alpha}]] - \mathbf{hom}[\mathbf{x}_{i2}, \Phi_2])^T (\mathbf{hom}[\mathbf{x}_{i1}, \Phi_1[\boldsymbol{\alpha}]] - \mathbf{hom}[\mathbf{x}_{i2}, \Phi_2]) \right]. \quad (16.35)$$

This criterion simplifies to solving the least squares problem  $|\mathbf{A}\boldsymbol{\alpha} - \mathbf{b}|^2$  where

$$\mathbf{A} = \begin{bmatrix} x'_{11} & y'_{11} & 1 \\ x'_{21} & y'_{21} & 1 \\ \vdots & \vdots & \vdots \\ x'_{I1} & y'_{I1} & 1 \end{bmatrix} \text{ and } \mathbf{b} = \begin{bmatrix} x'_{12} \\ x'_{22} \\ \vdots \\ x'_{I2} \end{bmatrix}, \quad (16.36)$$

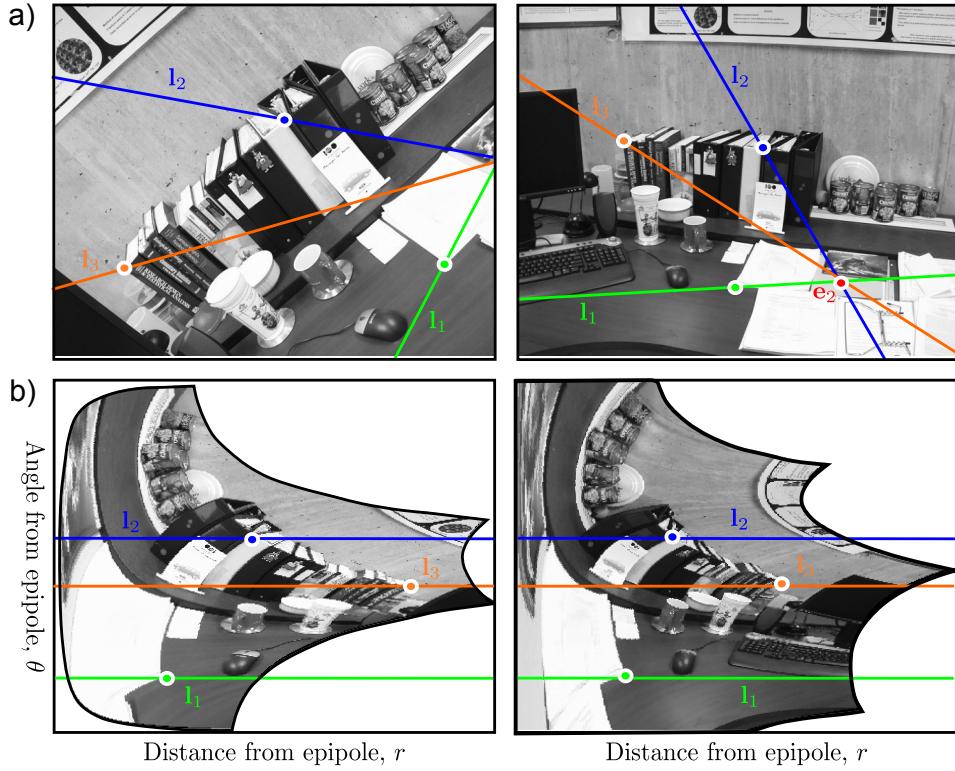
where the vectors  $\mathbf{x}'_{ij} = [x'_{ij}, y'_{ij}]^T$  are defined by

$$\begin{aligned} \mathbf{x}'_{i1} &= \mathbf{hom}[\mathbf{x}_{i1}, \Phi_2 \mathbf{M}] \\ \mathbf{x}'_{i2} &= \mathbf{hom}[\mathbf{x}_{i2}, \Phi_2]. \end{aligned} \quad (16.37)$$

This least squares problem can be solved using the standard approach (appendix C.7.1). Figure 16.9 shows example rectified images. After these transformations, the corresponding points are guaranteed to be on the same horizontal scanline, and dense stereo reconstruction can proceed.

### Decomposition of the fundamental matrix

The preceding algorithm requires the matrix  $\mathbf{M}$  from the decomposition of the fundamental matrix as  $\mathbf{F} = \mathbf{SM}$ , where  $\mathbf{S}$  is skew symmetric. A suitable way to



**Figure 16.10** Polar rectification. When the epipole is inside one of the images, the planar rectification method is no longer suitable. An alternative in this situation is to perform a nonlinear warp of each image, in which the two new dimension corresponds to the distance and angle from the epipole, respectively. This is known as *polar rectification*.

do this is to compute the singular value decomposition of the fundamental matrix  $\mathbf{F} = \mathbf{U}\mathbf{L}\mathbf{V}^T$ . We then define the matrices

$$\mathbf{L}' = \begin{bmatrix} l_{11} & 0 & 0 \\ 0 & l_{22} & 0 \\ 0 & 0 & \frac{l_{11}+l_{22}}{2} \end{bmatrix} \text{ and } \mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (16.38)$$

where  $l_{ii}$  denotes the  $i^{th}$  element from the diagonal of  $\mathbf{L}$ . Finally, we choose

$$\mathbf{M} = \mathbf{U}\mathbf{W}\mathbf{L}'\mathbf{V}^T. \quad (16.39)$$

### 16.5.2 Polar rectification

The planar rectification method described in section 16.5.1 is suitable when the epipole is sufficiently far outside the image. Since the basis of this method is to map the epipoles to infinity it cannot work when the epipole is inside the image, and distorts the image a great deal if it is close to the image. Under these circumstances, a *polar rectification* is preferred.

Polar rectification applies a nonlinear warp to each image so that corresponding points are mapped to the same scanline. Each new image is formed by re-sampling the original images so that the first new axis is the distance from the epipole and the second new axis is the angle from the epipole (figure 16.10). This approach can distort the image significantly but works for all camera configurations.

This method is conceptually simple, but should be implemented with caution; when the epipole lies within the camera image, it is important to ensure that the correct half of the epipolar line is aligned with the appropriate part of the other image. The reader is encouraged to consult the original description (Pollefeys *et al.* 1999b) before implementing this algorithm.

### 16.5.3 After rectification

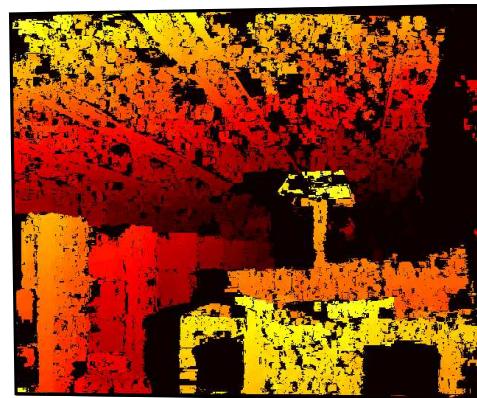
After rectification, the horizontal offset between every point in the first image and its corresponding point in the second image can be computed using a dense stereo algorithm (see sections 11.8.2 and 12.8.3). Typical results are shown in figure 16.11. Each point and its match are then warped back to their original positions (i.e., their image positions are ‘un-rectified’). For each pair of 2D points, the depth can then be computed using the algorithm of section 14.6.

Finally, we may wish to view the model from a novel direction. For the two-view case, a simple way to do this is to form a 3D triangular mesh and to texture this mesh using the information from one or the other image. If we have computed dense matches using a stereo matching algorithm, then the mesh can be computed from the perspective of one camera with two triangles per pixel and cut where there are sharp depth discontinuities. If we have only a sparse correspondence between the two images, then it is usual to triangulate the projections of the 3D points in one image using a technique such as Delaunay triangulation to form the mesh. The textured mesh can now be viewed from a novel direction using the standard computer graphics pipeline.

## 16.6 Multi-view reconstruction

So far, we have considered reconstruction based on two views of a scene. Of course, it is common to have more than two views. For example, we might want to build 3D models using a video sequence taken by a single moving camera, or equivalently from a video sequence from a static camera of a rigidly moving object (figure 16.12).

**Figure 16.11** Disparity. After rectification, the horizontal offset at each point can be computed using a dense stereo algorithm. Here we used the method of Sizintsev *et al.* (2010). Color indicates the horizontal shift (disparity) between images. Black regions indicate places where the matching was ambiguous or where the corresponding part of the scene was occluded. Given these horizontal correspondences, we undo the rectification to get a dense set of matching points and their offsets (now displaced in 2D). The 3D position can now be computed using the method from section 14.6.



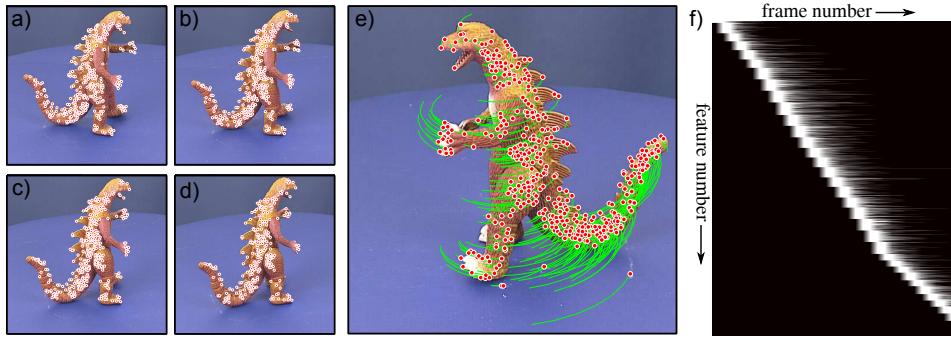
This problem is often referred to as *structure from motion* or *multi-view reconstruction*.

The problem is conceptually very similar to the two camera case. Once again, the solution ultimately relies on a nonlinear optimization in which we manipulate the camera position and the three-dimensional points to minimize the squared reprojection error (equation 16.2), and hence maximize the likelihood of the model. However, the multi-view case does bring several new aspects to the problem.

First, if we have a number of frames all taken with the same camera, there are now sufficient constraints to estimate the intrinsic parameters as well. We initialize the camera matrix with sensible values and add this to the final optimization. This process is known as *auto-calibration*. Second, matching points in video footage is easier, because the changes between adjacent frames tend to be small, and so the features can be explicitly tracked in two dimensions. However, it is usual for some points to be occluded in any given frame, and so we must keep track of which points are present at which time (figure 16.12f).

Third, there are now additional constraints on feature matching, which make it easier to eliminate outliers in the matching set of points. Consider a point that is matched between three frames. The point in the third frame will be constrained to lie on an epipolar line due to the first frame and another epipolar line due to the second frame: its position has to be at the intersection of the two lines and so is determined exactly. Unfortunately, this method will not work when the two predicted epipolar lines are the same as there is not a unique intersection. Consequently, the position in the third view is computed in a different way in practice. Just as we derived the fundamental matrix relation (equation 16.22) constraining the positions of matching points between two views, it is possible to derive a closed-form relation that constrains the positions across three images. The three-view analogue of the fundamental matrix is called the *tri-focal tensor*. It can be used to predict the position of the point in a third image given its position in the first and second images even when the epipolar lines are parallel. There is also a relation between four images, which is captured by the quadri-focal tensor, but there are no further relations between points in  $J > 5$  images.

#### Problems 16.9



**Figure 16.12** Multi-frame structure from motion. The goal is to construct a 3D model from a continuous video stream of a moving camera viewing a static object, or a static camera viewing a moving rigid object. a-d) Features are computed in each frame and tracked through the sequence. e) Features in current frame and their history. f) In each new frame, a number of new features is identified, and these are tracked until they are occluded or the correspondence is lost. Here, the white pixels indicate that a feature was present in a frame, and black pixels indicate that it was absent.

Finally, there are new ways to get initial estimates of the unknown quantities. It may not be practical to get the initial estimates of camera position by computing the transformation between adjacent frames and chaining these through the sequence. The translation between adjacent frames may be too small to reliably estimate the motion, and errors accrue as we move through the sequence. Moreover, it is difficult to maintain a consistent estimate of the (ambiguous) scale throughout. To this end, methods have been developed that simultaneously provide an initial estimate of all the camera positions and 3D points, some of which are based on factorization of a matrix containing all the  $(x, y)$  positions of every point tracked throughout the sequence.

Problem 16.10

### 16.6.1 Bundle adjustment

After finding the initial estimates of the 3D positions (structure) and the camera positions (motion), we must again resort to a large nonlinear optimization problem to fine-tune these parameters. With  $I$  tracked points over  $J$  frames, the problem is formulated as

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[ \sum_{i=1}^I \sum_{j=1}^J \log [Pr(\mathbf{x}_{ij} | \mathbf{w}_i, \boldsymbol{\Lambda}, \boldsymbol{\Omega}_j, \boldsymbol{\tau}_j)] \right] \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[ \sum_{i=1}^I \sum_{j=1}^J \log [\text{Norm}_{\mathbf{x}_{ij}}[\text{pinhole}[\mathbf{w}_i, \boldsymbol{\Lambda}, \boldsymbol{\Omega}_j, \boldsymbol{\tau}_j], \sigma^2 \mathbf{I}]] \right],\end{aligned}\quad (16.40)$$

where  $\boldsymbol{\theta}$  contains the unknown world points  $\{\mathbf{w}_i\}_{i=1}^I$ , the intrinsic matrix  $\boldsymbol{\Lambda}$ , and the extrinsic parameters  $\{\boldsymbol{\Omega}_j, \boldsymbol{\tau}_j\}_{j=1}^J$ . This optimization problem is known as *Euclidean bundle adjustment*. As for the two view case, it is necessary to constrain the overall scale of the solution in some way.

One way to solve this optimization problem is to use an alternating approach. We first improve the log likelihood with respect to each of the extrinsic sets of parameters  $\{\boldsymbol{\Omega}_j, \boldsymbol{\tau}_j\}$  (and possibly the intrinsic matrix  $\boldsymbol{\Lambda}$  if unknown), and then update each 3D position  $\mathbf{w}_i$ . This is known as *resection-intersection*. It seems attractive as it only involves optimizing over a small subset of parameters at any one time. However, this type of *coordinate ascent* is inefficient: it cannot take advantage of the large gains that come from varying all of the parameters at once.

To make progress, we note that the cost function is based on the normal distribution and so can be re-written in the least squares form

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} [\mathbf{z}^T \mathbf{z}], \quad (16.41)$$

where the vector  $\mathbf{z}$  contains the squared differences between the observed feature positions  $\mathbf{x}_{ij}$  and the positions  $\text{pinhole}[\mathbf{w}_i, \boldsymbol{\Lambda}, \boldsymbol{\Omega}_j, \boldsymbol{\tau}_j]$  predicted by the model with the current parameters:

$$\mathbf{z} = \begin{bmatrix} \mathbf{x}_{11} - \text{pinhole}[\mathbf{w}_1, \boldsymbol{\Lambda}, \boldsymbol{\Omega}_1, \boldsymbol{\tau}_1] \\ \mathbf{x}_{12} - \text{pinhole}[\mathbf{w}_1, \boldsymbol{\Lambda}, \boldsymbol{\Omega}_2, \boldsymbol{\tau}_2] \\ \vdots \\ \mathbf{x}_{IJ} - \text{pinhole}[\mathbf{w}_I, \boldsymbol{\Lambda}, \boldsymbol{\Omega}_J, \boldsymbol{\tau}_J] \end{bmatrix}. \quad (16.42)$$

The Gauss-Newton method (appendix B.2.3) is specialized to this type of problem and updates the current estimate  $\boldsymbol{\theta}^{[t]}$  of the parameters using:

$$\boldsymbol{\theta}^{[t]} = \boldsymbol{\theta}^{[t-1]} + \lambda (\mathbf{J}^T \mathbf{J})^{-1} \frac{\partial f}{\partial \boldsymbol{\theta}}, \quad (16.43)$$

where  $\mathbf{J}$  is the Jacobian matrix. The entry in the  $m^{th}$  row and  $n^{th}$  column of  $\mathbf{J}$  consists of the derivative of the  $m^{th}$  element of  $\mathbf{z}$  with respect to the  $n^{th}$  element of the parameter vector  $\boldsymbol{\theta}$ :

$$J_{mn} = \frac{\partial z_m}{\partial \theta_n}. \quad (16.44)$$

In a real structure from motion problem, there might be thousands of scene points, each with three unknowns, and also thousands of camera positions, each with six unknowns. At each stage of the optimization, we must invert  $\mathbf{J}^T \mathbf{J}$ , which

is a square matrix whose dimension is the same as the number of unknowns. When the number of unknowns is large, inverting this matrix becomes expensive.

However, it is possible to build a practical system by exploiting the sparse structure of  $\mathbf{J}^T \mathbf{J}$ . This sparsity results from the fact that every squared error term does not depend on every unknown. There is one contributing error term per observed 2D point and this depends only on the associated 3D point, the intrinsic parameters, and the camera position in that frame.

To exploit this structure, we order the elements of the Jacobian matrix as  $\mathbf{J} = [\mathbf{J}_w, \mathbf{J}_\Omega]$  where  $\mathbf{J}_w$  contains the terms that relate to the unknown world points  $\{\mathbf{w}_i\}_{i=1}^I$  in turn, and  $\mathbf{J}_\Omega$  contains the terms that relate to the unknown camera positions  $\{\boldsymbol{\Omega}_j, \boldsymbol{\tau}_j\}_{j=1}^J$ . For pedagogical reasons, we will assume that the intrinsic matrix  $\boldsymbol{\Lambda}$  is known here and hence has no entries in the Jacobian. We see that the matrix to be inverted becomes

$$\mathbf{J}^T \mathbf{J} = \begin{bmatrix} \mathbf{J}_w^T \mathbf{J}_w & \mathbf{J}_w^T \mathbf{J}_\Omega \\ \mathbf{J}_\Omega^T \mathbf{J}_w & \mathbf{J}_\Omega^T \mathbf{J}_\Omega \end{bmatrix}. \quad (16.45)$$

We now note that the matrices in the top-left and bottom-right of this matrix are block-diagonal (different world points do not interact with one another, and neither do the parameters from different cameras). Hence, these two sub-matrices can be inverted very efficiently. The Schur complement relation (appendix C.8.2), allows us to exploit this fact to reduce the complexity of the larger matrix inversion.

The preceding description is only a sketch of a real bundle adjustment algorithm; in a real system, additional sparseness in  $\mathbf{J}^T \mathbf{J}$  would be exploited, a more sophisticated optimization method such as Levenberg-Marquardt would be employed, and a robust cost function would be used to reduce the effect of outliers.

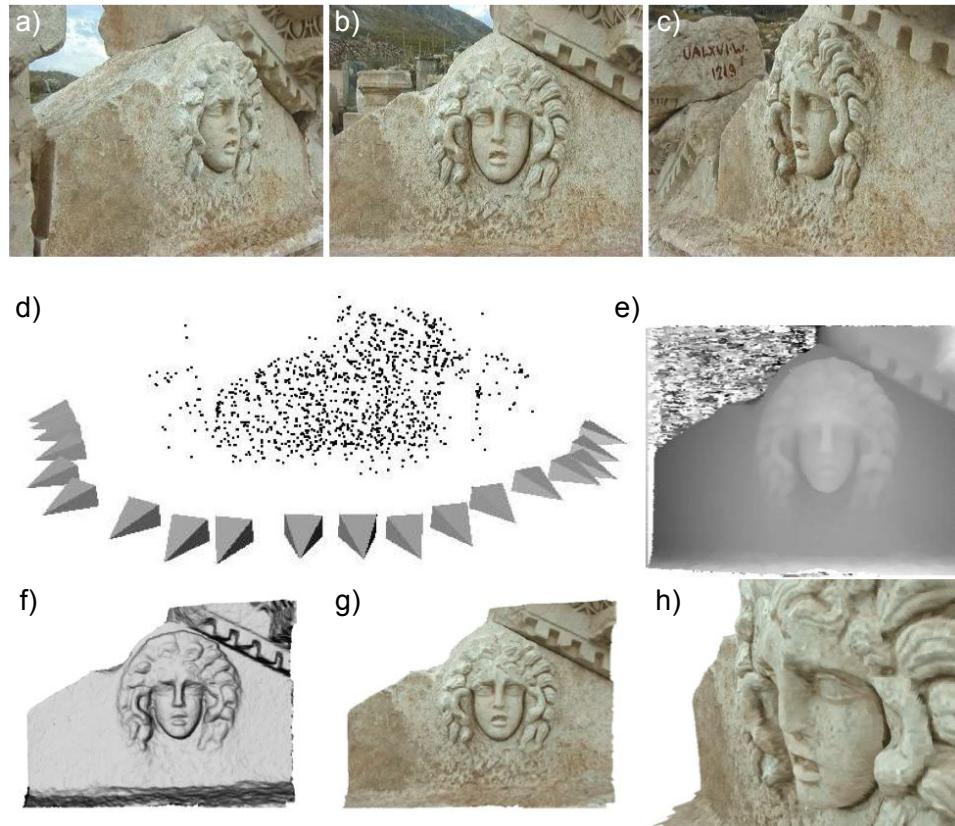
Problem 16.11

## 16.7 Applications

We first describe a typical pipeline for recovering a 3D mesh model from a sequence of video frames. We then discuss a system that can extract 3D information about a scene from images gathered by a search engine on the internet, and use this information to help navigate through the set of photos. Finally, we discuss a multi-camera system for capturing 3D objects that uses a volumetric representation and exploits a Markov random field prior to get a smooth reconstruction.

### 16.7.1 3D reconstruction pipeline

Pollefeys & Van Gool (2002) present a complete pipeline for constructing 3D models from a sequence of images taken from an uncalibrated hand-held camera (figure 16.13). In the first stage, they compute a set of interest points (corners) in each image. When the image data consist of individual still photos, these points are matched between images. When the image data consist of continuous video, they are tracked between frames. In either case, a sparse set of potential cor-



**Figure 16.13** 3D reconstruction pipeline. a-c) A 20 second video sequence of the camera panning around the medusa carving was captured. Every 20<sup>th</sup> frame was used for the reconstruction, three of which are shown here. d) Sparse reconstruction (points) and estimated camera positions (pyramids) after bundle adjustment procedure. e) Depth map after dense stereo matching. f) Shaded 3D mesh model. g-h) Two views of textured 3D mesh model. Adapted from Pollefeys & Van Gool (2002). ©2002 Wiley.

respondences is obtained. The multi-view relations are estimated using a robust procedure, and these are then used to eliminate outliers from the correspondence set.

To estimate the motion of the cameras, two images are chosen and a projective reconstruction is computed (i.e., a reconstruction that is ambiguous up to a 3D projective transformation because the intrinsic parameters are not known). For each of the other images in turn, the pose for the camera is determined relative to this reconstruction and the reconstruction is refined. In this way it is possible to incorporate views that have no common features with the original two frames.

A subsequent bundle adjustment procedure minimizes the re-projection errors



**Figure 16.14** Photo-tourism. a) A sparse 3D model of an object is computed from a set of photographs retrieved from the internet and the relative positions of the cameras (pyramids) are estimated. b) This 3D model is used as the basis of an interface that provides novel ways to explore the photo-collection by moving from image to image in 3D space. Adapted from Snavely *et al.* (2006). ©2006 ACM.

to get more accurate estimates of the camera positions and the points in the 3D world. At this stage, the reconstruction is still ambiguous up to a projective ambiguity and only now are initial estimates of the intrinsic parameters computed using a specialized procedure (see Pollefeys *et al.* 1999a). Finally, a full bundle adjustment method is applied; it simultaneously refines the estimates of the intrinsic parameters, camera positions, and 3D structure of the scene.

Successive pairs of images are rectified, and a dense set of disparities are computed using a multi-resolution dynamic programming technique. Given these dense correspondences, it is possible to compute an estimate of the 3D scene from the point of view of both cameras. A final estimate of the 3D structure relative to a reference frame is computed by fusing all of these independent estimates using a Kalman filter (see chapter 19).

For relatively simple scenes, a 3D mesh is computed by placing the vertices of the triangles in 3D space according to the values found in the depth map of the reference frame. The associated texture map can be retrieved from one or more of the original images. For more complex scenes, a single reference frame may not suffice and so several meshes are computed from different reference frames and fused together. Figures 16.13g-h show examples of the resulting textured 3D model of a Medusa head at the ancient site of Sagalassos in Turkey. More details concerning this pipeline can be found in Pollefeys *et al.* (2004).

### 16.7.2 Photo-tourism

Snavely *et al.* (2006) present a system for browsing a collection of images of an object that were gathered from the internet. A sparse 3D model of the object is

created by locating SIFT features in each image and finding a set of correspondences between pairs of images by computing the fundamental matrix using the eight-point algorithm with RANSAC.

A bundle adjustment procedure is then applied to estimate the camera positions and a sparse 3D model of the scene (figure 16.14a). This optimization procedure starts with only a single pair of images and gradually includes images based on their overlap with the current reconstruction, ‘re-bundling’ at each stage. The intrinsic matrix of each camera is also estimated in this step, but this is simplified by assuming that the center of projection is co-incident with the image center, that the skew is zero, and that the pixels are square, leaving a single focal length parameter. This is initialized in the optimization using information from the EXIF tags of the image when they are present. The bundle adjustment procedure was lengthy; for the model of Notre-Dame, it took two weeks to compute a model from 2635 photos of which 597 images were ultimately included. However, more recent approaches to reconstruction from internet photos such as that of Frahm *et al.* (2010) are considerably faster.

This sparse 3D model of the scene is exploited to create a set of tools for navigating around the set of photographs. For example, it is possible to

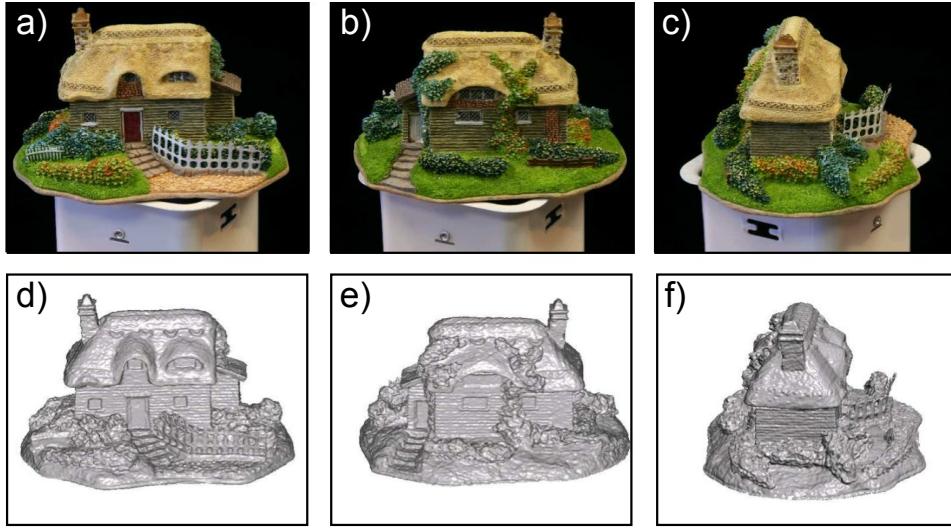
- select a particular view based on a 3D rendering (as in figure 16.14a),
- find images of the object that are similar to the current view,
- retrieve images of the object taken from the left or the right of the current position (effectively pan around the object),
- find images that are from a similar viewpoint but closer or further from the object (zoom into / away from the object),
- and annotate objects and have these annotations transferred to other images.

This system was extended by Snavely *et al.* (2008) to allow more natural interaction with the space of images. For example, in this system it is possible to pan smoothly around objects by warping the original photos, so that they appear to define a smooth path through space.

### 16.7.3 Volumetric graph cuts

The reconstruction pipeline described in section 16.7.1 has the potential disadvantage that it requires the merging of multiple meshes of the object computed from different viewpoints. Vogiatzis *et al.* (2007) presented a system that uses a volumetric representation of depth to avoid this problem. In other words the 3D space that we wish to reconstruct is divided into a 3D grid, and each constituent element (voxel) is simply labeled as being inside or outside the object. Hence, reconstruction can be viewed as a binary segmentation of the 3D space.

The relative positions of the cameras are computed using a standard bundle adjustment approach. However, the reconstruction problem is now formulated in terms of an energy function consisting of two terms and optimized using graph cuts. First, there is an occupation cost for labeling each voxel as either foreground



**Figure 16.15** Volumetric graph cuts. a-c) Three of the original photos used to build the 3D model. d-f) Renderings of the resulting model from similar viewpoints. Adapted from Vogiatzis *et al.* (2007). ©2007 IEEE.

or background. Second, there is a discontinuity cost for lying at the boundary between the two partitions. We will now examine each of these terms in more detail.

The cost of labeling a voxel as being within the object is set to a very high value if this voxel does not project into the silhouette of the object in each image (i.e., it is not within the visual hull). Conversely, it is assumed that concavities in the object do not extend beyond a fixed distance from this visual hull, so a very high cost is paid if voxels close to the center of the visual hull are labeled as being outside the object. For the remaining voxels, a data-independent cost is set that favors the voxel being part of the object and produces a ballooning tendency which counters the shrinking bias of the graph cut solution, which pays a cost at transitions between the object and the space around it.

The discontinuity cost for lying on the boundary of the object depends on the *photo-consistency* of the voxel; a voxel is deemed photo-consistent if it projects to positions with similar RGB values in all of the cameras from which it is visible. Of course, to evaluate this, we must estimate the set of cameras in which this point is visible. One approach to this is to approximate the shape of the object using the visual hull. However, Vogiatzis *et al.* (2007) propose a more sophisticated method in which each camera votes for the photo-consistency of the voxel based on its pattern of correlation with the other images.

The final optimization problem now takes the form of a sum of unary occupation costs and pairwise terms that encourage the final voxel label field to be smooth. These pairwise terms are modified by the discontinuity cost (an example

of using geodesic distance in graph cuts) so that the transition from foreground to background is more likely in regions where the photo-consistency is high. Figure 16.15 shows an example of a volumetric 3D model computed in this way.

## Discussion

This chapter has not introduced any truly new models; rather we have explored the ramifications of using multiple projective pinhole camera models simultaneously. It is now possible to use these ideas to reconstruct 3D models from camera sequences of rigid objects with well-behaved optical properties. However, 3D reconstruction in more general cases remains an open research problem.

## Notes

**Multi-view geometry:** For more information about general issues in multiview geometry consult the books by Faugeras *et al.* (2001), Hartley & Zisserman (2004) and Ma *et al.* (2004) and the online tutorial by Pollefeys (2002). A summary of multi-view relations was presented by Moons (1998).

**Essential and fundamental matrices:** The essential matrix was described by Longuet-Higgins (1981) and its properties were explored by Huang & Faugeras (1989), Horn (1990) and Maybank (1998) among others. The fundamental matrix was discussed in Faugeras (1992), Faugeras *et al.* (1992), Hartley (1992), and Hartley (1994). The eight-point algorithm for computing the essential matrix is due to Longuet-Higgins (1981). Hartley (1997) described a method for rescaling in the eight-point algorithm that improved its accuracy. Details of the seven-point algorithm for computing the fundamental matrix can be found in Hartley & Zisserman (2004). Nistér (2004) and Stewénius *et al.* (2006) describe methods for the relative orientation problem that work directly with five-point correspondences between the cameras.

**Rectification:** The planar rectification algorithm described in the text is adapted from the description in Hartley & Zisserman (2004). Other variations on planar rectification can be found in Fusiello *et al.* (2000), Loop & Zhang (1999) and Ma *et al.* (2004). The polar rectification procedure is due to Pollefeys *et al.* (1999b).

**Features and feature tracking:** The algorithms in this chapter rely on the computation of distinctive points in the image. Typically, these are found using the Harris corner detector (Harris & Stephens 1988) or the SIFT detector (Lowe 2004). A more detailed discussion of how these points are computed can be found in section 13.2. Methods for tracking points in smooth video sequences (as opposed to matching them across views with a wide baseline) are discussed in Lucas & Kanade (1981), Tomasi & Kanade (1991) and Shi & Tomasi (1994).

**Reconstruction pipelines:** Several authors have described pipelines for computing 3D structure based on a set of images of a rigid object including Fitzgibbon & Zisserman (1998), Pollefeys *et al.* (2004), Brown & Lowe (2005) and Agarwal *et al.* (2009). Newcombe & Davison (2010) present a recent system that runs at interactive speeds. A summary of this area can be found in Moons *et al.* (2009).

**Factorization:** Tomasi & Kanade (1992) developed an exact ML solution for the projection matrices and 3D points in a set of images based on factorization. This solution assumes that the projection process is affine (a simplification of the full pinhole model) and that every point is visible in every image. Sturm & Triggs (1996) developed a similar method that could be used for the full projective camera. Buchanan & Fitzgibbon (2005) discuss approaches to this problem when the complete set of data are not available.

**Bundle adjustment:** Bundle adjustment is a complex topic which is reviewed in Triggs *et al.* (1999). More recently Engels *et al.* (2006) discuss a real-time bundle adjustment approach that works with temporal sub-windows from a video sequence. Recent approaches to bundle adjustment have adopted a conjugate gradient optimization strategy (Byr öd & Åström 2010; Agarwal *et al.* 2010). A public implementation of bundle adjustment has been made available by Lourakis & Argyros (2009). A system which is cutting edge at the time of writing is described in Jeong *et al.* (2010), and recent methods that use multicore processing have also been developed (Wu *et al.* 2011).

**Multi-view reconstruction:** The stereo algorithms in chapters 11 and 12 compute an estimate of depth at each pixel of one or both of the input images. An alternative strategy

is to use an image-independent representation of shape. Examples of such representations include voxel occupancy grids (Vogiatzis *et al.* 2007; Kutulakos & Seitz 2000), level sets (Faugeras & Keriven 1998; Pons *et al.* 2007) and polygonal meshes (Fua & Leclerc 1995, Hernández & Schmitt 2004). Many multi-view reconstruction techniques also enforce the constraints imposed by the silhouettes (see section 14.7.2) on the final solution (e.g., Sinha & Pollefeys 2005; Sinha *et al.* 2007; Kolev & Cremers 2008). A review of multi-view reconstruction techniques can be found in Seitz *et al.* (2006).

## Problems

**Problem 16.1** Sketch the pattern of epipolar lines on the images in figure 16.17a.

**Problem 16.2** Show that the cross product relation can be written in terms of a matrix multiplication so that

$$\mathbf{a} \times \mathbf{b} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}.$$

**Problem 16.3** Consider figure 16.16. Write the direction of the three 3D vectors  $\mathbf{O}_1\mathbf{O}_2$ ,  $\mathbf{O}_1\mathbf{w}$ , and  $\mathbf{O}_2\mathbf{w}$  in terms of the observed image positions  $\mathbf{x}_1, \mathbf{x}_2$  and the rotation  $\boldsymbol{\Omega}$  and translation  $\boldsymbol{\tau}$  between the cameras. The scale of the vectors is unimportant.

The three vectors that you have found must be coplanar. The criterion for three 3D vectors  $\mathbf{a}, \mathbf{b}, \mathbf{c}$  being coplanar can be written as  $\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}) = 0$ . Use this criterion to derive the essential matrix.

**Problem 16.4** A clueless computer vision professor writes:

“The essential matrix is a  $3 \times 3$  matrix that relates image coordinates between two images of the same scene. It contains 8 independent degrees of freedom (it is ambiguous up to scale). It has rank 2. If we know the intrinsic matrices of the two cameras, we can use the essential matrix to recover the rotation and translation between the cameras exactly.”

Edit this statement to make it factually correct.

**Problem 16.5** The essential matrix relates points in two cameras so that

$$\mathbf{x}_2^T \mathbf{E} \mathbf{x}_1 = 0$$

is given by

$$\mathbf{E} = \begin{bmatrix} 0 & 0 & 10 \\ 0 & 0 & 0 \\ -10 & 0 & 0 \end{bmatrix}.$$

What is the epipolar line in image 2 corresponding to the point  $x_1 = [1, -1, 1]$ ? What is the epipolar line in image 2 corresponding to the points  $x_1 = [-5, -2, 1]$ ? Determine the position of the epipole in image 2. What can you say about the motion of the cameras?

**Problem 16.6** Show that we can retrieve the essential matrix by multiplying together the expressions from the decomposition (equations 16.19) as  $\mathbf{E} = \boldsymbol{\tau} \times \boldsymbol{\Omega}$ .

**Problem 16.7** Derive the fundamental matrix relation:

$$\tilde{\mathbf{x}}_2^T \mathbf{A}_2^{-T} \mathbf{E} \mathbf{A}_1^{-1} \tilde{\mathbf{x}}_1 = 0.$$

**Problem 16.8** I intend to compute the fundamental matrix using the eight-point algorithm. Unfortunately, my data set is polluted by 30% outliers. How many iterations of the RANSAC algorithm will I need to run to have a 99% probability of success (i.e., computing the fundamental matrix from eight inliers at least once)? How many iterations will I need if I use an algorithm based on seven points?

**Problem 16.9** We are given the fundamental matrix  $\mathbf{F}_{13}$  relating images 1 and 3 and the fundamental matrix  $\mathbf{F}_{23}$  relating images 2 and 3. I am now given corresponding points  $\mathbf{x}_1$  and  $\mathbf{x}_2$  in images 1 and 2, respectively. Derive a formula for the position of the corresponding point in image 3.

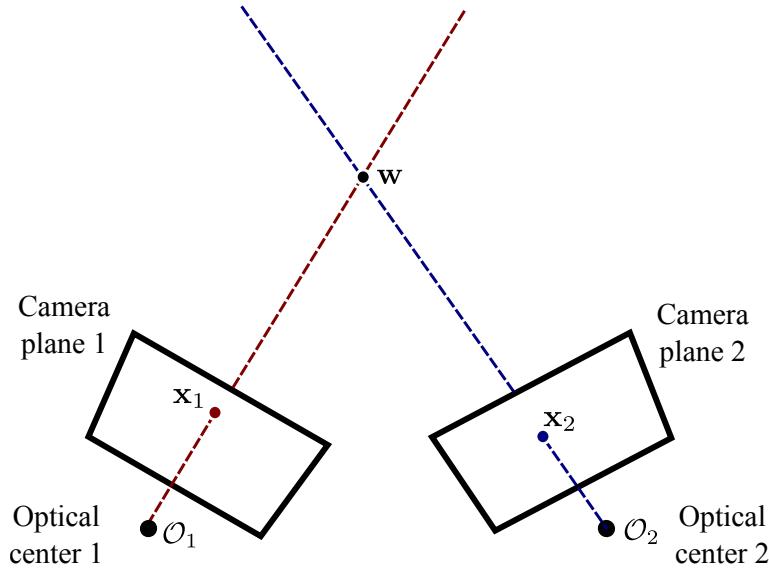
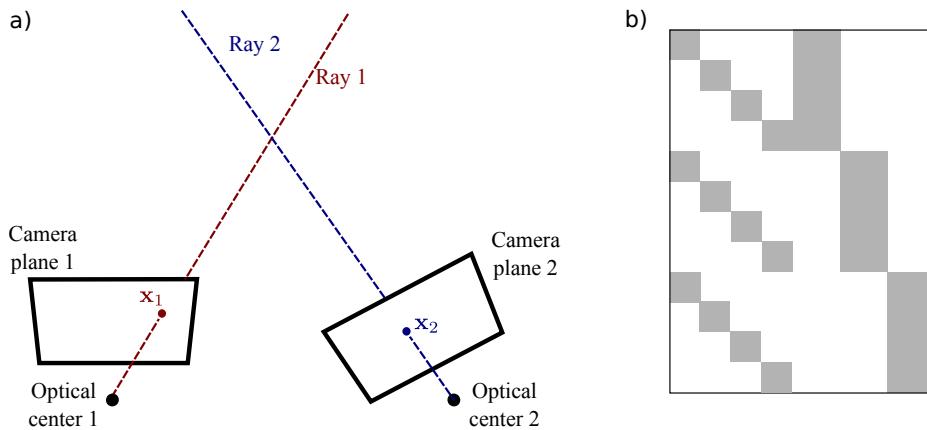


Figure 16.16 Figure for problem 16.3.

**Problem 16.10** Tomasi-Kanade factorization. In the orthographic camera (figure 14.19), the projection process can be described as

$$\begin{aligned} \begin{bmatrix} x \\ y \end{bmatrix} &= \begin{bmatrix} \phi_x & \gamma & \delta_x \\ 0 & \phi_y & \delta_y \end{bmatrix} \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} & \tau_x \\ \omega_{21} & \omega_{22} & \omega_{23} & \tau_y \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \pi_{11} & \pi_{12} & \pi_{13} \\ \pi_{21} & \pi_{22} & \pi_{23} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} + \begin{bmatrix} \tau'_x \\ \tau'_y \end{bmatrix}, \end{aligned}$$

or in matrix form



**Figure 16.17** a) Figure for problem 16.1. c) Figure for problem 16.11. Gray regions represent non-zero entries in this portrait Jacobian matrix.

$$\mathbf{x} = \boldsymbol{\Pi}\mathbf{w} + \boldsymbol{\tau}'$$

Now consider a data matrix  $\mathbf{X}$  containing the positions  $\{\mathbf{x}_{ij}\}_{i,j=1}^{IJ}$  of  $J$  points as seen in  $I$  images so that

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_{11} & \mathbf{x}_{12} & \dots & \mathbf{x}_{iJ} \\ \mathbf{x}_{21} & \mathbf{x}_{22} & \dots & \mathbf{x}_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_{I1} & \mathbf{x}_{I2} & \dots & \mathbf{x}_{IJ}, \end{bmatrix}$$

and where  $\mathbf{x}_{ij} = [x_{ij}, y_{ij}]^T$ .

- (i) Show that the matrix  $\mathbf{X}$  can be written in the form

$$\mathbf{X} = \mathbf{P}\mathbf{W} + \mathbf{T}$$

where  $\mathbf{P}$  contains all of the  $I$   $3 \times 2$  projection matrices  $\{\boldsymbol{\Pi}_i\}_{i=1}^I$ ,  $\mathbf{W}$  contains all of the  $J$  3D world positions  $\{\mathbf{w}_j\}_{j=1}^J$  and  $\mathbf{T}$  contains the translation vectors  $\{\boldsymbol{\tau}'_i\}_{i=1}^I$ .

- (ii) Devise an algorithm to recover the matrices  $\mathbf{P}$ ,  $\mathbf{W}$  and  $\mathbf{T}$  from the measurements  $\mathbf{X}$ . Is your solution unique?

**Problem 16.11** Consider a Jacobian that has a structure of non-zero entries as shown in figure 16.17b. Draw an equivalent image that shows the structure of the non-zero entries in the matrix  $\mathbf{J}^T \mathbf{J}$ . Describe how you would use the Schur complement relation to invert this matrix efficiently.

# **Part VI**

# **Models for vision**



# Part VI: Models for vision

In the final part of this book, we discuss four families of models. There is very little new theoretical material; these models are straight applications of the learning and inference techniques introduced in the first nine chapters. Nonetheless, this material addresses some of the most important machine vision applications: shape modeling, face recognition, tracking and object recognition.

In chapter 17, we discuss models that characterize the shape of objects. This is a useful goal in itself as knowledge of shape can help localize or segment an object. Furthermore, shape models can be used in combination with models for the RGB values to provide a more accurate generative account of the observed data.

In chapter 18, we investigate models that distinguish between the identities of objects and the style in which they are observed; a prototypical example of this type of application would be face recognition. Here the goal is to build a generative model of the data that can separate critical information about identity from the irrelevant image changes due to pose, expression and lighting.

In chapter 19, we discuss a family of models for tracking visual objects through time sequences. These are essentially graphical models based on chains such as those discussed in chapter 11. However, there are two main differences. First, we focus here on the case where the unknown variable is continuous rather than discrete. Second, we do not usually have the benefit of observing the full sequence; we must make a decision at each time based on information from only the past.

Finally, in chapter 20, we consider models for object and scene recognition. An important recent discovery is that good object recognition performance can be achieved using a discrete representation where the image is characterized as an unstructured histogram of *visual words*. Hence, this chapter considers models where the observed data are discrete.

It is notable that all of these families of models are generative; it has proven difficult to integrate complex knowledge about the structure of visual problem into discriminative models.



# Chapter 17

## Models for shape

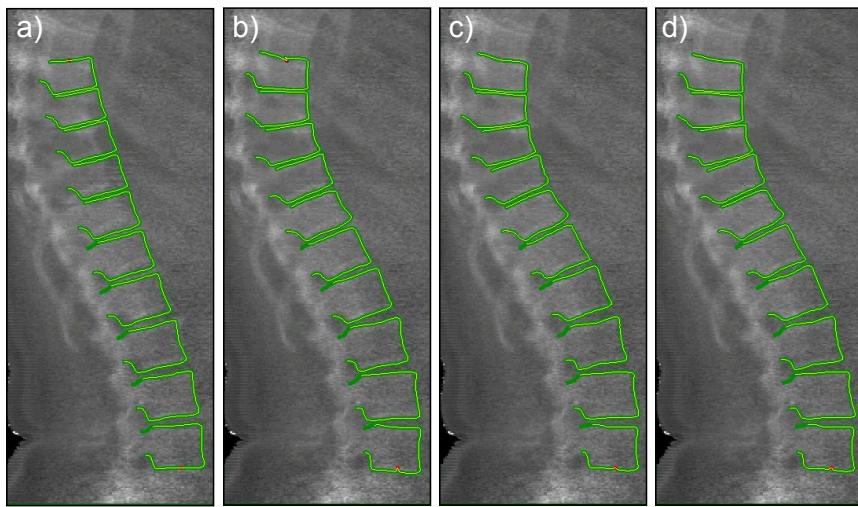
This chapter concerns models for 2D and 3D shape. The motivation for shape models is twofold. First, we may wish to identify exactly which pixels in the scene belong to a given object. One approach to this *segmentation* problem, is to model the outer contour of the object (i.e., the shape) explicitly. Second, the shape may provide information about the identity or other characteristics of the object: it can be used as an intermediate representation for inferring higher-level properties.

Unfortunately, modeling the shape of an object is challenging; we must account for deformations of the object, the possible absence of some parts of the object and even changes in the object topology. Furthermore, the object may be partially occluded, making it difficult to relate the shape model to the observed data.

One possible approach to establishing 2D object shape is to use a *bottom-up approach*; here, a set of boundary fragments are identified using an edge detector (section 13.2.1) and the goal is to connect these fragments to form a coherent object contour. Unfortunately, achieving this goal has proved surprisingly elusive. In practice, the edge detector finds extraneous edge fragments that are not part of the object contour, and misses others that are part of the true contour. Hence it is difficult to connect the edge fragments in a way that correctly reconstructs the contour of an object.

The methods in this chapter adopt the *top-down approach*. Here, we impose prior information about the object that constrains the possible contour shapes and hence reduces the search space. We will investigate a range of different types of prior information; in some models this will be very weak (e.g., the object boundary is smooth) and in others it will be very strong (e.g., the object boundary is a 2D projection of a particular 3D shape).

To motivate these models, consider the problem of fitting a 2D geometric model of the spine to medical imaging data (figure 17.1). Our goal is to characterize this complex shape with only a few parameters, which could subsequently be used as a basis for diagnosing medical problems. This is challenging because the local edge information in the image is weak. However, in our favor we have very strong prior knowledge about the possible shapes of the spine.



**Figure 17.1** Fitting a spine model. a) The spine model is initialized at a fixed position in the image. b-d) The model then adapts to the image until it describes the data as well as possible. Models that ‘crawl’ across the image in this way are known as *active shape models*. Figure provided by Tim Cootes and Martin Roberts.

## 17.1 Shape and its representation

Before we can introduce concrete models for identifying shape in images, we should define exactly what we mean by the word ‘shape’. A commonly-used definition comes from Kendall (1984) who states that shape “is all the geometrical information that remains when location, scale and rotational effects are filtered out from an object.” In other words, the shape consists of whatever geometric information is invariant to a similarity transform. Depending on the situation, we may generalize this definition to other transformation types such as Euclidean or affine.

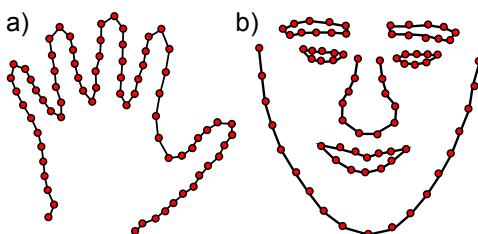
Problem 17.1

We will also need a way to represent the shape. One approach is to directly define an algebraic expression that describes the contour. For example, a *conic* defines points  $\mathbf{x} = [x, y]^T$  as lying on the contour if they obey

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} \alpha & \beta & \gamma \\ \beta & \delta & \epsilon \\ \gamma & \epsilon & \zeta \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0. \quad (17.1)$$

This family of shapes includes circles, ellipses, parabolas, and hyperbolas, where the choice of shape depends on the parameters  $\boldsymbol{\theta} = \{\alpha, \beta, \gamma, \delta, \epsilon, \zeta\}$ .

Algebraic models are attractive in that they provide a closed form expression for the contour, but their applicability is extremely limited; it is difficult to define a mathematical expression that describes a family of complex shapes such as the spine in figure 17.1. It *is* possible to model complex objects as superimposed collections



**Figure 17.2** Landmark points. Object shape can be represented with sets of landmark points. a) Here the landmark points (red dots) define a single open contour that describes the shape of a hand. b) In this example the landmark points are connected into sets of open and closed contours that describe the regions of the face.

of geometric primitives like conics, but the resulting models are unwieldy and lose many of the desirable properties of the closed form representation.

Most of the models in this chapter adopt a different approach; they define the shape using a set of *landmark points* (figure 17.2). One way to think of landmark points is as a set of discrete samples from one or more underlying continuous contours. The connectivity of the landmark points varies according to the model: they may be ordered and hence represent a single continuous contour, ordered with wrapping, so they represent a closed contour or may have a more complex organization so that they can represent a collection of closed and open contours.

The contour can be reconstituted from the landmark points by interpolating between them according to the connectivity. For example, in figure 17.2, the contours have been formed by connecting adjacent landmark points with straight lines. In more sophisticated schemes, the landmark points may indirectly determine the position of a smooth curve by acting as the control points of a spline model.

## 17.2 Snakes

As a base model, we will consider *parametric contour models*, which are sometimes also referred to as *active contour models* or *snakes*. These provide only very weak a priori geometric information; they assume that we know the topology of the contour (i.e., open or closed) and that it is smooth but nothing else. Hence they are suitable for situations where little is known about the contents of the image. We will consider a closed contour defined by a set of  $N$  2D landmark points  $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_N]$ , which are unknown.

Our goal is to find the configuration of landmark points that best explains the shape of an object in the image. We construct a generative model for the landmark positions, which is determined by a likelihood term (which indicates the agreement of this configuration with the image data) and a prior term (which encompasses our prior knowledge about the frequency with which different configurations occur).

The likelihood  $Pr(\mathbf{x}|\mathbf{W})$  of the RGB image data  $\mathbf{x}$  given the landmark points  $\mathbf{W}$  should be high when the landmark points  $\mathbf{w}$  lie on or close to edges in the image, and low when they lie in flat regions. One possibility for the likelihood is hence

$$Pr(\mathbf{x}|\mathbf{W}) \propto \prod_{n=1}^N \exp [\text{sobel}[\mathbf{x}, \mathbf{w}_n]], \quad (17.2)$$

where the function  $\text{sobel}[\mathbf{x}, \mathbf{w}]$  returns the magnitude of the Sobel edge operator (i.e., the square root of the sum of the squared responses to the horizontal and vertical Sobel filters – see section 13.1.3) at 2D position  $\mathbf{w}$  in the image.

This likelihood will be high when the landmark points are on a contour and low otherwise, as required. However, it has a rather serious disadvantage in practice; in completely flat parts of the image, the Sobel edge operator returns a value of zero. Consequently, if the landmark points lie in flat parts of the image, there is no information about which way to move them to improve the fit (figure 17.3a-b).

A better approach is to find a set of discrete edges using the Canny edge detector (section 13.2.1) and then compute a *distance transform*; each pixel is allotted a value according to its squared distance from the nearest edge pixel. The likelihood now becomes

$$Pr(\mathbf{x}|\mathbf{W}) \propto \prod_{n=1}^N \exp [-(\text{dist}[\mathbf{x}, \mathbf{w}_n])^2], \quad (17.3)$$

where the function  $\text{dist}[\mathbf{x}, \mathbf{w}]$  returns the value of the distance transform at position  $\mathbf{w}$  in the image. Now the likelihood is large when the landmark points all fall close to edges in the image (where the distance transform is low) and smoothly decreases as the distance between the landmark points and the edges increases (figure 17.3c-d). In addition to being pragmatic, this approach also has an attractive interpretation; the ‘squared distance’ objective function is equivalent to assuming that the measured position of the edge is a noisy estimate of the true edge position and that the noise is additive and normally distributed.

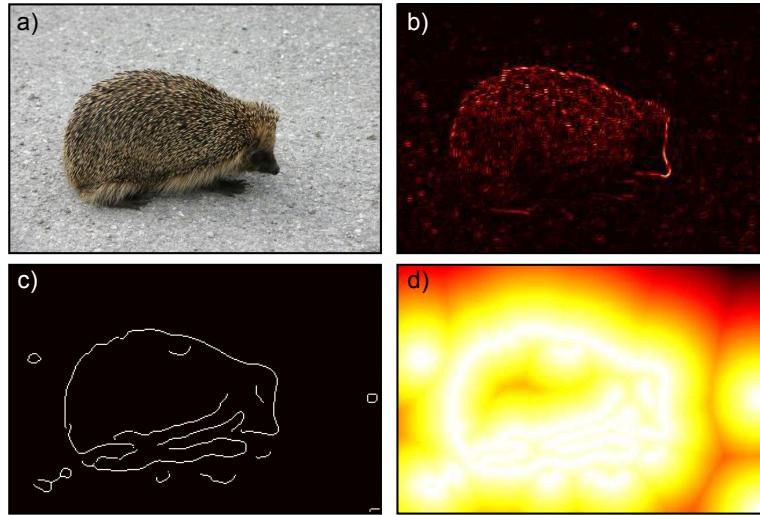
If we were to find the landmark points  $\mathbf{W}$  based on this criterion alone, then each point would be separately attracted to a strong edge in the image and the result would probably not form a coherent shape; they might even all be attracted to the same position. To avoid this problem and complete the model, we define a prior that favors smooth contours with low curvature. There are various ways to do this, but one possibility is to choose a prior with two terms

$$Pr(\mathbf{W}) \propto \prod_{n=1}^N \exp [\alpha \text{space}[\mathbf{w}, n] + \beta \text{curve}[\mathbf{w}, n]], \quad (17.4)$$

where the scalars  $\alpha$  and  $\beta$  control the relative contribution of these terms.

The first term encourages the spacing of the points around the contour to be even; the function  $\text{space}[\mathbf{w}, n]$  returns a high value if spacing between the  $n^{th}$  contour point  $\mathbf{w}_n$  and its neighbors is close to the average spacing between neighboring points along the contour:

$$\begin{aligned} \text{space}[\mathbf{w}, n] &= \\ &- \left( \frac{\sum_{n=1}^N \sqrt{(\mathbf{w}_n - \mathbf{w}_{n-1})^T (\mathbf{w}_n - \mathbf{w}_{n-1})}}{N} - \sqrt{(\mathbf{w}_n - \mathbf{w}_{n-1})^T (\mathbf{w}_n - \mathbf{w}_{n-1})} \right)^2, \end{aligned} \quad (17.5)$$



**Figure 17.3** Likelihood for landmark points. a) Original image. b) Output of Sobel edge operator - one possible scheme is to assign the landmark points a high likelihood if the Sobel response is strong at their position. This encourages the landmark points to lie on boundaries in the image, but the response is flat in regions away from the boundaries, and this makes it difficult to apply gradient methods to fit the model. c) Results of applying Canny edge detector. d) Negative distance from nearest Canny edge. This function is also high at boundaries in the image but varies smoothly in regions away from the boundaries.

where we assume that the contour is closed so that  $\mathbf{w}_0 = \mathbf{w}_N$ .

The second term in the prior,  $\text{curve}[\mathbf{w}, n]$ , returns larger values when the curvature is small and so encourages the contour to be smooth. It is defined as

Problem: 17.3

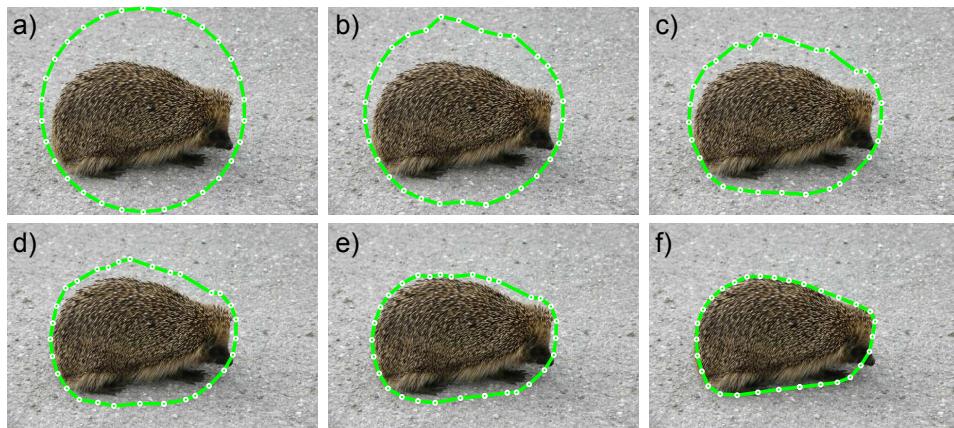
$$\text{curve}[\mathbf{w}, n] = -(\mathbf{w}_{n-1} - 2\mathbf{w}_n + \mathbf{w}_{n+1})^T (\mathbf{w}_{n-1} - 2\mathbf{w}_n + \mathbf{w}_{n+1}), \quad (17.6)$$

where again we assume that the contour is closed so  $\mathbf{w}_0 = \mathbf{w}_N$  and  $\mathbf{w}_{N+1} = \mathbf{w}_1$ .

There are only two parameters in this model (the weighting terms,  $\alpha$  and  $\beta$ ). These can be learned from training examples, but for simplicity, we will assume that they were set by hand so there is no need for an explicit learning procedure.

### 17.2.1 Inference

In inference, we observe a new image  $\mathbf{x}$  and try to fit the points  $\{\mathbf{w}_i\}$  on the contour so that they describe the image as well as possible. To this end, we use a maximum a posteriori criterion



**Figure 17.4** Snakes. The snake is defined by a series of connected landmark points. a-f) As the optimization proceeds and the posterior probability of these points increases, the snake contour crawls across the image. The objective function is chosen so that the landmark points are attracted to edges in the image, but also try to remain equidistant from one another and form a shape with low curvature.

$$\begin{aligned}\hat{\mathbf{W}} = \operatorname{argmax}_{\mathbf{W}} [Pr(\mathbf{W}|\mathbf{x})] &= \operatorname{argmax}_{\mathbf{W}} [Pr(\mathbf{x}|\mathbf{W})Pr(\mathbf{W})] \\ &= \operatorname{argmax}_{\mathbf{W}} [\log[Pr(\mathbf{x}|\mathbf{W})] + \log[Pr(\mathbf{W})]] .\end{aligned}\quad (17.7)$$

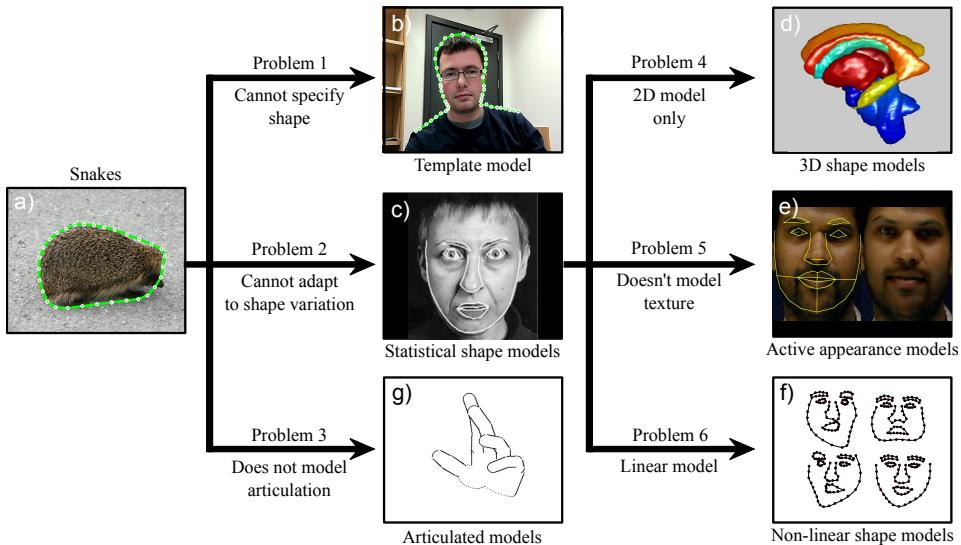
The optimum of this objective function cannot be found in closed form and we must apply a general nonlinear optimization procedure such as the Newton method (appendix B). The number of unknowns is twice the number of the landmark points as each point has an  $x$  and  $y$  coordinate.

#### Problem 17.4

An example of this fitting procedure is illustrated in figure 17.4. As the minimization proceeds, the contour crawls around the image, seeking the set of landmark points with highest posterior probability. For this reason, this type of contour model is sometimes known as a *snake* or *active contour model*.

The final contour in figure 17.4 fits snugly around the outer boundary of the hedgehog. However, it has not correctly delineated the nose region; this is an area of high curvature and the scale of the nose is smaller than the distance between adjacent landmark points. The model can be improved by using a likelihood that depends on the image at positions along the contour between the landmark points. In this way, we can develop a model that has few unknowns but that depends on the image in a dense way.

A second problem with this model is that the optimization procedure can get stuck in local optima. One possibility is to restart the optimization from a number of different initial conditions, and choose the final solution with the highest



**Figure 17.5** Shape models. a) The snake model only assumes that the contour is smooth. b) The template model assumes a priori knowledge of the object shape. c) Active shape models are a compromise in which some information about the object class is known, but the model can adapt to the particular image. d-f) We consider three extensions to the active shape models that describe 3D shape, simultaneously model intensity variation, and describe more complex shape variation, respectively. g) Finally, we investigate models in which a priori information about the structure of an articulated object is provided.

probability. Alternatively, it is possible to modify the prior to make inference more tractable. For example, the spacing term can be redefined as

$$\text{space}[\mathbf{w}, n] = - \left( \mu_s - \sqrt{(\mathbf{w}_n - \mathbf{w}_{n-1})^T (\mathbf{w}_n - \mathbf{w}_{n-1})} \right)^2, \quad (17.8)$$

which encourages the spacing of the points to be close to a pre-defined value  $\mu_s$ , and hence encourages solutions at a certain scale. This small change makes inference in the model much easier: the prior is now a 1D Markov random field, with cliques that consist of each element and its two neighbors (due to the term in the curve  $[\bullet, \bullet]$ ). The problem can now be discretized and solved efficiently using dynamic programming methods (see section 11.8.4).

### 17.2.2 Problems with the snake model

The simple snake model described previously has a number of limitations; it embodies only weak information about the smoothness of the object boundary and as

such:

- It is not useful where we know the object shape but not its position within the image.
- It is not useful where we know the class of the object (e.g. a face) but not the particular instance (e.g., whose face).
- The snake model is 2D and does not understand that some contours are created by projecting a 3D surface through the camera model.
- It cannot model articulated objects such as the human body.

We remedy these various problems in the subsequent parts of this chapter (figure 17.5). In section 17.3, we investigate a model in which we know exactly the shape we are looking for and the only problem is to find its position in the image. In sections 17.4-17.8, we investigate models that describe the statistical variation in a class of objects and can find unseen examples of the same class. Finally, in section 17.9, we discuss articulated models.

### 17.3 Shape templates

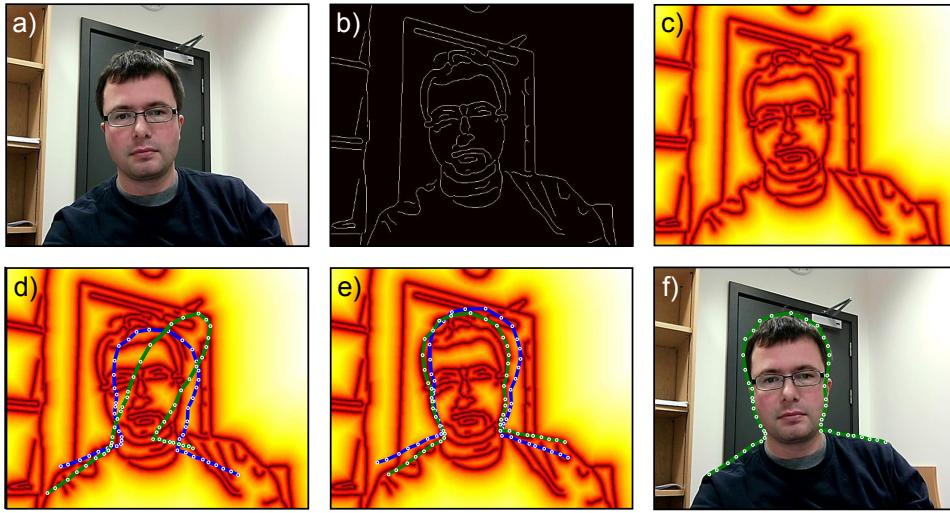
We will now consider *shape template* models. These impose the strongest possible form of geometric information; it is assumed that we know the shape exactly. So, whereas the snake model started with a circular configuration and adapted this to fit the image, the template model starts with the correct shape of the object and merely tries to identify its position, scale, and orientation in the image. More generally, the problem is to determine the parameters  $\Psi$  of the transformation that maps this shape onto the current image.

We will now develop a generative model that determines the likelihood of the observed image data as a function of these transformation parameters. The underlying representation of the shape is again a set  $\mathbf{W} = \{\mathbf{w}_n\}_{n=1}^N$  of 2D landmark points which are now assumed known. However, to explain the observed data, the points must be mapped into the image by a 2D transformation  $\text{trans}[\mathbf{w}, \Psi]$  where  $\Psi$  contains the parameters of the transformation model. For example, with a similarity transformation,  $\Psi$  would consist of the rotation angle, scaling factor, and 2D translation vector.

As with the snake model, we choose the likelihood of image data  $\mathbf{x}$  to be dependent on the negative distance from the closest edge in the image

$$Pr(\mathbf{x}|\mathbf{W}, \Psi) \propto \prod_{n=1}^N \exp \left[ -(\text{dist}[\mathbf{x}, \text{trans}[\mathbf{w}_n, \Psi]])^2 \right], \quad (17.9)$$

where the function  $\text{dist}[\mathbf{x}, \mathbf{w}]$  returns the distance transform of the image  $\mathbf{x}$  at position  $\mathbf{w}$ . Again the likelihood is larger when the landmark points all fall in regions where the distance transform is low (i.e., close to edges in the image).



**Figure 17.6** Shape templates. Here the shape of the object is known and only the affine transformation relating the shape to the image is unknown.  
 a) Original image. b) Results of applying Canny edge detector. c) Distance transformed image. The intensity represents the distance to the nearest edge. d) Fitting a shape template. The template is initialized with a randomly chosen affine transformation (blue curve). After optimization (green curve), the landmark points, which define the curve, have moved toward positions with lower values in the distance image. In this case, the fitting procedure has converged to a local optimum and the correct silhouette has not been identified. e) When we start the optimization from nearer the true optimum, it converges to the global maximum. f) Final fit of template.

### 17.3.1 Inference

The only unknown variables in the template model are the transformation parameters  $\Psi$ . For simplicity, we will assume that we have no prior knowledge of these parameters and adopt the maximum likelihood approach in which we maximize the log-likelihood  $L$ :

$$\begin{aligned}\hat{\Psi} = \underset{\Psi}{\operatorname{argmax}} [L] &= \underset{\Psi}{\operatorname{argmax}} [\log [Pr(\mathbf{x}|\mathbf{W}, \Psi)]] \\ &= \underset{\Psi}{\operatorname{argmax}} \left[ \sum_{n=1}^N -(\text{dist}[\mathbf{x}, \text{trans}[\mathbf{w}_n, \Psi]])^2 \right]. \quad (17.10)\end{aligned}$$

There is no closed form solution to this problem, so we must rely on nonlinear optimization. To this end, we must take the derivative of the objective function with respect to the unknowns, and for this we employ the chain rule:

$$\frac{\partial L}{\partial \Psi} = - \sum_{n=1}^N \sum_{j=1}^2 \frac{\partial(\text{dist}[\mathbf{x}, \mathbf{w}'_n])^2}{\partial w'_{jn}} \frac{\partial w'_{jn}}{\partial \Psi}, \quad (17.11)$$

where  $\mathbf{w}'_n = \text{trans}[\mathbf{w}_n, \Psi]$  is the transformed point, and  $w'_{jn}$  is the  $j^{th}$  entry in this 2D vector.

The first term on the right hand side of equation 17.11 is easy to compute. The derivative of the distance image can be approximated in each direction by evaluating horizontal or vertical derivative filters (section 13.1.3) at the current position  $\mathbf{w}'_n$ . In general this will not exactly fall on the center of a pixel and so the derivative values should be interpolated from nearby pixels. The second term depends on the transformation in question.

Figure 17.6 illustrates the fitting procedure for a template model based on an affine transform. As the optimization proceeds the contour crawls over the image as it tries to find a more optimal position. Unfortunately, there is no guarantee that the optimization will converge to the true position. As for the snake model, one way to deal with this problem is to restart the optimization from many different places and choose the solution with the overall maximum log likelihood. Alternatively, we could initialize the template position more intelligently; for example, in this case, the initial position might be based on the output of a face detector. It is also possible to restrict the possible solutions by imposing prior knowledge about the possible transformation parameters and using a maximum a posteriori formulation.

### 17.3.2 Inference with iterative closest point algorithm

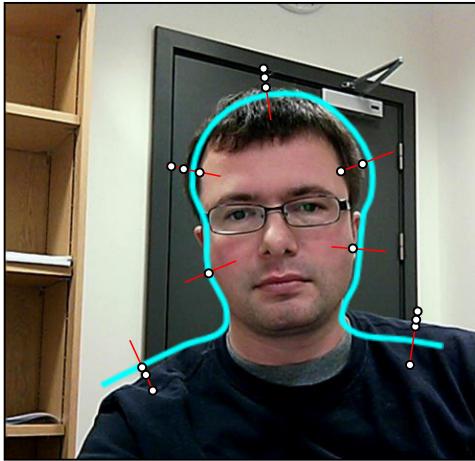
In chapter 15 we saw that the transformation mapping one set of points to another can be computed in closed form for several common families of transformations. However, the template model cannot be fit in closed form because we do not know which edge points in the image correspond to each landmark point in the model.

This suggests a different approach to inference for this model. The *iterative closest point (ICP)* algorithm alternately matches points in the image to the landmark points and computes the best transformation. More precisely:

- Each landmark point  $\mathbf{w}_n$  is transformed into the image as  $\mathbf{w}'_n = \text{trans}[\mathbf{w}_n, \Psi]$  using the current parameters  $\Psi$ .
- Each transformed point  $\mathbf{w}'_n$  is associated with the *closest* image point  $\mathbf{y}_n$  that lies on an edge.
- The transformation parameters  $\Psi$  that best map the landmark points  $\{\mathbf{w}_n\}_{n=1}^N$  to the image points  $\{\mathbf{y}_n\}_{n=1}^N$  are computed in closed form.

This procedure is repeated until convergence. As the optimization proceeds, the choice of closest points changes – a process known as *data association* – and so the computed transformation parameters also evolve.

A variation of this approach considers matching the landmark points to edges in a direction that is perpendicular to the contour (figure 17.7). This means that the search for the nearest edge point is now only in 1D, and this can also make



**Figure 17.7** Iterative closest point algorithm. We associate each landmark point (positions where the red normals join the blue contour) with a single edge point in the image. In this example, we search along the normal direction to the contour (red lines). There are usually several points identified by an edge detector along each normal (circles). In each case, we choose the closest – a process known as data association. We compute the transformation that best maps the landmark points to these closest edge positions. This moves the contour and potentially changes the closest points in the next iteration.

the fitting more robust in some circumstances. This is most practical for smooth contour models where the normals can be computed in closed form.

## 17.4 Statistical shape models

The template model is useful when we know exactly what the object is. Conversely, the snake model is useful when we have very little prior information about the object. In this section we describe a model that lies in between these two extremes. *Statistical shape models*, *active shape models*, or *point distribution models* describe the variation within a class of objects, and so can adapt to an individual shape from that class even if they have not seen this specific example before.

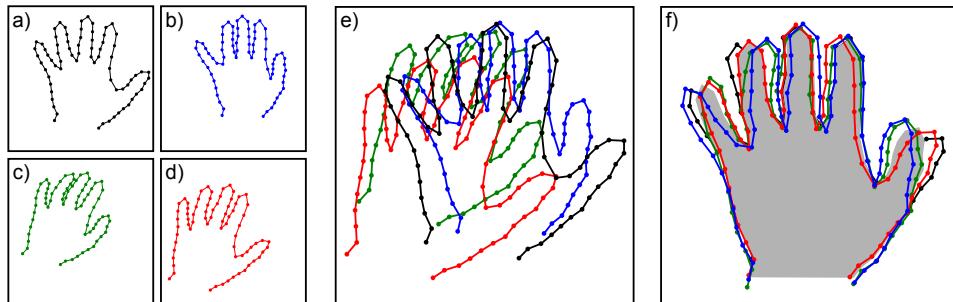
As with the template and snake models, the shape is described in terms of the positions of the  $N$  landmark points  $\{\mathbf{w}_n\}_{n=1}^N$ , and the likelihood of these points depends on their proximity to edges in the image. For example, we might choose the likelihood of the  $i^{th}$  training image data  $\mathbf{x}_i$  to be

$$Pr(\mathbf{x}_i|\mathbf{w}_i) \propto \prod_{n=1}^N \exp [-(\text{dist}[\mathbf{x}_i, \text{trans}[\mathbf{w}_{in}, \Psi_i]])^2], \quad (17.12)$$

where  $\mathbf{w}_{in}$  is the  $n^{th}$  landmark point in the  $i^{th}$  training image and  $\text{dist}[\bullet, \bullet]$  is a function that computes the distance to the nearest Canny edge in the image.

However, we now define a more sophisticated prior model over the landmark positions, which are characterized as a compound vector  $\mathbf{w}_i = [\mathbf{w}_{i1}^T, \mathbf{w}_{i2}^T, \dots, \mathbf{w}_{iN}^T]^T$  containing all of the  $x$ - and  $y$ -positions of the landmark points in the  $i^{th}$  image. In particular, we model the density  $Pr(\mathbf{w}_i)$  as a normal distribution so that

$$Pr(\mathbf{w}_i) = \text{Norm}_{\mathbf{w}_i}[\boldsymbol{\mu}, \boldsymbol{\Sigma}], \quad (17.13)$$



**Figure 17.8** Generalized Procrustes analysis. a-d) Four training shapes. e) Superimposing the training shapes shows that they are not aligned well. f) The goal of generalized Procrustes analysis is simultaneously to align all of the training shapes with respect to a chosen transformation family. Here, the images are aligned with respect to a similarity transformation (gray region illustrates mean shape). After this procedure, the remaining variation is described by a statistical shape model.

where the mean  $\mu$  captures the average shape, and the covariance  $\Sigma$  captures how different instances vary around this mean.

#### 17.4.1 Learning

In learning, our goal is to estimate the parameters  $\theta = \{\mu, \Sigma\}$  based on training data. Each training example consists of a set of landmark points that have been hand-annotated on one of the training images.

Unfortunately, the training examples are not usually geometrically aligned when we receive them. In other words, we receive the transformed data examples  $\mathbf{w}'_i = [\mathbf{w}'_1^T, \mathbf{w}'_2^T, \dots, \mathbf{w}'_N^T]^T$ , where

$$\mathbf{w}'_{in} = \text{trans}[\mathbf{w}_{in}, \Psi_i]. \quad (17.14)$$

Before we can learn the parameters  $\mu$  and  $\Sigma$  of the normal, we must align the examples using the inverse of this transformation

$$\mathbf{w}_{in} = \text{trans}[\mathbf{w}'_{in}, \Psi_i^-], \quad (17.15)$$

where  $\{\Psi_i^-\}_{i=1}^I$  are the parameters of the inverse transformations.

#### Alignment of training examples

The method for aligning the training examples is known as *generalized Procrustes analysis* (figure 17.8) and exploits the ‘chicken and egg’ structure of the underlying problem; if we knew the mean shape  $\mu$ , then it would be easy to estimate the parameters  $\{\Psi_i^-\}_{i=1}^I$  of the transformations that best map the observed points to

this mean. Similarly, if we knew these transformations, we could easily compute the mean shape by transforming the points and taking the mean of the resulting shapes. Generalized Procrustes analysis takes an alternating approach to this problem in which we repeatedly

1. Update the transformations using the criterion:

$$\hat{\Psi}_i^- = \operatorname{argmin}_{\Psi_i^-} \left[ \sum_{n=1}^N |\operatorname{trans}[\mathbf{w}'_{in}, \Psi_i^-] - \boldsymbol{\mu}_n|^2 \right], \quad (17.16)$$

where  $\boldsymbol{\mu} = [\boldsymbol{\mu}_1^T, \boldsymbol{\mu}_2^T, \dots, \boldsymbol{\mu}_N^T]^T$ . This can be achieved in closed form for common transformation families such as the Euclidean, similarity, or affine transforms (see section 15.2).

2. Update the mean template

$$\hat{\boldsymbol{\mu}} = \operatorname{argmin}_{\boldsymbol{\mu}} \left[ \sum_{n=1}^N |\operatorname{trans}[\mathbf{w}'_{in}, \Psi_i^-] - \boldsymbol{\mu}_n|^2 \right]. \quad (17.17)$$

In practice, to optimize this criterion we inversely transform each set of points using equation 17.15 and take the average of the resulting shape vectors. It is important to normalize the mean vector  $\boldsymbol{\mu}$  after this stage to uniquely define the absolute scale.

Typically, we would initialize the mean vector  $\boldsymbol{\mu}$  to be one of the training examples, and iterate between these steps until there was no further improvement.

After convergence, we can fit the statistical model

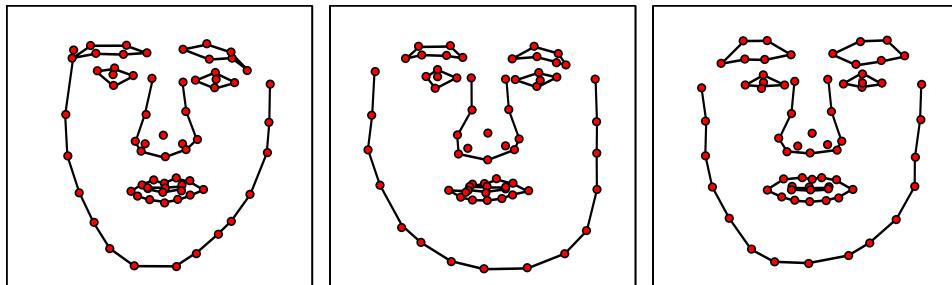
$$Pr(\mathbf{w}_i) = \operatorname{Norm}_{\mathbf{w}_i}[\boldsymbol{\mu}, \boldsymbol{\Sigma}]. \quad (17.18)$$

We already know the mean  $\boldsymbol{\mu}$ . We can compute the covariance  $\boldsymbol{\Sigma}$  using the maximum likelihood approach from the aligned shapes  $\{\mathbf{w}_i\}_{i=1}^I$ . Figure 17.9 visualizes a shape model for the human face that was learned using this technique.

### 17.4.2 Inference

In inference, we fit the model to a new image. The simplest way to do this is to take a brute force optimization approach in which we estimate the unknown landmark points  $\mathbf{w} = \{\mathbf{w}_n\}_{n=1}^N$ , and the parameters  $\boldsymbol{\Psi}$  of the transformation model so that

$$\hat{\mathbf{w}} = \operatorname{argmax}_{\mathbf{w}} \left[ \max_{\boldsymbol{\Psi}} \left[ \sum_{n=1}^N -(\operatorname{dist}[\mathbf{x}_i, \operatorname{trans}[\mathbf{w}_n, \boldsymbol{\Psi}]])^2 + \log[\operatorname{Norm}_{\mathbf{w}}[\boldsymbol{\mu}, \boldsymbol{\Sigma}]] \right] \right]. \quad (17.19)$$



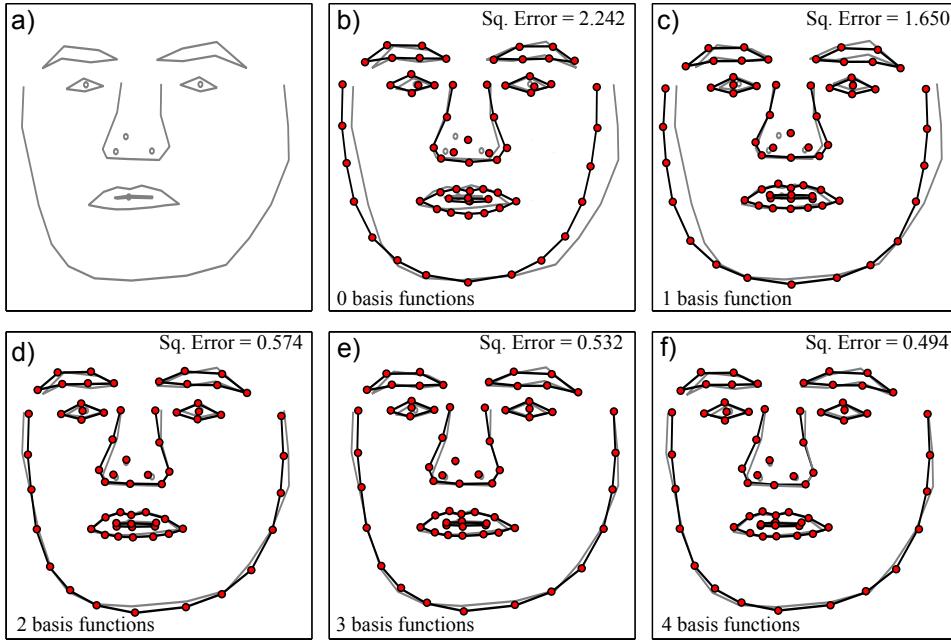
**Figure 17.9** Statistical model of face shape. Three samples drawn from normally distributed model of landmark vectors  $\mathbf{w}$ . After generating each 136 dimensional vector, it was reshaped to create a  $68 \times 2$  matrix containing 68 ( $x,y$ ) points which are plotted in the figure. The samples look like plausible examples of face shapes.

One way to optimize this objective function is to alternate between estimating the transformation parameters and the landmark points. For fixed landmark points  $\{\mathbf{w}_n\}_{n=1}^N$  we are effectively fitting a shape template model and we can use the methods of sections 17.3.1 to 17.3.2 to find the transformation parameters  $\Psi$ . For fixed transformation parameters, it is possible to estimate the landmark points by nonlinear optimization of the objective function.

As with the template model, the statistical shape model ‘crawls’ across the image as the optimization discovers improved ways to make the model agree with the image. However, unlike the template model, it can adapt its shape to match the idiosyncrasies of the particular object in this image as in figure 17.1.

Unfortunately this statistical shape model has some practical disadvantages due to the number of variables involved. In the fitting procedure, we must optimize over the landmark points themselves. If there are  $N$  landmark points, then there are  $2N$  variables over which to optimize. For the face model in figure 15.9 there would be 136 variables, and the resulting optimization is quite costly. Moreover, we will require a large number of training examples to accurately estimate the covariance  $\Sigma$  of these variables.

Furthermore, it is not clear that all these parameters are needed; the classes of objects that are suited to this normally distributed model (hands, faces, spines, etc.) are quite constrained in their shape variation, and so many of the parameters of this full model are merely describing noise in the training shape annotation. In the next section we describe a related model which uses fewer unknown variables (and so can be fit more efficiently) and fewer parameters (and so can be learned from less data).



**Figure 17.10** Approximation of face by weighting basis functions. a) Original face. b) Approximating original (gray) by mean face  $\mu$ . c) Approximating original by mean face plus optimal weighting  $h_{i1}$  of basis function  $\phi_1$ . d-f) Adding further weighted basis functions. As more terms are added, the approximation becomes closer to the original. With only four basis functions, the model can explain 78% of the variation of the original face around the mean.

## 17.5 Subspace shape models

The subspace shape model exploits the structure inherent in the covariance of the landmark points. In particular, it assumes that the shape vectors  $\{\mathbf{w}_i\}_{i=1}^I$  all lie very close to a  $K$ -dimensional linear subspace (see figure 7.19) and describes the shape vectors as resulting from the process

$$\mathbf{w}_i = \mu + \Phi \mathbf{h}_i + \epsilon_i, \quad (17.20)$$

where  $\mu$  is the mean shape,  $\Phi = [\phi_1, \phi_2, \dots, \phi_K]$  is a portrait matrix containing in its columns  $K$  basis functions  $\{\phi_k\}_{k=1}^K$  that define the subspace, and  $\epsilon_i$  is an additive normal noise term with spherical covariance  $\sigma^2 \mathbf{I}$ . The term  $\mathbf{h}_i$  is a  $K \times 1$  hidden variable where each element is responsible for weighting one of the basis functions. This can be seen more clearly by re-writing equation 17.20 as

$$\mathbf{w}_i = \boldsymbol{\mu} + \sum_{k=1}^K \phi_k h_{ik} + \epsilon_i, \quad (17.21)$$

where  $h_{ik}$  is the  $k^{th}$  element of the vector  $\mathbf{h}_i$ .

The principle of the subspace model is to approximate the shape vector by the deterministic part of this process so that

$$\mathbf{w}_i \approx \boldsymbol{\mu} + \sum_{k=1}^K \phi_k h_{ik}, \quad (17.22)$$

and so now we can represent the  $2N \times 1$  vector  $\mathbf{w}_i$  using only the  $K \times 1$  vector  $\mathbf{h}_i$ .

For constrained data sets such as the spine, hand, and face models, it is remarkable how good this approximation can be, even when  $K$  is set to quite a small number. For example, in figure 17.10 the face is very well approximated by taking a weighted sum of only  $K = 4$  basis functions. We will exploit this phenomenon when we fit the shape model; we now optimize over the weights of the basis functions rather than the landmark points themselves, and this results in considerable computational savings.

Of course, the approximation in figure 17.10 only works because (i) we have selected a set of basis functions  $\Phi$  that are suitable for representing faces and (ii) we have chosen the weights  $\mathbf{h}_i$  appropriately to describe this face. We will now take a closer look at the model and how to find these quantities.

### 17.5.1 Probabilistic principal component analysis

The particular subspace model that we will apply here is known as *probabilistic principal component analysis* or PPCA for short. To define the model, we re-express equation 17.20 in probabilistic terms:

$$Pr(\mathbf{w}_i | \mathbf{h}_i, \boldsymbol{\mu}, \Phi, \sigma^2) = \text{Norm}_{\mathbf{w}_i} [\boldsymbol{\mu} + \Phi \mathbf{h}_i, \sigma^2 \mathbf{I}], \quad (17.23)$$

where  $\boldsymbol{\mu}$  is a  $2N \times 1$  mean vector,  $\Phi$  is a  $2N \times K$  matrix containing  $K$  basis functions in its columns, and  $\sigma^2$  controls the degree of additive noise. In the context of this model, the basis functions are known as *principal components*. The  $K \times 1$  hidden variable  $\mathbf{h}_i$  weights the basis functions and determines the final positions on the subspace, before the additive noise component is added.

To complete the model, we also define a prior over the hidden variable  $\mathbf{h}_i$ , and we choose a spherical normal distribution for this:

$$Pr(\mathbf{h}_i) = \text{Norm}_{\mathbf{h}_i} [\mathbf{0}, \mathbf{I}]. \quad (17.24)$$

By marginalizing the joint distribution  $Pr(\mathbf{w}_i, \mathbf{h}_i)$  with respect to the hidden variable  $\mathbf{h}_i$ , we can retrieve the prior density  $Pr(\mathbf{w}_i)$ , and this is given by

$$\begin{aligned}
Pr(\mathbf{w}_i) &= \int Pr(\mathbf{w}_i|\mathbf{h}_i)Pr(\mathbf{h}_i)d\mathbf{h}_i \\
&= \int \text{Norm}_{\mathbf{w}_i}[\boldsymbol{\mu} + \boldsymbol{\Phi}\mathbf{h}_i, \sigma^2\mathbf{I}] \text{Norm}_{\mathbf{h}_i}[\mathbf{0}, \mathbf{I}]d\mathbf{h}_i \\
&= \text{Norm}_{\mathbf{w}_i}[\boldsymbol{\mu}, \boldsymbol{\Phi}\boldsymbol{\Phi}^T + \sigma^2\mathbf{I}].
\end{aligned} \tag{17.25}$$

This algebraic result is not obvious; however, it has a simple interpretation. The prior over the landmark points  $\mathbf{w}_i$  is once more normally distributed, but now the covariance is divided into two parts: the term  $\boldsymbol{\Phi}\boldsymbol{\Phi}^T$ , which explains the variation in the subspace (due to shape changes), and the term  $\sigma^2\mathbf{I}$ , which explains any remaining variation in the data (mainly noise in the training points).

### 17.5.2 Learning

The PPCA model is very closely related to factor analysis (section 7.6). The only difference is that the noise term  $\sigma^2\mathbf{I}$  is spherical in the PPCA model, but has the diagonal form in factor analysis. Surprisingly, this difference has important implications; it is possible to learn the PPCA model in closed form whereas the factor analysis model needs an iterative strategy such as the EM algorithm.

In learning we are given a set of aligned training data  $\{\mathbf{w}_i\}_{i=1}^I$  where  $\mathbf{w}_i = [\mathbf{w}_{i1}^T, \mathbf{w}_{i2}^T, \dots, \mathbf{w}_{iN}^T]^T$  is a vector containing all of the  $x$  and  $y$  positions of the landmark points in the  $i^{th}$  example. We wish to estimate the parameters  $\boldsymbol{\mu}$ ,  $\boldsymbol{\Phi}$  and  $\sigma^2$  of the PPCA model.

To this end, we first set the mean parameter  $\boldsymbol{\mu}$  to be the mean of the training examples  $\mathbf{w}_i$ :

$$\boldsymbol{\mu} = \frac{\sum_{i=1}^I \mathbf{w}_i}{I}. \tag{17.26}$$

We then form a matrix  $\mathbf{W} = [\mathbf{w}_1 - \boldsymbol{\mu}, \mathbf{w}_2 - \boldsymbol{\mu}, \dots, \mathbf{w}_I - \boldsymbol{\mu}]$  containing the zero-centered data and compute the singular value decomposition of  $\mathbf{WW}^T$

$$\mathbf{WW}^T = \mathbf{UL}^2\mathbf{U}^T, \tag{17.27}$$

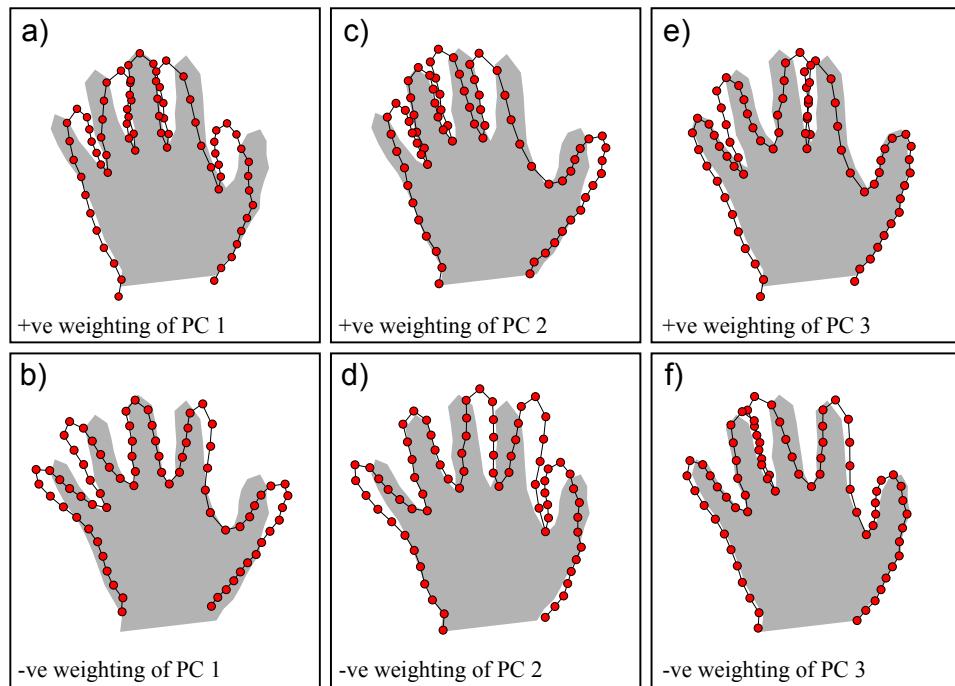
where  $\mathbf{U}$  is an orthogonal matrix and  $\mathbf{L}^2$  is diagonal. For a model that explains  $D$  dimensional data with  $K$  principal components, we compute the parameters using

$$\begin{aligned}
\hat{\sigma}^2 &= \frac{1}{D-K} \sum_{j=K+1}^D L_{jj}^2 \\
\hat{\boldsymbol{\Phi}} &= \mathbf{U}_K (\mathbf{L}_K^2 - \hat{\sigma}^2 \mathbf{I})^{1/2},
\end{aligned} \tag{17.28}$$

where  $\mathbf{U}_K$  denotes a truncation of  $\mathbf{U}$  where we have retained only the first  $K$  columns,  $\mathbf{L}_K^2$  represents a truncation of  $\mathbf{L}^2$  to retain only the first  $K$  columns and  $K$  rows, and  $L_{jj}$  is the  $j^{th}$  element from the diagonal of  $\mathbf{L}$ .

Algorithm 17.2

Problem 17.7



**Figure 17.11** Principal components for hand model. a-b) Varying the first principal component. In panel (a) we have added a multiple  $\lambda$  of the first principal component  $\phi_1$  to the mean vector  $\mu$ . In panel (b) we have subtracted the same multiple of the first principal component from the mean. In each case the shaded area indicates the mean vector. The first principal component has a clear interpretation: it controls the opening and closing of the fingers. Panels (c-d) and (e-f) show similar manipulations of the second and third principal components, respectively.

If the dimensionality  $D$  of the data is very high, then the eigenvalue decomposition of the  $D \times D$  matrix  $\mathbf{W}\mathbf{W}^T$  will be computationally expensive. If the number of training examples  $I$  is less than the dimensionality  $D$ , then a more efficient approach is to compute the singular value decomposition of the  $I \times I$  scatter matrix  $\mathbf{W}^T\mathbf{W}$

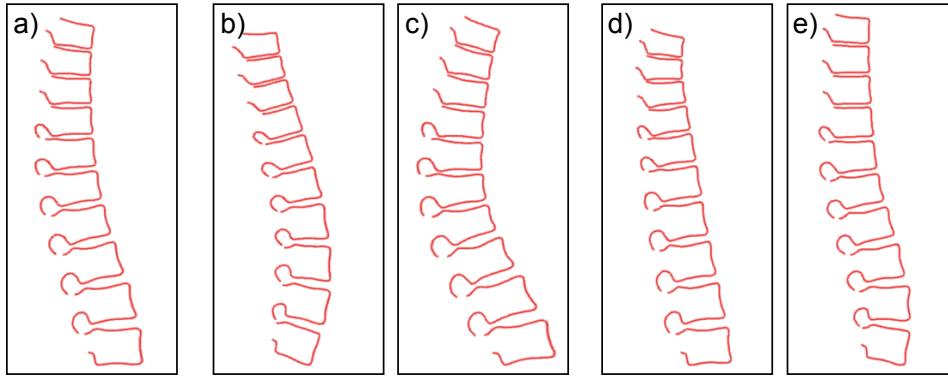
$$\mathbf{W}^T\mathbf{W} = \mathbf{V}\mathbf{L}^2\mathbf{V}^T, \quad (17.29)$$

and then re-arrange the SVD relation  $\mathbf{W} = \mathbf{U}\mathbf{L}\mathbf{V}^T$  to compute  $\mathbf{U}$ .

There are two things to notice about the estimated basis functions  $\Phi$ :

Problem 17.8

1. The basis functions (principal components) in the columns of  $\Phi$  are all orthogonal to one another. This can be easily seen because the solution for  $\Phi$  (equation 17.28) is the product of the truncated orthogonal matrix  $\mathbf{U}_K$  and the diagonal matrix  $(\mathbf{L}_K^2 - \hat{\sigma}^2\mathbf{I})^{1/2}$ .



**Figure 17.12** Learned spine model. a) Mean spine shape. b-c) Manipulating first principal component. d-e) Manipulating second principal component. Figure provided by Tim Cootes.

2. The basis functions are ordered: the first column of  $\Phi$  represents the direction in the space of  $\mathbf{w}$  that contained the most variance, and each subsequent direction explains less variation. This is a consequence of the SVD algorithm, which orders the elements of  $\mathbf{L}^2$  so that they decrease.

We could visualize the PPCA model by drawing samples from the marginal density (equation 17.25). However, the properties of the basis functions permit a more systematic way to examine the model. In figure 17.11 we visualize the PPCA model by manipulating the hidden variable  $\mathbf{h}_i$  and then illustrating the vector  $\boldsymbol{\mu} + \Phi\mathbf{h}_i$ . We can choose  $\mathbf{h}_i$  to elucidate each basis function  $\{\phi_k\}_{k=1}^K$  in turn. For example, by setting  $\mathbf{h}_i = \pm[1, 0, 0, 0, \dots, 0]$  we investigate the first basis function.

Figure 17.11 shows that the principal components  $\phi_k$  sometimes have surprisingly clear interpretations. For example, the first principal component clearly encodes the opening and closing of the fingers. A second example that visualizes the spine model from figure 17.1 is illustrated in figure 17.12.

### 17.5.3 Inference

In inference, we fit the shape to a new image by manipulating the weights  $\mathbf{h}$  of the basis functions  $\Phi$ . A suitable objective function is

$$\hat{\mathbf{h}} = \underset{\mathbf{h}}{\operatorname{argmax}} \left[ \underset{\Psi}{\max} \left[ \sum_{n=1}^N \left( -\frac{(\operatorname{dist}[\mathbf{x}_i, \operatorname{trans}[\boldsymbol{\mu}_n + \Phi_n \mathbf{h}, \Psi]])^2}{\sigma^2} \right) + \log[\operatorname{Norm}_{\mathbf{h}}[\mathbf{0}, \mathbf{I}]] \right] \right], \quad (17.30)$$

where  $\boldsymbol{\mu}_n$  contains the two elements of  $\boldsymbol{\mu}$  that pertain to the  $n^{th}$  point and  $\Phi_n$  contains the two rows of  $\Phi$  that pertain to the  $n^{th}$  point. There are a number of

ways to optimize this model, including a straightforward nonlinear optimization over the unknowns  $\mathbf{h}$  and  $\Psi$ . We will briefly describe an iterative closest point approach. This consists of iteratively repeating these steps:

- The current landmark points are computed as  $\mathbf{w} = \boldsymbol{\mu} + \Phi\mathbf{h}$ .
- Each landmark point is transformed into the image as  $\mathbf{w}'_n = \text{trans}[\mathbf{w}_n, \Psi]$ .
- Each transformed point  $\mathbf{w}'_n$  is associated with the *closest* edge point  $\mathbf{y}_n$  in the image.
- The transformation parameters  $\Psi$  that best map the original landmark points  $\{\mathbf{w}_n\}_{n=1}^N$  to the edge points  $\{\mathbf{y}_n\}_{n=1}^N$  are computed.
- Each point is transformed again using the updated parameters  $\Psi$ .
- The *closest* edge points  $\{\mathbf{y}_n\}_{n=1}^N$  are found once more.
- The hidden variables  $\mathbf{h}$  are updated (see below).

We repeat these steps until convergence. After optimization, the landmark points can be recovered as  $\mathbf{w} = \boldsymbol{\mu} + \Phi\mathbf{h}$ .

In the last step of the iterative algorithm we must update the hidden variables. This can be achieved using the objective function:

$$\begin{aligned}\hat{\mathbf{h}} &= \underset{\mathbf{h}}{\operatorname{argmax}} \left[ \sum_{n=1}^N \log[Pr(\mathbf{y}_n|\mathbf{h}), \Psi] + \log[Pr(\mathbf{h})] \right] \\ &= \underset{\mathbf{h}}{\operatorname{argmax}} \left[ \sum_{n=1}^N -(\mathbf{y}_n - \text{trans}[\boldsymbol{\mu}_n + \Phi_n \mathbf{h}, \Psi])^2 / \sigma^2 - \log[\mathbf{h}^T \mathbf{h}] \right],\end{aligned}\quad (17.31)$$

where  $\boldsymbol{\mu}_n$  contains the two elements of  $\boldsymbol{\mu}$  associated with the  $n^{th}$  landmark point and  $\Phi_n$  contains the two rows of  $\Phi$  associated with the  $n^{th}$  landmark point. If the transformation is linear so that  $\text{trans}[\mathbf{w}_n, \Psi]$  can be written in the form  $\mathbf{A}\mathbf{w}_n + \mathbf{b}$ , then this update can be computed in closed form and is given by

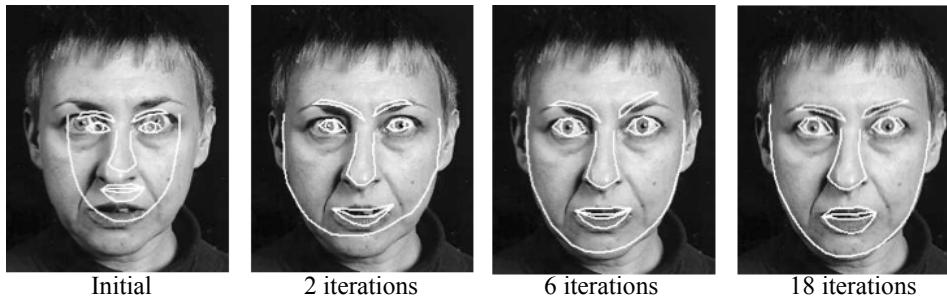
$$\hat{\mathbf{h}} = \left( \sigma^2 \mathbf{I} + \sum_{n=1}^N \Phi_n^T \mathbf{A}^T \mathbf{A} \Phi_n \right)^{-1} \sum_{n=1}^N \mathbf{A} \Phi_n (\mathbf{y}_n - \mathbf{A} \boldsymbol{\mu} - \mathbf{b}). \quad (17.32)$$

**Problem 17.9**

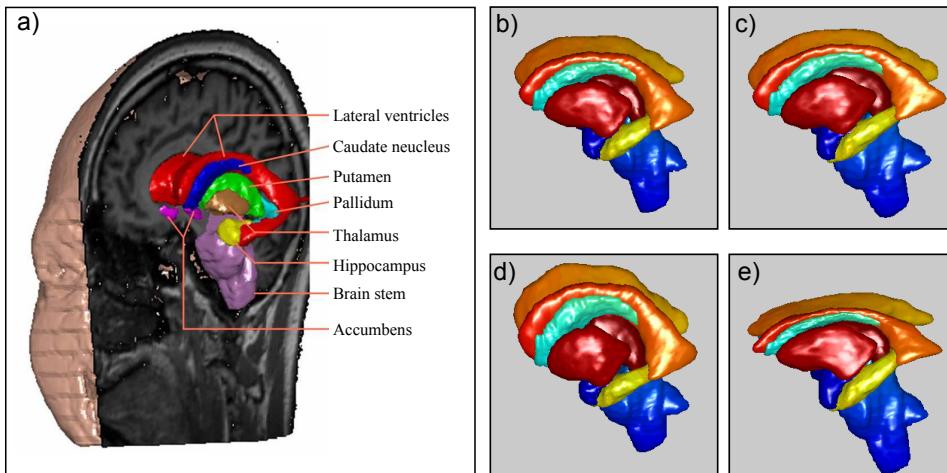
**Problem 17.10**  
**Problem 17.11**

Examples of performing inference in subspace shape models are illustrated in figures 17.1 and 17.13. As the optimization proceeds, the shape model moves across the surface of the image and adapts itself to the shape of the object. For this reason, these models are often referred to as *active shape models*.

We note that there are many variations of this model and many strategies that help fitting the model robustly. One particularly weak aspect of the model is that it assumes that each landmark point maps to a generic edge in the image; it does not even distinguish between the polarity or orientation of this edge, let alone take advantage of other nearby image information that might help identify the correct position. A more sensible approach is to build a generative model that describes the likelihood of the local image data when the  $n^{th}$  feature  $\mathbf{w}_n$  is present. A second important concept is *coarse-to-fine* fitting, in which a coarse model is fitted to a



**Figure 17.13** Several iterations of fitting a subspace shape model to a face image. After convergence, both the global transformation of the model and the details of the shape are correct. When a statistical shape model is fit to an image in this way, it is referred to as an *active shape model*. Figure provided by Tim Cootes.



**Figure 17.14** Three-dimensional statistical shape model. a) The model describes regions of the human brain. It is again defined by a set of landmark points. These are visualized by triangulating them to form a surface. b-c) Changing the weighting of the first principal component. d-e) Changing weighting of second component. Adapted from Babalola *et al.* (2008).

low resolution image. The result is then used as a starting point for a more detailed model at a higher resolution. In this way, it is possible to increase the probability of converging to the true fit without getting stuck in a local minimum.

## 17.6 Three-dimensional shape models

The subspace shape model can be easily extended to three dimensions. For 3D data, the model works in almost exactly the same way as it did in 2D. However, the landmark points  $\mathbf{w}$  become  $3 \times 1$  vectors determining the position within 3D space, and the global transformation that maps these into the image must also be generalized to 3D. For example, a 3D affine transformation encompasses 3D translations, rotations, shearing and scaling in 3D, and is determined by 12 parameters.

Finally, the likelihood must also be adapted for 3D. A trivial way to do this would be to create the 3D analogue of an edge operator by taking the root mean square of three derivative filters in the coordinate directions. We then construct an expression so that the likelihood is high in regions where the 3D edge operator returns a high value. An example of a 3D shape model is shown in figure 17.14.

## 17.7 Statistical models for shape and appearance

In chapter 7 we discussed the use of subspace models to describe the pixel intensities of a class of images such as faces. However, it was concluded that these were very poor models of this high-dimensional data. In this section, we consider models that describe both the intensity of the pixels and the shape of the object simultaneously. Moreover, they describe correlations between these aspects of the image: the shape tells us something about the intensity values and vice-versa (figure 17.15). When we fit these models to new images, they deform and adapt to the shape and intensity of the image, and so they are known as *active appearance models*.

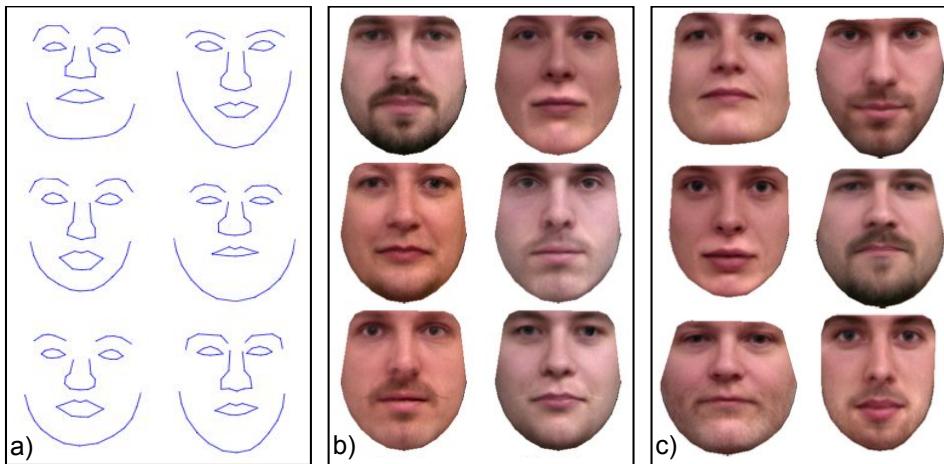
As before, we characterize the shape with a vector of  $N$  landmark points  $\mathbf{w} = [\mathbf{w}_1^T, \mathbf{w}_2^T, \dots, \mathbf{w}_N^T]$ . However, we now also describe a model of the pixel intensity values  $\mathbf{x}$  where this vector contains the concatenated RGB data from the image. The full model can best be described as a sequence of conditional probability statements:

$$\begin{aligned} Pr(\mathbf{h}_i) &= \text{Norm}_{\mathbf{h}_i}[\mathbf{0}, \mathbf{I}] \\ Pr(\mathbf{w}_i|\mathbf{h}_i) &= \text{Norm}_{\mathbf{w}_i}[\boldsymbol{\mu}_w + \boldsymbol{\Phi}_w \mathbf{h}_i, \sigma_w^2 \mathbf{I}] \\ Pr(\mathbf{x}_i|\mathbf{w}_i, \mathbf{h}_i) &= \text{Norm}_{\mathbf{x}_i}[\text{warp}[\boldsymbol{\mu}_x + \boldsymbol{\Phi}_x \mathbf{h}_i, \mathbf{w}_i, \boldsymbol{\Psi}_i], \sigma_x^2 \mathbf{I}]. \end{aligned} \quad (17.33)$$

This is quite a complex model, so we will break it down into its constituent parts. At the core of the model is a hidden variable  $\mathbf{h}_i$ . This can be thought of as a low-dimensional explanation that underpins both the shape and the pixel intensity values. In the first equation, we define a prior over this hidden variable.

In the second equation, the shape data  $\mathbf{w}$  is created by weighting a set of basis functions  $\boldsymbol{\Phi}_w$  by the hidden variable and adding a mean vector  $\boldsymbol{\mu}_w$ . The result is corrupted with spherically distributed normal noise with covariance  $\sigma_w^2 \mathbf{I}$ . This is exactly the same as for the statistical shape model.

The third equation describes how the observed pixel values  $\mathbf{x}_i$  in the  $i^{th}$  image depend on the shape  $\mathbf{w}_i$ , the global transformation parameters  $\boldsymbol{\Psi}_i$  and the hidden

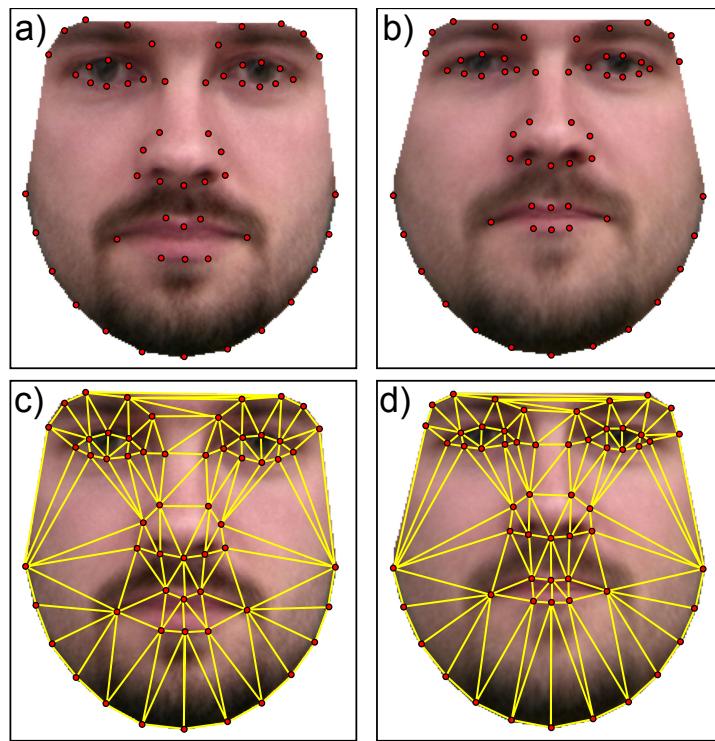


**Figure 17.15** Modeling both shape and texture. We learn a model in which we a) parameterize shape using a subspace model (results of manipulating weights of shape basis functions shown) and b) parameterize intensity values for fixed shape using a different subspace model (results of manipulating weights of texture basis functions shown). c) The subspace models are connected in that the weightings of the basis functions (principal components) in each model are always the same. In this way correlations between shape and texture are described (results of manipulating weights of basis functions for both shape and texture together shown). Adapted from Stegmann (2002)

variable  $\mathbf{h}_i$ . There are three stages in this process:

1. The intensity values are generated for the mean shape  $\boldsymbol{\mu}_w$ ; pixel values are described as a weighted sum  $\boldsymbol{\mu}_x + \Phi_x \mathbf{h}_i$  of a second set of basis functions  $\Phi_x$  which is added to a mean intensity  $\boldsymbol{\mu}_x$ .
2. These generated intensity values are then warped to the final desired shape; the operation  $\text{warp}[\boldsymbol{\mu}_x + \Phi_x \mathbf{h}_i, \mathbf{w}_i, \Psi_i]$  warps the resulting intensity image  $\boldsymbol{\mu}_x + \Phi_x \mathbf{h}_i$  to the desired shape based on the landmark points  $\mathbf{w}_i$  and the global transformation parameters  $\Psi_i$ .
3. Finally, the observed data  $\mathbf{x}_i$  is corrupted by normally distributed noise with spherical covariance  $\sigma_x^2 \mathbf{I}$ .

Notice that both the intensity values and the shape depend on the same underlying hidden variable  $\mathbf{h}_i$ . This means that the model can describe correlations in the shape and appearance; for example, the texture might change to include teeth when the mouth region expands. Figure 17.15 shows examples of manipulating the shape and texture components of a face model and illustrates the correlation between the two. Notice that the resulting images are much sharper than the factor analysis model for faces (figure 7.22); by explicitly accounting for the shape component, we get a superior model of the texture.



**Figure 17.16** Piecewise affine warping. a) The texture is synthesized for a fixed canonical shape. b) This shape is then warped to create the final image. c) One technique for warping the image is to use a piecewise affine transformation. We first triangulate the landmark points. d) Then each triangle undergoes a separate affine transformation such that the three points that define it move to their final positions. Adapted from Stegmann (2002).

### Warping images

A simple approach to warping the images is to triangulate the landmark points and then use a piecewise affine warp; each triangle is warped from the canonical position to the desired final position using a separate affine transformation (figure 17.16). The coordinates of the canonical triangle vertices are held in  $\mu$ . The coordinates of the triangle vertices after the warp are held in  $\text{trans}[\mu_w + \Phi_w h_i, \Psi_i]$ .

Problem 17.13

#### 17.7.1 Learning

In learning we are given a set of  $I$  images in which we know the transformed landmark points  $\{\mathbf{w}_i\}_{i=1}^I$  and the associated warped and transformed pixel data  $\{\mathbf{x}_i\}_{i=1}^I$  and we aim to learn the parameters  $\{\mu_w, \Phi_w, \sigma_w^2, \mu_x, \Phi_x, \sigma_x^2\}$ . The model is

too complex to learn directly; we take the approach of simplifying it by eliminating (i) the effect of the transformation on the landmark points, and (ii) both the warp and the transformation on the image data. Then we estimate the parameters in this simplified model.

To eliminate the effect of the transformations  $\{\Psi_i\}_{i=1}^I$  on the landmark points, we perform generalized Procrustes analysis. To eliminate the effect of the transformation and warps and the observed images, we now warp each training image to the average shape  $\mu_w = \sum_{i=1}^I \mathbf{w}_i / I$  using piecewise affine transformations.

The result of these operations is to generate training data consisting of aligned sets of landmark points  $\{\mathbf{w}_i\}_{i=1}^I$  representing the shape (similar to figure 17.15a) and a set of face images  $\{\mathbf{x}_i\}_{i=1}^I$  that all have the same shape (similar to figure 17.15b). These data are explained using the simpler model:

$$\begin{aligned} Pr(\mathbf{h}_i) &= \text{Norm}_{\mathbf{h}_i}[\mathbf{0}, \mathbf{I}] \\ Pr(\mathbf{w}_i|\mathbf{h}_i) &= \text{Norm}_{\mathbf{w}_i}[\boldsymbol{\mu}_w + \boldsymbol{\Phi}_w \mathbf{h}_i, \sigma_w^2 \mathbf{I}] \\ Pr(\mathbf{x}_i|\mathbf{h}_i) &= \text{Norm}_{\mathbf{x}_i}[\boldsymbol{\mu}_x + \boldsymbol{\Phi}_x \mathbf{h}_i, \sigma_x^2 \mathbf{I}]. \end{aligned} \quad (17.34)$$

To learn the parameters, we write the last two equations in the generative form

$$\begin{bmatrix} \mathbf{w}_i \\ \mathbf{x}_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mu}_w \\ \boldsymbol{\mu}_x \end{bmatrix} + \begin{bmatrix} \boldsymbol{\Phi}_w \\ \boldsymbol{\Phi}_x \end{bmatrix} \mathbf{h}_i + \begin{bmatrix} \boldsymbol{\epsilon}_{wi} \\ \boldsymbol{\epsilon}_{xi} \end{bmatrix}, \quad (17.35)$$

where  $\boldsymbol{\epsilon}_{wi}$  is a normally distributed noise term with spherical covariance  $\sigma_w^2 \mathbf{I}$  and  $\boldsymbol{\epsilon}_{xi}$  is a normally distributed noise term with spherical covariance  $\sigma_x^2 \mathbf{I}$ .

We now observe that the system is very similar to the standard form of a PPCA model or factor analyzer  $\mathbf{x}' = \boldsymbol{\mu}' + \boldsymbol{\Phi}' \mathbf{h} + \boldsymbol{\epsilon}'$ . Unlike PPCA, the noise term is structured and contains two values ( $\sigma_w^2$  and  $\sigma_x^2$ ). However, each dimension of the data does not have a separate variance as for factor analysis.

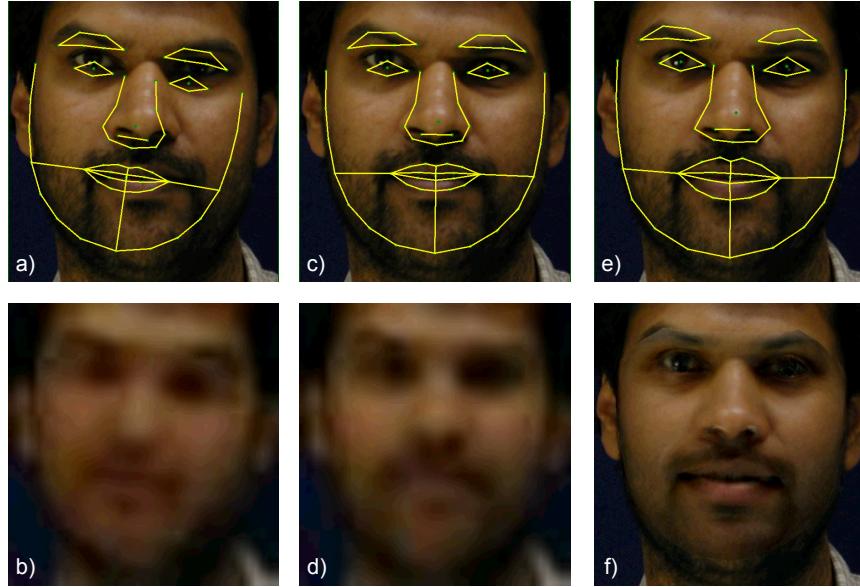
Unfortunately, there is no closed form solution as there was for the PPCA model. This model can be learned with a modified version of the EM algorithm for factor analysis (see section 7.6.2), where the update step for the variance terms  $\sigma^2$  and  $\sigma_x^2$  differs from the usual equation.

### 17.7.2 Inference

In inference we fit the model to new data by finding the values of the hidden variable  $\mathbf{h}$ , which are responsible for both the shape and appearance of the image. This low-dimensional representation of the object could then be used as an input to a second algorithm that analyzes its characteristics. For example, in a face model it might be used as the basis for discriminating gender.

Inference in this model can be simplified by assuming that the landmark points lie exactly on the subspace so that we have the deterministic relation  $\mathbf{w}_i = \boldsymbol{\mu} + \boldsymbol{\Phi}\mathbf{h}$ . This means that the likelihood of the observed data given the hidden variable  $\mathbf{h}$  can be expressed as:

$$Pr(\mathbf{x}|\mathbf{h}) = \text{Norm}_{\mathbf{x}}[\text{warp}[\boldsymbol{\mu}_x + \boldsymbol{\Phi}_x \mathbf{h}, \boldsymbol{\mu}_w + \boldsymbol{\Phi}_w \mathbf{h}, \boldsymbol{\Psi}], \sigma_x^2 \mathbf{I}]. \quad (17.36)$$



**Figure 17.17** Fitting a statistical model of shape and appearance. a) Shape model at start of fitting process superimposed on the observed image. b) Shape and appearance model (synthesized image  $\mathbf{x}$ ) at start of fitting process. c-d) After several iterations. e-f) At the end of the fitting procedure. The synthesized face (f) looks very similar to the observed image in (a),(c), and (e). Such models are known as *active appearance models* as they can be seen to adapt to the image. Figure provided by Tim Cootes.

We adopt a maximum likelihood procedure and note that this criterion is based on the normal distribution so the result is a least squares cost function:

$$\begin{aligned} \hat{\mathbf{h}}, \hat{\Psi} &= \underset{\mathbf{h}, \Psi}{\operatorname{argmax}} [\log [Pr(\mathbf{x}|\mathbf{h})]] \\ &= \underset{\mathbf{h}, \Psi}{\operatorname{argmin}} [(\mathbf{x} - \mathbf{warp}[\boldsymbol{\mu}_x + \boldsymbol{\Phi}_x \mathbf{h}, \boldsymbol{\mu}_w + \boldsymbol{\Phi}_w \mathbf{h}, \boldsymbol{\Psi}])^T \\ &\quad (\mathbf{x} - \mathbf{warp}[\boldsymbol{\mu}_x + \boldsymbol{\Phi}_x \mathbf{h}, \boldsymbol{\mu}_w + \boldsymbol{\Phi}_w \mathbf{h}, \boldsymbol{\Psi}])] . \end{aligned} \quad (17.37)$$

Denoting the unknown quantities by  $\boldsymbol{\theta} = \{\mathbf{h}, \boldsymbol{\Psi}\}$ , we observe that this cost function takes the general form of  $f[\boldsymbol{\theta}] = \mathbf{z}[\boldsymbol{\theta}]^T \mathbf{z}[\boldsymbol{\theta}]$  and can hence be optimized using the Gauss-Newton method (appendix B.2.3). We initialize the unknown quantities to some sensible values and then iteratively update these values using the relation

$$\boldsymbol{\theta}^{[t]} = \boldsymbol{\theta}^{[t-1]} + \lambda(\mathbf{J}^T \mathbf{J})^{-1} \frac{\partial f}{\partial \boldsymbol{\theta}}, \quad (17.38)$$

where  $\mathbf{J}$  is the Jacobian matrix. The entry in the  $m^{th}$  row and  $n^{th}$  column of  $\mathbf{J}$  consists of the derivative of the  $m^{th}$  element of  $\mathbf{z}$  with respect to the  $n^{th}$  element

of the parameter vector  $\theta$ :

$$J_{mn} = \frac{\partial z_m}{\partial \theta_n}. \quad (17.39)$$

Figure 17.17 shows an example of a statistical shape and appearance model being fit to face data. The spine model in figure 17.1 was also a model of this kind although only the shape component was shown. As usual, the success of this fitting procedure relies on having a good starting point for the optimization process and a coarse-to-fine strategy can help the optimization converge.

## 17.8 Non-Gaussian statistical shape models

The statistical shape model discussed in section 17.4 is effective for objects where the shape variation is relatively constrained and is well described by the normally distributed prior. However, in some situations a normal distribution will not suffice, and we must turn to more complex models. One possibility is to use a mixture of PPCAs, and this is straightforward to implement. However, we will use this opportunity to introduce an alternative model for describing non-Gaussian densities.

Problem 17.12

The *Gaussian process latent variable model* or *GPLVM* is a density model that can model complex non-normal distributions. The GPLVM extends the PPCA model so that the hidden variables  $\mathbf{h}_i$  are transformed through a fixed nonlinearity before being weighted by the basis functions  $\Phi$ .

### 17.8.1 PPCA as regression

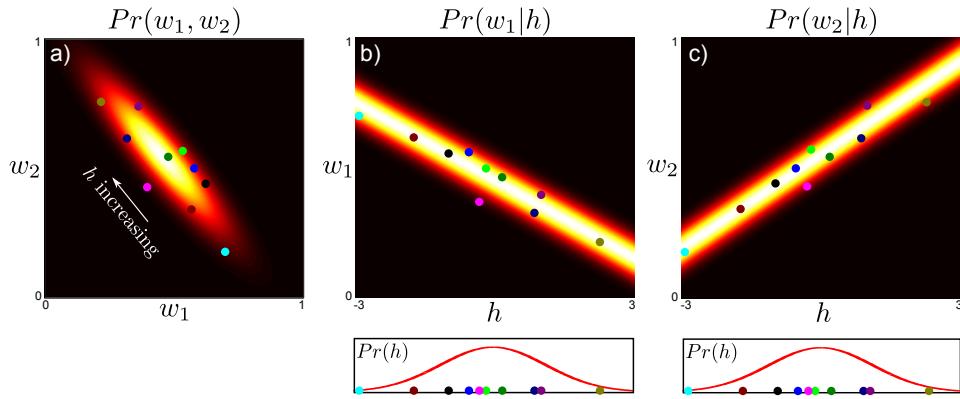
To help understand the GPLVM, we will reconsider subspace models in terms of regression. Consider the PPCA model, which can be expressed as

$$\begin{aligned} Pr(\mathbf{w}|\boldsymbol{\mu}, \Phi, \sigma^2) &= \int Pr(\mathbf{w}, \mathbf{h}|\boldsymbol{\mu}, \Phi, \sigma^2)d\mathbf{h} \\ &= \int Pr(\mathbf{w}|\mathbf{h}, \boldsymbol{\mu}, \Phi, \sigma^2)Pr(\mathbf{h})d\mathbf{h} \\ &= \int \text{Norm}_{\mathbf{w}}[\boldsymbol{\mu} + \Phi\mathbf{h}, \sigma^2\mathbf{I}]\text{Norm}_{\mathbf{h}}[\mathbf{0}, \mathbf{I}]d\mathbf{h}. \end{aligned} \quad (17.40)$$

The first term in the last line of this expression has a close relationship to linear regression (section 8.1); it is a model for predicting  $\mathbf{w}$  given the variable  $\mathbf{h}$ . Indeed if we consider just the  $d^{th}$  element  $w_d$  of  $\mathbf{w}$  then this term has the form

$$Pr(w_d|\mathbf{h}, \boldsymbol{\mu}, \Phi, \sigma^2) = \text{Norm}_{w_d}[\mu_d + \phi_{d\bullet}^T \mathbf{h}, \sigma^2], \quad (17.41)$$

where  $\mu_d$  is the  $d^{th}$  dimension of  $\boldsymbol{\mu}$  and  $\phi_{d\bullet}^T$  is the  $d^{th}$  row of  $\Phi$ ; this is exactly the linear regression model for  $w_d$  against  $\mathbf{h}$ .



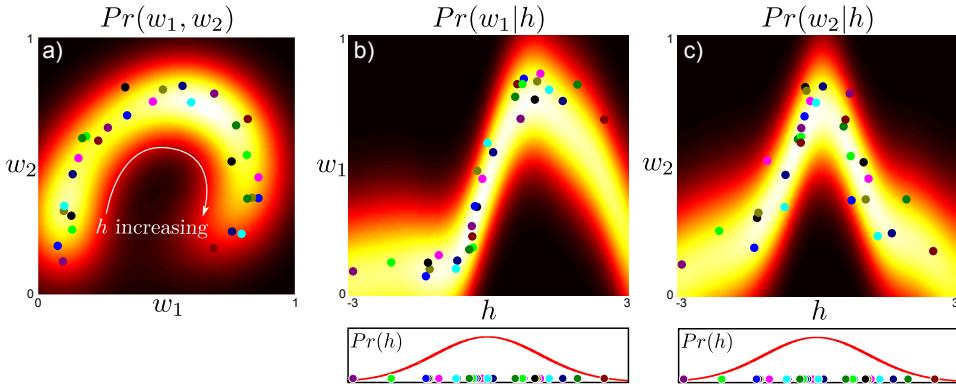
**Figure 17.18** PPCA model as regression. a) We consider a 2D data set which is explained by a PPCA model with a single hidden variable. The data are explained by a 2D normal distribution with mean  $\mu$  and covariance  $\phi\phi^T + \sigma^2\mathbf{I}$ . b) One way to think of this PPCA model is that the 2D data are explained by two underlying regression models. The first data dimension  $w_1$  is formed from a regression against  $h$  and c) the second data dimension  $w_2$  is formed from a different regression against the same values  $h$ .

This insight provides a new way of looking at the model. Figure 17.18 shows a 2D data set  $\{w_i\}_{i=1}^I$ , which is explained by a set of 1D hidden variables  $\{h_i\}_{i=1}^I$ . Each dimension of the 2D data  $\mathbf{w}$  is created by a different regression model, but in each case we are regressing against the common set of hidden variables  $\{h_i\}_{i=1}^I$ . So, the first dimension  $w_1$  of  $\mathbf{w}$  is described as  $\mu_1 + \phi_{1\bullet}^T h$  and the second dimension  $w_2$  is modeled as  $\mu_2 + \phi_{2\bullet}^T h$ . The common underlying hidden variable induces the correlation we see in the distribution  $Pr(\mathbf{w})$ .

Now let us consider how the overall density is formed. For a fixed value of  $h$  we get a prediction for  $w_1$  and a prediction for  $w_2$  each of which has additive normal noise with the same variance (figures 17.18b-c). The result is a 2D spherical normal distribution in  $\mathbf{w}$ . To create the density, we integrate over all possible values of  $h$ , weighted by a the normal prior; the final density is hence an infinite weighted sum of the 2D spherical normal distributions predicted by each value of  $h$ , and this happens to have the form of the normal distribution with non-spherical covariance  $\phi\phi^T + \sigma^2\mathbf{I}$  which is seen in figure 17.18a.

### 17.8.2 Gaussian process latent variable model

This interpretation of PPCA provides an obvious approach to describing more complex densities; we simply replace the linear regression model with a more sophisticated nonlinear regression model. As the name suggests, the Gaussian process latent variable model or GPLVM makes use of the Gaussian process regression model (see section 8.5).



**Figure 17.19** Gaussian process latent variable model (GPLVM) as regression.  
a) We consider a 2D dataset that is explained by a GPLVM model with a single variable. b) One way to consider this model is that the 2D data are explained by two underlying regression models. The first data dimension  $w_1$  is formed from a Gaussian process regression against the hidden variable  $h$ . c) The second data dimension  $w_2$  is formed from a different Gaussian process regression model against the same values  $h$ .

Figure 17.19 illustrates the GPLVM. Each dimension of the density in figure 17.19a once again results from a regression against a common underlying variable  $h$ . However, in this model, the two regression curves are nonlinear (figures 17.19b and c) and this accounts for the complexity of the original density.

There are two major practical changes in the GPLVM:

1. In Gaussian process regression, we marginalize over the regression parameters  $\mu$  and  $\Phi$  so we do not have to estimate these in the learning procedure.
2. Conversely, it is no longer possible to marginalize over the hidden variables  $\mathbf{h}$  in closed form; we must estimate the hidden variables during the training procedure. The inability to marginalize over the hidden variables also creates some difficulties in evaluating the final density.

We will now consider learning and inference in this model in turn.

### Learning

In the original Gaussian process regression model (section 8.5), we aimed to predict the univariate world states  $\mathbf{w} = [w_1, w_2, \dots, w_I]^T$  from multivariate data  $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_I]$ . The parameter vector  $\phi$  was marginalized out of the model, and the noise parameter  $\sigma^2$  was found by maximizing the marginal likelihood:

$$\begin{aligned}
\hat{\sigma}^2 &= \operatorname{argmax}_{\sigma^2} [Pr(\mathbf{w}|\mathbf{X}, \sigma^2)] \\
&= \operatorname{argmax}_{\sigma^2} \left[ \int Pr(\mathbf{w}|\mathbf{X}, \Phi, \sigma^2) Pr(\Phi) d\Phi \right] \\
&= \operatorname{argmax}_{\sigma^2} [\text{Norm}_{\mathbf{w}}[\mathbf{0}, \sigma_p^2 \mathbf{K}[\mathbf{X}, \mathbf{X}] + \sigma^2 \mathbf{I}]],
\end{aligned} \tag{17.42}$$

where  $\sigma_p^2$  controls the prior variance of the parameter vector  $\phi$  and  $\mathbf{K}[\bullet, \bullet]$  is the chosen kernel function.

In the GPLVM we have a similar situation. We aim to predict multivariate world values  $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_I]^T$  from multivariate hidden variables  $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_I]$ . Once more, we marginalize the basis functions  $\Phi$  out of the model and maximize over the noise parameters  $\sigma^2$ . However, this time, we do not know the values of the hidden variables  $\mathbf{H}$  that we are regressing against; these must be simultaneously estimated, giving the objective function:

$$\begin{aligned}
\hat{\mathbf{H}}, \hat{\sigma}^2 &= \operatorname{argmax}_{\mathbf{H}, \sigma^2} [Pr(\mathbf{W}, \mathbf{H}, \sigma^2)] \\
&= \operatorname{argmax}_{\mathbf{H}, \sigma^2} \left[ \int Pr(\mathbf{W}|\mathbf{X}, \Phi, \sigma^2) Pr(\Phi) Pr(\mathbf{H}) d\Phi \right] \\
&= \operatorname{argmax}_{\mathbf{H}, \sigma^2} \left[ \prod_{d=1}^D \text{Norm}_{\mathbf{w}_{d\bullet}}[\mathbf{0}, \sigma_p^2 \mathbf{K}[\mathbf{H}, \mathbf{H}] + \sigma^2 \mathbf{I}] \prod_{i=1}^I \text{Norm}_{\mathbf{h}_i}[\mathbf{0}, \mathbf{I}] \right],
\end{aligned} \tag{17.43}$$

where there is one term in the first product for each of the  $D$  dimensions of the world and one term in the second product for each of the training examples.

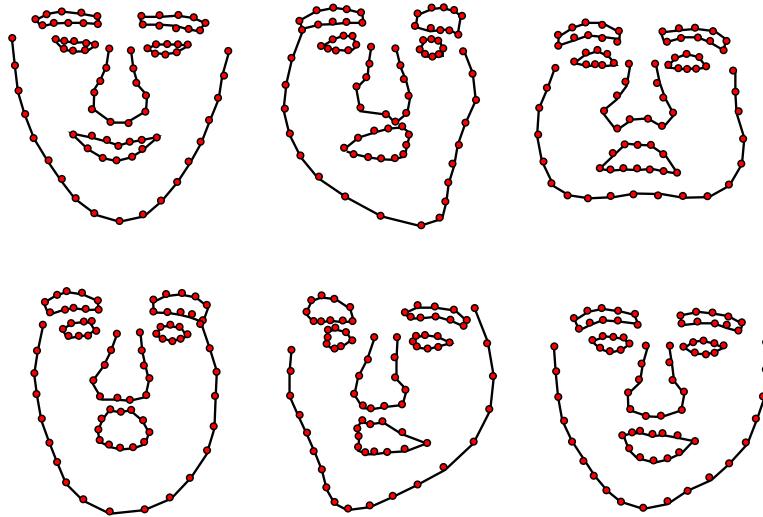
Unfortunately, there is no closed form solution to this optimization problem; to learn the model we must use one of the general-purpose nonlinear optimization techniques discussed in appendix B. Sometimes, the kernel  $\mathbf{K}[\bullet, \bullet]$  also contains parameters, and these should be simultaneously optimized.

### Inference

For a new value of the hidden variable  $\mathbf{h}^*$  the distribution over the  $d^{th}$  dimension of the world  $w_d^*$  is given by the analogue of equation 8.24:

$$\begin{aligned}
Pr(w_d^* | \mathbf{h}^*, \mathbf{H}, \mathbf{W}) &= \\
\text{Norm}_{w_d^*} \left[ \frac{\sigma_p^2}{\sigma^2} \mathbf{K}[\mathbf{h}^*, \mathbf{H}] \mathbf{w}_{d\bullet} - \frac{\sigma_p^2}{\sigma^2} \mathbf{K}[\mathbf{h}^*, \mathbf{H}] \left( \mathbf{K}[\mathbf{H}, \mathbf{H}] + \frac{\sigma_p^2}{\sigma^2} \mathbf{I} \right)^{-1} \mathbf{K}[\mathbf{H}, \mathbf{H}] \mathbf{w}_{d\bullet}, \right. \\
&\quad \left. \sigma_p^2 \mathbf{K}[\mathbf{h}^*, \mathbf{h}^*] - \sigma_p^2 \mathbf{K}[\mathbf{h}^*, \mathbf{H}] \left( \mathbf{K}[\mathbf{H}, \mathbf{H}] + \frac{\sigma_p^2}{\sigma^2} \mathbf{I} \right)^{-1} \mathbf{K}[\mathbf{H}, \mathbf{h}^*] + \sigma^2 \right].
\end{aligned} \tag{17.44}$$

To sample from the model, we select the hidden variable  $\mathbf{h}^*$  from the prior and then predict a probability distribution over the landmarks  $\mathbf{w}^*$  using this equation.



**Figure 17.20** Samples from a non-Gaussian face model based on a GPLVM. The samples represent more significant distortions to the shape than could be realistically described by the original statistical shape model. Adapted from Huang *et al.* (2011). ©2011 IEEE.

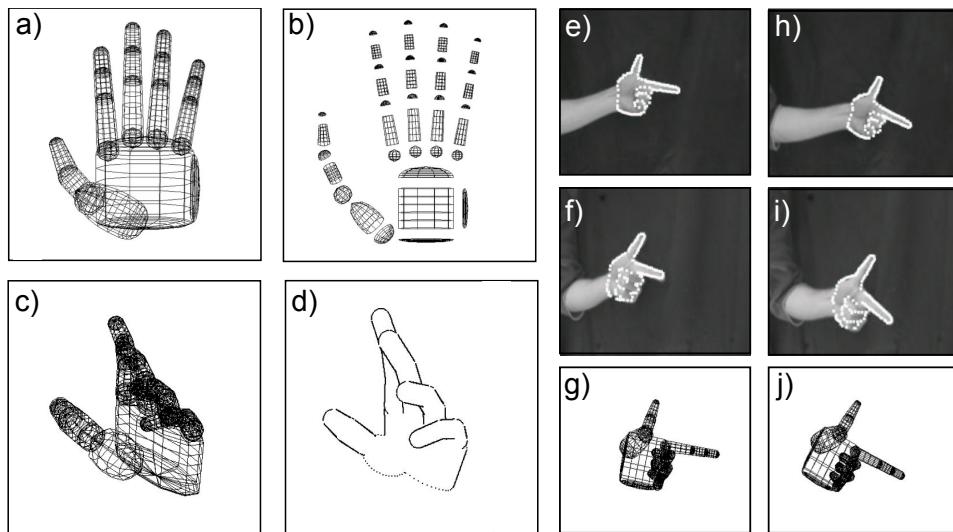
To assess the probability of the a new sample  $\mathbf{w}^*$ , we should use the relation

$$Pr(\mathbf{w}) = \prod_{d=1}^D \int Pr(w_d^* | \mathbf{h}^*, \mathbf{H}, \mathbf{W}) Pr(\mathbf{h}^*) d\mathbf{h}^*. \quad (17.45)$$

Unfortunately, this integral cannot be computed in closed form. One possibility is to maximize over  $\mathbf{h}^*$  rather than marginalize over it. Another possibility is to approximate the density  $Pr(\mathbf{h}^*)$  by a set of delta functions at the positions of the training data  $\{\mathbf{h}_i\}_{i=1}^I$  and we can then replace the integral with a sum over the individual predictions from these examples.

### Application to shape models

Figure 17.20 illustrates several examples of sampling from a shape model for a face that is based on the GPLVM. This more sophisticated model can cope with modeling larger variations in the shape than the original PPCA, which was based on a single normal distribution.



**Figure 17.21** Articulated model for a human hand. a) A three dimensional model for a human hand is constructed from 39 truncated quadrics. b) Exploded view. c) The model has 27 degrees of freedom that control the joint angles. d) It is easy to project this model into an image and find the outer and occluding contours, which can then be aligned with contours in an image. e-f) Two views of a hand taken simultaneously from different cameras g) Estimated state of hand model. h-j) Two more views and another estimate of the position. Adapted from Stenger *et al.* (2001a). ©2001 IEEE.

## 17.9 Articulated models

Statistical shape models work well when the shape variation is relatively small. However, there are other situations where we have much stronger a priori knowledge about the object. For example, in a body model, we know that there are two arms and two legs and that these are connected in a certain way to the main body. An *articulated model* parameterizes the model in terms of the joint angles and the overall transformation relating one root component to the camera.

The core idea of an articulated model is that the transformations of the parts are cumulative; the position of the foot depends on the position of the lower leg, which depends on the position of the upper leg and so on. This is known as a *kinematic chain*. To compute the global transformation of the foot relative to the camera, we chain the transformations relating each of the body parts in the appropriate order.

There are many approaches to constructing articulated models. They may be two dimensional (e.g., the pictorial structures discussed in chapter 11) or exist in three dimensions. We will consider a 3D hand model that is constructed from *truncated quadrics*. The quadric is the 3D generalization of the conic (see section 17.1) and can represent cylinders, spheres, ellipsoids, a pair of planes, and other shapes in 3D. Points in 3D which lie on the surface of the quadric satisfy the relation

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} \psi_1 & \psi_2 & \psi_3 & \psi_4 \\ \psi_2 & \psi_5 & \psi_6 & \psi_7 \\ \psi_3 & \psi_6 & \psi_8 & \psi_9 \\ \psi_4 & \psi_7 & \psi_9 & \psi_{10} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = 0. \quad (17.46)$$

Figure 17.21 illustrates a hand model that was constructed from a set of 39 *quadrics*. Some of these quadrics are truncated; to make a tube shape of finite length, a cylinder, or ellipsoid is clipped by a pair of planes in 3D so that only parts of the quadric that are between the planes are retained. The pair of planes is represented by a second quadric, so each part of the model is actually represented by two quadrics. This model has 27 degrees of freedom, 6 for the global hand position, 4 for the pose of each finger, and 5 for the pose of the thumb.

The quadric is a sensible choice because its projection through a pinhole camera takes the form of a conic and can be computed in closed form. Typically, an ellipsoid (represented by a quadric) would project down to an ellipse in the image (represented by a conic), and we can find a closed form expression for the parameters of this conic in terms of the quadric parameters.

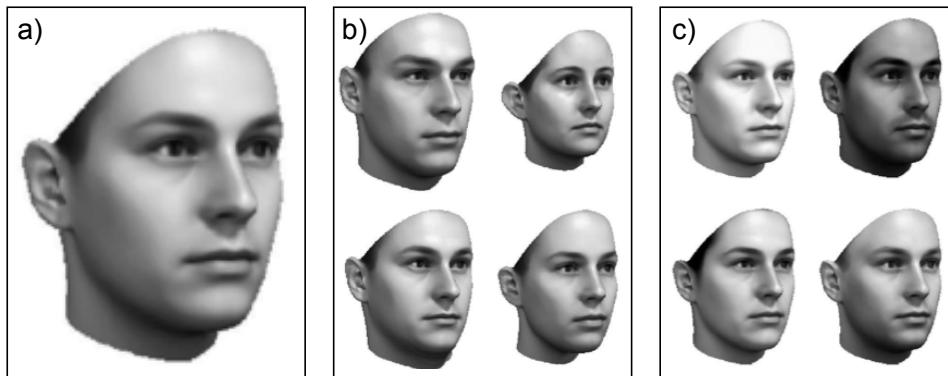
Problem 17.14

Given the camera position relative to the model, it is possible to project the collection of quadrics that form the 3D model into the camera image. Self occlusion can be handled neatly by testing the depth along each ray and not rendering the resulting conic if the associated quadric lies behind another part of the model. This leads to a straightforward method for fitting the model to an image of the object (i.e., finding the joint angles and overall pose relative to the camera). We simulate a set of contours for the model (as in figure 17.21d), and then evaluate an expression for the likelihood that increases when these match the observed edges in the image. To fit the model, we simply optimize this cost function.

Unfortunately, this algorithm is prone to converging to local minima; it is hard to find a good starting point for the optimization. Moreover, the visual data may be genuinely ambiguous in a particular image, and there may be more than one configuration of the object that is compatible with the observed image. The situation becomes more manageable if we view the object from more than one camera (as in figure 17.21e-f and h-i) as much of the ambiguity is resolved. Fitting the model is also easier when we are tracking the model through a series of frames; we can initialize the model fitting at each time based on the known position of the hand at the previous time. This kind of temporal model is investigated in chapter 19.

## 17.10 Applications

We will now look at two applications that extend the ideas of this chapter into 3D. First, we will consider a face model that is essentially a 3D version of the active appearance model. Second, we will discuss a model for the human body that combines the ideas of the articulated model and the subspace representation of shape.



**Figure 17.22** 3D morphable model of a face. The model was trained from laser scans of 200 individuals and is represented as a set of 70,000 vertex positions and an associated texture map. a) Mean face. b) As for the 2D subspace shape model, the final shape is described as a linear combination of basis shapes (principal components). In this model, however, these basis shapes are three dimensional. The figure shows the effect of changing the weighting of these basis functions while keeping the texture constant. c) The texture is also modeled as a linear combination of basis shapes. The figure shows the effect of changing the texture while keeping the shape constant. Adapted from Blanz & Vetter (2003). ©2003 IEEE.

### 17.10.1 3D morphable models

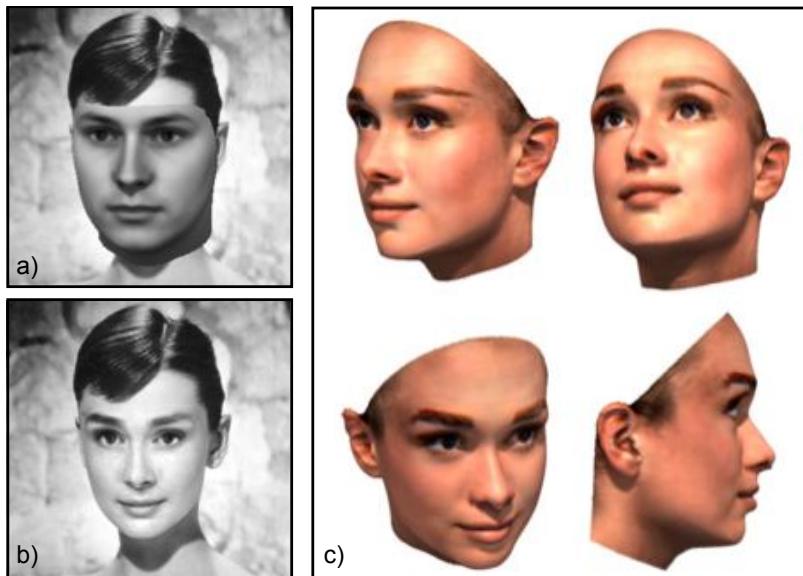
Blanz & Vetter (1999) developed a statistical model of the 3D shape and appearance of faces. Their model was based on 200 laser scans. Each face was represented by approximately 70,000 3D vertices and an RGB texture map. The captured faces were preprocessed so that the global 3D transformation between them was removed, and the vertices were registered using a method based on optical flow.

A statistical shape model was built where the 3D vertices now take on the role of the landmark points. As with most of the statistical shape models in this chapter, this model was based on a linear combination of basis functions (principal components). Similarly, the texture maps were described as a linear combination of a set of basis images (principal components). Figure 17.22 illustrates the mean face and the effect of varying the shape and texture components independently.

The model, as described so far, is thus a 3D version of the model for shape and appearance that we described in section 17.7. However, in addition, Blanz & Vetter (1999) model the rendering process using the *Phong shading model* which includes both ambient and directional lighting effects.

To fit this model to a photograph of a face, the square error between the observed pixel intensities and those predicted from the model is minimized. The goal then is to manipulate the parameters of the model so that the rendered image matches the observed image as closely as possible. These parameters include:

- the weightings of the basis functions that determine the shape,

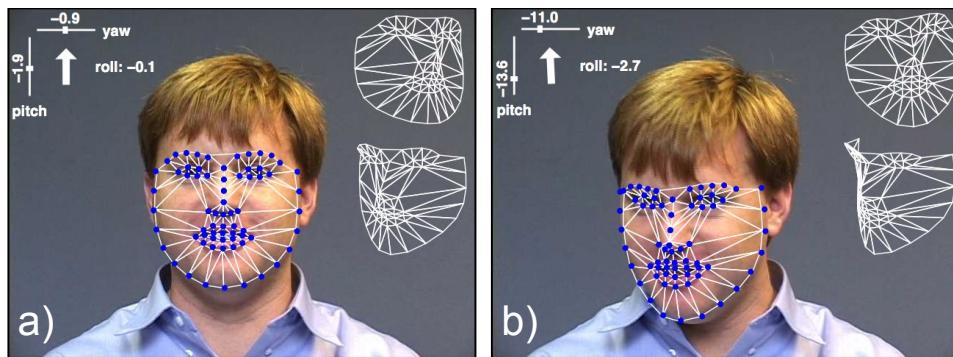


**Figure 17.23** Fitting a 3D morphable model to a real image of Audrey Hepburn. The goal is to find the parameters of the 3D model that best describe the 2D face. Once we have done this, we can manipulate the image. For example we could re-light the face or view it from different angles. a) Simulated image from model with initial parameters. b) Simulated image from model after fitting procedure. This closely replicates the original image in terms of both texture and shape. c) Images from fitted model generated under several different viewing conditions. Adapted from Blanz & Vetter (1999). ©1999 ACM.

- the weightings of the basis functions that determine the texture,
- the relative position of the camera and the object,
- the RGB intensities of the ambient and directed light, and
- the offsets and gains for each image RGB channel.

Other parameters such as the camera distance, light direction, and surface shininess were fixed by hand. In practice, the fitting was accomplished using a nonlinear optimization technique. Figure 17.23 illustrates the process of fitting the model to a real image. After convergence, the shape and texture closely replicate the original image. At the end of this process, we have full knowledge of the shape and texture of the face. This can now be viewed from different angles, re-lit and even have realistic shadows superimposed.

Blanz and Vetter (Blanz & Vetter 2003; Blanz *et al.* 2005) applied this model to face recognition. In the simplest case, they described the fitted face using a vector containing the weighting functions for both the shape and texture. Two faces can be compared by examining the distance between the vector associated with each.



**Figure 17.24** Real-time facial tracking using a 3D active appearance model. a-b) Two examples from tracking sequence. The pose of the face is indicated in the top left-hand corner. The superimposed mesh shown to the right of the face illustrates two different views of the shape component of the model. Adapted from Matthews *et al.* (2007). ©2007 Springer.

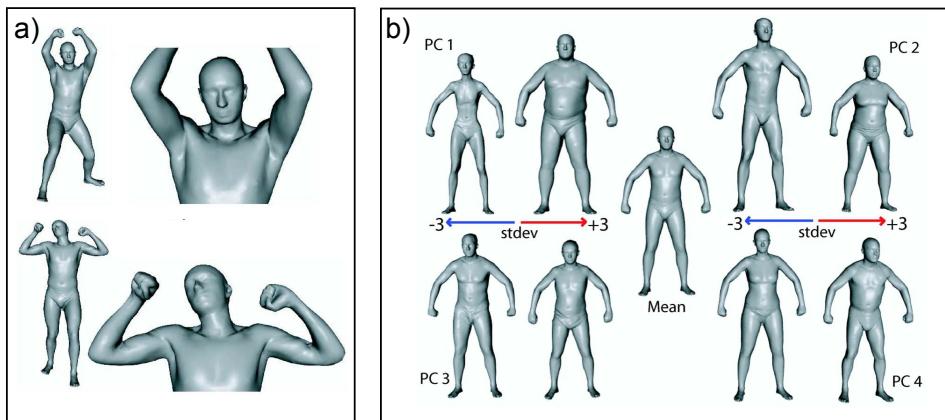
This method has the advantage that the faces can be originally presented in very different lighting conditions and with very different poses as neither of these factors is reflected in the final representation. However, the method is limited in practice by the model-fitting procedure, which does not always converge for real images that may suffer from complex lighting conditions and partial occlusion.

Matthews *et al.* (2007) presented a simplified version of the same model; this was still 3D but had a sparser mesh and did not include a reflectance model. However, they describe an algorithm for fitting this face model to video sequences that can run at more than 60 frames per second. This permits real-time tracking of the pose and expression of faces (figure 17.24). This technique has been used to capture facial expressions for CGI characters in movies and video games.

### 17.10.2 3D Body Model

Anguelov *et al.* (2005) presented a 3D body model that combines an articulated structure with a subspace model. The articulated model describes the skeleton of the body, and the subspace model describes the variation of the shape of the individual around this skeleton (figure 17.25).

At its core, this model is represented by a set of triangles that define the surface of the body. The model is best explained in terms of generation in which each triangle undergoes a series of transformations. First, the position is deformed in a way that depends on the configuration of the nearest joints in the body. The deformation is determined using a regression model and creates subtle effects such as the deformation of muscles. Second, the position of the triangle is deformed according to a PCA model which determines the individual's characteristics (body shape, etc.). Finally, the triangle is warped in 3D space depending on the position

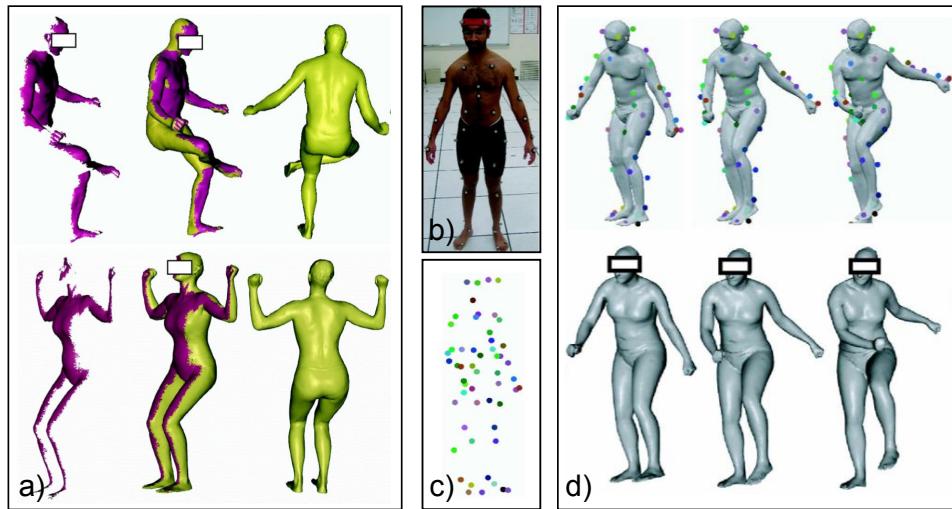


**Figure 17.25** 3D body model. a) The final position of the skin surface is regressed against the nearest joint angles; this produces subtle effects such as the bulging of muscles. b) The final position of the skin surface also depends on a PCA model which describes between-individual variation in body shape. Adapted from Anguelov *et al.* (2005). ©2005 ACM.

of the skeleton.

Two applications of this model are shown in figure 17.26. First, the model can be used to fill in missing parts of partial scans of human beings. Many scanners cannot capture a full 3D model at once; they may only capture the front of the object and so several scans must be combined to get a full model. This is problematic for moving objects such as human beings. Even for models that can capture 360° of shape, there are often missing or noisy parts of the data. Figure 17.26a shows an example of fitting the model to a partial scan; the position of the skeleton and the weighting of the PCA components are adapted until the synthesized shape agrees with the partial scan. The remaining part of the synthesized shape plausibly fills in the missing elements.

Figure 17.26b-d illustrates the use of the model for motion capture based animation. An actor in a motion capture studio has his body position tracked, and this body position can be used to determine the position of the skeleton of the 3D model. The regression model then adjusts the vertices of the skin model appropriately to model muscle deformation while the PCA model allows the identity of the resulting model to vary. This type of system can be used to generate character animations for video games and movies.



**Figure 17.26** Applications of 3D body model. a) Interpolation of partial scans. In each case, the purple region represents the original scan and the green region is the interpolated part after fitting a 3D body model (viewed from two angles). b) Motion capture animation. An actor is tracked in a conventional motion capture studio. c) This results in the known position of a number of markers on the body. d) These markers are used to position the skeletal aspect of the model. The PCA part of the model can control the shape of the final body (two examples shown). Adapted from Anguelov *et al.* (2005). ©2005 ACM.

## Discussion

In this chapter, we have presented a number of models for describing the shape of visual objects. These ideas relate closely to the subsequent chapters in this book. These shape models are often tracked in video sequences and the machinery for this tracking is developed in chapter 19. Several of the shape models have a subspace (principal component) representation at their core. In chapter 18 we investigate models that exploit this type of representation for identity recognition.

## Notes

**Snakes and active contour models:** Snakes were first introduced by Kass *et al.* (1987). Various modifications have been made to encourage them to converge to a sensible answer including the addition of a ballooning term (Cohen 1991) and a new type of external force field called gradient vector flow (Xu & Prince 1998). The original description treated the contour as a continuous object, but subsequent work also considered it as discrete and used greedy algorithms or dynamic programming methods for optimization (Amini *et al.* 1990; Williams & Shah 1992). Subsequent work has investigated the use of prior information about object shape and led to active contour models (Cootes *et al.* 1995). This is still an open research area (e.g., Bergtholdt *et al.* 2005; Freifeld *et al.* 2010).

The contour models discussed in this chapter are known as parametric because the shape is explicitly represented. A summary of early work on parametric active contours can be found in Blake & Isard (1998). A parallel strand of research investigates implicit or non-parametric contours in which the contour is implicitly defined by the level sets of a function defined on the image domain (Malladi *et al.* 1994; Caselles *et al.* 1997). There has also been considerable interest in applying prior knowledge to these models (Leventon *et al.* 2000; Rousson & Paragios 2002).

**Bottom-up models:** This chapter has concerned top-down approaches to contour detection in which a generative model for the object is specified that explains the observed edges in the image. However, there has also been a recent surge of research progress into bottom-up approaches, in which edge fragments are combined to form coherent shapes. For example, Opelt *et al.* (2006) introduced the “Boundary fragment model” in which pairs of edge fragments vote for the position of the centroid of the object and the object is detected by finding the position with the most support. Shotton *et al.* (2008a) present a similar model that incorporates scale invariance and which searches through local regions of the image to identify objects that form only a small part of a larger scene. Pairwise constraints between features were introduced by Leordeanu *et al.* (2007). Other work has investigated reconstructing pieces of the contour as combinations of local geometric primitives such as line segments and ellipses (Chia *et al.* 2010).

**Subspace models:** The statistical models in this chapter are based on subspace models such as probabilistic PCA (Tipping 2001) although in their original presentation they used regular (non-probabilistic) PCA (see section 13.4.2). The models could equally have been built using factor analysis (Rubin & Thayer 1982). This has the disadvantage that it cannot be learned in closed form, but can cope with modeling the joint distribution of quantities that are expressed in different units (e.g., shape and texture in active appearance models). Non-linear generalizations of PCA (Schölkopf *et al.* 1998) and factor analysis (Lawrence 2005) have extended these statistical models to the non-Gaussian case.

**Active shape and appearance models** More details about active shape models can be found in Cootes *et al.* (1995). More details about active appearance models can be found in Cootes *et al.* (2001) and Stegmann (2002). Jones & Soatto (2005) presented an interesting extension to active appearance models in which the objects was modeled as a number of superimposed layers. Recent interest in active appearance models has focussed on improving the efficiency of fitting algorithms (Matthews & Baker 2004; Matthews *et al.* 2007; Amberg *et al.* 2009). They have been applied to many tasks including face recognition, face pose estimation and expression recognition (Lanitis *et al.* 1997) and lip reading (Matthews *et al.* 2002). Several authors have investigated nonlinear approaches including systems based on mixture models (Cootes & Taylor 1997), kernel PCA (Romdhani *et al.* 1999) and the GPLVM (Huang *et al.* 2011).

**3D Morphable models:** Morphable models for faces were first introduced by Blanz & Vetter (1999) and were subsequently applied to editing images and video (Blanz *et al.* 2003), for face recognition (Blanz & Vetter 2003; Blanz *et al.* 2005) and for tracking 3D faces (Matthews *et al.* 2007). A related model has been developed for vehicles (Leotta & Mundy 2011).

**Body Tracking:** Generative models for tracking human bodies have been developed based on a number of representations including cylinders (Hogg 1983), ellipsoids (Bregler & Malik 1998), stick men (Mori *et al.* 2004), and meshes (Shakhnarovich *et al.* 2003). As well as 3D models, attempts have also been made to fit purely 2D models (e.g., Felzenszwalb & Huttenlocher 2005, and see Rehg *et al.* 2003). Some research has focused on multi-camera setups which help disambiguate the observed data (e.g., Gavrila & Davis 1996). Models for tracking the body in time sequences have attracted a lot of attention (e.g., Deutscher *et al.* 2000; Sidenbladh *et al.* 2000 and see chapter 19). Recent work has attempted to leverage knowledge of the physics of the body movement to improve the results (Brubaker *et al.* 2010). There are also a number of approaches to body tracking based on regression (see chapter 8). Reviews of human motion tracking can be found in Forsyth *et al.* (2006), Moeslund *et al.* (2006), Poppe (2007) and Sigal *et al.* (2010).

**Body models for graphics:** The work of Anguelov *et al.* (2005), who combined the skeletal and statistical shape models, was preceeded by Allen *et al.* (2003) and Seo & Magnenat-Thalmann (2003), who also applied PCA to skeletal models for the human body although they do not include a component that models deformations due to muscle movement.

**Hand models:** Many authors have developed models for tracking hands including Rehg & Kanade (1994), Rehg & Kanade (1995), Heap & Hogg (1996), Stenger *et al.* (2001a), Wu *et al.* (2001), and Lu *et al.* (2003). De La Gorce *et al.* (2008) present a very sophisticated model that describes texture, shading, and self-occlusions of the hand.

## Problems

**Problem 17.1** A conic is defined as the set of points where

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} \alpha & \beta & \gamma \\ \beta & \delta & \epsilon \\ \gamma & \epsilon & \zeta \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0,$$

or

$$\tilde{\mathbf{x}}^T \mathbf{C} \tilde{\mathbf{x}} = 0.$$

Use MATLAB to draw the 2D function  $\tilde{\mathbf{x}}^T \mathbf{C} \tilde{\mathbf{x}}$  and identify the set of positions where this function is zero for the following matrices:

$$\mathbf{C}_1 = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -1 \end{bmatrix} \quad \mathbf{C}_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -2 \end{bmatrix} \quad \mathbf{C}_3 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

**Problem 17.2** Devise an efficient algorithm to compute the distance transform. The algorithm should take a binary image and return at each pixel the city block distance to

the nearest non-zero element of the original image. The city block distance  $d$  between pixels  $(x_1, y_1)$  and pixel  $(x_2, y_2)$  is defined as

$$d = |x_1 - x_2| + |y_1 - y_2|.$$

**Problem 17.3** Consider a prior that is based in the curvature term:

$$\text{curve}[\mathbf{w}, n] = -(\mathbf{w}_{n-1} - 2\mathbf{w}_n + \mathbf{w}_{n+1})^T (\mathbf{w}_{n-1} - 2\mathbf{w}_n + \mathbf{w}_{n+1}).$$

If landmark point  $\mathbf{w}_1 = [100, 100]$  and landmark point  $\mathbf{w}_3$  is at position  $\mathbf{w}_3 = [200, 300]$ , what position  $\mathbf{w}_2$  will minimize the function  $\text{curve}[\mathbf{w}, 2]$ ?

**Problem 17.4** If the snake as described in section 17.2 is initialized in an empty image, how would you expect it to evolve during the fitting procedure?

**Problem 17.5** The spacing element of the snake prior (equation 17.5) encourages all of the control points of the snake to be the equidistant. An alternative approach is to give the snake a tendency to shrink (so that it collapses around objects). Write out an alternative expression for the spacing term that accomplishes this goal.

**Problem 17.6** Devise a method to find the ‘best’ weight vector  $\mathbf{h}$  given a new vector  $\mathbf{w}$  and the parameters  $\{\boldsymbol{\mu}, \boldsymbol{\Phi}, \sigma^2\}$  of the PPCA model (see figure 17.10).

**Problem 17.7** Show that if the singular value decomposition of a matrix  $\mathbf{W}$  can be written as  $\mathbf{W} = \mathbf{U}\mathbf{L}\mathbf{V}^T$ , then it follows that

$$\begin{aligned} \mathbf{W}\mathbf{W}^T &= \mathbf{U}\mathbf{L}^2\mathbf{U}^T \\ \mathbf{W}^T\mathbf{W} &= \mathbf{V}\mathbf{L}^2\mathbf{V}^T. \end{aligned}$$

**Problem 17.8** Devise a method to learn the PPCA model using the EM algorithm, giving details of both the E- and M-steps. Are you guaranteed to get the same answer as the method based on the SVD?

**Problem 17.9** Show that the maximum a posteriori solution for the hidden weight variable  $\mathbf{h}$  is as given in equation 17.32.

**Problem 17.10** You are given a set of 100 male faces and 100 female faces. By hand you mark 50 landmark points on each image. Describe how to use this data to develop a generative approach to gender classification based on shape alone. Describe both the training process and how you would infer the gender for a new face that does not contain landmark points.

**Problem 17.11** Imagine that we have learned a point distribution for the shape of the human face. Now we see a new face where everything below the nose is occluded by a scarf. How could you exploit the model to estimate both the positions of the landmark points in the top half of the face and the landmark points in the (missing) bottom half of the face?

**Problem 17.12** An alternative approach to building a nonlinear model of shape is to use a mixture model. Describe an approach to training a statistical shape model based on the mixture of probabilistic principal component analyzers. How would you fit this model to a new image?

**Problem 17.13** One way to warp one image to another is to implement a piecewise affine warp. Assume that we have a number of points in image 1 and their corresponding points in image 2. We first triangulate each set of points in the same way. We now represent the

position  $\mathbf{x}_1$  in image 1 as a weighted sum of the three vertices of the triangle  $\mathbf{a}_1, \mathbf{b}_1, \mathbf{c}_1$  that it lies in so that

$$\mathbf{x}_1 = \alpha\mathbf{a}_1 + \beta\mathbf{b}_1 + \gamma\mathbf{c}_1,$$

where the weights are constrained to be positive with  $\alpha + \beta + \gamma = 1$ . These weights are known as *barycentric coordinates*.

To find the position in the second image, we then compute the position relative to the three vertices  $\mathbf{a}_2, \mathbf{b}_2, \mathbf{c}_2$  of the warped triangle so that

$$\mathbf{x}_2 = \alpha\mathbf{a}_2 + \beta\mathbf{b}_2 + \gamma\mathbf{c}_2.$$

How can we compute the weights  $\alpha, \beta, \gamma$ ? Devise a method to warp the whole image in this manner.

**Problem 17.14** Consider an ellipsoid in 3D space that is represented by the quadric

$$\tilde{\mathbf{w}}^T \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{b}^T & c \end{bmatrix} \tilde{\mathbf{w}} = 0,$$

where  $\mathbf{A}$  is a  $3 \times 3$  matrix,  $\mathbf{b}$  is a  $3 \times 1$  vector, and  $c$  is a scalar.

For a normalized camera we can write the world point  $\tilde{\mathbf{w}}$  in terms of the image point  $\tilde{\mathbf{x}}$  as  $\tilde{\mathbf{w}} = [\tilde{\mathbf{x}}^T, s]^T$  where  $s$  is a scaling factor that determines the distance along the ray  $\tilde{\mathbf{x}}$ .

- (i) Combine these conditions to produce a criterion that must be true for an image point  $\tilde{\mathbf{x}}$  to lie within the projection of the conic.
- (ii) The edge of the image of the conic is the locus of points for which there is a single solution for the distance  $s$ . Outside the conic there is no real solution for  $s$  and inside it there are two possible solutions corresponding to the front and back face of the quadric. Use this intuition to derive an expression for the conic in terms of  $\mathbf{A}, \mathbf{b}$  and  $\mathbf{c}$ . If the camera has intrinsic matrix  $\Lambda$ , what would the new expression for the conic be?

## Chapter 18

# Models for style and identity

In this chapter we discuss a family of models that explain observed data in terms of several underlying causes. These causes can be divided into three types: the identity of the object, the style in which it is observed, and the remaining variation.

To motivate these models, consider *face recognition*. For a facial image, the identity of the face (i.e., whose face it is) obviously influences the observed data. However, the style in which the face is viewed is also important. The pose, expression, and illumination are all style elements that might be modeled. Unfortunately, many other things also contribute to the final observed data: the person may have applied cosmetics, put on glasses, grown a beard, or dyed their hair. These myriad contributory elements are too difficult to model, and so are explained with a generic noise term.

In face recognition tasks, our goal is to infer whether the identities of face images are the same or different. For example, in *face verification*, we aim to infer a binary variable  $w \in \{0, 1\}$ , where  $w = 0$  indicates that the identities differ and  $w = 1$  indicates that they are the same. This task is extremely challenging when there are large changes in pose, illumination, or expression; the change in the image due to style may dwarf the change due to identity (figure 18.1).

The models in this chapter are generative, so the focus is on building separate density models over the observed image data  $\mathbf{x}$  for the cases where the faces do and don't have the same identity. They are all *subspace models* and describe data as a linear combination of basis vectors. We have previously encountered several models of this type, including factor analysis (section 7.6) and PPCA (section 17.5.1). The models in this chapter are most closely related to factor analysis, so we will start by reviewing this.

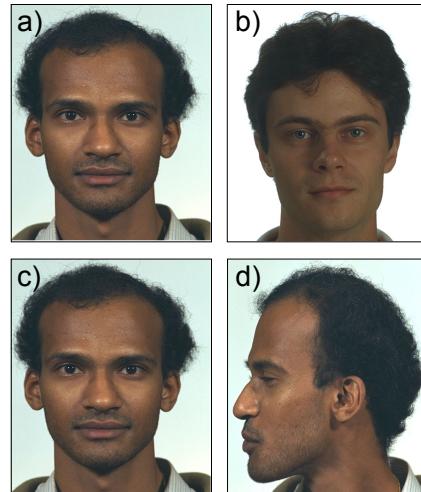
### Factor analysis

Recall that the factor analysis model explained the  $i^{th}$  data example  $\mathbf{x}_i$  as

$$\mathbf{x}_i = \boldsymbol{\mu} + \boldsymbol{\Phi}\mathbf{h}_i + \boldsymbol{\epsilon}_i, \quad (18.1)$$

where  $\boldsymbol{\mu}$  is the overall mean of the data. The matrix  $\boldsymbol{\Phi} = [\phi_1, \phi_2, \dots, \phi_K]$  contains  $K$  factors in its columns. Each factor can be thought of as a basis vector in a

**Figure 18.1** Face recognition. In face recognition the goal is to draw inferences about the identities of face images. This is difficult because the style in which the picture was taken can have a more drastic effect on the observed data than the identities themselves. For example, the images in a-b) are more similar to one another by most measures than the images in c-d) because the style (pose) has changed in the latter case. Nonetheless, the identities in a-b) are different but the identities in c-d) are the same. We must build models that tease apart the contributions of identity and style to make accurate inferences about whether the identities match.



high-dimensional space, and so together they define a subspace. The  $K$  elements of the hidden variable  $\mathbf{h}_i$  weight the  $K$  factors to explain the observed deviations of the data from the mean. Remaining differences that cannot be explained in this way are ascribed to additive noise  $\boldsymbol{\epsilon}_i$  which is normally distributed with diagonal covariance  $\boldsymbol{\Sigma}$ .

In probabilistic terms, we write

$$\begin{aligned} Pr(\mathbf{x}_i|\mathbf{h}_i) &= \text{Norm}_{\mathbf{x}_i}[\boldsymbol{\mu} + \boldsymbol{\Phi}\mathbf{h}_i, \boldsymbol{\Sigma}] \\ Pr(\mathbf{h}_i) &= \text{Norm}_{\mathbf{h}_i}[\mathbf{0}, \mathbf{I}], \end{aligned} \quad (18.2)$$

where we have also defined a suitable prior over the hidden variable  $\mathbf{h}_i$ . Ancestral sampling from this model is illustrated in figure 18.2.

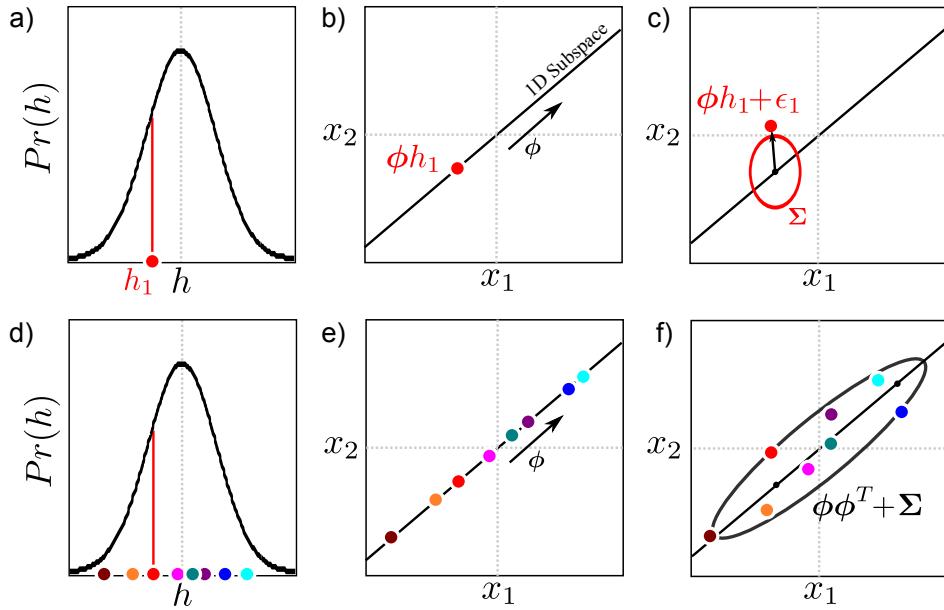
We can compute the likelihood of observing a new data example by marginalizing over the hidden variable, to get a final probability model

$$\begin{aligned} Pr(\mathbf{x}_i) &= \int Pr(\mathbf{x}_i, \mathbf{h}_i) d\mathbf{h}_i = \int Pr(\mathbf{x}_i|\mathbf{h}_i)Pr(\mathbf{h}_i) d\mathbf{h}_i \\ &= \text{Norm}_{\mathbf{x}_i}[\boldsymbol{\mu}, \boldsymbol{\Phi}\boldsymbol{\Phi}^T + \boldsymbol{\Sigma}]. \end{aligned} \quad (18.3)$$

To learn this model from training data  $\{\mathbf{x}_i\}_{i=1}^I$ , we use the expectation maximization algorithm. In the E-step, we compute the posterior distribution  $Pr(\mathbf{h}_i|\mathbf{x}_i)$  over each hidden variable  $\mathbf{h}_i$ ,

$$Pr(\mathbf{h}_i|\mathbf{x}_i) = \text{Norm}_{\mathbf{h}_i}[(\boldsymbol{\Phi}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\Phi} + \mathbf{I})^{-1} \boldsymbol{\Phi}^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_i - \boldsymbol{\mu}), (\boldsymbol{\Phi}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\Phi} + \mathbf{I})^{-1}]. \quad (18.4)$$

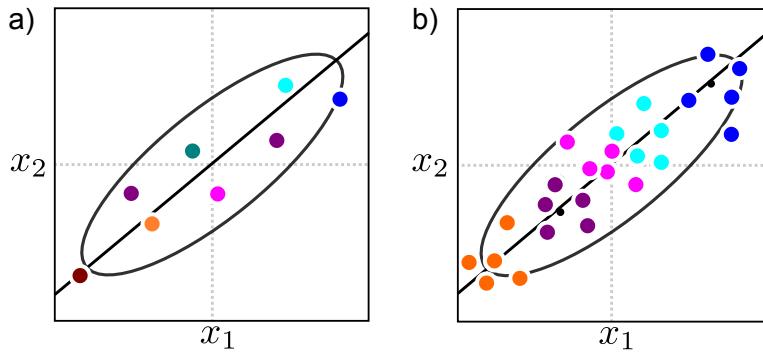
In the M-step we update the parameters as



**Figure 18.2** Ancestral sampling from factor analyzer. In both this figure and other subsequent figures in this chapter, we assume that the mean  $\mu$  is zero. a) To generate from a factor analyzer defined over 2D data we first choose the hidden variable  $\mathbf{h}_i$  from the normally distributed prior. Here  $h_i$  is a 1D variable and a small negative value is drawn. b) For each case we weight the factors  $\Phi$  by the hidden variable. This generates a point on the subspace (here a 1D subspace indicated by black line). c) Then we add the noise term  $\epsilon_i$ , which is normally distributed with covariance  $\Sigma$ . Finally, we would add a mean term  $\mu$  (not shown). d-f) This process is repeated many times. The final distribution of the data in (f) is a normal distribution that is oriented along the subspace. Deviations from this subspace are due to the noise term. The final covariance is  $\Phi\Phi^T + \Sigma$ .

$$\begin{aligned}
 \hat{\mu} &= \frac{\sum_{i=1}^I \mathbf{x}_i}{I} \\
 \hat{\Phi} &= \left( \sum_{i=1}^I (\mathbf{x}_i - \hat{\mu}) \mathbf{E}[\mathbf{h}_i]^T \right) \left( \sum_{i=1}^I \mathbf{E}[\mathbf{h}_i \mathbf{h}_i^T] \right)^{-1} \\
 \hat{\Sigma} &= \frac{1}{I} \sum_{i=1}^I \text{diag} \left[ (\mathbf{x}_i - \hat{\mu})(\mathbf{x}_i - \hat{\mu})^T - \hat{\Phi} \mathbf{E}[\mathbf{h}_i] (\mathbf{x}_i - \hat{\mu})^T \right], \quad (18.5)
 \end{aligned}$$

where the expectations  $E[\mathbf{h}_i]$  and  $E[\mathbf{h}_i \mathbf{h}_i^T]$  over the hidden variable are extracted from the posterior distribution computed in the E-step. More details about factor analysis can be found in section 7.6.



**Figure 18.3** Subspace model vs. subspace identity model. a) The subspace model generates data that are roughly aligned along a subspace (here a 1D subspace defined by the black line) as illustrated in figure 18.2. b) In the identity subspace model, the overall data distribution is the same, but there is additional structure: points that belong to the same identity (same color) are generated in the same region of space.

## 18.1 Subspace identity model

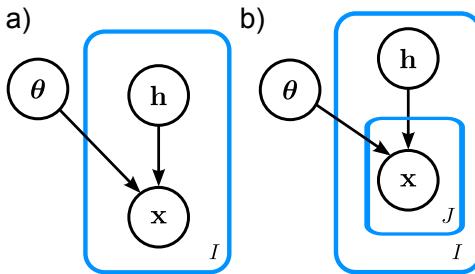
The factor analysis model provides a good description of the pixel data  $\mathbf{x}$  in frontal face images: they really do lie close to a linear subspace (see figure 7.22). However, this description of the data does not account for identity. For images which have the same style (e.g., pose, lighting), we expect faces which have the same identity to lie in a similar part of the space (figure 18.3), but there is no mechanism to accomplish this in the original model.

We now extend the factor analysis model to take account of data examples which are known to have the same identity and show how to exploit this to make inferences about the identity of new data examples. We adopt the notation  $\mathbf{x}_{ij}$  to denote the  $j^{th}$  of  $J$  observed data examples from the  $i^{th}$  of  $I$  identities (individuals). In real-world data sets it is unlikely that we will have exactly  $J$  examples for every individual and the models we present do not require this, but this assumption simplifies the notation.

The generative explanation for the observed data  $\mathbf{x}_{ij}$  is now

$$\mathbf{x}_{ij} = \boldsymbol{\mu} + \boldsymbol{\Phi}\mathbf{h}_i + \boldsymbol{\epsilon}_{ij}, \quad (18.6)$$

where all of the terms have the same interpretations as before. The key difference is that now all of the  $J$  data examples from the same individual are formed by taking the *same* linear combination  $\mathbf{h}_i$  of the basis functions  $\phi_1 \dots \phi_K$ . However, a different noise term is added for each data example, and this explains the differences between the  $J$  face images of a given individual. We can write this in probabilistic form as



**Figure 18.4** Graphical models for subspace model and subspace identity model. a) In the subspace model (factor analysis) there is one data example  $x_i$  per hidden variable  $h_i$  and some other parameters  $\theta = \{\mu, \Phi, \Sigma\}$  that describe the subspace. b) In the subspace identity model, there are  $J$  data examples  $x_{ij}$  per hidden variable  $h_i$  and all of these  $J$  examples have the same identity.

$$\begin{aligned} Pr(\mathbf{h}_i) &= \text{Norm}_{\mathbf{h}_i}[\mathbf{0}, \mathbf{I}] \\ Pr(\mathbf{x}_{ij}|\mathbf{h}_i) &= \text{Norm}_{\mathbf{x}_{ij}}[\boldsymbol{\mu} + \boldsymbol{\Phi}\mathbf{h}_i, \boldsymbol{\Sigma}], \end{aligned} \quad (18.7)$$

where as before we have defined a prior over the hidden variables. The graphical models for both factor analysis and the subspace identity model are illustrated in figure 18.4. Figure 18.5 illustrates ancestral sampling from the subspace identity model; as desired this produces data points that lie close together when the identity is the same.

One way to think of this is that we have decomposed the variance in the model into two parts. The *between-individual* variation explains the differences between data due to different identities and the *within-individual* variation explains the differences between data examples due to all other factors. The data density for a single data point remains

$$Pr(\mathbf{x}_{ij}) = \text{Norm}_{\mathbf{x}_{ij}}[\boldsymbol{\mu}, \boldsymbol{\Phi}\boldsymbol{\Phi}^T + \boldsymbol{\Sigma}]. \quad (18.8)$$

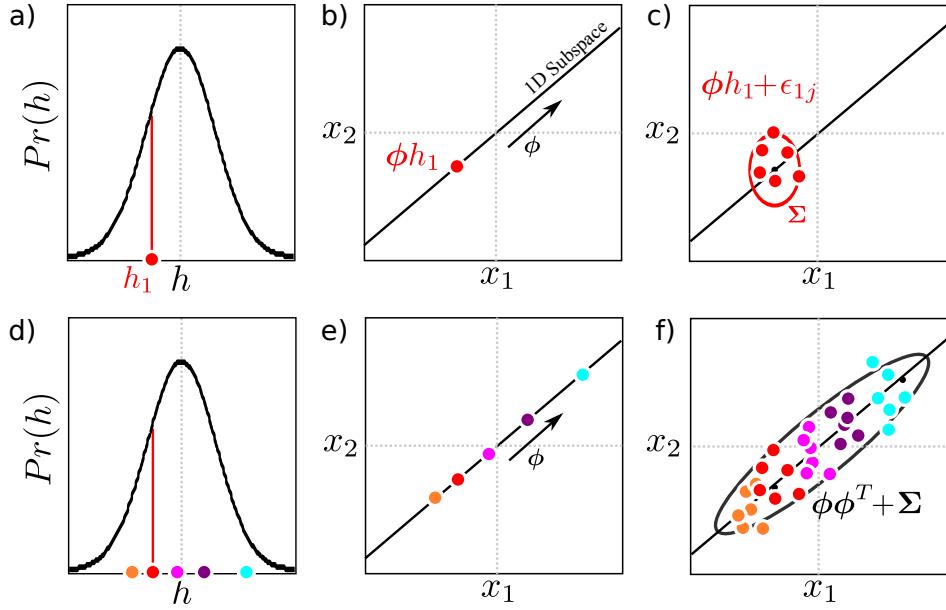
However, the two components of the variance now have clear interpretations. The term  $\boldsymbol{\Phi}\boldsymbol{\Phi}^T$  corresponds to the between-individual variation, and the term  $\boldsymbol{\Sigma}$  is the within-individual variation.

### 18.1.1 Learning

Before we consider how to use this model to draw inferences about identity in face recognition tasks, we will briefly discuss how to learn the parameters  $\theta = \{\mu, \Phi, \Sigma\}$ . As for the factor analysis model, we exploit the EM algorithm to iteratively increase a bound on the log likelihood. In the E-step we compute the posterior probability distribution over each of the hidden variables  $\mathbf{h}_i$  given all of the data  $\mathbf{x}_{i\bullet} = \{\mathbf{x}_{ij}\}_{j=1}^J$  associated with that particular identity,

Algorithm 18.1

Problem 18.1

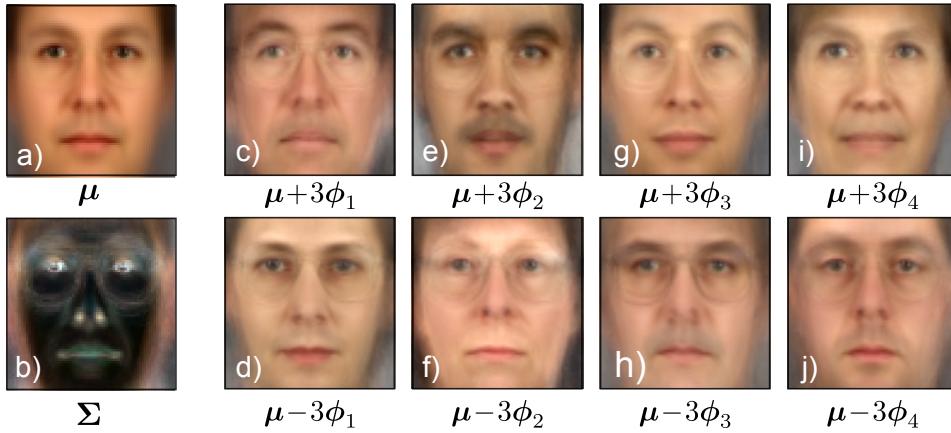


**Figure 18.5** Ancestral sampling from identity subspace model. a) To generate from this model we first choose the hidden variable  $\mathbf{h}_i$  from the normally distributed prior. Here  $h_i$  is a 1D variable and a small negative number is drawn. b) We weight the factors  $\Phi$  by the hidden variable. This generates a point on the subspace. c) Then we add different noise terms  $\{\epsilon_{ij}\}_{j=1}^J$  to create each of the  $J$  examples  $\{\mathbf{x}_{ij}\}_{j=1}^J$ . In each case, the noise is normally distributed with covariance  $\Sigma$ . Finally, we would add a mean term  $\mu$  (not shown). d-f) This process is repeated several times. The final distribution of the data in f) is a normal distribution with covariance  $\Phi\Phi^T + \Sigma$ . However, it is structured so that points with the same hidden variable (identity) are close to one another.

$$\begin{aligned} Pr(\mathbf{h}_i | \mathbf{x}_{i\bullet}) &= \frac{\prod_{j=1}^J Pr(\mathbf{x}_{ij} | \mathbf{h}_i) Pr(\mathbf{h}_i)}{\int \prod_{j=1}^J Pr(\mathbf{x}_{ij} | \mathbf{h}_i) Pr(\mathbf{h}_i) d\mathbf{h}_i} \\ &= \text{Norm}_{\mathbf{h}_i} \left[ \left( J\Phi^T \Sigma^{-1} \Phi + \mathbf{I} \right)^{-1} \Phi^T \Sigma^{-1} \sum_{j=1}^J (\mathbf{x}_{ij} - \mu), \left( J\Phi^T \Sigma^{-1} \Phi + \mathbf{I} \right)^{-1} \right]. \end{aligned} \quad (18.9)$$

From this we extract the moments that will be needed in the M-step:

$$\begin{aligned} E[\mathbf{h}_i] &= (J\Phi^T \Sigma^{-1} \Phi + \mathbf{I})^{-1} \Phi^T \Sigma^{-1} \sum_{j=1}^J (\mathbf{x}_{ij} - \mu) \\ E[\mathbf{h}_i \mathbf{h}_i^T] &= (J\Phi^T \Sigma^{-1} \Phi + \mathbf{I})^{-1} + E[\mathbf{h}_i] E[\mathbf{h}_i]^T. \end{aligned} \quad (18.10)$$



**Figure 18.6** Subspace identity model parameters. These parameters were learned from  $J = 3$  images of  $I = 195$  individuals from the XM2VTS data set. a) Estimated mean  $\mu$ . b) Estimated covariance,  $\Sigma$ . c-l) Four of 32 subspace directions explored by adding and subtracting multiples of each dimension to the mean.

In the M-step we update the parameters using the relations

Problem 18.2

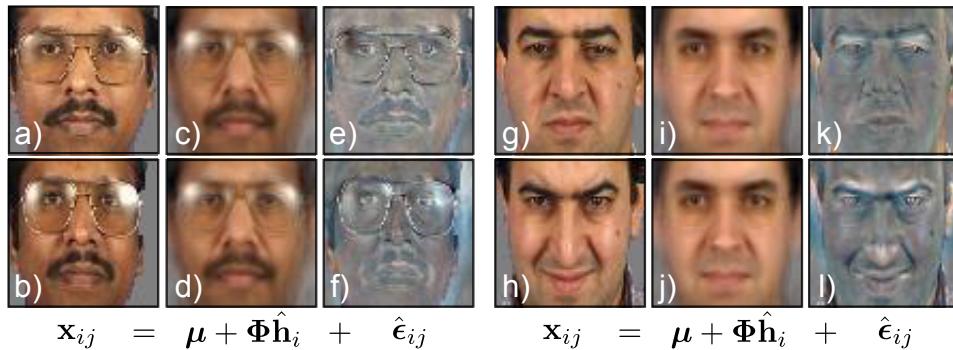
$$\begin{aligned}\hat{\mu} &= \frac{\sum_{i=1}^I \sum_{j=1}^J \mathbf{x}_{ij}}{IJ} \\ \hat{\Phi} &= \left( \sum_{i=1}^I \sum_{j=1}^J (\mathbf{x}_{ij} - \hat{\mu}) \mathbf{E}[\mathbf{h}_i]^T \right) \left( \sum_{i=1}^I J \mathbf{E}[\mathbf{h}_i \mathbf{h}_i^T] \right)^{-1} \\ \Sigma &= \frac{1}{IJ} \sum_{i=1}^I \sum_{j=1}^J \text{diag} \left[ (\mathbf{x}_{ij} - \hat{\mu})(\mathbf{x}_{ij} - \hat{\mu})^T - \hat{\Phi} \mathbf{E}[\mathbf{h}_i] (\mathbf{x}_{ij} - \hat{\mu})^T \right],\end{aligned}\quad (18.11)$$

which were generated by taking the derivative of the EM bound with respect to the relevant quantities, equating the results to zero, and re-arranging. We alternate the E- and M-steps until the log-likelihood of the data no longer increases.

Figure 18.6 shows parameters learned from  $70 \times 70$  pixel face images from the XM2VTS database. A model with a  $K=32$  dimensional hidden space was learned with 195 identities and 3 images per person. The subspace directions capture major changes that correlate with identity. For example, ethnicity and gender are clearly represented. The noise describes whatever remains. It is most prominent around high-contrast features such as the eyes.

In figure 18.7 we decompose pairs of matching images into their identity and noise components. To accomplish this, we compute the MAP hidden variable  $\hat{\mathbf{h}}_i$ . The posterior over  $\mathbf{h}$  is normal (equation 18.9) and so the MAP estimate is simply the mean of this normal. We can then visualize the identity component

Problem 18.3



**Figure 18.7** Fitting the subspace identity model to new data. a-b) Original images  $\mathbf{x}_{i1}$  and  $\mathbf{x}_{i2}$ . These faces can be decomposed into the sum of c-d) an identity component and e-f) a within-individual noise component. To decompose the image in this way, we computed the MAP estimate  $\hat{\mathbf{h}}_i$  of the hidden variable and set the identity component to be  $\boldsymbol{\mu} + \boldsymbol{\Phi} \hat{\mathbf{h}}_i$ . The noise comprises whatever cannot be explained by the identity. g-l) A second example.

$\boldsymbol{\mu} + \boldsymbol{\Phi} \hat{\mathbf{h}}_i$  which is the same for each image of the same individual and looks like a prototypical view of that person. We can also visualize the within-individual noise  $\hat{\boldsymbol{\epsilon}}_{ij} = \mathbf{x}_{ij} - \boldsymbol{\mu} - \boldsymbol{\Phi} \hat{\mathbf{h}}_i$ , which explains how each image of the same person differs.

### 18.1.2 Inference

We will now discuss how to exploit the model to make inferences about new faces that were not part of the training data set. In face verification problems, we observe two data examples  $\mathbf{x}_1$  and  $\mathbf{x}_2$  and wish to infer the state of the world  $w \in \{0, 1\}$  where  $w = 0$  denotes the case where the data examples have different identities and  $w = 1$  denotes the case where the data examples have the same identity.

This is a generative model, and so we calculate the posterior  $Pr(w|\mathbf{x}_1, \mathbf{x}_2)$  over the world state using Bayes' rule

$$Pr(w=1|\mathbf{x}_1, \mathbf{x}_2) = \frac{Pr(\mathbf{x}_1, \mathbf{x}_2|w=1)Pr(w=1)}{\sum_{n=0}^1 Pr(\mathbf{x}_1, \mathbf{x}_2|w=n)Pr(w=n)}. \quad (18.12)$$

To compute this we need the prior probabilities  $Pr(w=0)$  and  $Pr(w=1)$  of the data examples having different identities or the same identity. In the absence of any other information we might set these both to 0.5. We also need expressions for the likelihoods  $Pr(\mathbf{x}_1, \mathbf{x}_2|w=0)$  and  $Pr(\mathbf{x}_1, \mathbf{x}_2|w=1)$ .

We will first consider the likelihood  $Pr(\mathbf{x}_1, \mathbf{x}_2|w=0)$  when the two data points have different identities. Here, each image is explained by a different hidden variable and so the generative equation looks like

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\Phi} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Phi} \end{bmatrix} \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \end{bmatrix} + \begin{bmatrix} \boldsymbol{\epsilon}_1 \\ \boldsymbol{\epsilon}_2 \end{bmatrix}, \quad (18.13)$$

We note that this has the form of a factor analyzer:

$$\mathbf{x}' = \boldsymbol{\mu}' + \boldsymbol{\Phi}' \mathbf{h}' + \boldsymbol{\epsilon}'. \quad (18.14)$$

We can re-express this in probabilistic terms as

$$\begin{aligned} Pr(\mathbf{x}'|\mathbf{h}') &= \text{Norm}_{\mathbf{x}'}[\boldsymbol{\mu}' + \boldsymbol{\Phi}' \mathbf{h}', \boldsymbol{\Sigma}'] \\ Pr(\mathbf{h}') &= \text{Norm}_{\mathbf{h}'}[\mathbf{0}, \mathbf{I}], \end{aligned} \quad (18.15)$$

where  $\boldsymbol{\Sigma}'$  is defined as

$$\boldsymbol{\Sigma}' = \begin{bmatrix} \boldsymbol{\Sigma} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma} \end{bmatrix}. \quad (18.16)$$

We can now compute the likelihood  $Pr(\mathbf{x}_1, \mathbf{x}_2|w = 0)$  by writing the joint likelihood of the compound variables  $\mathbf{x}'$  and  $\mathbf{h}'$  and marginalizing over  $\mathbf{h}'$ , so that

$$\begin{aligned} Pr(\mathbf{x}_1, \mathbf{x}_2|w = 0) &= \int Pr(\mathbf{x}'|\mathbf{h}') Pr(\mathbf{h}') d\mathbf{h}' \\ &= \text{Norm}_{\mathbf{x}'}[\boldsymbol{\mu}', \boldsymbol{\Phi}' \boldsymbol{\Phi}'^T + \boldsymbol{\Sigma}'], \end{aligned} \quad (18.17)$$

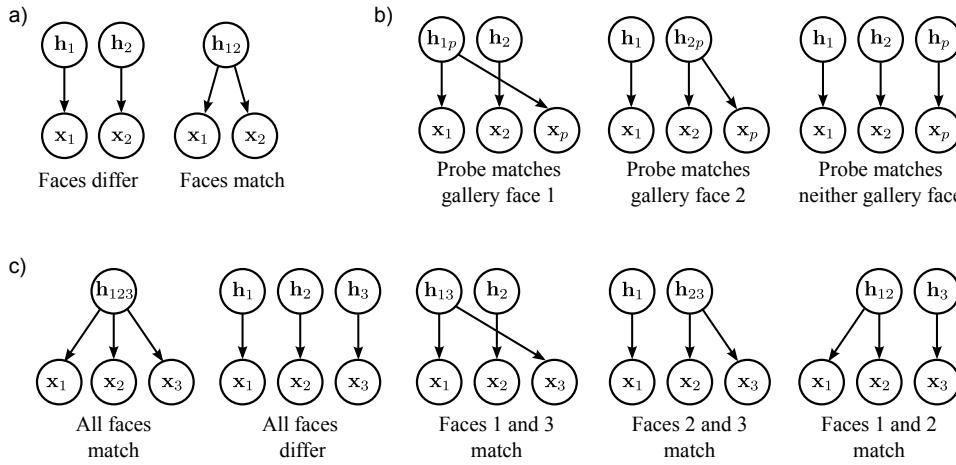
where we have used the standard factor analysis result for the integration.

For the case where the faces match ( $w = 1$ ), we know that both data examples must have been created from the same hidden variable. To compute the likelihood  $Pr(\mathbf{x}_1, \mathbf{x}_2|w = 1)$ , we write the compound generative equation

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\Phi} \\ \boldsymbol{\Phi} \end{bmatrix} \mathbf{h}_{12} + \begin{bmatrix} \boldsymbol{\epsilon}_1 \\ \boldsymbol{\epsilon}_2 \end{bmatrix}, \quad (18.18)$$

which we notice also has the form of a standard factor analyzer (equation 18.14) and so we can compute the likelihood using the same method.

One way to think about this process is that we are comparing the likelihood for two different models of the data (figure 18.8a). However, it should be noted that the model that categorizes the faces as different ( $w = 0$ ) has two variables ( $\mathbf{h}_1$  and  $\mathbf{h}_2$ ), whereas the model that categorizes the faces as the same ( $w = 1$ ) has only one ( $\mathbf{h}_{12}$ ). One might expect then that the model with more variables would always provide a superior explanation of the data. In fact this does not happen here, because we marginalized these variables out of the likelihoods, and so the final expressions do not include these hidden variables. This is an example of *Bayesian model selection*: it is valid to compare models with different numbers of parameters as long as they are marginalized out of the final solution.



**Figure 18.8** Inference as model comparison. a) Verification task. Given two faces  $\mathbf{x}_1$  and  $\mathbf{x}_2$  we must decide whether (i) they belong to different people and hence have separate hidden variables  $\mathbf{h}_1$ ,  $\mathbf{h}_2$  or (ii) they belong to the same person and hence share a single hidden variable  $\mathbf{h}_{12}$ . These two hypotheses are illustrated as the two graphical models. b) Open set identification task. We are given a library  $\{\mathbf{x}_i\}_{i=1}^I$  of faces that belong to different people and a probe face  $\mathbf{x}_p$ . In this case (where  $I = 2$ ), we must decide whether the probe matches (i) gallery face 1, (ii) gallery face 2, or (iii) none of the gallery faces. In closed set identification, we simply omit the latter model. c) Clustering task. Given three faces  $\mathbf{x}_1$ ,  $\mathbf{x}_2$ , and  $\mathbf{x}_3$  we must decide whether (i) they are all from the same person, (ii) all from different people, or (iii-v) two of the three match.

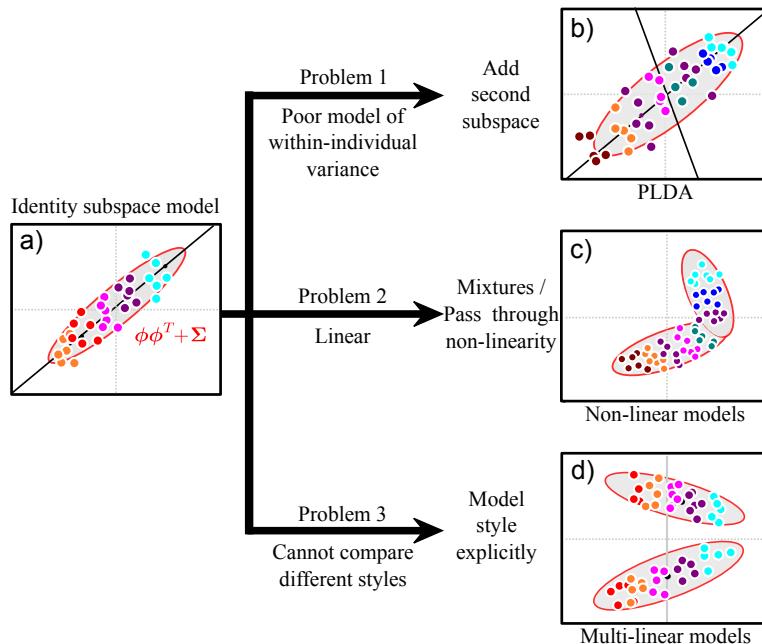
### 18.1.3 Inference in other recognition tasks

Face verification is only one of several possible face recognition problems. Others include:

- *Closed set identification:* find which one of  $N$  gallery faces matches a given probe face.
- *Open set identification:* choose one of  $N$  gallery faces that matches a probe or identify that there is no match in the gallery.
- *Clustering:* given  $N$  faces, find how many different people are present and which face belongs to which person.

#### Problem 18.4

All of these models can be thought of in terms of model comparison (figure 18.8). For example, consider a clustering task in which we have three faces  $\mathbf{x}_1$ ,  $\mathbf{x}_2$ ,  $\mathbf{x}_3$  and wish to know if (i) there are three different identities, or (ii) all of the images belong to the same person, or (iii) two images belong to the same person and the third belongs to someone different (distinguishing between the three different ways that



**Figure 18.9** Identity models. a) There are three limitations to the subspace identity model. b) First, it has an impoverished model of the within-individual noise. To remedy this we develop probabilistic linear discriminant analysis. c) Second, it is linear and can only describe the distribution of faces as a normal distribution. Hence, we develop nonlinear models based on mixtures and kernels. d) Third, it does not work well when there are large style changes. To cope with this, we introduce multi-linear models.

this can happen). The world can take five states  $w \in \{1, 2, 3, 4, 5\}$  corresponding to these five situations, and each is explained by a different compound generative equation. For example, if the first two images are the same person, but the third is different we would write

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu} \\ \boldsymbol{\mu} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\Phi} & \mathbf{0} \\ \boldsymbol{\Phi} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Phi} \end{bmatrix} \begin{bmatrix} \mathbf{h}_{12} \\ \mathbf{h}_3 \end{bmatrix} + \begin{bmatrix} \boldsymbol{\epsilon}_1 \\ \boldsymbol{\epsilon}_2 \\ \boldsymbol{\epsilon}_3 \end{bmatrix}, \quad (18.19)$$

which again has the form of a factor analyzer and so we can compute the likelihood using the method described earlier. We compare the likelihood for this model to the likelihoods for the other models using Bayes' rule with suitable priors.

Problem 18.5

### 18.1.4 Limitations of identity subspace model

The subspace identity model has three main limitations (figure 18.9).

1. The model of within-individual covariance (diagonal) is inadequate.
2. It is a linear model and cannot model non-Gaussian densities.
3. It cannot model large changes in style (e.g., frontal vs. profile faces).

**Problem 18.6**  
**Problem 18.7**

We tackle these problems by introducing probabilistic linear discriminant analysis (section 18.2), nonlinear identity models (section 18.3), and multi-linear models (sections 18.4-18.5), respectively.

## 18.2 Probabilistic linear discriminant analysis

The subspace identity model explains the data as the sum of a component due to the identity and an additive noise term. However, the noise term is rather simple: it describes the within-individual variation as a normal distribution with diagonal covariance. The estimated noise components that we visualized in figure 18.7 contain considerable structure, which suggests that modeling the within-individual variation at each pixel as independent is insufficient.

Probabilistic linear discriminant analysis (PLDA) uses a more sophisticated model for the within-individual variation. This model adds a new term to the generative equation that describes the within-individual variation as also lying on a subspace determined by a second factor matrix  $\Psi$ . The  $j^{th}$  image  $\mathbf{x}_{ij}$  of the  $i^{th}$  individual is now described as

$$\mathbf{x}_{ij} = \boldsymbol{\mu} + \Phi \mathbf{h}_i + \Psi \mathbf{s}_{ij} + \boldsymbol{\epsilon}, \quad (18.20)$$

where  $\mathbf{s}_{ij}$  is a hidden variable that represents the *style* of this face: it describes systematic contributions to the image from uncontrolled viewing parameters. Notice that it differs for each instance  $j$ , and so it tells us nothing about identity.

The columns of  $\Phi$  describe the space of between-individual variation and  $\mathbf{h}_i$  determines a point in this space. The columns of  $\Psi$  describe the space of within-individual variation, and  $\mathbf{s}_{ij}$  determines a point in this space. A given face is now modeled as the sum of a term  $\boldsymbol{\mu} + \Phi \mathbf{h}_i$  that derives from the identity of the individual, a term  $\Psi \mathbf{s}_{ij}$  that models the style of this particular image, and a noise term  $\boldsymbol{\epsilon}_{ij}$  that explains any remaining variation (figure 18.10).

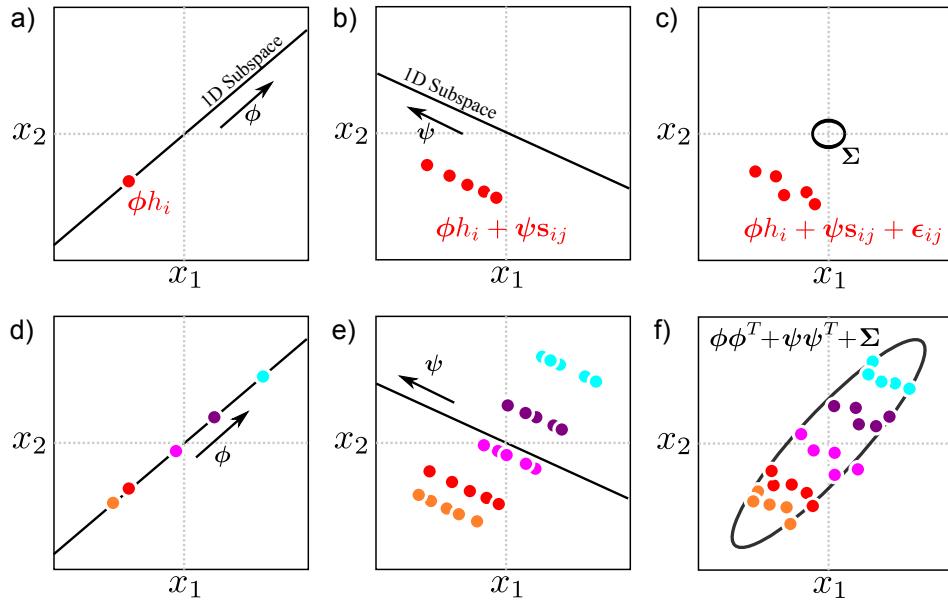
Once again, we can write the model in probabilistic terms:

$$\begin{aligned} Pr(\mathbf{h}_i) &= \text{Norm}_{\mathbf{h}_i}[\mathbf{0}, \mathbf{I}] \\ Pr(\mathbf{s}_{ij}) &= \text{Norm}_{\mathbf{s}_{ij}}[\mathbf{0}, \mathbf{I}] \\ Pr(\mathbf{x}_{ij} | \mathbf{h}_i, \mathbf{s}_{ij}) &= \text{Norm}_{\mathbf{x}_{ij}}[\boldsymbol{\mu} + \Phi \mathbf{h}_i + \Psi \mathbf{s}_{ij}, \Sigma], \end{aligned} \quad (18.21)$$

where now we have defined priors over both hidden variables.

### 18.2.1 Learning

In the E-step, we collect together all the  $J$  observations  $\{\mathbf{x}_{ij}\}_{j=1}^J$  associated with



**Figure 18.10** Ancestral sampling from PLDA model. a) We sample a hidden variable  $\mathbf{h}_i$  from the identity prior and use this to weight the between-individual factors  $\Phi$ . b) We sample  $J$  hidden variables  $\{\mathbf{s}_{ij}\}_{j=1}^J$  from the style prior and use these to weight the within-individual factors  $\Psi$ . c) Finally, we add normal noise with diagonal covariance,  $\Sigma$ . d-f) This process is repeated for several individuals. Notice that the clusters associated with each identity in f) are now oriented (compare to figure 18.5f); we have constructed a more sophisticated model of within-individual variation.

the same identity to form the compound system

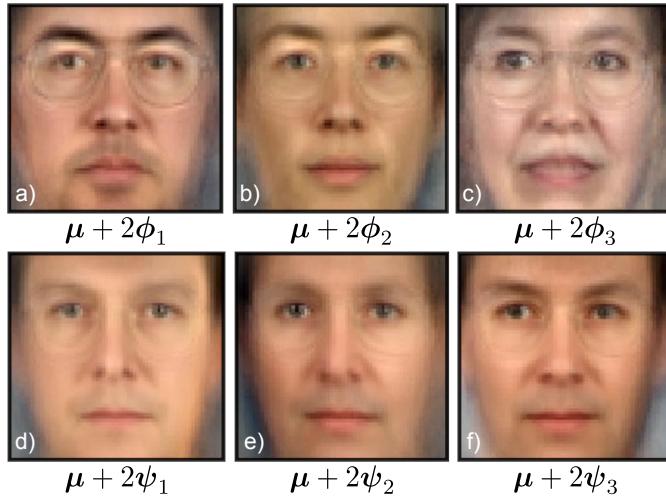
$$\begin{bmatrix} \mathbf{x}_{i1} \\ \mathbf{x}_{i2} \\ \vdots \\ \mathbf{x}_{iJ} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu} \\ \vdots \\ \boldsymbol{\mu} \end{bmatrix} + \begin{bmatrix} \Phi & \Psi & 0 & \dots & 0 \\ \Phi & 0 & \Psi & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \Phi & 0 & 0 & \dots & \Psi \end{bmatrix} \begin{bmatrix} \mathbf{h}_i \\ \mathbf{s}_{i1} \\ \mathbf{s}_{i2} \\ \vdots \\ \mathbf{s}_{iJ} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\epsilon}_{i1} \\ \boldsymbol{\epsilon}_{i2} \\ \vdots \\ \boldsymbol{\epsilon}_{iJ} \end{bmatrix}, \quad (18.22)$$

which takes the form of the original subspace identity model  $\mathbf{x}'_i = \boldsymbol{\mu}' + \Phi' \mathbf{h}'_i + \boldsymbol{\epsilon}'$ . We can hence compute the joint posterior probability distribution over all of the hidden variables in  $\mathbf{h}'$  using equation 18.9.

In the M-step we write a compound generative equation for each image,

$$\mathbf{x}_{ij} = \boldsymbol{\mu} + [\Phi \quad \Psi] \begin{bmatrix} \mathbf{h}_i \\ \mathbf{s}_{ij} \end{bmatrix} + \boldsymbol{\epsilon}_{ij}. \quad (18.23)$$

On noting that this has the form  $\mathbf{x}_{ij} = \boldsymbol{\mu} + \Phi'' \mathbf{h}''_{ij} + \boldsymbol{\epsilon}_{ij}$  of the standard factor analysis model, we can solve for the unknown parameters using equations 18.5.



**Figure 18.11** PLDA model. a-c) As we move around in the between-individual subspace  $\Phi$  the images look like different people. d-f) As we move around in the within-individual subspace  $\Psi$  the images look like the same person viewed in slightly different poses and under different illuminations. The PLDA model has successfully separated out contributions that correlate with identity from those that don't. Adapted from Li *et al.* (2011). ©2012 IEEE.

The computations require the expectations  $E[\mathbf{h}_{ij}'']$  and  $E[\mathbf{h}_{ij}''\mathbf{h}_{ij}''^T]$ , and these can be extracted from the posterior computed in the E-step.

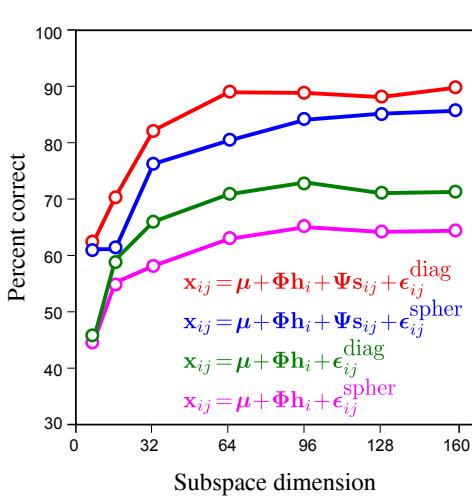
Figure 18.11 shows parameters learned from  $J = 3$  examples each of  $I = 195$  people from the XM2VTS database with 16 between-individual basis functions in  $\Phi$  and 16 within-individual basis functions in  $\Psi$ . The figure demonstrates that the model has distinguished these two components.

### 18.2.2 Inference

As for the subspace identity model, we perform inference by comparing the likelihoods of models using Bayes's rule. For example, in the verification task we compare models that explain the two data examples  $\mathbf{x}_1$  and  $\mathbf{x}_2$  as having either their own identities  $\mathbf{h}_1$  and  $\mathbf{h}_2$  or sharing a single identity  $\mathbf{h}_{12}$ . When the identities are different ( $w=0$ ), the data are generated as

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu} \end{bmatrix} + \begin{bmatrix} \Phi & \mathbf{0} & \Psi & \mathbf{0} \\ \mathbf{0} & \Phi & \mathbf{0} & \Psi \end{bmatrix} \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{s}_1 \\ \mathbf{s}_2 \end{bmatrix} + \begin{bmatrix} \boldsymbol{\epsilon}_1 \\ \boldsymbol{\epsilon}_2 \end{bmatrix}. \quad (18.24)$$

When the identities are the same ( $w=1$ ), the data are generated as

**Figure 18.12** Face recognition results.

The models were trained using three  $70 \times 70$  RGB images each from 195 people from the XM2VTS database and tested using two images each from 100 different people. A gallery was formed from one image of each of the test individuals. For each of the remaining 100 test images, the system had to identify the match in the gallery. Plots show % correct performance as a function of subspace dimensionality (number of columns in  $\Phi$  and  $\Psi$ ). Results show that as the noise model becomes more complex (adding within-individual subspace, using diagonal rather than spherical additive noise) the results improve systematically.

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu} \end{bmatrix} + \begin{bmatrix} \Phi & \Psi & 0 \\ \Phi & 0 & \Psi \end{bmatrix} \begin{bmatrix} \mathbf{h}_{12} \\ \mathbf{s}_1 \\ \mathbf{s}_2 \end{bmatrix} + \begin{bmatrix} \boldsymbol{\epsilon}_1 \\ \boldsymbol{\epsilon}_2 \end{bmatrix}. \quad (18.25)$$

Both of these formulae have the same form  $\mathbf{x}' = \boldsymbol{\mu}' + \Phi' \mathbf{h}' + \boldsymbol{\epsilon}'$  as the original factor analysis model, and so the likelihood of the data  $\mathbf{x}'$  after marginalizing out the hidden variables  $\mathbf{h}'$  is given by

$$Pr(\mathbf{x}') = \text{Norm}_{\mathbf{x}'}[\boldsymbol{\mu}', \Phi' \Phi'^T + \boldsymbol{\Sigma}'], \quad (18.26)$$

where the particular choice of  $\Phi'$  comes from equations 18.24 or 18.25, respectively. Other inference tasks concerning identity such as closed set recognition and clustering can be formulated in a similar way; we associate one value of the discrete world variable  $w = \{1, \dots, K\}$  with each possible configuration of identities, construct a generative model for each and compare the likelihoods via Bayes' rule.

Figure 18.12 compares closed-set identification performance for several models as a function of the subspace size (for the PLDA models, the size of  $\Psi$  and  $\Phi$  were always the same). The results are not state-of-the-art: the images were not properly pre-processed, and anyway, this data set is considered relatively unchallenging. Nonetheless, the pattern of results nicely demonstrates an important point: The %-correct classification improves as we increases the model's ability to describe within-individual noise: building more complex models is worth the time and effort!

## 18.3 Non-linear identity models

The models discussed so far describe the between-individual and within-individual variance by means of linear models and produce final densities that are normally

distributed. However, there is no particular reason to believe that the distribution of faces is normal. We now briefly discuss two methods to generalize the preceding models to the nonlinear case.

The first approach is to note that since the identity subspace model and PLDA are both valid probabilistic models, we can easily describe a more complex distribution in terms of mixtures of these elements. For example, a mixture of PLDAs model (figure 18.13) can be written as

$$\begin{aligned} Pr(c_i) &= \text{Cat}_{c_i}[\boldsymbol{\lambda}] \\ Pr(\mathbf{h}_i) &= \text{Norm}_{\mathbf{h}_i}[\mathbf{0}, \mathbf{I}] \\ Pr(\mathbf{s}_{ij}) &= \text{Norm}_{\mathbf{s}_{ij}}[\mathbf{0}, \mathbf{I}] \\ Pr(\mathbf{x}_{ij}|c_i, \mathbf{h}_i, \mathbf{s}_{ij}) &= \text{Norm}_{\mathbf{x}_{ij}}[\boldsymbol{\mu}_{c_i} + \boldsymbol{\Phi}_{c_i}\mathbf{h}_i + \boldsymbol{\Psi}_{c_i}\mathbf{s}_{ij}, \boldsymbol{\Sigma}_{c_i}], \end{aligned} \quad (18.27)$$

where  $c_i \in [1 \dots C]$  is a hidden variable that determines to which of the  $c$  clusters the data belong. Each cluster has different parameters, so the full model is nonlinear. To learn this model, we embed the existing learning algorithm inside a second EM loop that associates each identity with a cluster. In inference, we assume that faces must belong to the same cluster if they match.

A second approach is based on the Gaussian process latent variable model (see section 17.8). The idea is to induce a complex density by passing the hidden variable through a nonlinear function  $\mathbf{f}[\bullet]$  before using the result to weight the basis functions. For example, the generalization of the subspace identity model to the nonlinear case can be written as

$$\begin{aligned} Pr(\mathbf{h}_i) &= \text{Norm}_{\mathbf{h}_i}[\mathbf{0}, \mathbf{I}] \\ Pr(\mathbf{x}_{ij}|\mathbf{h}_i, \boldsymbol{\mu}, \boldsymbol{\Phi}, \boldsymbol{\Sigma}) &= \text{Norm}_{\mathbf{x}_{ij}}[\boldsymbol{\mu} + \boldsymbol{\Phi}\mathbf{f}[\mathbf{h}_i], \boldsymbol{\Sigma}]. \end{aligned} \quad (18.28)$$

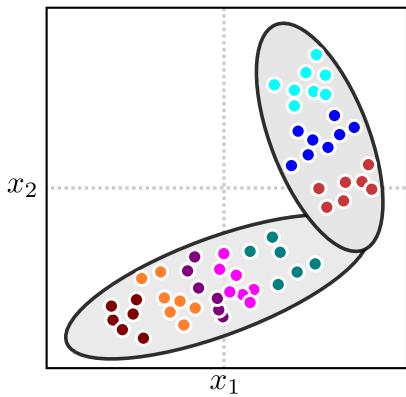
Although this model is conceptually simple, it is harder to work with in practice: it is no longer possible to marginalize over the hidden variables. However, the model is still linear with respect to the factor matrix  $\boldsymbol{\Phi}$ , and it is possible to marginalize over this and the mean  $\boldsymbol{\mu}$ , giving a likelihood term of the form

$$Pr(\mathbf{x}_{ij}|\mathbf{h}_i, \boldsymbol{\Sigma}) = \iint \text{Norm}_{\mathbf{x}_{ij}}[\boldsymbol{\mu} + \boldsymbol{\Phi}\mathbf{f}[\mathbf{h}_i], \boldsymbol{\Sigma}] d\boldsymbol{\mu} d\boldsymbol{\Phi}. \quad (18.29)$$

This model can be expressed in terms of inner products of the transformed hidden variables  $\mathbf{f}[\mathbf{h}]$  and so is amenable to kernelization. Unfortunately, because we cannot marginalize over  $\mathbf{h}_i$ , it is no longer possible exactly to compare model likelihoods directly in the inference stage. However, in practice there are ways to approximate this process.

## 18.4 Asymmetric bilinear models

The models that we have discussed so far are sufficient if the within-individual variation is small. However, there are other situations where the style of the data



**Figure 18.13** Mixture models. One way to create more complex models is to use a mixture of subspace identity models, or a mixture of PLDAs. A discrete variable is associated with each data point that indicates to which mixture component it belongs. If two faces belong to the same person, this must be the same; every image of the same person is associated with one mixture component. The within-individual model may also vary between components. Consequently, the within-individual variation may differ depending on the identity of the face.

may change considerably. For example, consider the problem of face recognition when some of the faces are frontal and others profile. Unfortunately, any given frontal face has more in common visually with other non-matching frontal faces than it does with a matching profile face.

Motivated by this problem, we now develop the *asymmetric bilinear model* for modeling identity and style: as before, we treat the identity  $\mathbf{h}_i$  as continuous, but now we treat the style  $s \in \{1 \dots S\}$  as discrete taking one of  $S$  possible values. For example, in the cross-pose face recognition example,  $s = 0$  might indicate a frontal face, and  $s = 1$  might indicate a profile face. The model is hence asymmetric as it treats identity and style differently. The expression of the identity depends on the style category so that the same identity may produce completely different data in different styles.

We adopt the notation  $\mathbf{x}_{ijs}$  to denote the  $j^{th}$  of  $J$  examples of the  $i^{th}$  of  $I$  identities in the  $s^{th}$  of  $S$  styles. The data are generated as

$$\mathbf{x}_{ijs} = \boldsymbol{\mu}_s + \Phi_s \mathbf{h}_i + \boldsymbol{\epsilon}_{ijs}, \quad (18.30)$$

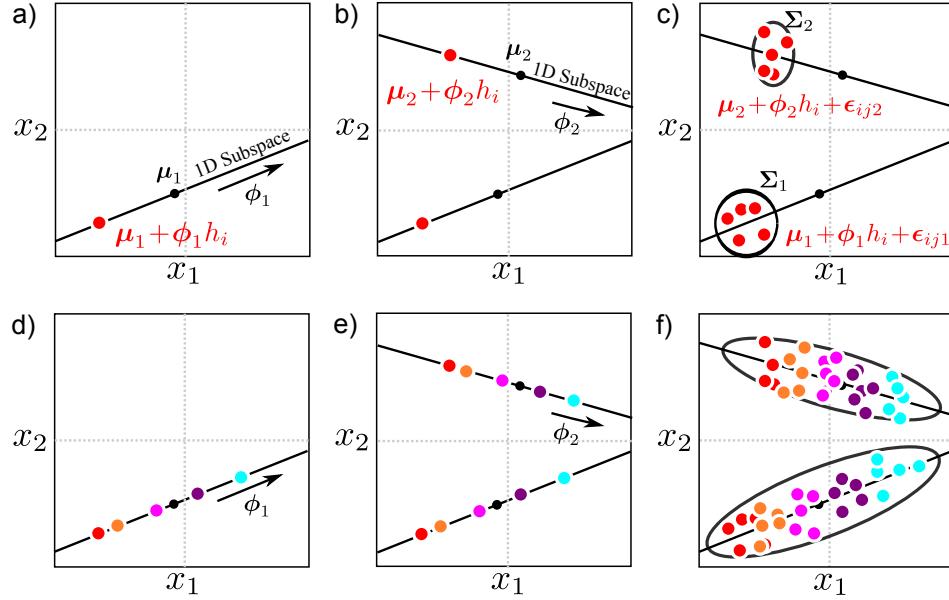
where  $\boldsymbol{\mu}_s$  is a mean vector associated with the  $s^{th}$  style,  $\Phi_s$  contains basis functions associated with the  $s^{th}$  style, and  $\boldsymbol{\epsilon}_{ijs}$  is additive normal noise with a covariance  $\Sigma_s$  that also depends on the style. When the noise covariances are spherical, this model is a probabilistic form of *canonical correlation analysis*. When the noise is diagonal, it is known as *tied factor analysis*. We will use the generic term *asymmetric bilinear model* to cover both situations.

Equation 18.30 is easy to parse; for a given individual, the identity  $\mathbf{h}_i$  is constant. The data are explained as a weighted linear sum of basis functions, where the weights determine the identity. However, the basis functions (and other aspects of the model) are now contingent on the style.

We can write the model in probabilistic terms as

$$\begin{aligned} Pr(s) &= \text{Cat}_s[\boldsymbol{\lambda}] \\ Pr(\mathbf{h}_i) &= \text{Norm}_{\mathbf{h}_i}[\mathbf{0}, \mathbf{I}] \\ Pr(\mathbf{x}_{ijs} | \mathbf{h}_i, s) &= \text{Norm}_{\mathbf{x}_{ijs}}[\boldsymbol{\mu}_s + \Phi_s \mathbf{h}_i, \Sigma_s], \end{aligned} \quad (18.31)$$

Problem 18.8



**Figure 18.14** Asymmetric bilinear model with two styles. a) We draw a hidden variable  $\mathbf{h}_i$  from the prior and use this to weight basis functions  $\Phi_1$  (one basis function  $\phi_1$  shown). The result is added to the mean  $\mu_1$ . b) We use the same value of  $\mathbf{h}_i$  to weight a second set of basis functions  $\Phi_2$  and add the result to  $\mu_2$ . c) We add normally distributed noise with a diagonal covariance  $\Sigma_s$  that depends on the style. d-f) When we repeat this procedure, it produces one normal distribution per style. The data within each style are structured so that nearby points have the same identity (color) and identities that are close in one cluster are also close in the other cluster.

where  $\lambda$  contains parameters that determine the probability of observing data in each style. Figure 18.14 demonstrates ancestral sampling from this model. If we marginalize over the identity parameter  $\mathbf{h}$  and the style parameter  $s$ , the overall data distribution (without regard to the structure of the style clusters) is a mixture of factor analyzers,

$$Pr(\mathbf{x}) = \sum_{s=1}^S \lambda_s \text{Norm}_{\mathbf{x}}[\mu_s, \Phi_s \Phi_s^T + \Sigma_s]. \quad (18.32)$$

### 18.4.1 Learning

Algorithm 18.3

For simplicity, we will assume that the styles of each training example are known and so it is also trivial to estimate the categorical parameters  $\lambda$ . As for the previous models in this chapter, we employ the EM algorithm.

In the E-step, we compute a posterior distribution over the hidden variable  $\mathbf{h}_i$  that represents the identity, using all of the training data for that individual regardless of the style. Employing Bayes' rule we have

$$Pr(\mathbf{h}_i | \mathbf{x}_{i\bullet\bullet}) = \frac{\prod_{j=1}^J \prod_{s=1}^S Pr(\mathbf{x}_{ijs} | \mathbf{h}_i) Pr(\mathbf{h}_i)}{\int \prod_{j=1}^J \prod_{s=1}^S Pr(\mathbf{x}_{ijs} | \mathbf{h}_i) Pr(\mathbf{h}_i) d\mathbf{h}_i}, \quad (18.33)$$

where  $\mathbf{x}_{i\bullet\bullet} = \{\mathbf{x}_{ijs}\}_{j,s=1}^{J,S}$  denotes all the data associated with the  $i^{th}$  individual.

One way to compute this is to write a compound generative equation for  $\mathbf{x}_{i\bullet\bullet}$ . For example, with  $J = 2$  images at each of  $S = 2$  styles we would have

$$\begin{bmatrix} \mathbf{x}_{i11} \\ \mathbf{x}_{i12} \\ \mathbf{x}_{i21} \\ \mathbf{x}_{i22} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \\ \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix} + \begin{bmatrix} \boldsymbol{\Phi}_1 \\ \boldsymbol{\Phi}_2 \\ \boldsymbol{\Phi}_1 \\ \boldsymbol{\Phi}_2 \end{bmatrix} \mathbf{h}_i + \begin{bmatrix} \boldsymbol{\epsilon}_{11} \\ \boldsymbol{\epsilon}_{12} \\ \boldsymbol{\epsilon}_{21} \\ \boldsymbol{\epsilon}_{22} \end{bmatrix}, \quad (18.34)$$

which has the same form as the identity subspace model,  $\mathbf{x}'_{ij} = \boldsymbol{\mu}' + \boldsymbol{\Phi}' \mathbf{h}_i + \boldsymbol{\epsilon}'_{ij}$ . We can hence compute the posterior distribution using equation 18.9 and extract the expected values needed for the M-step using equation 18.10.

In the M-step we update the parameters  $\boldsymbol{\theta}_s = \{\boldsymbol{\mu}_s, \boldsymbol{\Phi}_s, \boldsymbol{\Sigma}_s\}$  for each style separately using all of the relevant data. This gives the updates

$$\begin{aligned} \hat{\boldsymbol{\mu}}_s &= \frac{\sum_{i=1}^I \sum_{j=1}^J \mathbf{x}_{ijs}}{IJ} \\ \hat{\boldsymbol{\Phi}}_s &= \left( \sum_{i=1}^I \sum_{j=1}^J (\mathbf{x}_{ijs} - \hat{\boldsymbol{\mu}}_s) \mathbf{E}[\mathbf{h}_i]^T \right) \left( J \sum_{i=1}^I \mathbf{E}[\mathbf{h}_i \mathbf{h}_i^T] \right)^{-1} \\ \hat{\boldsymbol{\Sigma}}_s &= \frac{1}{IJ} \sum_{i=1}^I \sum_{j=1}^J \text{diag} \left[ (\mathbf{x}_{ijs} - \hat{\boldsymbol{\mu}}_s)^T (\mathbf{x}_{ijs} - \hat{\boldsymbol{\mu}}_s) - \hat{\boldsymbol{\Phi}}_s \mathbf{E}[\mathbf{h}_i] \mathbf{x}_{ijs}^T \right]. \end{aligned} \quad (18.35)$$

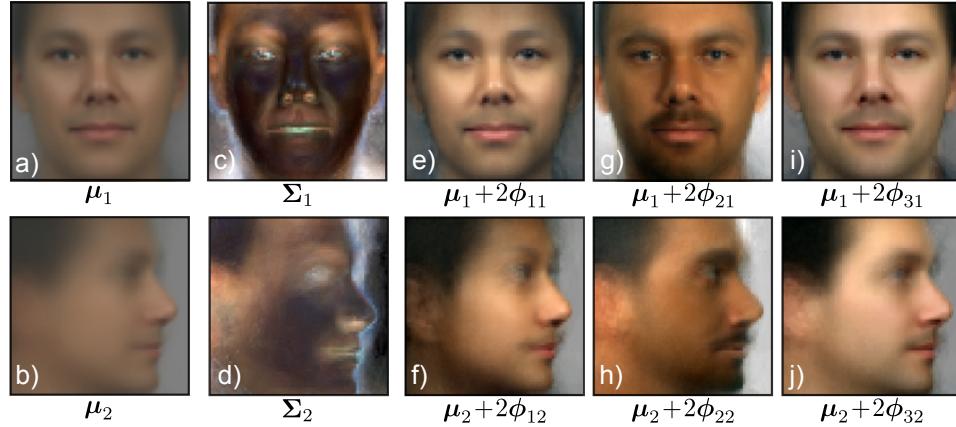
As usual, we iterate these two steps until the system converges and the log likelihood ceases to improve. Figure 18.15 shows examples of the learned parameters for a data set that includes faces at two poses.

### 18.4.2 Inference

There are a number of possible forms of inference in this model. These include:

1. Given  $\mathbf{x}$ , infer the style  $s \in \{1, \dots, S\}$ .
2. Given  $\mathbf{x}$ , infer the parameterized identity  $\mathbf{h}$ .
3. Given  $\mathbf{x}_1$  and  $\mathbf{x}_2$ , infer whether they have the same identity or not.
4. Given  $\mathbf{x}_1$  in style  $s_1$ , translate the style to  $s_2$  to create  $\hat{\mathbf{x}}_2$ .

We will consider each in turn.



**Figure 18.15** Learned parameters of asymmetric bilinear model with two styles (frontal and profile faces). This model was learned from one  $70 \times 70$  image of each style in 200 individuals from the FERET data set. a,b) Mean vector for each style. c-d) Diagonal covariance for each style. e-f) Varying first basis function in each style (notation  $\phi_{ks}$  denotes  $k^{th}$  basis function of  $s^{th}$  style). g-h) Varying second basis function in each style. i-l) Varying third basis function in each style. Manipulating the two sets of basis functions in the same way produces images that look like the same person, viewed in each of the styles. Adapted from Prince *et al.* (2008). ©2008 IEEE.

### Inferred style

The likelihood of the data given style  $s$  but regardless of identity  $\mathbf{h}$  is

$$\begin{aligned}
 Pr(\mathbf{x}|s) &= \int Pr(\mathbf{x}|\mathbf{h}, s)Pr(\mathbf{h}) d\mathbf{h} \\
 &= \int \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}_s + \boldsymbol{\Phi}_s \mathbf{h}, \boldsymbol{\Sigma}_s] Pr(\mathbf{h}) d\mathbf{h} \\
 &= \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}_s, \boldsymbol{\Phi}_s \boldsymbol{\Phi}_s^T + \boldsymbol{\Sigma}_s].
 \end{aligned} \tag{18.36}$$

The posterior  $Pr(s|\mathbf{x})$  over style  $s$  can be computed by combining this likelihood with the prior  $Pr(s)$  using Bayes' rule. The prior for style  $s$  is given by

$$Pr(s) = \text{Cat}_s[\boldsymbol{\lambda}] \tag{18.37}$$

### Inferred identity

The likelihood of the data for a fixed identity  $\mathbf{h}$  but regardless of style  $s$  is

$$\begin{aligned} Pr(\mathbf{x}|\mathbf{h}) &= \sum_{s=1}^S Pr(\mathbf{x}|\mathbf{h}, s)Pr(s) \\ &= \sum_{s=1}^S \text{Norm}_x[\boldsymbol{\mu}_s + \boldsymbol{\Phi}_s \mathbf{h}, \boldsymbol{\Sigma}_s] \lambda_s. \end{aligned} \quad (18.38)$$

The posterior over identity can now be combined with the prior  $Pr(\mathbf{h})$  using Bayes' rule and is given by

$$Pr(\mathbf{h}|\mathbf{x}) = \sum_{s=1}^S \lambda_s \text{Norm}_{\mathbf{h}_i}[(\boldsymbol{\Phi}_s^T \boldsymbol{\Sigma}_s^{-1} \boldsymbol{\Phi}_s + \mathbf{I})^{-1} \boldsymbol{\Phi}_s^T \boldsymbol{\Sigma}_s^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_s), (\boldsymbol{\Phi}_s^T \boldsymbol{\Sigma}_s^{-1} \boldsymbol{\Phi}_s + \mathbf{I})]. \quad (18.39)$$

Note that this posterior distribution is a mixture of Gaussians, with one component for each possible style.

### Identity matching

Given two examples  $\mathbf{x}_1, \mathbf{x}_2$ , compute the posterior probability that they have the same identity, even though they may be viewed in different styles. We will initially assume the styles are known and are  $s_1$  and  $s_2$ , respectively. We first build the compound model

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mu}_{s_1} \\ \boldsymbol{\mu}_{s_2} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Phi}_{s_1} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Phi}_{s_2} \end{bmatrix} \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \end{bmatrix} + \begin{bmatrix} \boldsymbol{\epsilon}_1 \\ \boldsymbol{\epsilon}_2 \end{bmatrix}, \quad (18.40)$$

which represents the case where the identities differ ( $w = 0$ ). We compute the likelihood by noting that this has the form  $\mathbf{x}' = \boldsymbol{\mu}' + \boldsymbol{\Phi}' \mathbf{h}' + \boldsymbol{\epsilon}'$  of the original factor analyzer, and so we can write

$$Pr(\mathbf{x}'|w = 0) = \text{Norm}_{\mathbf{x}'}[\boldsymbol{\mu}', \boldsymbol{\Phi}' \boldsymbol{\Phi}'^T + \boldsymbol{\Sigma}'], \quad (18.41)$$

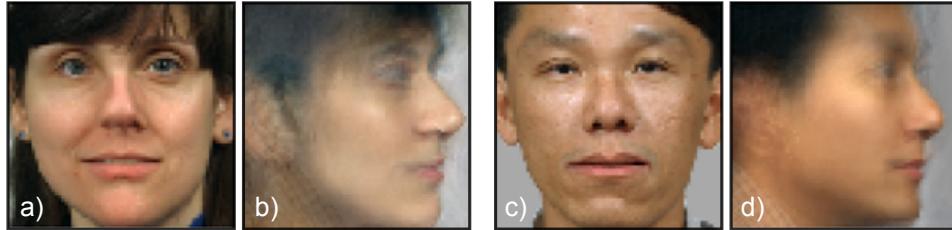
where  $\boldsymbol{\Sigma}'$  is a diagonal matrix containing the (diagonal) covariances of the elements of  $\boldsymbol{\epsilon}'$  (as in equation 18.16).

Similarly, we can build a system for when the identities match ( $w=1$ )

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mu}_{s_1} \\ \boldsymbol{\mu}_{s_2} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\Phi}_{s_1} \\ \boldsymbol{\Phi}_{s_2} \end{bmatrix} \mathbf{h}_{12} + \begin{bmatrix} \boldsymbol{\epsilon}_1 \\ \boldsymbol{\epsilon}_2 \end{bmatrix}, \quad (18.42)$$

and compute its likelihood  $Pr(\mathbf{x}'|w = 1)$  in the same way. The posterior probability  $Pr(w = 1|\mathbf{x}')$  can then be computed using Bayes' rule.

If we do not know the styles, then each likelihood term will become a mixture of Gaussians where each component has the form of equation 18.41. There will be one component for every one of the  $S^2$  combinations of the two styles. The mixing weights will be given by the probability of observing that combination so that  $Pr(s_1 = m, s_2 = n) = \lambda_m \lambda_n$ .



**Figure 18.16** Style translation based on asymmetric bilinear model from figure 18.15. a) Original face in style 1 (frontal). b) Translated to style 2 (profile). c-d) A second example. Adapted from Prince *et al.* (2008). ©2008 IEEE.

### Style translation

Finally, let us consider style translation. Given observed data  $\mathbf{x}$  in style  $s_1$  translate to the style  $s_2$  while maintaining the same identity. A simple way to get a point estimate of the translated styles is to first estimate the identity variable  $\mathbf{h}$  based on the observed image  $\mathbf{x}_{s_1}$ . To do this, we compute the posterior distribution

Algorithm 18.4

$$Pr(\mathbf{h}|\mathbf{x}, s_1) = \text{Norm}_{\mathbf{h}_i}[(\Phi_{s_1}^T \Sigma_{s_1}^{-1} \Phi_{s_1} + \mathbf{I})^{-1} \Phi_{s_1}^T \Sigma_{s_1}^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_{s_1}), (\Phi_{s_1}^T \Sigma_{s_1}^{-1} \Phi_{s_1} + \mathbf{I})]. \quad (18.43)$$

and then set  $\mathbf{h}$  to the MAP estimate

$$\begin{aligned} \hat{\mathbf{h}} &= \underset{\mathbf{h}}{\operatorname{argmax}} [Pr(\mathbf{h}|\mathbf{x}, s_1)] \\ &= (\Phi_{s_1}^T \Sigma_{s_1}^{-1} \Phi_{s_1} + \mathbf{I})^{-1} \Phi_{s_1}^T \Sigma_{s_1}^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_{s_1}), \end{aligned} \quad (18.44)$$

which is just the mean of this distribution.

We then generate the image in the second style as

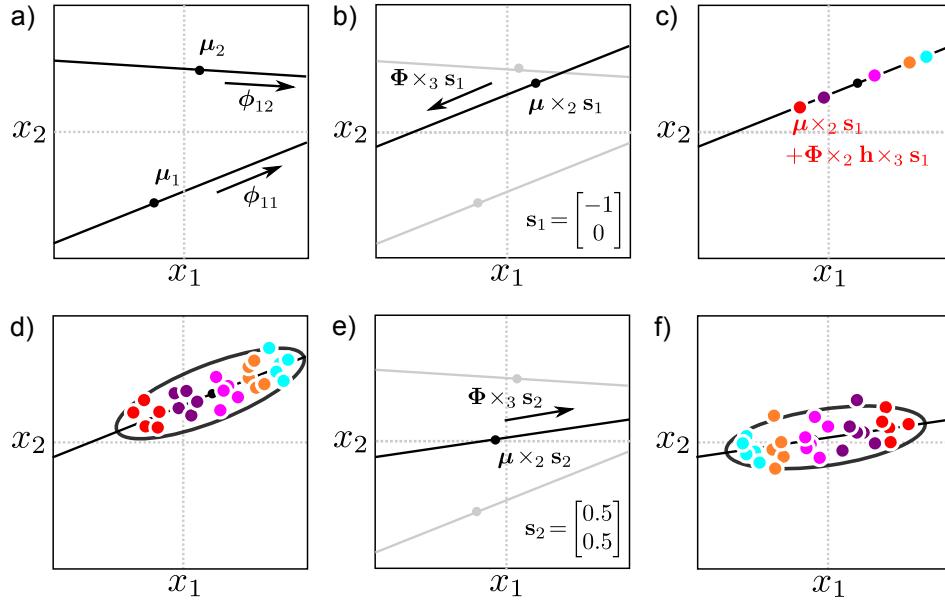
$$\mathbf{x}_{s_2} = \boldsymbol{\mu}_{s_2} + \Phi_{s_2} \hat{\mathbf{h}}, \quad (18.45)$$

which is the original generative equation with the noise term omitted.

## 18.5 Symmetric bilinear and multilinear models

As the name suggests, symmetric bilinear models treat both style and identity equivalently. Both are continuous variables, and the model is linear in each. To write these models in a compact way, it is necessary to introduce tensor product notation. In this notation (see appendix C.3), the generative equation for the subspace identity model (equation 18.6) is written as

$$\mathbf{x}_{ij} = \boldsymbol{\mu} + \Phi \times_2 \mathbf{h}_i + \boldsymbol{\epsilon}_{ij}, \quad (18.46)$$



**Figure 18.17** Ancestral sampling from symmetric bilinear model, with 1D identity and 2D style. a) In this model each style dimension consists of a 1D subspace identity model with a 1D subspace. b) For a given style vector  $s_1$ , we weight these models to create a new subspace identity model. c) We then generate from this by weighting the factor by the hidden variable  $h$  and d) adding noise to generate different instances of this identity in this style. e) A different weighting induced by the style vector  $s_2$  creates a different subspace model. f) Generation from the resulting subspace identity model.

where the notation  $\Phi \times_2 h_i$  means take the dot product of the second dimension of  $\Phi$  with  $h_i$ . Since  $\Phi$  was originally a 2D matrix, this returns a vector.

In the symmetric bilinear model, the generative equation for the  $j^{th}$  example of the  $i^{th}$  identity in the  $k^{th}$  style is given by

$$\mathbf{x}_{ijk} = \boldsymbol{\mu} + \Phi \times_2 h_i \times_3 s_k + \epsilon_{ijk}, \quad (18.47)$$

where  $h_i$  is a 1D vector representing the identity,  $s_k$  is a 1D vector representing the style and  $\Phi$  is now a 3D tensor. In the expression  $\Phi \times_2 h_i \times_3 s_k$  we take the dot product with two of these three dimensions, leaving a column vector as desired.

In probabilistic form, we write

$$\begin{aligned} Pr(h_i) &= \text{Norm}_{h_i}[\mathbf{0}, \mathbf{I}] \\ Pr(s_k) &= \text{Norm}_{s_k}[\mathbf{0}, \mathbf{I}] \\ Pr(\mathbf{x}_{ijk} | h_i, s_k) &= \text{Norm}_{\mathbf{x}_{ijk}}[\boldsymbol{\mu} + \Phi \times_2 h_i \times_3 s_k, \Sigma]. \end{aligned} \quad (18.48)$$

For a fixed style vector  $s_k$  this model is exactly a subspace identity model with

hidden variable  $\mathbf{h}_i$ . The choice of style determines the factors by weighting a set of basis functions to create them. It is also possible to make the mean vector depend on the style by using the model

$$\mathbf{x}_{ijk} = \boldsymbol{\mu} \times_2 \mathbf{s}_k + \boldsymbol{\Phi} \times_2 \mathbf{h}_i \times_3 \mathbf{s}_k + \boldsymbol{\epsilon}_{ijk}, \quad (18.49)$$

where  $\boldsymbol{\mu}$  is now a matrix with basis functions in the columns that are weighted by the style  $\mathbf{s}$ . Ancestral sampling from this model is illustrated in figure 18.17.

It is instructive to compare the asymmetric and symmetric bilinear models. In the asymmetric bilinear model, there were a discrete number of styles each of which generated data that individually looked like a subspace identity model, but the model induced a relationship between the position of an identity in one style cluster and in another. In the symmetric bilinear model, there is a continuous family of styles, that produces a continuous family of subspace identity models. Again the model induces a relationship between the position of an identity in each.

Up to this point, we have described the model as a subspace identity model for fixed style. The model is symmetric, and so it is possible to reverse the roles of the variables. For a fixed identity, the model looks like a subspace model where the basis functions are weighted by the variable  $\mathbf{s}_k$ . In other words, the model is linear in both sets of hidden variables when the other is fixed. It is *not*, however, simultaneously linear in both  $\mathbf{h}$  and  $\mathbf{s}$  together. These variables have a nonlinear interaction, and overall the model is nonlinear.

### 18.5.1 Learning

Unfortunately, is not possible to compute the likelihood of the bilinear model in closed form; we cannot simultaneously marginalize over both sets of hidden variables and compute

$$Pr(\mathbf{x}_{ijk}|\boldsymbol{\theta}) = \iint Pr(\mathbf{x}_{ijk}, \mathbf{h}_i, \mathbf{s}_k | \boldsymbol{\theta}) d\mathbf{h}_i d\mathbf{s}_k, \quad (18.50)$$

where  $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Phi}, \boldsymbol{\Sigma}\}$  represents all of the unknown parameters. The usual approach for learning models with hidden variables is to use the EM algorithm, but this is no longer suitable because we cannot compute the joint posterior distribution over the hidden variables  $Pr(\mathbf{h}_i, \mathbf{s}_k | \{\mathbf{x}_{ijk}\}_{j=1}^J)$  in closed form either.

For the special case of spherical additive noise  $\boldsymbol{\Sigma} = \sigma^2 \mathbf{I}$ , and complete data (where we see  $J$  examples of each of the  $I$  individuals in each of  $K$  styles), it is possible to solve for the parameters in closed form using a method similar to that used for PPCA (section 17.5.1). This technique relies on the *N-mode singular value decomposition*, which is the generalization of the SVD to higher dimensions.

For models with diagonal noise, we can approximate by maximizing over one of the hidden variables rather than marginalizing over them. For example, if we maximize over the style parameters so that



**Figure 18.18** Translation of styles using symmetric bilinear model. a) We learn the model from a set of images, where the styles (rows) and identities (columns) are known. Then we are given a new image which has a previously unseen identity and style. b) The symmetric bilinear model can estimate the identity parameters and simulate the image in new styles, or c) estimate the style parameters and simulate new identities. In both cases, the simulated results are close to the ground truth. Adapted from Tenenbaum & Freeman (2000). ©2000 MIT Press.

$$\boldsymbol{\theta} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[ \sum_{k=1}^K \max_{\mathbf{s}_k} \left[ \sum_{i=1}^I \sum_{j=1}^J \log \left[ \int Pr(\mathbf{x}_{ijk}, \mathbf{h}_i, \mathbf{s}_k | \boldsymbol{\theta}) d\mathbf{h}_i \right] \right] \right], \quad (18.51)$$

then the remaining model is linear in the hidden variables  $\mathbf{h}_i$ . It would hence be possible to apply an alternating approach in which we first fix the styles and learn the parameters with the EM algorithm and then fix the parameters and update the style parameters using optimization.

### 18.5.2 Inference

Various forms of inference are possible, including all of those discussed for the asymmetric bilinear model. We can, for example, make decisions about whether identities match by comparing different compound models. It is not possible to marginalize over both the identity and style variables in these models, and so we maximize over the style variable in a similar manner to the learning procedure. Similarly, we can translate from one style to another by estimating the identity

variable  $\mathbf{h}$  (and the current style variable  $\mathbf{s}$  if unknown) from the observed data. We then use the generative equation with a different style vector  $\mathbf{s}$  to simulate a new example in a different style.

The symmetric bilinear model has a continuous parameterization of style, and so it is also possible to perform a new translation task: given an example whose identity we have not previously seen *and* whose style we have not previously seen, we can translate either its style or identity as required. We first compute the current identity and style which can be done using a nonlinear optimization approach,

$$\hat{\mathbf{h}}, \hat{\mathbf{s}} = \underset{\mathbf{h}, \mathbf{s}}{\operatorname{argmax}} [Pr(\mathbf{x}|\boldsymbol{\theta}, \mathbf{h}, \mathbf{s})Pr(\mathbf{h})Pr(\mathbf{s})]. \quad (18.52)$$

Then we simulate new examples using the generative equation, modifying the style  $\mathbf{s}$  or identity  $\mathbf{h}$  as required. An example of this is shown in figure 18.18.

### 18.5.3 Multi-linear models

The symmetric bilinear model can be extended to create *multi-linear* or *multi-factor* models. For example, we might describe our data as depending on three hidden variables,  $\mathbf{h}, \mathbf{s}$  and  $\mathbf{t}$ , so the generative equation becomes

$$\mathbf{x}_{ijkl} = \boldsymbol{\mu} + \boldsymbol{\Phi} \times_2 \mathbf{h}_i \times_3 \mathbf{s}_k \times_4 \mathbf{t}_l + \boldsymbol{\epsilon}_{ijkl}, \quad (18.53)$$

and now the tensor  $\boldsymbol{\Phi}$  becomes four-dimensional. As in the symmetric bilinear model, it is not possible to marginalize over all of the hidden variables in closed form, and this constrains the possible methods for learning and inference.

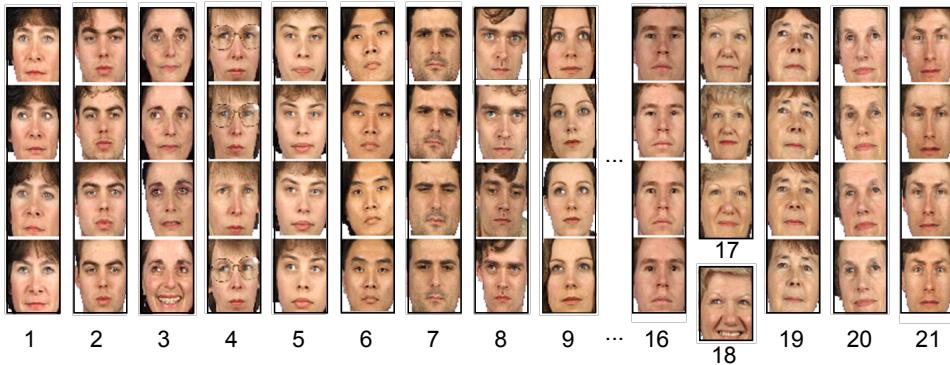
## 18.6 Applications

We have illustrated many of the models in this chapter with examples from face recognition. In this section, we will describe face recognition in more detail and talk about some of the practicalities of building a recognition system. Subsequently, we will discuss an application in which a visual texture is compactly represented as a multi-linear model. Finally, we will describe a nonlinear version of the multilinear model that can be used to synthesize animation data.

### 18.6.1 Face recognition

To provide a more concrete idea of how well these algorithms work in practice, we will discuss a recent application in detail. Li *et al.* (2011) present a recognition system based on probabilistic linear discriminant analysis.

Eight keypoints on each face were identified, and the images were registered using a piecewise affine warp. The final image size was  $400 \times 400$ . Feature vectors were extracted from the area of the image around each keypoint. The feature



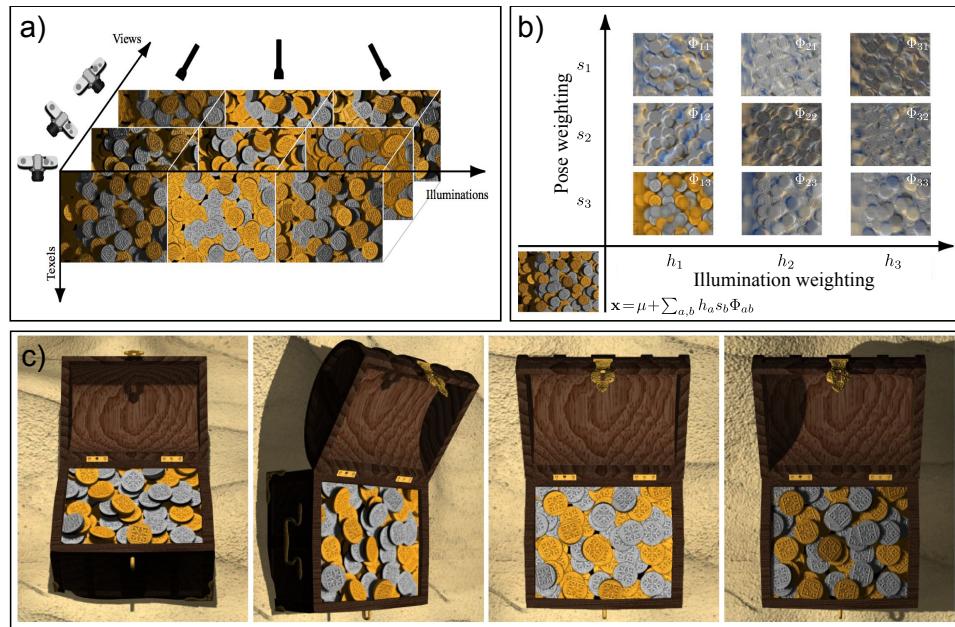
**Figure 18.19** Face clustering results from Li *et al.* (2011). The algorithm was presented with a set of 80 faces consisting of 4 pictures each of 20 people and clustered these almost perfectly; it correctly found 19 of these 20 groups of images but erroneously divided the data from one individual into two separate clusters. The algorithm works well for these frontal faces despite changes in expression (e.g., cluster 3), changes in hairstyle (e.g., cluster 9) and the addition or removal of glasses (e.g., cluster 4). Adapted from Li *et al.* (2011). ©2011 IEEE.

vectors consisted of image gradients at eight orientations and three scales at points in a  $6 \times 6$  grid centered on the keypoint. A separate recognition model was built for each keypoint and these were treated as independent in the final recognition decision.

The system was trained using only the first 195 individuals from the XM2VTS database and signal and noise subspaces of size 64. In testing, the algorithm was presented with 80 images taken from the remaining 100 individuals in the database and was required to cluster them into groups according to identity. There may be 80 images of the same person or 80 images of different people or any permutation between these extremes.

In principle it is possible to calculate the likelihood for each possible clustering of the data. Unfortunately, in practice there are far too many possible configurations. Hence, Li *et al.* (2011) adopted a greedy agglomerative strategy. They started with the hypothesis that there are 80 different individuals. They considered merging all pairs of individuals and chose the combination that increased the likelihood the most. They repeated this process until the likelihood could not be improved. Example clustering results are illustrated in figure 18.19 and are typical of state-of-the-art recognition algorithms; they cope relatively easily with frontal faces under controlled lighting conditions.

However, for more natural images the same algorithms struggle even with more sophisticated preprocessing. For example, Li *et al.* (2011) applied the PLDA model to face verification in the ‘Labeled Faces in the Wild’ dataset (Huang *et al.* 2007b), which contains images of famous people collected from the internet, and obtained an equal-error rate of approximately 10%. This is typical of the state of the art



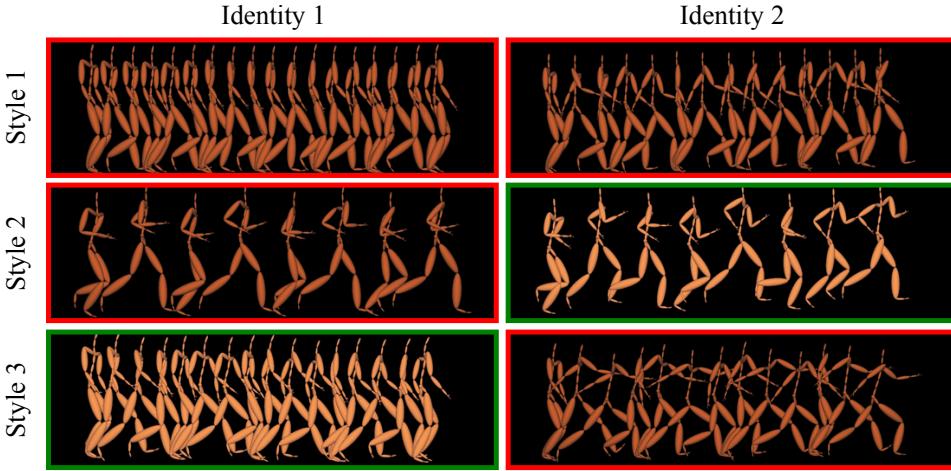
**Figure 18.20** Tensor textures. a) The training set consists of renderings of a set of coins viewed from several different directions and with several different lighting directions. b) A new texture (bottom left corner) is computed as a weighted sum of the learned basis functions stored in the 3D tensor  $\Phi$ . c) Several frames of a video sequence in which the texture is synthesized appropriately from the model. Adapted from Vasilescu & Terzopoulos (2004). ©2003 ACM.

at the time of writing and is much worse than for faces captured under controlled conditions.

### 18.6.2 Modeling texture

The interaction of light with a surface can be described by the bi-directional reflectance distribution function; essentially, this describes the outgoing light at each angle from the surface given incoming light at a particular angle to the surface. The bi-directional texture function (BTF) generalizes this model to also depend on the 2D position on the surface of the object. If we know the BTF, then we know how a textured surface will appear from every angle and under every lighting combination.

This function could be approximated by taking several thousand images of the surface viewed from different angles and under different lighting conditions. However, the resulting data are clearly highly redundant. Vasilescu & Terzopoulos (2004) described the BTF using a multi-linear model known as ‘TensorTextures’.



**Figure 18.21** Multi-factor GPLVM applied to animation synthesis. A three-factor model was learned with factors for identity, style, and position in the gait sequence. The figures shows training data (red boxes) and synthesized data from the learned model (green boxes). In each case, it manages to simulate the style and identity well. Adapted from Wang *et al.* (2007).

It contained style factors that represent the lighting and viewing directions (figure 18.20a-b). Although both of these quantities are naturally 2D, they represented them as vectors of size 21 and 37, respectively, where each training example was transformed to one coordinate axis.

Figure 18.20c shows images generated from the TensorTextures model; the hidden variables associated in each style were chosen by linearly interpolating between the hidden variables of nearby training examples, and the image was synthesized without the addition of noise. It can be seen that the TensorTextures model has learned a compact representation of the appearance variation under changes in viewpoint and illumination, including complex effects due to self-occlusion, inter-reflection, and self-shadowing.

### 18.6.3 Animation synthesis

Wang *et al.* (2007) developed a multi-factor model that depended nonlinearly on the identity and style components. Their approach was based on the Gaussian process latent variable model; the style and identity factors were transformed through nonlinear functions before weighting the tensor  $\Phi$ . In this type of model, the likelihood might be given by:

$$Pr(\mathbf{x}_{ijk}|\Sigma, \mathbf{h}_i, \mathbf{s}_k) = \iint \text{Norm}_{\mathbf{x}_{ijk}}[\boldsymbol{\mu} + \Phi \times_2 \mathbf{f}[\mathbf{h}_i] \times_3 \mathbf{g}[\mathbf{s}_k], \Sigma] d\Phi d\boldsymbol{\mu}, \quad (18.54)$$

where  $\mathbf{f}[\bullet]$  and  $\mathbf{g}[\bullet]$  are nonlinear functions that transform the identity and style parameters respectively. In practice, the tensor  $\Phi$  can be marginalized out of the final likelihood computation along with the overall mean  $\mu$ . This model can be expressed in terms of inner products of the identity and style parameters and can hence be kernelized. It is known as the *multi-factor Gaussian process latent variable model* or the *multi-factor GPLVM*.

Wang *et al.* (2007) used this model to describe human locomotion. A single pose was described as an 89-dimensional vector, which consisted of 43 joint angles, the corresponding 43 angular velocities and the global translational velocity. They built a model consisting of three factors; the identity of the individual, the gait of locomotion (walk, stride, or run), and the current state in the motion sequence. Each was represented as a 3D vector. They learned the model from human capture data using an RBF kernel.

Figure 18.21 shows the results of style translation in this model. The system can predict realistic body poses in styles that have not been observed in the training data. Since the system is generative, it can also be used to synthesize novel motion sequences for a given individual in which the gait varies over time.

## Discussion

In this chapter, we have examined a number of models that describe image data as a function of style and content variables. During training, these variables are forced to take the same value for examples where we know the style or content are the same. We have demonstrated a number of different forms of inference including identity recognition and style translation.

## Notes

**Face recognition:** For a readable introduction to face recognition consult Chellappa *et al.* (2010). For more details, consult the review paper by Zhao *et al.* (2003) or the edited book by Li & Jain (2005).

**Subspace methods for face recognition:** Turk & Pentland (2001) developed the *eigenfaces* method in which the pixel data were reduced in dimension by linearly projecting it onto a subspace corresponding to the principal components of the training data. The decision about whether two faces matched or not was based on the distance between these low dimensional representations. This approach quickly supplanted earlier techniques that had been based on measuring the relative distance between facial features (Brunelli & Poggio 1993).

The subsequent history of face recognition has been dominated by other *subspace methods*. Researchers have variously investigated the choice of basis functions (e.g., Bartlett *et al.* 1998; Belhumeur *et al.* 1997; He *et al.* 2005; Cai *et al.* 2007), analogous nonlinear techniques (Yang 2002), and the choice of distance metric (Perlibakas 2004). The relationship between different subspace models is discussed in (Wang & Tang 2004b). A review of subspace methods (without particular reference to face recognition) can be found in De La Torre (2011).

**Linear discriminant analysis:** A notable sub-category of these subspace methods consists of approaches based on linear discriminant analysis (LDA). The Fisherfaces algorithm (Belhumeur *et al.* 1997) projected face data to a space where the ratio of between-individual variation to within-individual variation was maximized. Fisherfaces is limited to directions in which at least some within-individual variance has been observed (the small-sample problem). The null-space LDA approach (Chen *et al.* 2000) exploited the signal in the remaining subspace. The Dual-Space LDA approach (Wang & Tang 2004a) combined these two sources of information.

**Probabilistic approaches:** The identity models in this chapter are probabilistic re-interpretations of earlier non-probabilistic techniques. For example, the subspace identity model is very similar to the eigenfaces algorithm (Turk & Pentland 2001) and probabilistic LDA is very similar to the Fisherfaces algorithm (Belhumeur *et al.* 1997). For more details about these probabilistic versions, consult Li *et al.* (2011) and Ioffe (2006) who presented a slightly different probabilistic LDA algorithm. There have also been many other probabilistic approaches to face recognition (Liu & Wechsler 1998; Moghaddam *et al.* 2000; Wang & Tang 2003; Zhou & Chellappa 2004).

**Alignment and pose changes:** An important part of most face recognition pipelines is to accurately identify facial features so that either (i) the face image can be aligned to a fixed template or (ii) the separate parts of the face can be treated independently (Wiskott *et al.* 1997; Moghaddam & Pentland 1997). Common methods to identify facial features include the use of active shape models (Edwards *et al.* 1998) or pictorial structures (Everingham *et al.* 2006; Li *et al.* 2010).

For larger pose changes, it may not be possible to warp the face accurately to a common template and explicit methods are required to compare the faces. These include fitting 3D morphable models to the images and then simulating a frontal image from a non-frontal one (Blanz *et al.* 2005), predicting the face at one pose from another using statistical methods (Gross *et al.* 2002; Lucey & Chen 2006) or using the tied factor analysis model discussed in this chapter (Prince *et al.* 2008). A review of face recognition across large pose changes can be found in Zhang & Gao (2009).

**Current work in face recognition:** It is now considered that face recognition for frontal faces in constant lighting and with no pose or expression changes is almost solved. Earlier databases that have these characteristics (e.g., Messer *et al.* 1999; Phillips *et al.* 2000) have now been supplanted by test databases containing more variation (Huang *et al.* 2007b).

Several recent trends have emerged in face recognition. These include a resurgence of interest in discriminative models (e.g., Wolf *et al.* 2009; Taigman *et al.* 2009; Kumar *et al.* 2009), learning metrics to discriminate identity (e.g., Nowak & Jurie 2007; Ferencz *et al.* 2008; Guillaumin *et al.* 2009; Nguyen & Bai 2010), the use of sparse representations (e.g., Wright *et al.* 2009), and a strong interest in preprocessing techniques. In particular, many current methods are based on Gabor features (e.g., Wang & Tang 2003), local binary patterns (Ojala *et al.* 2002; Ahonen *et al.* 2004), three-patch local binary patterns (Wolf *et al.* 2009), or SIFT features (Lowe 2004). Some of the most successful methods combine or select several different preprocessing techniques (Li *et al.* 2011; Taigman *et al.* 2009; Pinto & Cox 2011).

**Bilinear and multi-linear models:** Bilinear models were introduced to computer vision by Tenenbaum & Freeman (2000) and multi-linear models were explored by Vasilescu & Terzopoulos (2002). Kernelized multi-linear models were discussed by Li *et al.* (2005) and Wang *et al.* (2007). An alternative approach to nonlinear multi-factor models was presented in Elgammal & Lee (2004). The most common use of bilinear and multi-linear models in computer vision has been for face recognition in situations where the capture conditions vary (Grimes *et al.* 2003; Lee *et al.* 2005; Cuzzolin 2006; Prince *et al.* 2008).

## Problems

**Problem 18.1** Prove that the posterior distribution over the hidden variable in the subspace identity model is as given in equation 18.9.

**Problem 18.2** Show that the M-step updates for the subspace identity model are as given in equation 18.11.

**Problem 18.3** Develop a closed form solution for learning the parameters  $\{\boldsymbol{\mu}, \boldsymbol{\Phi}, \sigma^2\}$  of a subspace identity model where the noise is spherical:

$$Pr(\mathbf{x}_{ij}) = \text{Norm}_{\mathbf{x}_{ij}}[\boldsymbol{\mu}, \boldsymbol{\Phi}\boldsymbol{\Phi}^T + \sigma^2\mathbf{I}].$$

Hint: Assume you have exactly  $J = 2$  examples of each of the  $I$  training images and base your solution on probabilistic PCA.

**Problem 18.4** In a face clustering problem, how many possible models of the data are there with 2,3,4,10, and 100 faces?

**Problem 18.5** An alternative approach to face verification using the identity subspace model is to compute the probability of the observed data  $\mathbf{x}_1$  and  $\mathbf{x}_2$  under the models:

$$\begin{aligned} Pr(\mathbf{x}_1, \mathbf{x}_2 | w=0) &= Pr(\mathbf{x}_1)Pr(\mathbf{x}_2) \\ Pr(\mathbf{x}_1, \mathbf{x}_2 | w=1) &= Pr(\mathbf{x}_1)Pr(\mathbf{x}_2|\mathbf{x}_1). \end{aligned}$$

Write down expressions for the marginal probability terms  $Pr(\mathbf{x}_1)$ ,  $Pr(\mathbf{x}_2)$  and the conditional probability  $Pr(\mathbf{x}_2|\mathbf{x}_1)$ . How could you use these expressions to compute the posterior  $Pr(w|\mathbf{x}_1, \mathbf{x}_2)$  over the world state?

**Problem 18.6** Propose a version of the subspace identity model that is robust to outliers in the training data.

**Problem 18.7** Moghaddam *et al.* (2000) took a different probabilistic approach to face verification. They took the difference  $\mathbf{x}_\Delta = \mathbf{x}_2 - \mathbf{x}_1$  and modeled the likelihoods of this vector  $Pr(\mathbf{x}_\Delta|w = 0)$  and  $Pr(\mathbf{x}_\Delta|w = 1)$  when the two faces match or don't. Propose expressions for these likelihoods and discuss learning and inference in this model. Identify one possible disadvantage of this model.

**Problem 18.8** Develop a model that combines the advantages of PLDA and the asymmetric bilinear model; it should be able to model the within-individual covariance with a subspace, but also be able to compare data between disparate styles. Discuss learning and inference in your model.

**Problem 18.9** In the asymmetric bilinear model, how would you infer whether the style of two examples is the same or not, regardless of whether the images matched?



# Chapter 19

## Temporal models

This chapter concerns temporal models and tracking. The goal is to infer a sequence of world states  $\{\mathbf{w}_t\}_{t=1}^T$  from a noisy sequence of measurements  $\{\mathbf{x}_t\}_{t=1}^T$ . The world states are not independent; each is modeled as being contingent on the previous one. We exploit this statistical dependency to help estimate the state  $\mathbf{w}_t$ , even when the associated observation  $\mathbf{x}_t$  is partially or completely uninformative.

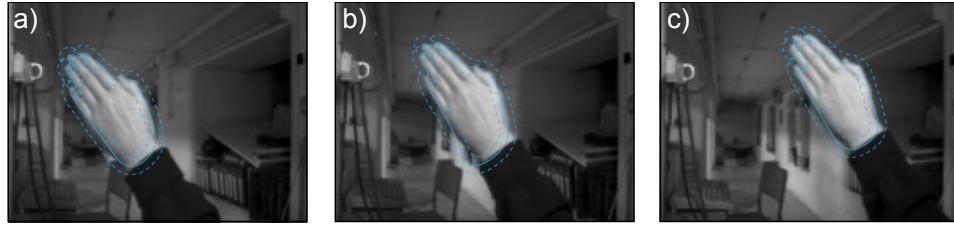
Since the states form a chain, the resulting models are similar to those in chapter 11. However, there are two major differences. First, in this chapter, we consider models where the world state is continuous rather than discrete. Second, the models here are designed for real-time applications; a judgment is made based on only the past and present measurements and without knowledge of those in the future.

A prototypical use of these temporal models is for *contour tracking*. Consider a parameterized model of an object contour (figure 19.1). The goal is to track this contour through a sequence of images so that it remains firmly attached to the object. A good model should be able to cope with non-rigid deformations of the object, background clutter, blurring, and occasional occlusion.

### 19.1 Temporal estimation framework

Each model in this chapter consists of two components.

- The *measurement model* describes the relationship between the measurements  $\mathbf{x}_t$  and the state  $\mathbf{w}_t$  at time  $t$ . We treat this as generative, and model the likelihood  $Pr(\mathbf{x}_t|\mathbf{w}_t)$ . We assume that the data at time  $t$  depend only on the state at time  $t$  and not those at any other time. In mathematical terms, we assume that  $\mathbf{x}_t$  is conditionally independent of  $\mathbf{w}_{1\dots t-1}$ , given  $\mathbf{w}_t$ .
- The *temporal model* describes the relationship between states. Typically, we make the *Markov assumption*: we assume that each state depends only upon its predecessor. More formally, we assume that  $\mathbf{w}_t$  is conditionally independent of the states  $\mathbf{w}_{1\dots t-2}$  given its immediate predecessor  $\mathbf{w}_{t-1}$ , and just model the relationship  $Pr(\mathbf{w}_t|\mathbf{w}_{t-1})$ .



**Figure 19.1** Contour tracking. The goal is to track the contour (solid blue line) through the sequence of images so that it remains firmly attached to the object (see chapter 17 for more information about contour models). The estimation of the contour parameters is based on a temporal model relating nearby frames, and local measurements from the image (e.g., between the dashed blue lines). Adapted from Blake & Isard (1998). ©1998 Springer.

Together, these assumptions lead to the graphical model shown in figure 19.2.

### 19.1.1 Inference

In inference, the general problem is to compute the marginal posterior distribution  $Pr(\mathbf{w}_t|\mathbf{x}_{1\dots t})$  over the world state  $\mathbf{w}_t$  at time  $t$  given all of the measurements  $\mathbf{x}_{1\dots t}$  up until this time. At time  $t=1$ , we have only observed a single measurement  $\mathbf{x}_1$ , so our prediction is based entirely on this datum. To compute the posterior distribution  $Pr(\mathbf{w}_1|\mathbf{x}_1)$ , Bayes' rule is applied:

$$Pr(\mathbf{w}_1|\mathbf{x}_1) = \frac{Pr(\mathbf{x}_1|\mathbf{w}_1)Pr(\mathbf{w}_1)}{\int Pr(\mathbf{x}_1|\mathbf{w}_1)Pr(\mathbf{w}_1) d\mathbf{w}_1}, \quad (19.1)$$

where the distribution  $Pr(\mathbf{w}_1)$  contains our prior knowledge about the initial state.

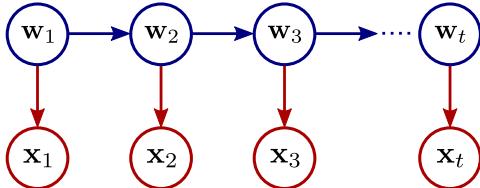
At time  $t=2$  we observe a second measurement  $\mathbf{x}_2$ . We now aim to compute the posterior distribution of the state at time  $t=2$  based on both  $\mathbf{x}_1$  and  $\mathbf{x}_2$ . Again, we apply Bayes' rule,

$$Pr(\mathbf{w}_2|\mathbf{x}_1, \mathbf{x}_2) = \frac{Pr(\mathbf{x}_2|\mathbf{w}_2)Pr(\mathbf{w}_2|\mathbf{x}_1)}{\int Pr(\mathbf{x}_2|\mathbf{w}_2)Pr(\mathbf{w}_2|\mathbf{x}_1) d\mathbf{w}_2}. \quad (19.2)$$

Notice that the likelihood term  $Pr(\mathbf{x}_2|\mathbf{w}_2)$  is only dependent on the current measurement (according to assumption stated earlier). The prior  $Pr(\mathbf{w}_2|\mathbf{x}_1)$  is now based on what we have learned from the previous measurement; the possible values of the state at this time depend on our knowledge of what happened at the previous time and how these are affected by the temporal model.

Generalizing this procedure to time  $t$ , we have

$$Pr(\mathbf{w}_t|\mathbf{x}_{1\dots t}) = \frac{Pr(\mathbf{x}_t|\mathbf{w}_t)Pr(\mathbf{w}_t|\mathbf{x}_{1\dots t-1})}{\int Pr(\mathbf{x}_t|\mathbf{w}_t)Pr(\mathbf{w}_t|\mathbf{x}_{1\dots t-1}) d\mathbf{w}_t}. \quad (19.3)$$



**Figure 19.2** Graphical model for Kalman filter and other temporal models in this chapter. This implies the following conditional independence relation: the state  $\mathbf{w}_t$  is conditionally independent of the states  $\mathbf{w}_{1\dots t-1}$  and the measurements  $\mathbf{x}_{1\dots t-1}$ , given the previous state  $\mathbf{w}_{t-1}$ .

To evaluate this, we must compute  $Pr(\mathbf{w}_t|\mathbf{x}_{1\dots t-1})$ , which represents our prior knowledge about  $\mathbf{w}_t$  before we look at the associated measurement  $\mathbf{x}_t$ . This prior depends on our knowledge  $Pr(\mathbf{w}_{t-1}|\mathbf{x}_{1\dots t-1})$  of the state at the previous time and the temporal model  $Pr(\mathbf{w}_t|\mathbf{w}_{t-1})$ , and is computed recursively as

$$Pr(\mathbf{w}_t|\mathbf{x}_{1\dots t-1}) = \int Pr(\mathbf{w}_t|\mathbf{w}_{t-1})Pr(\mathbf{w}_{t-1}|\mathbf{x}_{1\dots t-1}) d\mathbf{w}_{t-1}, \quad (19.4)$$

which is known as the *Chapman-Kolmogorov* relation. The first term in the integral represents the prediction for the state at time  $t$  given a known state  $\mathbf{w}_{t-1}$  at time  $t-1$ . The second term represents the uncertainty about what the state actually was at time  $t-1$ . The Chapman-Kolmogorov equation amalgamates these two pieces of information to predict what will happen at time  $t$ .

Hence, inference consists of two alternating steps. In the *prediction step*, we compute the prior  $Pr(\mathbf{w}_t|\mathbf{x}_{1\dots t-1})$  using the Chapman-Kolmogorov relation (equation 19.4). In the *measurement incorporation step*, we combine this prior with the new information from the measurement  $\mathbf{x}_t$  using Bayes's rule (equation 19.3).

### 19.1.2 Learning

The goal of learning is to estimate the parameters  $\boldsymbol{\theta}$  that determine the relationship  $Pr(\mathbf{w}_t|\mathbf{w}_{t-1})$  between adjacent states and the relationship  $Pr(\mathbf{x}_t|\mathbf{w}_t)$  between the state and the data, based on several observed time sequences.

If we know the states for these sequences, then this can be achieved using the maximum likelihood method. If the states are unknown, then they can be treated as *hidden variables*, and the model can be learned using the EM algorithm (section 7.8). In the E-step, we compute the posterior distribution over the states for each time sequence. This is a process similar to the inference method described earlier, except that it also uses data from later in the sequence (section 19.2.6). In the M-step, we update the EM bound with respect to  $\boldsymbol{\theta}$ .

The rest of this chapter focusses on inference in this type of temporal model. We will first consider the *Kalman filter*. Here the uncertainty over the world state is described by a normal distribution,<sup>1</sup> the relationship between the measurements

<sup>1</sup>In the original formulation of the Kalman filter, it was only assumed that the noise was white; however, if the distribution is normal, we can compute the exact marginal posterior distributions, so we favor this assumption.

and the world is linear with additive normal noise, and the relationship between the state at adjacent times is also linear with additive normal noise.

## 19.2 Kalman filter

To define the Kalman filter, we must specify the temporal and measurement models. First, the temporal model relates the states at times  $t-1$  and  $t$  and is given by

$$\mathbf{w}_t = \boldsymbol{\mu}_p + \boldsymbol{\Psi}\mathbf{w}_{t-1} + \boldsymbol{\epsilon}_p, \quad (19.5)$$

where  $\boldsymbol{\mu}_p$  is a  $D_w \times 1$  vector, which represents the mean change in the state and  $\boldsymbol{\Psi}$  is a  $D_w \times D_w$  matrix, which relates the mean of the state at time  $t$  to the state at time  $t-1$ . This is known as the *transition* matrix. The term  $\boldsymbol{\epsilon}_p$  is a realization of the transition noise, which is normally distributed with covariance  $\boldsymbol{\Sigma}_p$  and determines how closely related the states are at times  $t$  and  $t-1$ . Alternately, we can write this in probabilistic form:

$$Pr(\mathbf{w}_t | \mathbf{w}_{t-1}) = \text{Norm}_{\mathbf{w}_t}[\boldsymbol{\mu}_p + \boldsymbol{\Psi}\mathbf{w}_{t-1}, \boldsymbol{\Sigma}_p]. \quad (19.6)$$

Second, the measurement model relates the data  $\mathbf{x}_t$  at time  $t$  to the state  $\mathbf{w}_t$ ,

$$\mathbf{x}_t = \boldsymbol{\mu}_m + \boldsymbol{\Phi}\mathbf{w}_t + \boldsymbol{\epsilon}_m, \quad (19.7)$$

where  $\boldsymbol{\mu}_m$  is a  $D_x \times 1$  mean vector and  $\boldsymbol{\Phi}$  is a  $D_x \times D_w$  matrix relating the  $D_x \times 1$  measurement vector to the  $D_w \times 1$  state. The term  $\boldsymbol{\epsilon}_m$  is a realization of the measurement noise, which is normally distributed with covariance  $\boldsymbol{\Sigma}_m$ . In probabilistic notation, we have

$$Pr(\mathbf{x}_t | \mathbf{w}_t) = \text{Norm}_{\mathbf{x}_t}[\boldsymbol{\mu}_m + \boldsymbol{\Phi}\mathbf{w}_t, \boldsymbol{\Sigma}_m]. \quad (19.8)$$

Notice that the measurement equation is identical to the relation between the data and the hidden variable in the factor analysis model (section 7.6); here the state  $\mathbf{w}$  replaces the hidden variable  $\mathbf{h}$ . In the context of the Kalman filter, the dimension  $D_w$  of the state  $\mathbf{w}$  is often larger than the dimension  $D_x$  of the measurements  $\mathbf{x}$ , so  $\boldsymbol{\Phi}$  is a *landscape* matrix, and the measurement noise  $\boldsymbol{\Sigma}_m$  is not necessarily diagonal.

The form of both the temporal and measurement equations is the same: each is a normal probability distribution where the mean is a linear function of another variable and the variance is constant. This form has been carefully chosen because it ensures that if the marginal posterior  $Pr(\mathbf{w}_{t-1} | \mathbf{x}_{1\dots t-1})$  at time  $t-1$  was normal, then so is the marginal posterior  $Pr(\mathbf{w}_t | \mathbf{x}_{1\dots t})$  at time  $t$ . Hence, the inference procedure consists of a recursive updating of the means and variances of these normal distributions. We now elaborate on this procedure.

### 19.2.1 Inference

In inference, the goal is to compute the posterior probability  $Pr(\mathbf{w}_t | \mathbf{x}_{1\dots t})$  over the state  $\mathbf{w}_t$  given all of the measurements  $\mathbf{x}_{1\dots t}$  so far. As before, we apply the prediction and measurement-incorporation steps to recursively estimate  $Pr(\mathbf{w}_t | \mathbf{x}_{1\dots t})$  from  $Pr(\mathbf{w}_{t-1} | \mathbf{x}_{1\dots t-1})$ . The latter distribution is assumed to be normal with mean  $\boldsymbol{\mu}_{t-1}$  and variance  $\boldsymbol{\Sigma}_{t-1}$ .

In the prediction step, we compute the prior at time  $t$  using the Chapman-Kolmogorov equation

Problem 19.1

$$\begin{aligned} Pr(\mathbf{w}_t | \mathbf{x}_{1\dots t-1}) &= \int Pr(\mathbf{w}_t | \mathbf{w}_{t-1}) Pr(\mathbf{w}_{t-1} | \mathbf{x}_{1\dots t-1}) d\mathbf{w}_{t-1} \\ &= \int \text{Norm}_{\mathbf{w}_t}[\boldsymbol{\mu}_p + \boldsymbol{\Psi}\mathbf{w}_{t-1}, \boldsymbol{\Sigma}_p] \text{Norm}_{\mathbf{w}_{t-1}}[\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1}] d\mathbf{w}_{t-1} \\ &= \text{Norm}_{\mathbf{w}_t}[\boldsymbol{\mu}_p + \boldsymbol{\Psi}\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_p + \boldsymbol{\Psi}\boldsymbol{\Sigma}_{t-1}\boldsymbol{\Psi}^T] \\ &= \text{Norm}_{\mathbf{w}_t}[\boldsymbol{\mu}_+, \boldsymbol{\Sigma}_+], \end{aligned} \quad (19.9)$$

where we have denoted the predicted mean and variance of the state by  $\boldsymbol{\mu}_+$  and  $\boldsymbol{\Sigma}_+$ . The integral between lines 2 and 3 was solved by using equations 5.17 and 5.14 to rewrite the integrand as proportional to a normal distribution in  $\mathbf{w}_{t-1}$ . Since the integral of any pdf is one, the result is the constant of proportionality, which is itself a normal distribution in  $\mathbf{w}_t$ .

In the measurement incorporation step, we apply Bayes' rule,

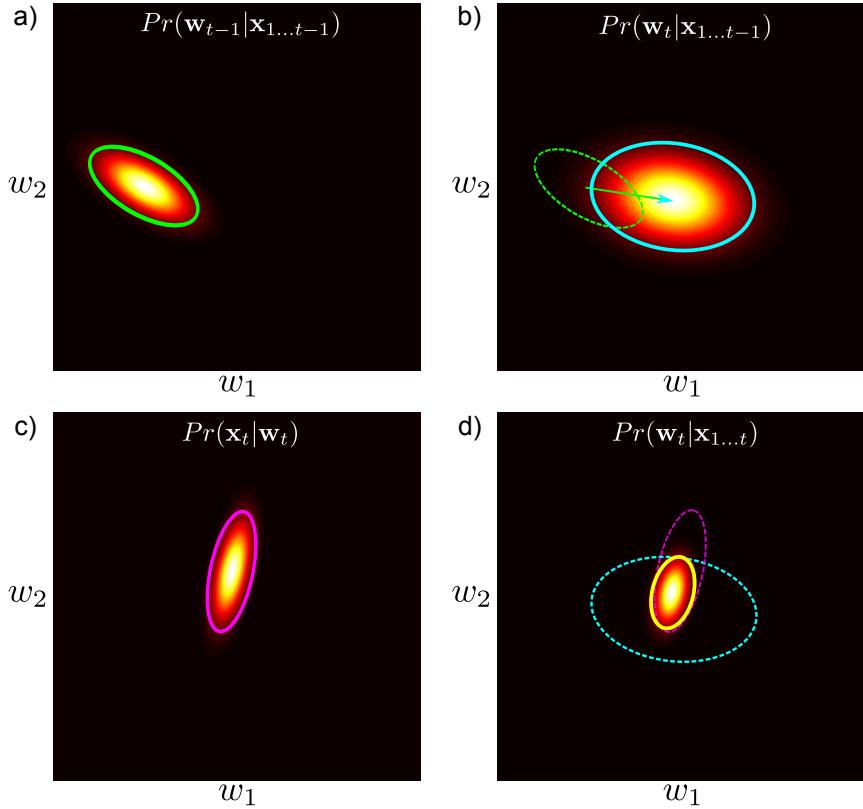
Problem 19.2

$$\begin{aligned} Pr(\mathbf{w}_t | \mathbf{x}_{1\dots t}) &= \frac{Pr(\mathbf{x}_t | \mathbf{w}_t) Pr(\mathbf{w}_t | \mathbf{x}_{1\dots t-1})}{Pr(\mathbf{x}_{1\dots t})} \\ &= \frac{\text{Norm}_{\mathbf{x}_t}[\boldsymbol{\mu}_m + \boldsymbol{\Phi}\mathbf{w}_t, \boldsymbol{\Sigma}_m] \text{Norm}_{\mathbf{w}_t}[\boldsymbol{\mu}_+, \boldsymbol{\Sigma}_+]}{Pr(\mathbf{x}_{1\dots t})} \\ &= \text{Norm}_{\mathbf{w}_t} \left[ \left( \boldsymbol{\Phi}^T \boldsymbol{\Sigma}_m^{-1} \boldsymbol{\Phi} + \boldsymbol{\Sigma}_+^{-1} \right)^{-1} \left( \boldsymbol{\Phi}^T \boldsymbol{\Sigma}_m^{-1} (\mathbf{x}_t - \boldsymbol{\mu}_m) + \boldsymbol{\Sigma}_+^{-1} \boldsymbol{\mu}_+ \right), \right. \\ &\quad \left. \left( \boldsymbol{\Phi}^T \boldsymbol{\Sigma}_m^{-1} \boldsymbol{\Phi} + \boldsymbol{\Sigma}_+^{-1} \right)^{-1} \right] \\ &= \text{Norm}_{\mathbf{w}_t}[\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t], \end{aligned} \quad (19.10)$$

where we have used equation 5.17 on the likelihood term and then combined the likelihood and prior using equation 5.14. The right-hand side is now proportional to a normal distribution in  $\mathbf{w}_t$ , and the constant of proportionality must be one to ensure that the posterior on the left hand side is a valid distribution.

Notice that the posterior  $Pr(\mathbf{w}_t | \mathbf{x}_{1\dots t})$  is normal with mean  $\boldsymbol{\mu}_t$  and covariance  $\boldsymbol{\Sigma}_t$ . We are in the same situation as at the start, so the procedure can be repeated for the next time step.

It can be shown that the mean of the posterior is a weighted sum of the values predicted by the measurements and the prior knowledge, and the covariance is smaller than either. However, this is not particularly obvious from the equations. In the following section we re-write the equation for the posterior in a form that makes these properties more obvious.



**Figure 19.3** Recursive inference in the Kalman filter. a) The posterior probability  $Pr(\mathbf{w}_{t-1}|\mathbf{x}_{1\dots t-1})$  of the state  $\mathbf{w}_{t-1}$ , given all of the measurements  $\mathbf{x}_{1\dots t-1}$  up to that time, takes the form of a normal distribution (green ellipse). b) In the prediction step, we apply the Chapman-Kolmogorov relation to estimate the prior  $Pr(\mathbf{w}_t|\mathbf{x}_{1\dots t-1})$  which is also a normal distribution (cyan ellipse). c) The measurement likelihood  $Pr(\mathbf{x}_t|\mathbf{w}_t)$  is proportional to a normal distribution (magenta ellipse). d) To compute the posterior probability  $Pr(\mathbf{w}_t|\mathbf{x}_{1\dots t})$ , we apply Bayes' rule by taking the product of the prior from panel (b) and the likelihood from panel (c) and normalizing. This yields a new normal distribution (yellow ellipse) and the procedure can begin anew.

### 19.2.2 Rewriting measurement incorporation step

The measurement incorporation step is rarely presented in the above form in practice. One reason for this is that the equations for  $\boldsymbol{\mu}_t$  and  $\boldsymbol{\Sigma}_t$  contain an inversion that is of size  $D_{\mathbf{w}} \times D_{\mathbf{w}}$ . If the world state is much higher dimensional than the observed data, then it would be more efficient to reformulate this as an inverse of size  $D_{\mathbf{x}} \times D_{\mathbf{x}}$ . To this end, we define the *Kalman gain* as

$$\mathbf{K} = \boldsymbol{\Sigma}_+ \boldsymbol{\Phi}^T (\boldsymbol{\Sigma}_m + \boldsymbol{\Phi} \boldsymbol{\Sigma}_+ \boldsymbol{\Phi}^T)^{-1}. \quad (19.11)$$

We will use this to modify the expressions for  $\boldsymbol{\mu}_t$  and  $\boldsymbol{\Sigma}_t$  from equation 19.10. Starting with  $\boldsymbol{\mu}_t$ , we apply the matrix inversion lemma (appendix C.8.4):

$$\begin{aligned} & (\boldsymbol{\Phi}^T \boldsymbol{\Sigma}_m^{-1} \boldsymbol{\Phi} + \boldsymbol{\Sigma}_+^{-1})^{-1} (\boldsymbol{\Phi}^T \boldsymbol{\Sigma}_m^{-1} (\mathbf{x}_t - \boldsymbol{\mu}_m) + \boldsymbol{\Sigma}_+^{-1} \boldsymbol{\mu}_+) \\ &= \mathbf{K}(\mathbf{x}_t - \boldsymbol{\mu}_m) + (\boldsymbol{\Phi}^T \boldsymbol{\Sigma}_m^{-1} \boldsymbol{\Phi} + \boldsymbol{\Sigma}_+^{-1})^{-1} \boldsymbol{\Sigma}_+^{-1} \boldsymbol{\mu}_+ \\ &= \mathbf{K}(\mathbf{x}_t - \boldsymbol{\mu}_m) + (\boldsymbol{\Sigma}_+ - \boldsymbol{\Sigma}_+ \boldsymbol{\Phi}^T (\boldsymbol{\Phi} \boldsymbol{\Sigma}_+ \boldsymbol{\Phi}^T + \boldsymbol{\Sigma}_m)^{-1} \boldsymbol{\Phi} \boldsymbol{\Sigma}_+) \boldsymbol{\Sigma}_+^{-1} \boldsymbol{\mu}_+ \\ &= \mathbf{K}(\mathbf{x}_t - \boldsymbol{\mu}_m) + \boldsymbol{\mu}_+ - \boldsymbol{\Sigma}_+ \boldsymbol{\Phi}^T (\boldsymbol{\Phi} \boldsymbol{\Sigma}_+ \boldsymbol{\Phi}^T + \boldsymbol{\Sigma}_m)^{-1} \boldsymbol{\Phi} \boldsymbol{\mu}_+ \\ &= \mathbf{K}(\mathbf{x}_t - \boldsymbol{\mu}_m) + \boldsymbol{\mu}_+ - \mathbf{K} \boldsymbol{\Phi} \boldsymbol{\mu}_+ \\ &= \boldsymbol{\mu}_+ + \mathbf{K}(\mathbf{x}_t - \boldsymbol{\mu}_m - \boldsymbol{\Phi} \boldsymbol{\mu}_+), \end{aligned} \quad (19.12)$$

The expression in brackets in the final line is known as the *innovation*, and is the difference between the actual measurements,  $\mathbf{x}_t$  and the predicted measurements  $\boldsymbol{\mu}_m + \boldsymbol{\Phi} \boldsymbol{\mu}_+$  based on the prior estimate of the state. It is easy to see why  $\mathbf{K}$  is termed the Kalman gain: it determines the amount that the measurements contribute to the new estimate in each direction in state space. If the Kalman gain is small in a given direction, then this implies that the measurements are unreliable relative to the prior and should not influence the mean of the state too much. If the Kalman gain is large in a given direction, then this suggests that the measurements are more reliable than the prior and should be weighted more highly.

We now return to the covariance term of equation 19.10. Using the matrix inversion lemma, we get

$$\begin{aligned} (\boldsymbol{\Phi}^T \boldsymbol{\Sigma}_m \boldsymbol{\Phi} + \boldsymbol{\Sigma}_+^{-1})^{-1} &= \boldsymbol{\Sigma}_+ - \boldsymbol{\Sigma}_+ \boldsymbol{\Phi}^T (\boldsymbol{\Phi} \boldsymbol{\Sigma}_+ \boldsymbol{\Phi}^T + \boldsymbol{\Sigma}_m)^{-1} \boldsymbol{\Phi} \boldsymbol{\Sigma}_+ \\ &= \boldsymbol{\Sigma}_+ - \mathbf{K} \boldsymbol{\Phi} \boldsymbol{\Sigma}_+ \\ &= (\mathbf{I} - \mathbf{K} \boldsymbol{\Phi}) \boldsymbol{\Sigma}_+, \end{aligned} \quad (19.13)$$

which also has a clear interpretation: the posterior covariance is equal to the prior covariance less a term that depends on the Kalman gain: we are always more certain about the state after incorporating information due to the measurement, and the Kalman gain modifies how much more certain we are. When the measurements are more reliable, the Kalman gain is high, and the covariance decreases more.

After these manipulations, we can rewrite equation 19.10 as

$$Pr(\mathbf{w}_t | \mathbf{x}_{1\dots t}) = \text{Norm}_{\mathbf{w}_t} [\boldsymbol{\mu}_+ + \mathbf{K}(\mathbf{x}_t - \boldsymbol{\mu}_m - \boldsymbol{\Phi} \boldsymbol{\mu}_+), (\mathbf{I} - \mathbf{K} \boldsymbol{\Phi}) \boldsymbol{\Sigma}_+]. \quad (19.14)$$

### 19.2.3 Inference summary

Developing the inference equations was rather long-winded, so here we summarize the inference process in the Kalman filter (see also figure 19.3). We aim to compute the marginal posterior probability  $Pr(\mathbf{w}_t | \mathbf{x}_{1\dots t})$  based on a normally distributed

Algorithm 19.1

estimate of the marginal posterior probability  $Pr(\mathbf{w}_{t-1}|\mathbf{x}_{1\dots t-1})$  at the previous time and a new measurement  $\mathbf{x}_t$ . If the posterior probability at time  $t-1$  has mean  $\boldsymbol{\mu}_{t-1}$  and variance  $\boldsymbol{\Sigma}_{t-1}$ , then the Kalman filter updates take the form

$$\begin{aligned} \text{State Prediction: } & \boldsymbol{\mu}_+ = \boldsymbol{\mu}_p + \boldsymbol{\Psi}\boldsymbol{\mu}_{t-1} & (19.15) \\ \text{Covariance Prediction: } & \boldsymbol{\Sigma}_+ = \boldsymbol{\Sigma}_p + \boldsymbol{\Psi}\boldsymbol{\Sigma}_{t-1}\boldsymbol{\Psi}^T \\ \text{State Update: } & \boldsymbol{\mu}_t = \boldsymbol{\mu}_+ + \mathbf{K}(\mathbf{x}_t - \boldsymbol{\mu}_m - \boldsymbol{\Phi}\boldsymbol{\mu}_+) \\ \text{Covariance Update: } & \boldsymbol{\Sigma}_t = (\mathbf{I} - \mathbf{K}\boldsymbol{\Phi})\boldsymbol{\Sigma}_+, \end{aligned}$$

where

$$\mathbf{K} = \boldsymbol{\Sigma}_+\boldsymbol{\Phi}^T(\boldsymbol{\Sigma}_m + \boldsymbol{\Phi}\boldsymbol{\Sigma}_+\boldsymbol{\Phi}^T)^{-1}. \quad (19.16)$$

In the absence of prior information at time  $t=1$ , it is usual to initialize the prior mean  $\boldsymbol{\mu}_0$  to any reasonable value and the covariance  $\boldsymbol{\Sigma}_0$  to a large multiple of the identity matrix representing our lack of knowledge.

#### 19.2.4 Example 1

The Kalman filter update equations are not very intuitive. To help understand the properties of this model, we present two toy examples. In the first case, we consider an object that is approximately circling a central point in two dimensions. The state consists of the two-dimensional position of the object. The actual set of states can be seen in figure 19.4a.

Let us assume that we do not have a good temporal model that describes this motion. Instead, we will assume the simplest possible model. The Brownian motion model assumes that the state at time  $t+1$  is similar to the state at time  $t$ :

$$Pr(\mathbf{w}_t|\mathbf{w}_{t-1}) = \text{Norm}_{\mathbf{w}_t}[\mathbf{w}_{t-1}, \sigma_p^2 \mathbf{I}]. \quad (19.17)$$

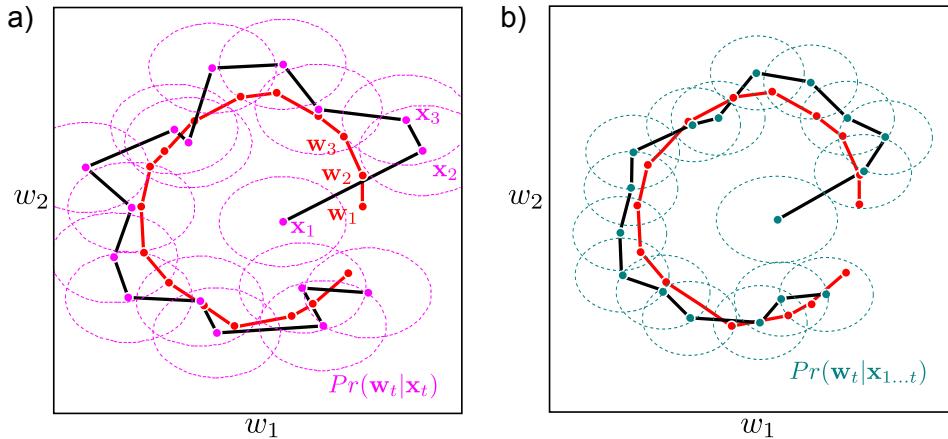
This is a special case of the more general formulation in equation 19.6. Notice that this model has no insight into the fact that the object is rotating.

For ease of visualization, we will assume that the observations are just noisy realizations of the true 2D state so that

$$Pr(\mathbf{x}_t|\mathbf{w}_t) = \text{Norm}_{\mathbf{x}_t}[\mathbf{w}_t, \boldsymbol{\Sigma}_m], \quad (19.18)$$

where  $\boldsymbol{\Sigma}_m$  is diagonal. This is a special case of the general formulation in 19.8.

In inference, the goal is to estimate the posterior probability over the state at each time step given the observed sequence so far. Figure 19.4b shows the sequence of posterior probabilities  $Pr(\mathbf{w}_t|\mathbf{x}_{1\dots t})$  over the state after running the Kalman recursions. It is now easy to understand why this is referred to as the Kalman *filter*: the MAP states (the peaks of the marginal posteriors) are smoothed relative to estimates from the measurements alone. Notice that the Kalman filter estimate has a lower covariance than the estimate due to the measurements alone



**Figure 19.4** Kalman filter example 1. The true state (red dots) evolves in a 2D circle. The measurements (magenta dots) consist of direct noisy observations of the state. a) Posterior probability  $Pr(\mathbf{w}_t | \mathbf{x}_t)$  of state using measurements alone (uniform prior assumed). The mean of each distribution is centered on the associated data point, and the covariance depends on the measurement noise. b) Posterior probabilities based on Kalman filter. Here the temporal equation describes the state as a random perturbation of the previous state. Although this is not a good model (the circular movement is not accounted for), the posterior covariances (blue ellipses) are smaller than without the Kalman filter (magenta ellipses in (a)) and the posterior means (blue dots) are closer to the true states (red dots) than the estimates due to the measurements alone (magenta dots in (a)).

(figure 19.4a). In this example, the inclusion of the temporal model makes the estimates both more accurate and more certain, even though the particular choice of temporal model is wrong.

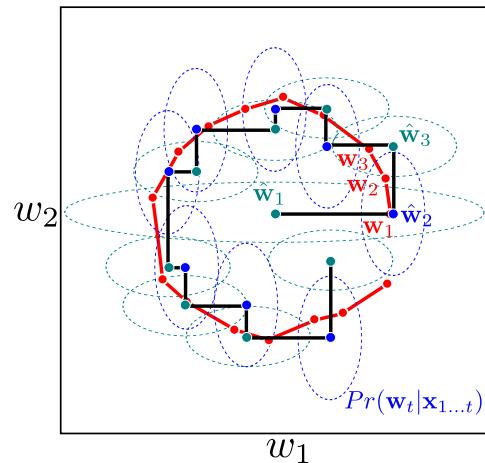
### 19.2.5 Example 2

Figure 19.5 shows a second example, where the setup is the same except that the observation equation differs at alternate time steps. At even time steps, we observe a noisy estimate of just the first dimension of the state  $\mathbf{w}$ . At odd time steps, we observe only a noisy estimate of the second dimension. The measurement equations are

$$\begin{aligned} Pr(x_t | \mathbf{w}_t) &= \text{Norm}_{x_t} \left[ [1 \ 0] \mathbf{w}_t, \sigma_m^2 \right], & \text{for } t = 1, 3, 5 \dots \\ Pr(x_t | \mathbf{w}_t) &= \text{Norm}_{x_t} \left[ [0 \ 1] \mathbf{w}_t, \sigma_m^2 \right], & \text{for } t = 2, 4, 6 \dots \end{aligned} \quad (19.19)$$

This is an example of a *non-stationary model*: the model changes over time.

**Figure 19.5** Kalman filter example 2.  
As for example 1, the true state (red points) proceeds in an approximate circle. We don't have access to this temporal model and just assume that the new state is a perturbation of the previous state. At even time steps, the measurement is a noisy realization of the first dimension (horizontal coordinate) of the state. At odd time steps, it is a noisy realization of the second dimension (vertical coordinate). The Kalman filter produces plausible estimates (blue and cyan points and ellipses) even though a single measurement is being used to determine a 2D state at any time step.



We use the same set of ground truth 2D states as for example 1, and simulate the associated measurements using equation 19.19. The posterior probability over the state was computed using the Kalman filter recursions with the relevant measurement matrix  $\Phi$  at each time step.

The results (figure 19.5) show that the Kalman filter maintains a good estimate of the 2D state, despite only having a 1D measurement at each time-step. At odd time steps, the variance in the first dimension decreases (due to the information from the measurement) but the variance in the second dimension increases (due to the uncertainty in the temporal model). At even time steps, the opposite occurs.

This model may seem esoteric, but it is common in many imaging modalities for different aspects of the measurements to arrive at different times. One option is to wait until a full set of measurements is present before estimating the state, but this means that many of them are out of date. Incorporation of each measurement when it arrives using a Kalman filter is a superior approach.

### 19.2.6 Smoothing

The preceding inference procedure is designed for real-time applications where the estimation of the state is based only on measurements from the past and present. However, there may be occasions where we wish to infer the state using measurements that lie in the future. In the parlance of Kalman filters, this is known as *smoothing*.

We consider two cases. The *fixed lag smoother* is still intended for on-line estimation, but it delays the decision about the state by a fixed number of time steps. The *fixed interval smoother* assumes that we have observed all of the measurements in the entire sequence before we make a judgment about the world state.

### Fixed lag smoother

The fixed lag smoother relies on a simple trick. To estimate the state delayed by  $\tau$  time steps, we *augment* the state vector to contain the delayed estimates of the previous  $\tau$  times. The time update equation now takes the form

$$\begin{bmatrix} \mathbf{w}_t \\ \mathbf{w}_t^{[1]} \\ \mathbf{w}_t^{[2]} \\ \vdots \\ \mathbf{w}_t^{[\tau]} \end{bmatrix} = \begin{bmatrix} \Psi & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{w}_{t-1} \\ \mathbf{w}_{t-1}^{[1]} \\ \mathbf{w}_{t-1}^{[2]} \\ \vdots \\ \mathbf{w}_{t-1}^{[\tau]} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\epsilon}_p \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}, \quad (19.20)$$

where the notation  $\mathbf{w}_t^{[m]}$  refers to the state at time  $t-m$  based on measurements up to time  $t$ , so  $\mathbf{w}_t^{[\tau]}$  is the quantity we wish to estimate. This state evolution equation clearly fits the general form of the Kalman filter (equation 19.5). It can be parsed as follows: in the first equation of this system, the temporal model is applied to the current estimate of the state. In the remaining equations, the delayed states are created by simply copying the previous state.

The associated measurement equation is

$$\mathbf{x}_t = [\Phi \quad \mathbf{0} \quad \mathbf{0} \quad \dots \quad \mathbf{0}] \begin{bmatrix} \mathbf{w}_t \\ \mathbf{w}_t^{[1]} \\ \mathbf{w}_t^{[2]} \\ \vdots \\ \mathbf{w}_t^{[\tau]} \end{bmatrix} + \boldsymbol{\epsilon}_m, \quad (19.21)$$

which fits the form of the Kalman filter measurement model (equation 19.7). The current measurements are based on the current state, and the time-delayed versions play no part. Applying the Kalman recursions with these equations computes not just the current estimate of the state, but also the time-delayed estimates.

### Fixed interval smoother

The fixed interval smoother consists of a backward set of recursions that estimate the marginal posterior distributions  $Pr(\mathbf{w}_t | \mathbf{x}_{1\dots T})$  of the state at each time step, taking into account all of the measurements  $\mathbf{x}_{1\dots T}$ . In these recursions, the marginal posterior distribution  $Pr(\mathbf{w}_t | \mathbf{x}_{1\dots T})$  of the state at time  $t$  is updated, and, based on this result, the marginal posterior  $Pr(\mathbf{w}_{t-1} | \mathbf{x}_{1\dots T})$  at time  $t-1$  is updated and so on. We denote the mean and variance of the marginal posterior  $Pr(\mathbf{w}_t | \mathbf{x}_{1\dots T})$  at time  $t$  by  $\boldsymbol{\mu}_{t|T}$  and  $\boldsymbol{\Sigma}_{t|T}$ , respectively, and use

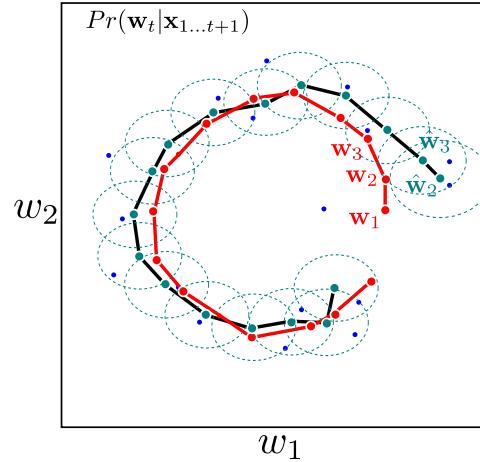
$$\begin{aligned} \boldsymbol{\mu}_{t|T} &= \boldsymbol{\mu}_t + \mathbf{C}_t (\boldsymbol{\mu}_{t+1|T} - \boldsymbol{\mu}_{+|t}) \\ \boldsymbol{\Sigma}_{t|T} &= \boldsymbol{\Sigma}_t + \mathbf{C}_t (\boldsymbol{\Sigma}_{t+1|T} - \boldsymbol{\Sigma}_{+|t}) \mathbf{C}_t^T, \end{aligned} \quad (19.22)$$

where the notation  $\boldsymbol{\mu}_{+|t}$  and  $\boldsymbol{\Sigma}_{+|t}$  denotes the mean and variance of the posterior distribution  $Pr(\mathbf{w}_t | \mathbf{x}_{1\dots t-1})$  of the state at time  $t$  based on the measurements up

**Problem 19.7**

**Algorithm 19.2**

**Figure 19.6** Fixed lag Kalman smoothing. The estimated state (light blue dots) and the associated covariance (light blue ellipses) were estimated using a lag of 1 time step: each estimate is based on all of the data in the past, the present observation, and the observation one step into the future. The resulting estimated states have a smaller variance than for the standard Kalman filter (compare to figure 19.4b) and are closer to the true state (red dots). In effect, the estimate averages out the noise in the measurements (dark blue dots). The cost of this improvement is that there is a delay of one time step.



to time  $t-1$  (i.e., what we denoted as  $\mu_+$  and  $\Sigma_+$  during the forward Kalman filter recursions) and

$$\mathbf{C}_t = \Sigma_{t|t} \Psi^T \Sigma_{+|t}^{-1}. \quad (19.23)$$

It can be shown that the backward recursions correspond to the backward pass of sum-product belief propagation in the Kalman filter graphical model.

### 19.2.7 Temporal and measurement models

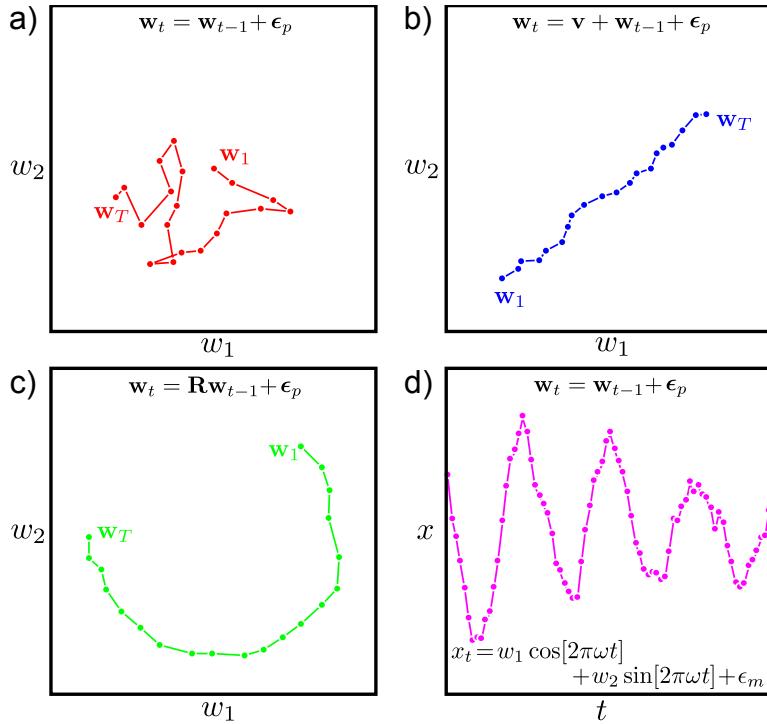
The choice of temporal model in the Kalman filter is restricted to be linear and is dictated by the matrix  $\Psi$ . Despite this limitation, it is possible to build a surprisingly versatile family of models within this framework. In this section we review a few of the best known examples.

1. *Brownian motion:* The simplest model is Brownian motion (figure 19.7a) in which the state is operated upon by the identity matrix so that

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \boldsymbol{\epsilon}_p. \quad (19.24)$$

2. *Geometric transformations:* The family of linear filters includes geometric transformations such as rotations, stretches and shears. For example, choosing  $\Psi$  so that  $\Psi^T \Psi = \mathbf{I}$  and  $|\Psi| = 1$  creates a rotation around the origin (figure 19.7b): this is the true temporal model in figures 19.4-19.6.
3. *Velocities and accelerations:* The Brownian motion model can be extended by adding a constant velocity  $\mathbf{v}$  to the motion model (figure 19.7c) so that

$$\mathbf{w}_t = \mathbf{v} + \mathbf{w}_{t-1} + \boldsymbol{\epsilon}_p. \quad (19.25)$$



**Figure 19.7** Temporal and measurement models. a) Brownian motion model. At each time step the state is randomly perturbed from the previous position, so a drunken walk through state space occurs. b) Constant velocity model. At each time step, a constant velocity is applied in addition to the noise. c) Transformation model. At each time step the state is rotated about the origin and noise is added. d) Oscillatory measurements. These quasi-sinusoidal measurements are created from a two-dimensional state that itself undergoes Brownian motion. The two elements of the state control the sinusoidal and co-sinusoidal parts of the measurements.

To incorporate a changing velocity, we can enhance the state vector to include the velocity term so that

$$\begin{bmatrix} \mathbf{w}_t \\ \dot{\mathbf{w}}_t \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{w}_{t-1} \\ \dot{\mathbf{w}}_{t-1} \end{bmatrix} + \epsilon_p. \quad (19.26)$$

This has the natural interpretation that the velocity term  $\dot{\mathbf{w}}$  contributes an offset to the state at each time. However, the velocity is itself uncertain and is related by a Brownian motion model to the velocity at the previous time step. For the measurement equation, we have

$$\mathbf{x}_t = [\mathbf{I} \quad \mathbf{0}] \begin{bmatrix} \mathbf{w}_t \\ \dot{\mathbf{w}}_t \end{bmatrix} + \epsilon_m. \quad (19.27)$$

In other words, the measurements at time  $t$  do not directly depend on the velocity term, only the state itself. This idea can be easily extended to include an acceleration term as well.

4. *Oscillatory data:* Some data are naturally oscillatory. To describe oscillatory 1D data, we could use a state containing a  $2 \times 1$  state vector  $\mathbf{w}$  using Brownian motion as the temporal model. We then implement the non-stationary measurement equation

$$\mathbf{x}_t = [\cos[2\pi\omega t] \quad \sin[2\pi\omega t]] \mathbf{w}_t + \boldsymbol{\epsilon}_m. \quad (19.28)$$

For a fixed state  $\mathbf{w}$ , this model produces noisy sinusoidal data. As the state varies due to the Brownian motion of the temporal model, the phase and amplitude of the quasi-sinusoidal output will change (19.7d).

### 19.2.8 Problems with the Kalman filter

Although the Kalman filter is a flexible tool, it has a number of shortcomings (figure 19.8). Most notably,

- it requires the temporal and measurement equations to be linear, and
- it assumes that the marginal posterior is unimodal and can be well captured by a mean and covariance; hence, it can only ever have one hypothesis about the position of the object.

In the following sections, we discuss models that address these problems. The *extended Kalman filter* and the *unscented Kalman filter* both allow nonlinear state update and measurement equations. *Particle filtering* abandons the use of the normal distribution and describes the state as a complex multi-modal distribution.

## 19.3 Extended Kalman filter

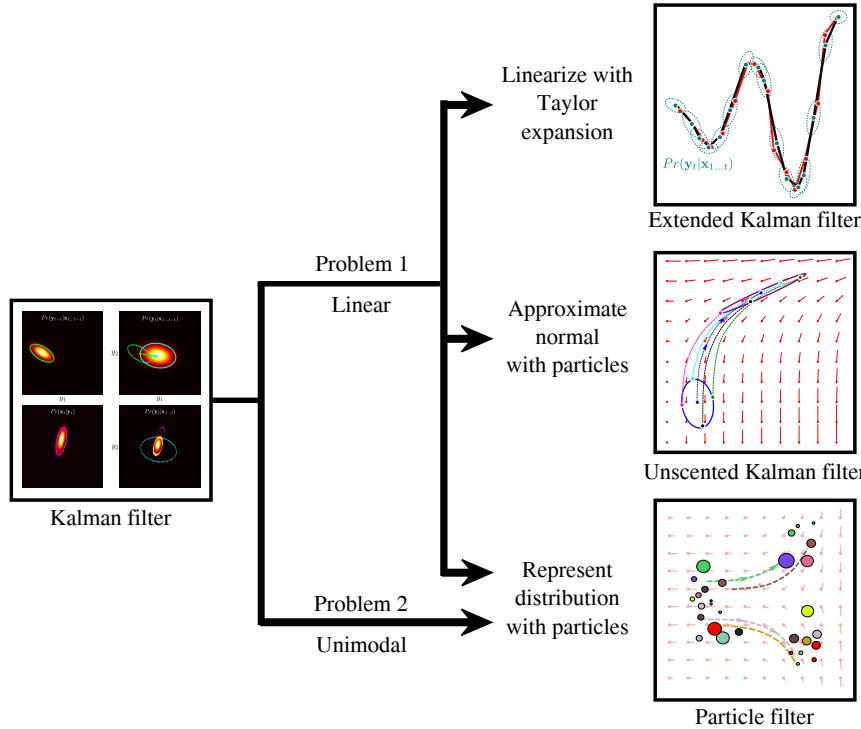
The extended Kalman filter (EKF) is designed to cope with more general temporal models, where the relationship between the states at time  $t$  is an arbitrary nonlinear function  $\mathbf{f}[\bullet, \bullet]$  of the state at the previous time step and a stochastic contribution  $\boldsymbol{\epsilon}_p$

$$\mathbf{w}_t = \mathbf{f}[\mathbf{w}_{t-1}, \boldsymbol{\epsilon}_p], \quad (19.29)$$

where the covariance of the noise term  $\boldsymbol{\epsilon}_p$  is  $\Sigma_p$  as before. Similarly, it can cope with a nonlinear relationship  $\mathbf{g}[\bullet, \bullet]$  between the state and the measurements

$$\mathbf{x}_t = \mathbf{g}[\mathbf{w}_t, \boldsymbol{\epsilon}_m], \quad (19.30)$$

where the covariance of  $\boldsymbol{\epsilon}_m$  is  $\Sigma_m$ .



**Figure 19.8** Temporal models for tracking. The Kalman filter has two main problems: first, it requires the temporal and measurement models to be linear. This problem is directly addressed by the extended and unscented Kalman filters. Second, it represents the uncertainty with a normal distribution that is unimodal, and so cannot maintain multiple hypotheses about the state. This problem is tackled by particle filters.

The extended Kalman filter works by taking linear approximations to the non-linear functions at the peak  $\mu_t$  of the current estimate using the Taylor expansion. If the function is not too nonlinear, then this approximation will adequately represent the function in the region of the current estimate and we can proceed as usual. We define the Jacobian matrices,

Algorithm 19.3

$$\begin{aligned}
\Psi &= \left. \frac{\partial \mathbf{f}[\mathbf{w}_{t-1}, \boldsymbol{\epsilon}_p]}{\partial \mathbf{w}_{t-1}} \right|_{\boldsymbol{\mu}_{t-1}, \mathbf{0}} \\
\Upsilon_p &= \left. \frac{\partial \mathbf{f}[\mathbf{w}_{t-1}, \boldsymbol{\epsilon}_p]}{\partial \boldsymbol{\epsilon}_p} \right|_{\boldsymbol{\mu}_{t-1}, \mathbf{0}} \\
\Phi &= \left. \frac{\partial \mathbf{g}[\mathbf{w}_t, \boldsymbol{\epsilon}_m]}{\partial \mathbf{w}_t} \right|_{\boldsymbol{\mu}_+, \mathbf{0}} \\
\Upsilon_m &= \left. \frac{\partial \mathbf{g}[\mathbf{w}_t, \boldsymbol{\epsilon}_m]}{\partial \boldsymbol{\epsilon}_m} \right|_{\boldsymbol{\mu}_+, \mathbf{0}}
\end{aligned} \tag{19.31}$$

where the notation  $|_{\boldsymbol{\mu}_+, \mathbf{0}}$  denotes that the derivative is computed at position  $\mathbf{w} = \boldsymbol{\mu}_+$  and  $\boldsymbol{\epsilon} = \mathbf{0}$ . Note that we have overloaded the meaning of  $\Phi$  and  $\Psi$  here. Previously they represented the linear transforms between states at adjacent times, and between the state and the measurements, respectively. Now they represent the local linear approximation of the nonlinear functions relating these quantities.

The update equations for the extended Kalman filter are

$$\begin{aligned}
\text{State Prediction:} \quad \boldsymbol{\mu}_+ &= \mathbf{f}[\boldsymbol{\mu}_{t-1}, \mathbf{0}] \\
\text{Covariance Prediction:} \quad \boldsymbol{\Sigma}_+ &= \boldsymbol{\Psi} \boldsymbol{\Sigma}_{t-1} \boldsymbol{\Psi}^T + \boldsymbol{\Upsilon}_p \boldsymbol{\Sigma}_p \boldsymbol{\Upsilon}_p^T \\
\text{State Update:} \quad \boldsymbol{\mu}_t &= \boldsymbol{\mu}_+ + \mathbf{K}(\mathbf{x}_t - \mathbf{g}[\boldsymbol{\mu}_+, \mathbf{0}]) \\
\text{Covariance Update:} \quad \boldsymbol{\Sigma}_t &= (\mathbf{I} - \mathbf{K}\Phi)\boldsymbol{\Sigma}_+
\end{aligned} \tag{19.32}$$

where

$$\mathbf{K} = \boldsymbol{\Sigma}_+ \boldsymbol{\Phi}^T (\boldsymbol{\Upsilon}_m \boldsymbol{\Sigma}_m \boldsymbol{\Upsilon}_m^T + \boldsymbol{\Phi} \boldsymbol{\Sigma}_+ \boldsymbol{\Phi}^T)^{-1}. \tag{19.33}$$

In the context of fixed interval smoothing, the results can be improved by performing several passes back and forth through the data, re-linearizing around the previous estimates of the state in each sweep. This is called the *iterated extended Kalman filter*.

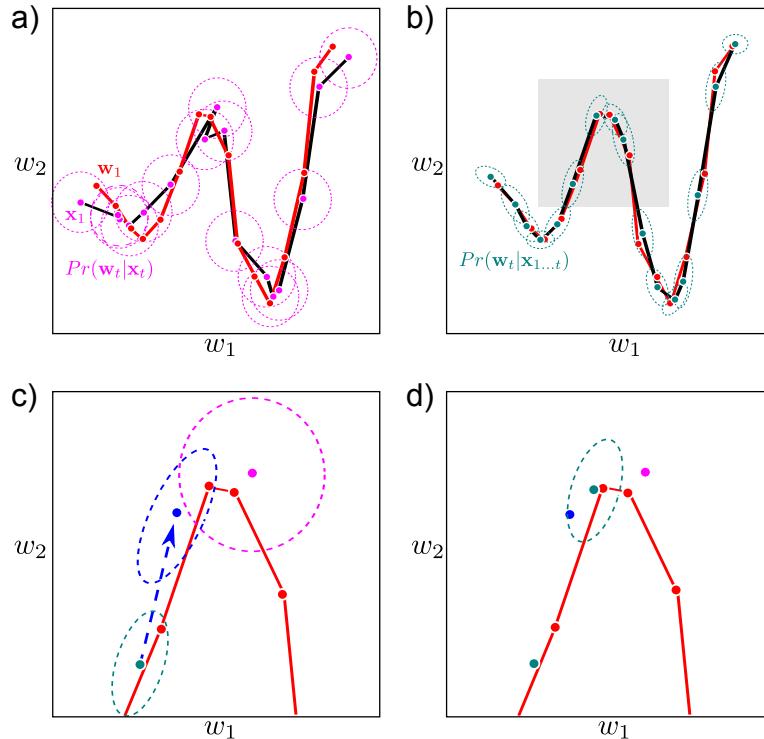
Algorithm 19.4

In conclusion, the extended Kalman filter is conceptually simple, but only copes with relatively benign nonlinearities. It is a heuristic solution to the nonlinear tracking problem and may diverge from the true solution.

### 19.3.1 Example

Figure 19.9 shows an worked example of the extended Kalman filter. In this case, the time update model is nonlinear, but the observation model is still linear:

$$\begin{aligned}
\mathbf{w}_t &= \mathbf{f}[\mathbf{w}_{t-1}, \boldsymbol{\epsilon}_p] \\
\mathbf{x}_t &= \mathbf{w} + \boldsymbol{\epsilon}_m,
\end{aligned} \tag{19.34}$$



**Figure 19.9** The extended Kalman filter (EKF). a) The temporal model that transforms the state (red dots) here is nonlinear. Each observed data point (magenta dots) is a noisy copy of the state at that time. Based on the data alone, the posterior probability over the state is given by the magenta ellipses. b) Estimated marginal posterior distributions using the extended Kalman filter: the estimates are more certain and more accurate than using the measurements alone. c) Close up of shaded region from (b). The EKF makes a normal prediction (dark blue ellipse) based on the previous state (light blue ellipse) and the linearized motion model. The measurement makes a different prediction (magenta ellipse). d) The EKF combines these two predictions to get an improved estimate (light blue ellipse).

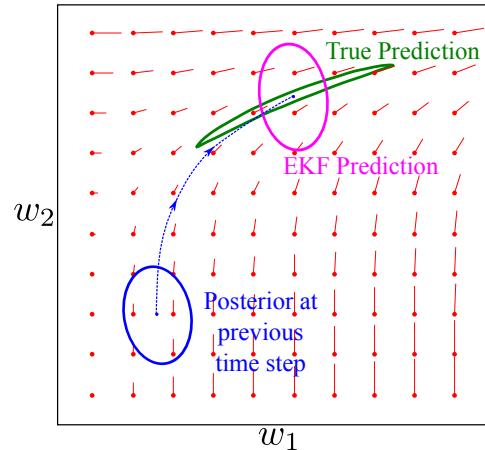
where  $\epsilon_p$  is a normal noise term with covariance  $\Sigma_p$ ,  $\epsilon_m$  is a normal noise term with covariance  $\Sigma_m$ , and the nonlinear function  $\mathbf{f}[\bullet, \bullet]$  is given by

$$\mathbf{f}[\mathbf{w}, \epsilon_p] = \begin{bmatrix} w_1 \\ w_1 \sin[w_1] + \epsilon_p \end{bmatrix}. \quad (19.35)$$

The Jacobian matrices for this model are easily computed. The matrices  $\boldsymbol{\Upsilon}_p, \boldsymbol{\Phi}$ , and  $\boldsymbol{\Upsilon}_m$  are all equal to the identity. The Jacobian matrix  $\boldsymbol{\Psi}$  is

$$\boldsymbol{\Psi} = \begin{bmatrix} 1 & 0 \\ \sin[w_1] + w_1 \cos[w_1] & 0 \end{bmatrix}. \quad (19.36)$$

**Figure 19.10** Problems with the EKF.  
A 2D temporal function  $f[\bullet, \bullet]$  operates on the state (red lines indicate gradient direction). In the prediction step, the previous estimate (blue ellipse) should be passed through this function (to create distorted green ellipse) and noise added (not shown). The EKF approximates this by passing the mean through the function and updating the covariance based on a linear approximation of the function at the previous state estimate. Here, the function changes quite nonlinearly, and so this approximation is bad, and the predicted covariance (magenta ellipse) is inaccurate.



The results of using this system for tracking are illustrated in figure 19.9. The extended Kalman filter successfully tracks this nonlinear model giving results that are both closer to the true state and more confident than estimates based on the individual measurements alone. The EKF works well here because the nonlinear function is smooth and departs from linearity slowly relative to the time steps: consequently, the linear approximation is reasonable.

## 19.4 Unscented Kalman filter

The extended Kalman filter is only reliable if the linear approximations to the functions  $f[\bullet, \bullet]$  or  $g[\bullet, \bullet]$  describe them well in the region of the current position. Figure 19.10 shows a situation where the local properties of the temporal function are not representative and so the EKF estimate of the covariance is inaccurate.

Algorithm 19.5

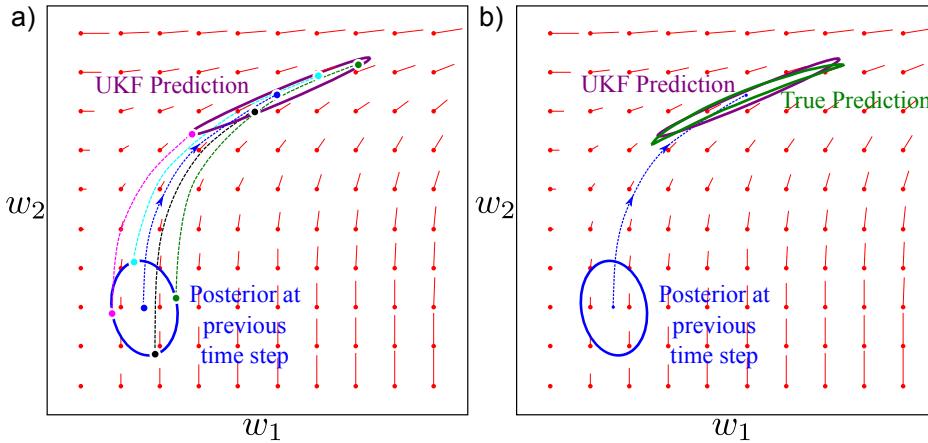
The unscented Kalman filter (UKF) is a derivative-free approach that partially circumvents this problem. It is suited to nonlinear models with additive, normally distributed noise, so that the temporal and measurement equations are

$$\begin{aligned} \mathbf{w}_t &= \mathbf{f}[\mathbf{w}_{t-1}] + \boldsymbol{\epsilon}_p \\ \mathbf{x}_t &= \mathbf{g}[\mathbf{w}_t] + \boldsymbol{\epsilon}_m. \end{aligned} \quad (19.37)$$

As such it is slightly more restricted in its applicability compared to the extended Kalman filter. However, it can cope with more extreme nonlinearities in the state evolution and measurement equations than the EKF.

To understand how the UKF works, consider a nonlinear temporal model. Given the mean and covariance of the normally distributed posterior distribution  $Pr(\mathbf{w}_{t-1}|\mathbf{x}_{1\dots t-1})$  over the state at time  $t-1$ , we wish to predict a normally distributed prior  $Pr(\mathbf{w}_t|\mathbf{x}_{1\dots t-1})$  over the state at time  $t$ . We proceed by

- approximating the normally distributed posterior  $Pr(\mathbf{w}_{t-1}|\mathbf{x}_{1\dots t-1})$  with a



**Figure 19.11** Temporal update for unscented Kalman filter. a) In the temporal update step, the posterior distribution from the previous time step (which is a normal) is approximated by a weighted set of point masses that collectively have the same mean and covariance as this normal. Each of these is passed through the temporal model (dashed lines). A normal prediction is formed by estimating the mean and the covariance of the transformed point masses. In a real system, the variance would subsequently be inflated to account for additive uncertainty in the position. b) The resulting prediction is quite close to the true prediction and for this case is much better than that of the extended Kalman filter (see figure 19.10).

set of point masses that are deterministically chosen so that they have the same mean  $\mu_{t-1}$  and covariance  $\Sigma_{t-1}$  as the original distribution,

- passing each of the point masses through the nonlinear function, and then
- setting the predicted distribution  $Pr(\mathbf{w}_t|\mathbf{x}_{1\dots t-1})$  at time  $t$  to be a normal distribution with the mean and covariance of the transformed point masses.

This process is illustrated in figure 19.11.

As for the extended Kalman filter, the predicted state  $Pr(\mathbf{w}_t|\mathbf{x}_{1\dots t-1})$  in the UKF is a normal distribution. However, this normal distribution is a provably better approximation to the true distribution than that provided by the EKF. A similar approach is applied to cope with a nonlinearity in the measurement equations. We will now consider each of these steps in more detail.

#### 19.4.1 State evolution

As for the standard Kalman filter, the goal of the state evolution step is to form a prediction  $Pr(\mathbf{w}_t|\mathbf{x}_{1\dots t-1})$  about the state at time  $t$  by applying the temporal model to the posterior distribution  $Pr(\mathbf{w}_{t-1}|\mathbf{x}_{1\dots t-1})$  at the previous time step.

This marginal posterior distribution is normal with mean  $\boldsymbol{\mu}_{t-1}$  and covariance  $\boldsymbol{\Sigma}_{t-1}$ , and the prediction will also be normal with mean  $\boldsymbol{\mu}_+$  and covariance  $\boldsymbol{\Sigma}_+$ .

We proceed as follows. We approximate the posterior distribution at the previous time step by a weighted sum of  $2D_{\mathbf{w}}+1$  delta functions, where  $D_{\mathbf{w}}$  is the dimensionality of the state, so that

$$\begin{aligned} Pr(\mathbf{w}_{t-1} | \mathbf{x}_{1\dots t-1}) &= \text{Norm}_{\mathbf{w}_{t-1}}[\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1}] \\ &\approx \sum_{j=0}^{2D_{\mathbf{w}}} a_j \delta[\mathbf{w}_{t-1} - \hat{\mathbf{w}}^{[j]}], \end{aligned} \quad (19.38)$$

where the weights  $\{a_j\}_{j=0}^{2D_{\mathbf{w}}}$  are positive and sum to one. In this context, the delta functions are referred to as *sigma points*. The positions  $\{\hat{\mathbf{w}}^{[j]}\}_{j=0}^{2D_{\mathbf{w}}}$  of the sigma points are carefully chosen so that

$$\begin{aligned} \boldsymbol{\mu}_{t-1} &= \sum_{j=0}^{2D_{\mathbf{w}}} a_j \hat{\mathbf{w}}^{[j]} \\ \boldsymbol{\Sigma}_{t-1} &= \sum_{j=0}^{2D_{\mathbf{w}}} a_j (\hat{\mathbf{w}}^{[j]} - \boldsymbol{\mu}_{t-1})(\hat{\mathbf{w}}^{[j]} - \boldsymbol{\mu}_{t-1})^T. \end{aligned} \quad (19.39)$$

One possible scheme is to choose sigma points

$$\begin{aligned} \hat{\mathbf{w}}^{[0]} &= \boldsymbol{\mu}_{t-1} \\ \hat{\mathbf{w}}^{[j]} &= \boldsymbol{\mu}_{t-1} + \sqrt{\frac{D_{\mathbf{w}}}{1-a_0}} \boldsymbol{\Sigma}_{t-1}^{1/2} \mathbf{e}_j \quad \text{for all } j \in \{1 \dots D_{\mathbf{w}}\} \\ \hat{\mathbf{w}}^{[D_{\mathbf{w}}+j]} &= \boldsymbol{\mu}_{t-1} - \sqrt{\frac{D_{\mathbf{w}}}{1-a_0}} \boldsymbol{\Sigma}_{t-1}^{1/2} \mathbf{e}_j \quad \text{for all } j \in \{1 \dots D_{\mathbf{w}}\}, \end{aligned} \quad (19.40)$$

where  $\mathbf{e}_j$  is the unit vector in the  $j^{th}$  direction. The associated weights are chosen so that  $a_0 \in [0, 1]$  and

$$a_j = \frac{1-a_0}{2D_{\mathbf{w}}}, \quad (19.41)$$

where the choice of the weight  $a_0$  of the first sigma point determines how far the remaining sigma points are from the mean.

We pass the sigma points through the nonlinearity, to create a new set of samples  $\hat{\mathbf{w}}_+^{[j]} = \mathbf{f}[\hat{\mathbf{w}}^{[j]}]$  that collectively form a prediction for the state. We then compute the mean and the variance of the predicted distribution  $Pr(\mathbf{w}_t | \mathbf{x}_{1\dots t-1})$  from the mean and variance of the transformed points so that

$$\begin{aligned}\boldsymbol{\mu}_+ &= \sum_{j=0}^{2D_w} a_j \hat{\mathbf{w}}_+^{[j]} \\ \boldsymbol{\Sigma}_+ &= \sum_{j=0}^{2D_w} a_j (\hat{\mathbf{w}}_+^{[j]} - \boldsymbol{\mu}_+) (\hat{\mathbf{w}}_+^{[j]} - \boldsymbol{\mu}_+)^T + \boldsymbol{\Sigma}_p,\end{aligned}\quad (19.42)$$

where we have added an extra term  $\boldsymbol{\Sigma}_p$  to the predicted covariance to account for the additive noise in the temporal model.

### 19.4.2 Measurement incorporation

The measurement incorporation process in the UKF uses a similar idea: we approximate the predicted distribution  $Pr(\mathbf{w}_t | \mathbf{x}_{1\dots t-1})$  as a set of delta functions or sigma points

$$\begin{aligned}Pr(\mathbf{w}_t | \mathbf{x}_{1\dots t-1}) &= \text{Norm}_{\mathbf{w}_{t-1}}[\boldsymbol{\mu}_+, \boldsymbol{\Sigma}_+] \\ &\approx \sum_{j=0}^{2D_w} a_j \delta[\mathbf{w}_t - \hat{\mathbf{w}}^{[j]}],\end{aligned}\quad (19.43)$$

where we choose the centers of the sigma points and the weights so that

$$\begin{aligned}\boldsymbol{\mu}_+ &= \sum_{j=0}^{2D_w} a_j \hat{\mathbf{w}}^{[j]} \\ \boldsymbol{\Sigma}_+ &= \sum_{j=0}^{2D_w} a_j (\hat{\mathbf{w}}^{[j]} - \boldsymbol{\mu}_+) (\hat{\mathbf{w}}^{[j]} - \boldsymbol{\mu}_+)^T.\end{aligned}\quad (19.44)$$

For example, we could use the scheme outlined in equations 19.40 and 19.41.

Then the sigma points are passed through the measurement model  $\hat{\mathbf{x}}^{[j]} = \mathbf{g}[\hat{\mathbf{w}}^{[j]}]$  to create a new set of points  $\{\hat{\mathbf{x}}^{[j]}\}_{j=0}^{2D_w}$  in the measurement space. We compute the mean and covariance of these predicted measurements using the relations

$$\begin{aligned}\boldsymbol{\mu}_x &= \sum_{j=0}^{2D_w} a_j \hat{\mathbf{x}}^{[j]} \\ \boldsymbol{\Sigma}_x &= \sum_{j=0}^{2D_w} a_j (\hat{\mathbf{x}}^{[j]} - \boldsymbol{\mu}_x) (\hat{\mathbf{x}}^{[j]} - \boldsymbol{\mu}_x)^T + \boldsymbol{\Sigma}_m\end{aligned}\quad (19.45)$$

The measurement incorporation equations are now

$$\begin{aligned}\boldsymbol{\mu}_t &= \boldsymbol{\mu}_+ + \mathbf{K}(\mathbf{x}_t - \boldsymbol{\mu}_x) \\ \boldsymbol{\Sigma}_t &= \boldsymbol{\Sigma}_+ - \mathbf{K}\boldsymbol{\Sigma}_x\mathbf{K}^T,\end{aligned}\quad (19.46)$$

where the Kalman gain  $\mathbf{K}$  is now re-defined as

$$\mathbf{K} = \left( \sum_{j=0}^{2D_w} a_j (\hat{\mathbf{w}}^{[j]} - \boldsymbol{\mu}_+)^T (\hat{\mathbf{x}}^{[j]} - \boldsymbol{\mu}_x)^T \right) \boldsymbol{\Sigma}_x^{-1}. \quad (19.47)$$

As for the prediction step, it can be shown that the UKF approximation is better than that of the EKF.

## 19.5 Particle filtering

The extended and unscented Kalman filters can partially cope with nonlinear temporal and measurement models. However, they both represent the uncertainty over the state as a normal distribution. They are hence ill-equipped to deal with situations where the true probability distribution over the state is multi-modal. Figure 19.12 illustrates a temporal model that maps nearby states into two distinct regions. In this situation, neither the EKF nor the UKF suffice: the EKF models only one of the resulting clusters and the UKF tries to model both with a single normal model, assigning a large probability to the empty region between the clusters.

Particle filtering resolves this problem by representing the probability density as a set of particles in the state space. Each particle can be thought of as representing a hypothesis about the possible state. When the state is tightly constrained by the data, all of these particles will lie close to one another. In more ambiguous cases, they will be widely distributed or clustered into groups of competing hypotheses.

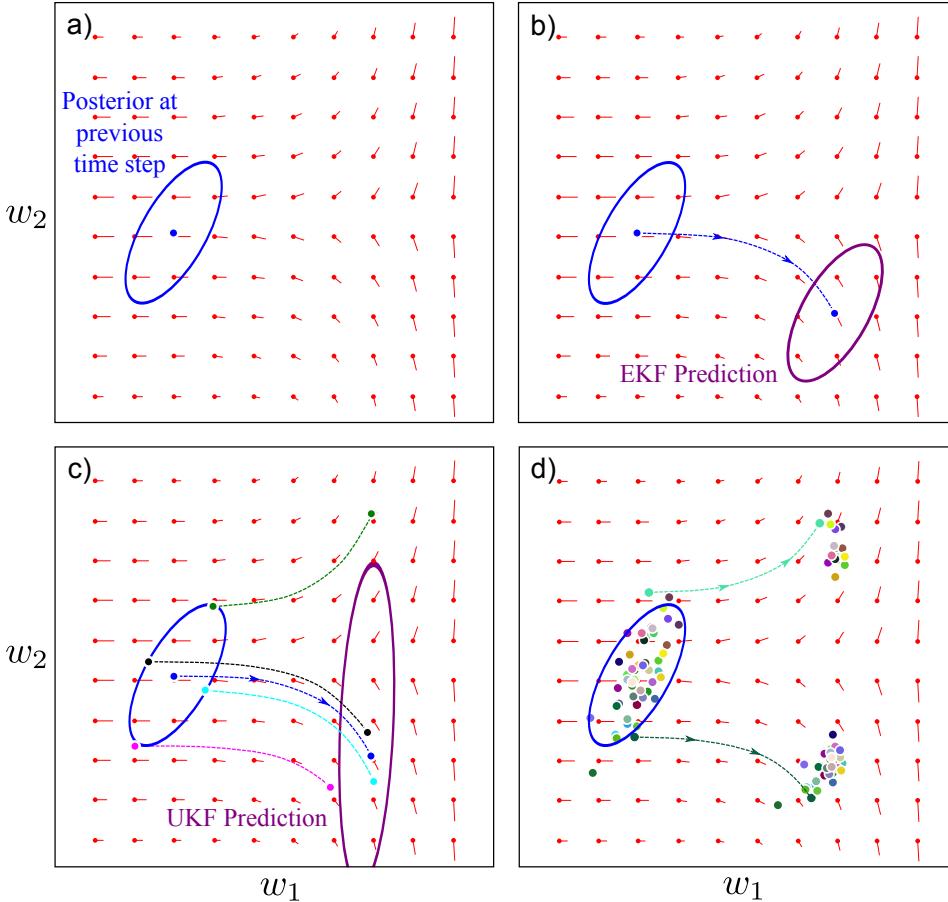
The particles can be evolved through time or projected down to simulate measurements regardless of how nonlinear the functions are. The latter exercise leads us to another nice property of the particle filter: since the state is multi-modal, so are the predicted measurements. In turn, the measurement density may also be multi-modal. In vision systems, this means that the system copes much better with clutter in the scene (figure 19.14). As long as some of the predicted measurements agree with the measurement density, the tracker should remain stable.

One of the simplest particle filter methods is the *conditional density propagation* or *condensation* algorithm. The probability distribution  $Pr(\mathbf{w}_{t-1}|\mathbf{x}_{1\dots t-1})$  is represented by a weighted sum of  $J$  weighted particles:

$$Pr(\mathbf{w}_{t-1}|\mathbf{x}_{1\dots t-1}) = \sum_{j=1}^J a_j \delta[\mathbf{w}_{t-1} - \hat{\mathbf{w}}_{t-1}^{[j]}], \quad (19.48)$$

where the weights are positive and sum to one. Each particle represents a hypothesis about the state, and the weight of the particle indicates our confidence in that

Algorithm 19.6



**Figure 19.12** The need for particle filtering. a) Consider this new temporal update function (change of state with time indicated by red pointers), which bifurcates around the horizontal midline. b) With the EKF, the predicted distribution after the time update is toward the bottom, as the initial mean was below the bifurcation. The linear approximation is not good here, and so the covariance estimate is inaccurate. c) In the UKF, one of the sigma points that approximates the prior is above the midline, so it is moved upward (green dashed line), whereas the others are below the midline and are moved downward (other dashed lines). The estimated covariance is very large. d) We can get an idea of the true predicted distribution by sampling the posterior at the previous time-step, and passing the samples through the nonlinear temporal model. It is clear that the resulting distribution is bimodal and can never be well approximated with a normal distribution. The particle filter represents the distribution in terms of particles throughout the tracking process, and so it can describe multimodal distributions like this.

hypothesis. Our goal is to compute the probability distribution  $Pr(\mathbf{w}_t | \mathbf{x}_{1\dots t})$  at the next time step, which will be represented in a similar fashion. As usual, this process is divided into a time evolution and a measurement incorporation step, which we now consider in turn.

### 19.5.1 Time evolution

In the time evolution step, we create  $J$  predictions  $\hat{\mathbf{w}}_+^{[j]}$  for the time evolved state. Each is represented by an *unweighted particle*. We create each prediction by resampling so that we

- choose an index  $n \in \{1 \dots J\}$  of the original weighted particles where the probability is according to the weights. In other words we draw a sample from  $\text{Cat}_n[\mathbf{a}]$ , and
- draw the sample  $\hat{\mathbf{w}}_+^{[j]}$  from the temporal update distribution  $Pr(\mathbf{w}_t | \mathbf{w}_{t-1} = \hat{\mathbf{w}}_{t-1}^{[n]})$  (equation 19.37).

In this process, the final unweighted particles  $\hat{\mathbf{w}}_+^{[j]}$  are created from the original weighted particles  $\hat{\mathbf{w}}_{t-1}^{[j]}$  according to the weights  $\mathbf{a} = [a_1, a_2 \dots a_J]$ . Hence, the highest weighted original particles may contribute repeatedly to the final set, and the lowest weighted ones may not contribute at all.

### 19.5.2 Measurement incorporation

In the measurement incorporation step we weight the new set of particles according to how well they agree with the observed data. To this end, we

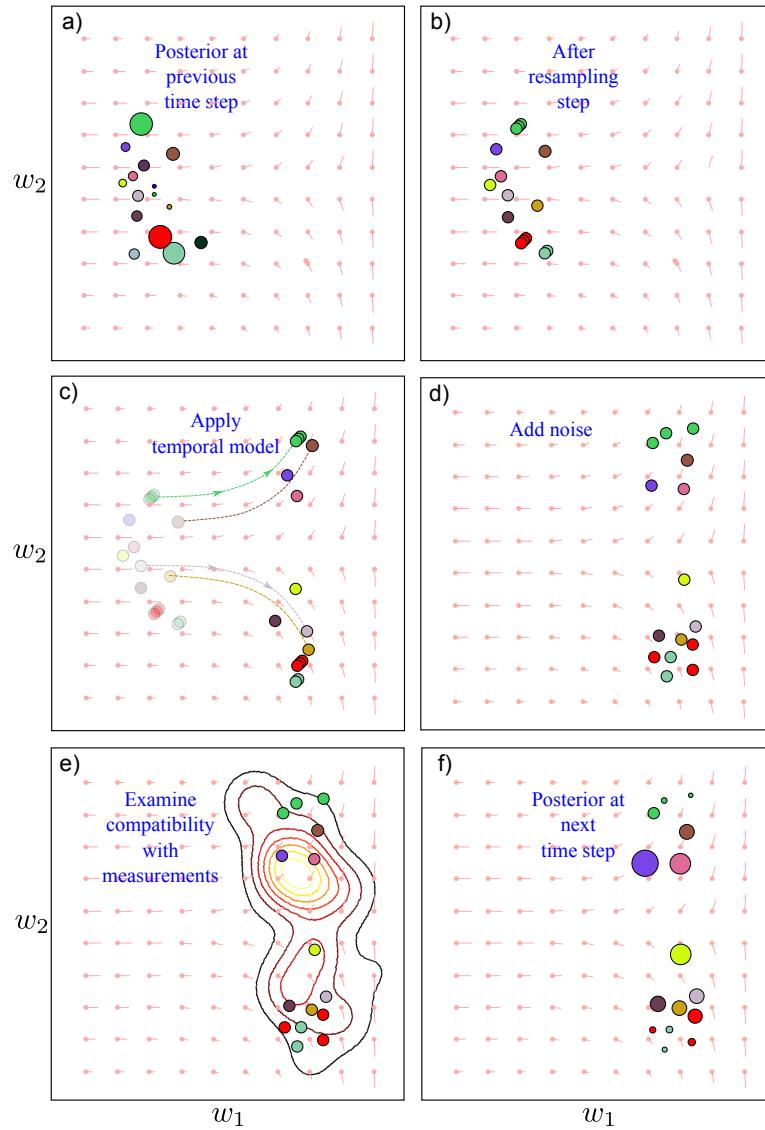
- pass the particles through the measurement model  $\hat{\mathbf{x}}_+^{[j]} = \mathbf{g}[\hat{\mathbf{w}}_+^{[j]}]$ .
- weight the particles according to their agreement with the observation density. For example, with a Gaussian measurement model, we could use

$$a_j \propto \text{Norm}_{\mathbf{x}_t}[\hat{\mathbf{x}}_+^{[j]}, \Sigma_m]. \quad (19.49)$$

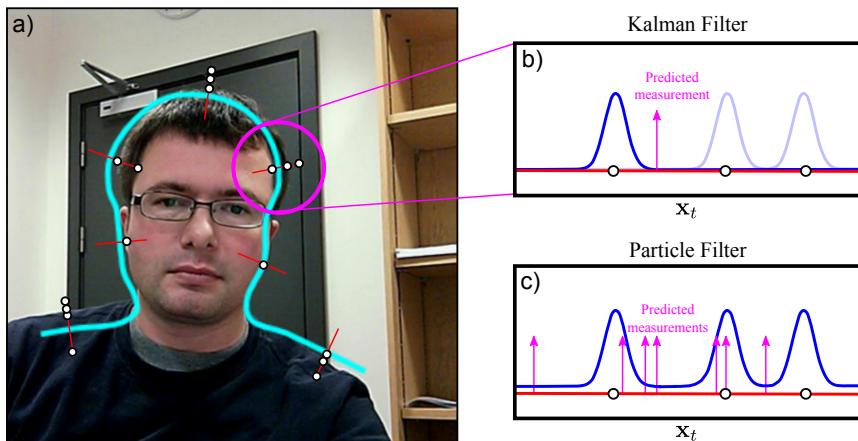
- normalize the resulting weights  $\{a_j\}_{j=1}^J$  to sum to one.
- Finally, we set the new states  $\hat{\mathbf{w}}_t^{[j]}$  to the predicted states  $\hat{\mathbf{w}}_+^{[j]}$  and the new weights to  $a_j$ .

Figure 19.13 demonstrates the action of the particle filter. The filter copes elegantly with the multi-modal probability distribution. It is also suitable for situations where it is not obvious which aspect of the data is the true measurement. This is known as the *data association* problem (figure 19.14).

The main disadvantage of particle filters is their cost: in high dimensions, a very large number of particles may be required to get an accurate representation of the true distribution over the state.



**Figure 19.13** The condensation algorithm. a) The posterior at the previous step is represented as a set of weighted particles. b) The particles are re-sampled according to their weights to produce a new set of unweighted particles. c) These particles are passed through the nonlinear temporal function. d) Noise is added according to the temporal model. e) The particles are passed through the measurement model and compared to the measurement density. f) The particles are re-weighted according to their compatibility with the measurements, and the process can begin again.

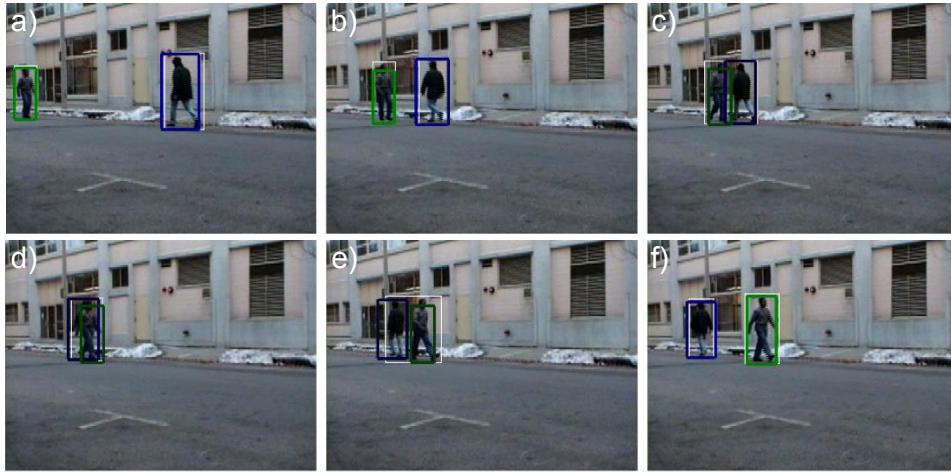


**Figure 19.14** Data association. a) Consider the problem of tracking this contour. The contour from the previous frame is shown in cyan, and the problem is to incorporate the measurements in this frame. These measurements take the form of edges along one-dimensional slices perpendicular to the contour. In most cases, there are multiple possible edges that might be due to the contour. b) In the Kalman filter, the measurement density is constrained to be normal: we are forced to choose between the different possible hypotheses. A sensible way to do this is to select the one that is closest to the predicted measurement. This is known as data association. c) In the particle filter, there are multiple predicted measurements and it is not necessary that the measurement density is normal. We can effectively take into account all of the possible measurements.

### 19.5.3 Extensions

There are many variations and extensions to the particle filter. In many schemes, the particles are not re-sampled on each iteration, but only occasionally so the main scheme operates with weighted particles. The re-sampling process can also be improved by applying *importance sampling*. Here, the new samples are generated in concert with the measurement process, so that particles that agree with the measurements are more likely to be produced. This helps prevent the situation where none of the unweighted samples after the prediction stage agree with the measurements.

The process of *Rao-Blackwellization* partitions the state into two subsets of variables. The first subset is tracked using a particle filter, but the other subset is conditioned on the first subset and evaluated analytically using a process more like the Kalman filter. Careful choice of these subsets can result in a tracking algorithm that is both efficient and accurate.



**Figure 19.15** Pedestrian tracking results. a-f) Six frames from a sequence in which a 3D bounding plane of constant depth was used to track each person. Adapted from Rosales & Sclaroff (1999). ©1999 IEEE.

## 19.6 Applications

Tracking algorithms can be used in combination with any model that is being re-estimated over a time sequence. For example, they are often used in combination with 3D body models to track the pose of a person in video footage. In this section, we will describe three example applications. First we will consider tracking the 3D position of a pedestrian in a scene. Second, we will describe *simultaneous localization and mapping (SLAM)* in which both a 3D representation of a scene and the pose of the camera viewing it are estimated through a time sequence. Finally, we will consider contour tracking for an object with complex motion against a cluttered background.

### 19.6.1 Pedestrian Tracking

Rosales & Sclaroff (1999) described a system to track pedestrians in 2D from a fixed static camera using the extended Kalman filter. For each frame, the measurements  $\mathbf{x}$  consisted of a 2D bounding box  $\mathbf{x} = \{x_1, y_1, x_2, y_2\}$  around the pedestrian. This was found by segmenting the pedestrian from the background using a background subtraction model, and finding a connected region of foreground pixels.

The state of the world  $\mathbf{w}$  was considered to be the 3D position of a bounding box of constant depth  $\{u_1, v_1, u_2, v_2, w\}$  and the velocities associated with these five quantities. The state update equation assumed first-order Newtonian dynamics in 3D space. The relationship between the state and measurements was

Problem 19.10

$$\begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \end{bmatrix} \frac{1}{1+w} + \epsilon, \quad (19.50)$$

which mimics the nonlinear action of the pinhole camera.

Results from the tracking procedure are illustrated in figure 19.15. The system successfully tracks pedestrians and was extended to cope with occlusions (which are predicted by the EKF based on the estimated trajectory of the objects). In the full system, the estimated bounding boxes are used to warp the images of the pedestrians to a fixed size, and the resulting registered images were used as input to an action recognition algorithm

### 19.6.2 Monocular SLAM

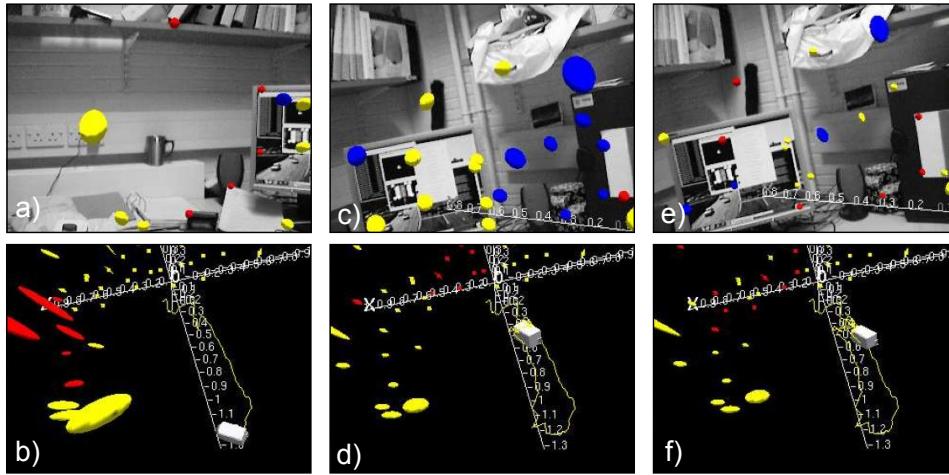
The goal of *simultaneous localization and mapping* or *SLAM* is to build a map of an unknown environment based on measurements from a moving sensor (usually attached to a robot), and to establish where the sensor is in the world at any given time. Hence, the system must simultaneously estimate the structure of the world and its own position and orientation within that model. SLAM was originally based on range sensors, but in the last decade it has become possible to build systems that work in real-time based on a monocular camera. In essence, it is a real-time version of the sparse 3D reconstruction algorithms discussed in chapter 16.

Davison *et al.* (2007) presented one such system which was based on the extended Kalman filter. Here the world state  $\mathbf{w}$  contains

- the camera position  $(u, v, w)$ ,
- the orientation of the camera as represented by a 4D quaternion  $\mathbf{q}$ ,
- the velocity and angular velocity, and
- a set of 3D points  $\{\mathbf{p}_k\}$  where  $\mathbf{p}_k = [p_{uk}, p_{vk}, p_{wk}]$ . This set generally expands during the sequence as more parts of the world are seen.

The state update equation modifies the position and orientation according to their respective velocities, and allows these velocities themselves to change. The observation equation maps each 3D point through a pinhole camera model to create a predicted 2D image position (i.e., similar to equation 19.50). The actual measurements consist of interest points in the image, which are uniquely identified and associated with the predicted measurements by considering their surrounding region. The system is very efficient as it is only necessary to search the image in regions close to the 2D point predicted by the measurement equation from the current state.

The full system is more complex, and includes special procedures for initializing new feature points, modeling the local region of each point as a plane, selecting which features to measure in this frame, and managing the resulting map by deleting extraneous features to reduce the complexity. A series of frames from the system is illustrated in figure 19.16.

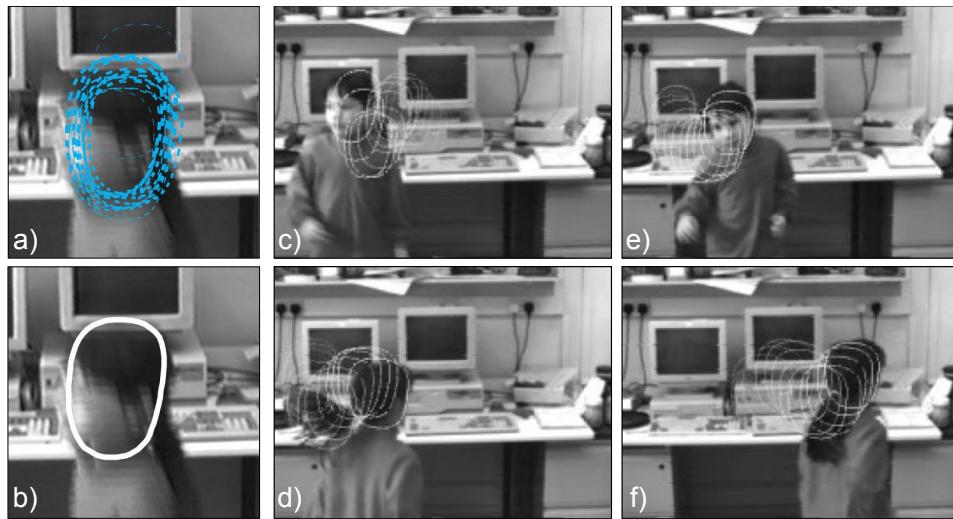


**Figure 19.16** Monocular SLAM model of Davison *et al.* (2007). a) Current view of world with visual features and their 3D covariance ellipse superimposed. Yellow points are features in the model that were measured. Blue points are features in the model that were not detected in this frame. Red points are new features that can potentially be incorporated into the model. b) Model of world at the time this frame was captured, showing the position of the camera and the position and uncertainty of the visual features. c-f) Two more pairs of images and associated models from later in the same time sequence. Notice that the uncertainty in the feature positions gradually decreases over time. Adapted from Davison *et al.* (2007). ©2007 IEEE.

### 19.6.3 Tracking a contour through clutter

In section 17.3 we discussed fitting a template shape model to an image. It was noted that this was a challenging problem as the template frequently fell into local minima when the object was observed in clutter. In principle, this problem should be easier in a temporal sequence; if we know the template position in the current frame, then we can hypothesize quite accurately where it is in the next frame. Nonetheless, tracking methods based on the Kalman filter still fail to maintain a lock on the object; sooner or later, part of the contour becomes erroneously associated with the background and the tracking fails.

Blake & Isard (1998) describe a system based on the condensation algorithm that can cope with tracking a contour undergoing rapid motions against a cluttered background. The state of the system consists of the parameters of an affine transform that maps the template into the scene and the measurements are similar to those illustrated in figure 19.14. The state is represented as 100 weighted particles, each of which represents a different hypothesis about the current position of the template. This system was used to track the head of a child dancing in front of a cluttered background (figure 19.17).



**Figure 19.17** Tracking a contour through clutter. a) The representation of uncertainty about the contour consists of a set of weighted particles, each of which represents a hypothesis about the current position. In this panel, their weight indicates our relative belief that the system takes this state. b) The overall estimate of the current position can be illustrated by taking a weighted mean of the hypotheses. c-f) Four frames of sequence tracked by condensation algorithm showing the current estimate and the estimates from previous frames. Adapted from Blake & Isard (1998). ©1998 Springer.

## Discussion

In this chapter we have discussed a family of models that allow us to estimate a set of continuous parameters throughout a time sequence, by exploiting the temporal coherence of the underlying state. In principle, these methods can be applied to any model in this book that estimates a set of parameters from a single frame. These models have a close relationship with the chain-based models discussed in chapter 11. The latter models used discrete parameters and generally estimated the state from the whole sequence rather than just the observed data up until the current time.

## Notes

**Applications of tracking:** Tracking models are used for a variety of tasks in vision including tracking pedestrians (Rosales & Sclaroff 1999; Beymer & Konolige 1999), contours (Terzopoulos & Szeliski 1992; Blake *et al.* 1993; Blake *et al.* 1995; Blake & Isard 1996; Blake & Isard 1998), points (Broida & Chellappa 1986), 3D hand models (Stenger *et al.* 2001b) and 3D body models (Wang *et al.* 2008). They have also been used for activity recognition (Vaswani *et al.* 2003), estimating depth (Matthies *et al.* 1989), SLAM (Davison *et al.* 2007), and object recognition (Zhou *et al.* 2004). Reviews of tracking methods and applications can be found in Blake (2006) and Yilmaz *et al.* (2006). Approaches to tracking the human body are reviewed in Poppe (2007).

**Tracking models:** The Kalman filter was originally developed by Kalman (1960) and Kalman & Bucy (1961). The unscented Kalman filter was developed by Julier & Uhlmann (1997). The condensation algorithm is due to Blake & Isard (1996). For more information about the Kalman filter and its variants, consult Maybeck (1990), Gelb (1974) and Jazwinski (1970). Roweis & Ghahramani (1999) provide a unifying review of linear models which provides information about how the Kalman filter relates to other methods. Arulampalam *et al.* (2002) provide a detailed review of the use of particle filters. A summary of tracking models and different approaches to inference within them can be found in Minka (2002).

There are a large number of variants on the standard Kalman filter. Many of these involve switching between different state space models or propagating mixtures (Shumway & Stoffer 1991; Ghahramani & Hinton 1996b; Murphy 1998; Chen & Liu 2000; Isard & Blake 1998). A notable recent extension has been to develop a nonlinear tracking algorithm based on the GPLVM (Wang *et al.* 2008).

One topic that has not been discussed in this chapter is how to learn the parameters of the tracking models. In practice, it is not uncommon to set these parameters by hand. However, information about learning in these temporal models can be found in Shumway & Stoffer (2007), Ghahramani & Hinton (1996a), Roweis & Ghahramani (2001), and Oh *et al.* (2005).

**Simultaneous localization and mapping (SLAM):** Simultaneous localization and mapping has its roots in the robotics community who were concerned with the representation of spatial uncertainty by vehicles exploring an environment (Durrant-Whyte 1988; Smith & Cheeseman 1987) although the term SLAM was coined much later by Durrant-Whyte *et al.* (1996). Smith *et al.* (1990) had the important insight that the errors in mapped positions were correlated due to uncertainty in the camera position. The roots of vision-based SLAM are to be found in the pioneering work of Harris & Pike (1987), Ayache (1991), and Beardsley *et al.* (1995).

SLAM systems are usually based on either the extended Kalman filter (Guivant & Nebot 2001; Leonard & Feder 2000; Davison *et al.* 2007) or a Rao-Blackwellized particle filter (Montemerlo *et al.* 2002; Montemerlo *et al.* 2003; Sim *et al.* 2005). However, it currently the subject of some debate as to whether a tracking method of this type is necessary at all, or whether repeated bundle adjustments on tactically chosen subsets of 3D points will suffice (Strasdat *et al.* 2010).

Recent examples of efficient visual SLAM systems can be found in Nistér *et al.* (2004), Davison *et al.* (2007), Klein & Murray (2007), Mei *et al.* (2009), and Newcombe *et al.* (2011), and most of these include a bundle adjustment procedure into the algorithm. Current research issues in SLAM include how to efficiently match features in the image with features in the current model (Handa *et al.* 2010) and how to close loops (i.e.,

recognize that the robot has returned to a familiar place) in a map (Newman & Ho 2005). A review of SLAM techniques can be found in Durrant-Whyte & Bailey (2006) and Bailey & Durrant-Whyte (2006).

## Problems

**Problem 19.1** Prove the Chapman-Kolmogorov relation:

$$\begin{aligned} Pr(\mathbf{w}_t | \mathbf{x}_{1\dots t-1}) &= \int Pr(\mathbf{w}_t | \mathbf{w}_{t-1}) Pr(\mathbf{w}_{t-1} | \mathbf{x}_{1\dots t-1}) d\mathbf{w}_{t-1} \\ &= \int \text{Norm}_{\mathbf{w}_t}[\boldsymbol{\mu}_p + \boldsymbol{\Psi}\mathbf{w}_{t-1}, \boldsymbol{\Sigma}_p] \text{Norm}_{\mathbf{w}_{t-1}}[\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1}] d\mathbf{w}_{t-1} \\ &= \text{Norm}_{\mathbf{w}_t}[\boldsymbol{\mu}_p + \boldsymbol{\Psi}\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_p + \boldsymbol{\Psi}\boldsymbol{\Sigma}_{t-1}\boldsymbol{\Psi}^T] \end{aligned}$$

**Problem 19.2** Derive the measurement incorporation step for the Kalman filter. In other words, show that

$$\begin{aligned} Pr(\mathbf{w}_t | \mathbf{x}_{1\dots t}) &= \frac{Pr(\mathbf{x}_t | \mathbf{w}_t) Pr(\mathbf{w}_t | \mathbf{w}_{1\dots t-1}, \mathbf{x}_{1\dots t})}{Pr(\mathbf{x}_{1\dots t})} \\ &= \frac{\text{Norm}_{\mathbf{x}_t}[\boldsymbol{\mu}_m + \boldsymbol{\Phi}\mathbf{w}_t, \boldsymbol{\Sigma}_m] \text{Norm}_{\mathbf{w}_t}[\boldsymbol{\mu}_+, \boldsymbol{\Sigma}_+]}{Pr(\mathbf{x}_{1\dots t})} \\ &= \text{Norm}_{\mathbf{w}_t} \left[ \left( \boldsymbol{\Phi}^T \boldsymbol{\Sigma}_m^{-1} \boldsymbol{\Phi} + \boldsymbol{\Sigma}_+^{-1} \right)^{-1} \left( \boldsymbol{\Phi}^T \boldsymbol{\Sigma}_m^{-1} (\mathbf{x}_t - \boldsymbol{\mu}_m) + \boldsymbol{\Sigma}_+^{-1} \boldsymbol{\mu}_+ \right), \right. \\ &\quad \left. \left( \boldsymbol{\Phi}^T \boldsymbol{\Sigma}_m^{-1} \boldsymbol{\Phi} + \boldsymbol{\Sigma}_+^{-1} \right)^{-1} \right]. \end{aligned}$$

**Problem 19.3** Consider a system similar to the Kalman filter but where the prior based on the previous time step is a mixture of  $K$  Gaussians

$$Pr(\mathbf{w}_t | \mathbf{x}_{1\dots t-1}) = \sum_{k=1}^K \lambda_k \text{Norm}_{\mathbf{w}_t}[\boldsymbol{\mu}_{+k}, \boldsymbol{\Sigma}_{+k}].$$

What will happen in the subsequent measurement incorporation step? What will happen in the next time update step?

**Problem 19.4** Consider a model where there are two possible temporal update equations, represented by state transition matrices  $\boldsymbol{\Psi}_1$  and  $\boldsymbol{\Psi}_2$  and the system periodically switches from one regime to the other. Write a set of equations that describe this model and discuss a strategy for max-marginals inference.

**Problem 19.5** In a Kalman filter model, discuss how you would compute the joint posterior distribution  $Pr(\mathbf{w}_{1\dots T} | \mathbf{x}_{1\dots T})$  over all of the unknown world states. What form will this posterior distribution take? In the Kalman filter we choose to compute the marginal posteriors instead. Why is this?

**Problem 19.6** Apply the sum-product algorithm (section 11.4.3) to the Kalman filter model and show that the result is equivalent to applying the Kalman filter recursions.

**Problem 19.7** Prove the Kalman smoother recursions:

$$\begin{aligned}\boldsymbol{\mu}_{t|T} &= \boldsymbol{\mu}_t + \mathbf{C}_t(\boldsymbol{\mu}_{t+1|T} - \boldsymbol{\mu}_{+|t}) \\ \boldsymbol{\Sigma}_{t|T} &= \boldsymbol{\Sigma}_t + \mathbf{C}_t(\boldsymbol{\Sigma}_{t+1|T} - \boldsymbol{\Sigma}_{+|t})\mathbf{C}_t^T,\end{aligned}$$

where

$$\mathbf{C}_t = \boldsymbol{\Sigma}_{t|t}\boldsymbol{\Psi}^T\boldsymbol{\Sigma}_{+|t}^{-1}.$$

Hint: It may help to examine the proof of the forward-backward algorithm for HMMs (section 11.4.2).

**Problem 19.8** Discuss how you would learn the parameters of the Kalman filter model given training sequences consisting of (i) both the known world state and the observed data, and (ii) just the observed data alone.

**Problem 19.9** In the unscented Kalman filter we represented a Gaussian with mean  $\boldsymbol{\mu}$  and covariance  $\boldsymbol{\Sigma}$  with a set of weights

$$\begin{aligned}\hat{\mathbf{w}}^{[0]} &= \boldsymbol{\mu}_{t-1} \\ \hat{\mathbf{w}}^{[j]} &= \boldsymbol{\mu}_{t-1} + \sqrt{\frac{D_{\mathbf{w}}}{1-a_0}}\boldsymbol{\Sigma}_{t-1}^{1/2}\mathbf{e}_j \quad \text{for all } j \in \{1 \dots D_{\mathbf{w}}\} \\ \hat{\mathbf{w}}^{[D_{\mathbf{w}}+j]} &= \boldsymbol{\mu}_{t-1} - \sqrt{\frac{D_{\mathbf{w}}}{1-a_0}}\boldsymbol{\Sigma}_{t-1}^{1/2}\mathbf{e}_j \quad \text{for all } j \in \{1 \dots D_{\mathbf{w}}\},\end{aligned}$$

with associated weights

$$a_j = \frac{1-a_0}{2D_{\mathbf{w}}}.$$

Show that the mean and covariance of these points are indeed  $\boldsymbol{\mu}_{t-1}$  and  $\boldsymbol{\Sigma}_{t-1}$  so that

$$\begin{aligned}\boldsymbol{\mu}_{t-1} &= \sum_{j=0}^{2D_{\mathbf{w}}} a_j \hat{\mathbf{w}}^{[j]} \\ \boldsymbol{\Sigma}_{t-1} &= \sum_{j=0}^{2D_{\mathbf{w}}} a_j (\hat{\mathbf{w}}^{[j]} - \boldsymbol{\mu}_{t-1})(\hat{\mathbf{w}}^{[j]} - \boldsymbol{\mu}_{t-1})^T.\end{aligned}$$

**Problem 19.10** The extended Kalman filter requires the Jacobian matrix describing how small changes in the data create small changes in the measurements. Compute the Jacobian matrix for the measurement model for the pedestrian-tracking application (equation 19.50).



# Chapter 20

## Models for visual words

In most of the models in this book, the observed data are treated as continuous. Hence, for generative models the data likelihood is usually based on the normal distribution. In this chapter, we explore generative models that treat the observed data as discrete. The data likelihoods are now based on the categorical distribution; they describe the probability of observing the different possible values of the discrete variable.

As a motivating example for the models in this chapter, consider the problem of *scene classification* (figure 20.1). We are given example training images of different scene categories (e.g., office, coastline, forest, mountain) and we are asked to learn a model that can classify new examples. Studying the scenes in figure 20.1 demonstrates how challenging a problem this is. Different images of the same scene may have very little in common with one another, yet we must somehow learn to identify them as the same. We will also discuss object recognition, which has many of the same characteristics; the appearance of an object such as a tree, bicycle, or chair can vary dramatically from one image to another, and we must somehow capture this variation.

The key to modeling these complex scenes is to encode the image as a collection of *visual words*, and use the frequencies with which these words occur as the substrate for further calculations. We start this chapter by describing this transformation.

### 20.1 Images as collections of visual words

To encode an image in terms of visual words, we need first to establish a *dictionary*. This is computed from a large set of training images that are unlabeled, but known to contain examples of all of the scenes or objects that will ultimately be classified. To compute the dictionary, we take the following steps:

1. For every one of the  $I$  training images, select a set of  $J_i$  spatial locations. One possibility is to identify interest points (section 13.2) in the image. Al-



**Figure 20.1** Scene recognition. The goal of scene recognition is to assign a discrete category to an image according to the type or content. In this case, the data includes images of a) street scenes, b) the sea, and c) forests. Scene recognition is a useful precursor to object recognition; if we know that the scene is a street, then the probability of a car being present is high, but the probability of a boat being present is small. Unfortunately, scene recognition is quite a challenging task in itself. Different examples from the same scene class may have very little in common visually.

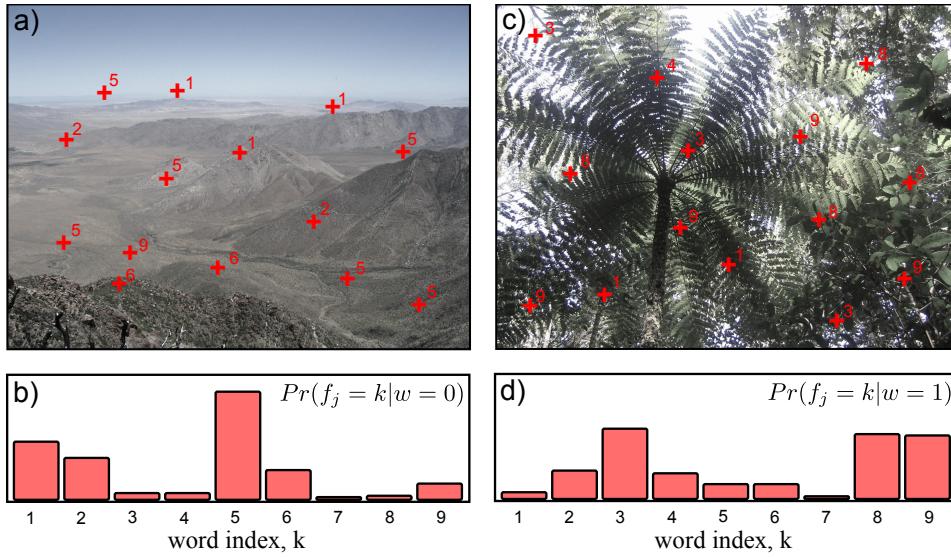
ternately, the image can be sampled in a regular grid.

2. Compute a descriptor at each spatial location in each image that characterizes the surrounding region with a low dimensional vector. For example, we might compute the SIFT descriptor (section 13.3.2).
3. Cluster all of these descriptor vectors into  $K$  groups using a method such as the K-means algorithm (section 13.4.4).
4. The means of the  $K$  clusters are used as the  $K$  prototype vectors in the dictionary.

Typically, several hundred thousand descriptors would be used to compute the dictionary, which might consist of several hundred prototype words.

Having computed the dictionary, we are now in a position to take a new image and convert it into a set of visual words. To compute the visual words, we take the following steps:

1. Select a set of  $J$  spatial locations in the image using the same method as for the dictionary.
2. Compute the descriptor at each of the  $J$  spatial locations.
3. Compare each descriptor to the set of  $K$  prototype descriptors in the dictionary and find the closest prototype (visual word).



**Figure 20.2** Scene recognition using bags of words. a) A set of interest points are found in this desert scene and a descriptor is calculated at each. These descriptors are compared to a dictionary containing  $K$  prototypes and the index of the nearest prototype is chosen (red numbers). Here  $K = 9$  but in real applications it might be several hundred. b) The scene-type ‘desert’ implies a certain distribution over the observed visual words. c) A second image containing a jungle scene and the associated visual words. d) The scene type ‘jungle’ implies a different distribution over the visual words. A new image can be classified as belonging to one scene type or another by assessing the likelihood that the observed visual words were drawn from the ‘desert’ or ‘jungle’ distribution.

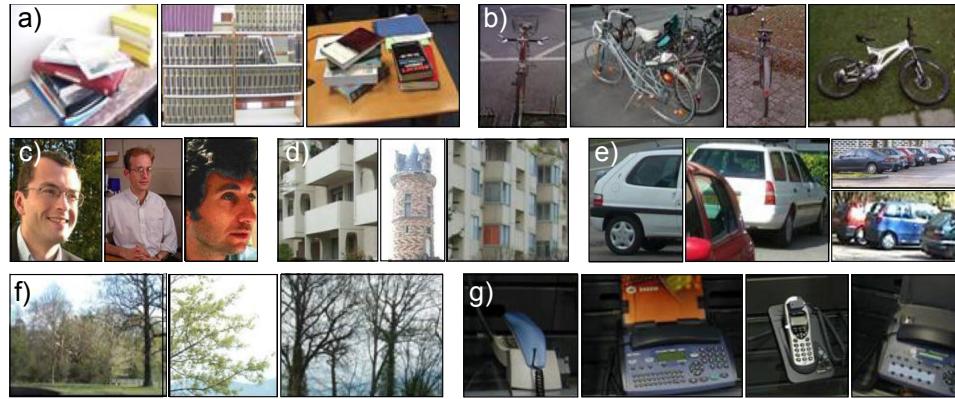
4. Assign to this location a discrete index that corresponds to the index of the closest word in the dictionary.

After computing the visual words, the data  $\mathbf{x}$  from a single image consist of a set  $\mathbf{x} = \{f_j, x_j, y_j\}_{j=1}^J$  of  $J$  word indices  $f_j \in \{1 \dots K\}$ , and their 2D image positions  $(x_j, y_j)$ . This is a highly compressed representation that nonetheless contains the critical information about the image appearance and layout. In the remaining part of the chapter we will develop a series of generative models that attempt to explain the pattern of this data when different objects or scenes are present.

## 20.2 Bag of words

One of the simplest possible representations for an image in terms of visual words is the *bag of words*. Here we entirely discard the spatial information held in the

Algorithm 20.1



**Figure 20.3** Object recognition using bag of words. Csurka *et al.* (2004) built a generative bag of visual words model to distinguish between examples of a) books, b) bicycles, c) people, d) buildings, e) cars, f) trees, and g) phones. Despite the wide variety of visual appearance within each class, they achieved 72% correct classification. By applying a discriminative approach to the same problem, they managed to improve performance further.

word positions  $(x_j, y_j)$  and just retain the word indices  $f_j$  so that the observed data are  $\mathbf{x} = \{f_j\}_{j=1}^J$ . In other words, the image is simply represented by the frequency with which each word appears. It is assumed that different types of object or scene will tend to contain different words and that this can be exploited to perform scene or object recognition (figure 20.2).

More formally, the goal is to infer a discrete variable  $w \in \{1, 2, \dots, N\}$  indicating which of  $N$  classes is present in this image. We take a generative approach. Since the data  $\{f_j\}_{j=1}^J$  are discrete, we describe its probability with a categorical distribution and make the parameters  $\boldsymbol{\lambda}$  of this distribution a function of the discrete world state.

$$\begin{aligned} Pr(\mathbf{x}|w = n) &= \prod_{j=1}^J \text{Cat}_{f_j}[\boldsymbol{\lambda}_n] \\ &= \prod_{k=1}^K \lambda_{kn}^{T_k}, \end{aligned} \quad (20.1)$$

where  $T_k$  is the total number of times that the  $k^{th}$  word was observed, so that

$$T_k = \sum_{j=1}^J \delta[f_j - k]. \quad (20.2)$$

We will now consider the learning and inference algorithms for this model.

### 20.2.1 Learning

In learning, our goal is to estimate the parameters  $\{\boldsymbol{\lambda}_n\}_{n=1}^N$  based on labeled pairs  $\{\mathbf{x}_i, w_i\}$  of the observed data  $\mathbf{x}_i = \{f_{ij}\}_{j=1}^{J_i}$  and the world state  $w_i$ . We note that the  $n^{th}$  parameter vector  $\boldsymbol{\lambda}_n$  is used only when the world state  $w_i = n$ . Hence, we can learn each parameter vector separately; we learn the parameter  $\boldsymbol{\lambda}_n$  from the subset  $\mathcal{S}_n$  of training images where  $w_i = n$ .

Making use of the results in section 4.5, we see that if we apply a Dirichlet prior with uniform parameter  $\boldsymbol{\alpha} = [\alpha, \alpha, \dots, \alpha]$ , then the MAP estimate of the categorical parameters is given by

$$\hat{\lambda}_{nk} = \frac{\sum_{i \in \mathcal{S}_n} T_{ik} + \alpha - 1}{\sum_{k=1}^K (\sum_{i \in \mathcal{S}_n} T_{ik} + \alpha - 1)}, \quad (20.3)$$

where  $\lambda_{nk}$  is the  $k^{th}$  entry in the categorical distribution for the  $n^{th}$  class, and  $T_{ik}$  is the total number of times that word  $k$  was observed in the  $i^{th}$  training image.

### 20.2.2 Inference

To infer the world state, we apply Bayes' rule:

$$Pr(w = n | \mathbf{x}) = \frac{Pr(\mathbf{x} | w = n) Pr(w = n)}{\sum_{k=1}^K Pr(\mathbf{x} | w = k) Pr(w = k)}, \quad (20.4)$$

where we allocate suitable prior probabilities  $Pr(w = n)$  according to the relative frequencies with which each world type is present.

#### Discussion

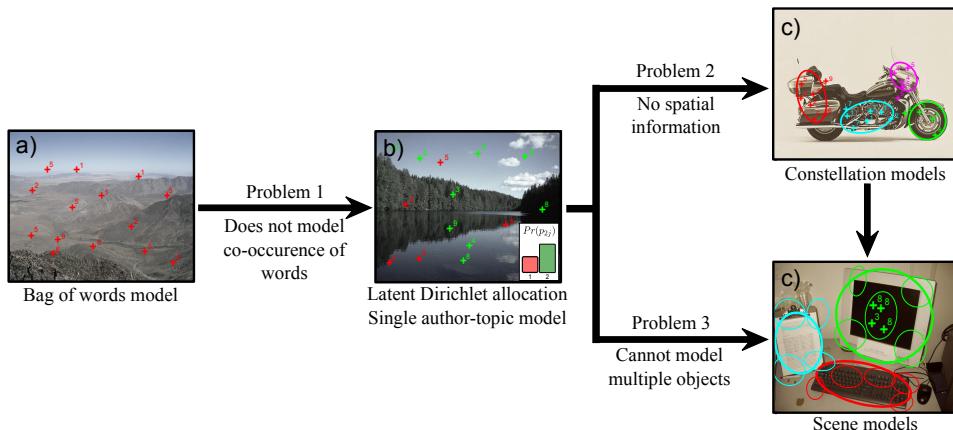
Despite discarding all spatial information, the bag of words model works remarkably well for object recognition. For example, Csurka *et al.* (2004) achieved 72% correct performance at classifying images of the seven classes found in figure 20.3. It should be noted that it is possible to improve performance further by treating the vector  $\mathbf{z} = [T_1, T_2, \dots, T_K] / \sum_k T_k$  of normalized word frequencies as continuous and subjecting it to a kernelized discriminative classifier (see chapter 9). Regardless, we will continue to investigate the (more theoretically interesting) generative approach.

Problem 20.1

### 20.2.3 Problems with the bag of words model

There are a number of drawbacks to the generative bag of words model:

- It assumes that the words are generated independently, although this is not necessarily true. The presence of a particular visual word tells us about the likelihood of observing other words.
- It ignores spatial information. Consequently, when applied to object recognition, it cannot tell us where the object is in the image.



**Figure 20.4** Problems with bag of words model. a) The bag of words model is quite effective for object and scene recognition, but it can be improved upon by b) modeling the cooccurrence of visual words (creating the latent Dirichlet allocation model). c) This model can be extended to describe the relative positions of different parts of the object (creating a constellation model) and d) extended again to describe the relative position of objects in the scene (creating a scene model).

- It is unsuited to describing multiple objects in a single image.

We devote the remaining part of this chapter to building a series of generative models that improve on these weaknesses (figure 20.4).

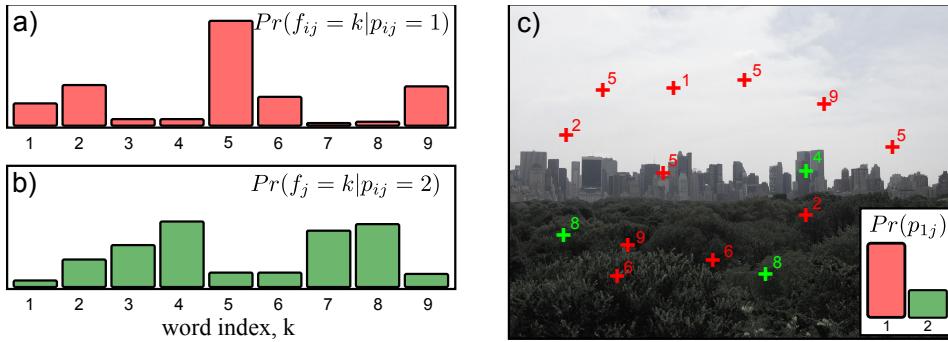
### 20.3 Latent Dirichlet allocation

Algorithm 20.2

We will now develop an intermediate model known as *latent Dirichlet allocation*. This model has limited utility for visual applications in its most basic form, but it underpins more interesting models that are discussed subsequently.

There are two important differences between the bag of words and latent Dirichlet allocation models. First, the bag of words model describes the relative frequency of visual words in a single image, whereas latent Dirichlet allocation describes the occurrence of visual words across a number of images. Second, the bag of words model assumes that each word in the image is generated completely independently; having observed word  $f_{i1}$ , we are none the wiser about word  $f_{i2}$ . However, in the latent Dirichlet allocation model, a hidden variable associated with each image induces a more complex distribution over the word frequencies.

Latent Dirichlet allocation can be best understood by analogy to text documents. Each document is considered as a certain mixture of *topics*. For example, this book might contain the topics ‘machine learning’, ‘vision’ and ‘computer sci-



**Figure 20.5** Latent Dirichlet allocation. This model treats each word as belonging to one of  $M$  different parts (here  $M = 2$ ). a) The probability of observing the different words, given that the part is 1, is described by a categorical distribution. b) The probability of observing the different words, given that the part is 2, is described by a different categorical distribution. c) In each image, the tendency for the observed words to belong to each part is different. In this case part 1 is more likely than part 2, so most of the words belong to part 1 (are red as opposed to green).

ence' in proportions of 0.3, 0.5, and 0.2, respectively. Each topic defines a probability distribution over words; the words ‘image’ and ‘pixel’ might be more probable under the topic of vision, and the words ‘algorithm’ and ‘complexity’ might be more probable under the topic of computer science.

To generate a word, we first choose a topic according to the topic probabilities for the current document. Then we choose a word according to a distribution that depends on the chosen topic. Notice how this model induces correlations between the probability of observing different words. For example, if we see the word ‘image’, then this implies that the topic ‘vision’ has a significant probability and hence observing the word ‘pixel’ becomes more likely.

Now let us convert these ideas back to the vision domain. The document becomes an image, and the words become visual words. The topic does not have an absolutely clear interpretation, but we will refer to it as a *part*. It is a cluster of visual words that tend to co-occur in images. They may or may not be spatially close to one another in the image, and they may or may not correspond to an actual ‘part’ of an object (figure 20.5).

Formally, the model represents the words in an image as a mixture of categorical distributions. The mixing weights depend on the image, but the parameters of the categorical distributions are shared across all of the images:

$$\begin{aligned} Pr(p_{ij}) &= \text{Cat}_{p_{ij}}[\boldsymbol{\pi}_i] \\ Pr(f_{ij}|p_{ij}) &= \text{Cat}_{f_{ij}}[\boldsymbol{\lambda}_{p_{ij}}], \end{aligned} \quad (20.5)$$

where  $i$  indexes the image and  $j$  indexes the word. The first equation says that the part label  $p_{ij} \in \{1, 2, \dots, M\}$  associated with the  $j^{th}$  word in the  $i^{th}$  image

is drawn from a categorical distribution with parameters  $\boldsymbol{\pi}_i$  that are unique to this image. The second equation says that the actual choice of visual word  $f_{ij}$  is a categorical distribution where the parameters  $\boldsymbol{\lambda}_{p_{ij}}$  depend on the part. For short, we will refer to  $\{\boldsymbol{\pi}_i\}_{i=1}^I$  and  $\{\boldsymbol{\lambda}_m\}_{m=1}^M$  as the part probabilities and the word probabilities, respectively.

The final density over the words comes from marginalizing over the part labels which are hidden variables, so that

$$Pr(f_{ij}) = \sum_{m=1}^M Pr(f_{ij}|p_{ij}=m)Pr(p_{ij}=m). \quad (20.6)$$

To complete the model, we define priors on the parameters  $\{\boldsymbol{\pi}_i\}_{i=1}^I$ ,  $\{\boldsymbol{\lambda}_m\}_{m=1}^M$  where  $I$  is the number of images and  $M$  is the total number of parts. In each case, we choose the conjugate Dirichlet prior with a uniform parameter vector so that

$$\begin{aligned} Pr(\boldsymbol{\pi}_i) &= \text{Dir}_{\boldsymbol{\pi}_i}[\boldsymbol{\alpha}] \\ Pr(\boldsymbol{\lambda}_m) &= \text{Dir}_{\boldsymbol{\lambda}_m}[\boldsymbol{\beta}], \end{aligned} \quad (20.7)$$

where  $\boldsymbol{\alpha} = [\alpha, \alpha, \dots, \alpha]$  and  $\boldsymbol{\beta} = [\beta, \beta, \dots, \beta]$ . The associated graphical model is shown in figure 20.6.

Notice that latent Dirichlet allocation is a density model for the data in a set of images. It does not involve a ‘world’ term that we wish to infer. In the subsequent models, we will re-introduce the world term and use the model for inference in visual problems. However, for now we will concentrate on how to learn the relatively simple latent Dirichlet allocation model.

### 20.3.1 Learning

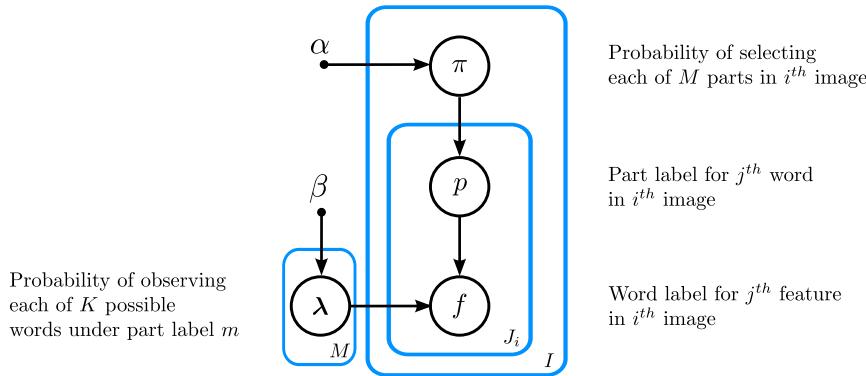
In learning, the goal is to estimate the part probabilities  $\{\boldsymbol{\pi}_i\}_{i=1}^I$  for each of the  $I$  training images and the word probabilities for each of the  $M$  parts  $\{\boldsymbol{\lambda}_m\}_{m=1}^M$  based on a set of training data  $\{f_{ij}\}_{i=1,j=1}^{I,J_i}$ , where  $J_i$  denotes the number of words found in the  $i^{th}$  image.

If we knew the values of the hidden part labels  $\{p_{ij}\}_{i=1,j=1}^{I,J_i}$ , then it would be easy to learn the unknown parameters. Adopting the approach of section 4.5, the exact expressions would be:

$$\begin{aligned} \hat{\pi}_{im} &= \frac{\sum_j \delta[p_{ij} - m] + \alpha}{\sum_{j,m} \delta[p_{ij} - m] + M\alpha} \\ \hat{\lambda}_{mk} &= \frac{\sum_{i,j} \delta[p_{ij} - m] \delta[f_{ij} - k] + \beta}{\sum_{i,j,k} \delta[p_{ij} - m] \delta[f_{ij} - k] + K\beta}. \end{aligned} \quad (20.8)$$

Problem 20.2

Unfortunately, we do not know these part labels, so we cannot use this direct technique. One possible approach would be to adopt the EM algorithm in which we alternately compute the posterior distribution over the part labels and update



**Figure 20.6** Graphical model for latent Dirichlet allocation. The likelihood of the  $j^{th}$  word  $f_{ij}$  in the  $i^{th}$  image taking each of the  $K$  different values depends on which of  $M$  parts it belongs to, and this is determined by the associated part label  $p_{ij}$ . The tendency of the part label to take different values is different for each image, and is determined by the parameters  $\pi_i$ . The hyperparameters  $\alpha$  and  $\beta$  determine the Dirichlet priors over the part probabilities respectively.

the parameters. Unfortunately, this is also problematic; all of the part labels  $\{p_{ij}\}_{j=1}^{J_i}$  in the  $i^{th}$  image share a parent  $\pi_i$  in the graphical model. This means we cannot treat them as independent. In theory, we could compute their joint posterior distribution, but there may be several hundred words per image, each of which takes several hundred values, and so this is not practical.

Hence, our strategy will be to:

- write an expression for the posterior distribution over the part labels,
- develop an MCMC method to draw samples from this distribution, and then
- use the samples to estimate the parameters.

These three steps are expanded upon in the next three sections.

### Posterior distribution over part labels

The posterior distribution over the part labels  $\mathbf{p} = \{p_{ij}\}_{i=1,j=1}^{I,J_i}$  results from applying Bayes' rule:

$$Pr(\mathbf{p}|\mathbf{f}) = \frac{Pr(\mathbf{f}|\mathbf{p})Pr(\mathbf{p})}{\sum_{\mathbf{f}} Pr(\mathbf{f}|\mathbf{p})Pr(\mathbf{p})}, \quad (20.9)$$

where  $\mathbf{f} = \{f_{ij}\}_{i=1,j=1}^{I,J_i}$  denotes the observed words.

The two terms in the numerator depend on the word probabilities  $\{\lambda_m\}_{m=1}^M$  and the part probabilities  $\{\pi_i\}_{i=1}^I$ , respectively. However, since each of these quantities has a conjugate prior, we can marginalize over them and remove them from the computation entirely. Hence, the likelihood  $Pr(\mathbf{f}|\mathbf{p})$  can be written as

$$\begin{aligned}
Pr(\mathbf{f}|\mathbf{p}) &= \int \prod_{i=1}^I \prod_{j=1}^{J_i} Pr(f_{ij}|p_{ij}, \boldsymbol{\lambda}_{1\dots M}) Pr(\boldsymbol{\lambda}_{1\dots M}) d\boldsymbol{\lambda}_{1\dots M} \\
&= \left( \frac{\Gamma[K\beta]}{\Gamma[\beta]^K} \right)^M \prod_{m=1}^M \frac{\prod_{k=1}^K \Gamma \left[ \sum_{i,j} \delta[f_{ij} - k] \delta[p_{ij} - m] + \beta \right]}{\Gamma \left[ \sum_{i,j,k} \delta[f_{ij} - k] \delta[p_{ij} - m] + K\beta \right]},
\end{aligned} \tag{20.10}$$

and the prior can be written as  
Problem 20.3

$$\begin{aligned}
Pr(\mathbf{p}) &= \prod_{i=1}^I \int \prod_{j=1}^{J_i} Pr(p_{ij}|\boldsymbol{\pi}_i) Pr(\boldsymbol{\pi}_i) d\boldsymbol{\pi}_i \\
&= \left( \frac{\Gamma[M\alpha]}{\Gamma[\alpha]^M} \right)^I \prod_{i=1}^I \frac{\prod_{m=1}^M \Gamma \left[ \sum_j \delta[p_{ij} - m] + \alpha \right]}{\Gamma \left[ \sum_{j,m} \delta[p_{ij} - m] + M\alpha \right]},
\end{aligned} \tag{20.11}$$

where we exploited conjugate relations to help solve the integral as in section 4.5.3.

Unfortunately, we cannot compute the denominator of equation 20.9 as this involves summing over every possible assignment of the word labels  $\mathbf{f}$ . Consequently, we can only compute the posterior probability for part labels  $\mathbf{p}$  up to an unknown scaling factor. We encountered a similar situation before in the MRF labeling problem (chapter 12). In that case there was a polynomial time algorithm to find the MAP estimate, but here that is not possible; the cost function for this problem cannot be expressed as a sum of unary and pairwise terms.

### Drawing samples from posterior distribution

To make progress, we will use a Monte Carlo Markov chain method to generate a set of samples  $\{\mathbf{p}^{[1]}, \mathbf{p}^{[2]}, \dots, \mathbf{p}^{[T]}\}$  from the posterior distribution. More specifically, we will use a Gibbs sampling approach (see section 10.7.2) in which we update each part label  $p_{ij}$  in turn. To do this, we compute the posterior probability of the current part label assuming that all of the others are fixed and then draw a sample from this distribution. We repeat this for every part label to generate a new sample of  $\mathbf{p}$ . This posterior probability of a single part label assuming that the others are fixed has  $M$  elements which are computed as

$$Pr(p_{ij} = m | \mathbf{p}_{\setminus ij}, \mathbf{f}) = \frac{Pr(p_{ij} = m, \mathbf{p}_{\setminus ij}, \mathbf{f})}{\sum_{m=1}^M Pr(p_{ij} = m, \mathbf{p}_{\setminus ij}, \mathbf{f})}, \tag{20.12}$$

where the notation  $\mathbf{p}_{\setminus ij}$  denotes all of the elements of  $\mathbf{p}$  except  $p_{ij}$ . To estimate this, we must compute joint probabilities  $Pr(\mathbf{f}, \mathbf{p}) = Pr(\mathbf{f}|\mathbf{p})Pr(\mathbf{p})$  using equations 20.10 and 20.11. In practice, the resulting expression simplifies considerably to

$$\Pr(p_{ij} = m | \mathbf{p}_{\setminus ij}, \mathbf{f}) \propto \left( \frac{\sum_{a,b \setminus i,j} \delta[f_{ab} - f_{ij}] \delta[p_{ab} - m] + \beta}{\sum_k \sum_{a,b \setminus i,j} \delta[f_{ab} - k] \delta[p_{ab} - m] + K\beta} \right) \left( \frac{\sum_{b \setminus j} \delta[p_{ib} - m] + \alpha}{\sum_m \sum_{b \setminus j} \delta[p_{ib} - m] + M\alpha} \right) \quad (20.13)$$

where the notation  $\sum_{a,b \setminus i,j}$  means sum over all values of  $\{a, b\}$  except  $i, j$ . Although it looks rather complex, this expression has a simple interpretation. The first term is the probability of observing the word  $f_{ij}$  given that part  $p_{ij} = m$ . The second term is the proportion of the time that part  $m$  is present in the current document.

To sample from the posterior distribution, we initialize the part labels  $\{p_{ij}\}_{i=1,j=1}^{I,J_i}$ , and alternately update each part label in turn. After a reasonable burn in period (several thousand iterations over all of the variables), the resulting samples can be assumed to be drawn from the posterior. We then take a subset of samples from this chain, where each is separated by a reasonable distance, to ensure that their correlation is low.

### Using samples to estimate parameters

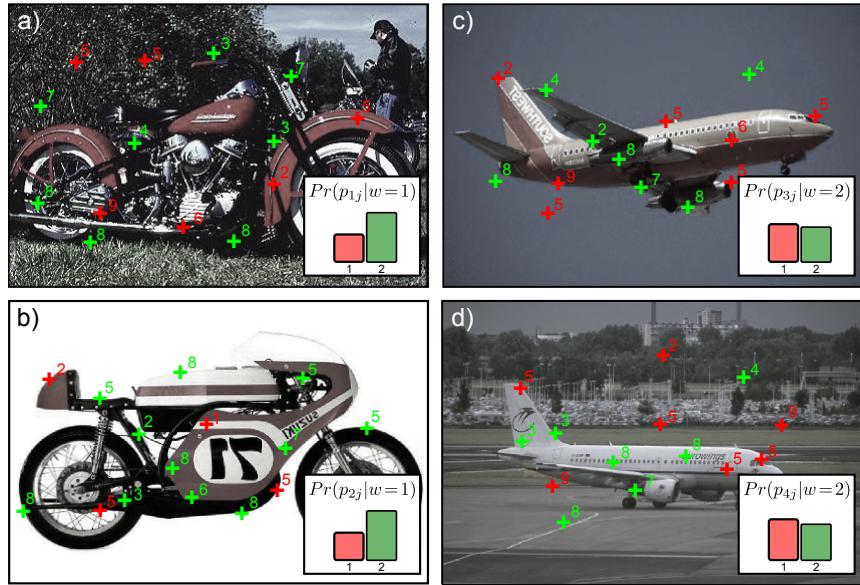
Finally, we now estimate the unknown parameters using the expressions

$$\begin{aligned} \hat{\pi}_{im} &= \frac{\sum_{t,j} \delta[p_{ij}^{[t]} - m] + \alpha}{\sum_{t,j,m} \delta[p_{ij}^{[t]} - m] + M\alpha} \\ \hat{\lambda}_{mk} &= \frac{\sum_{t,i,j} \delta[p_{ij}^{[t]} - m] \delta[f_{ij} - k] + \beta}{\sum_{t,i,j,k} \delta[p_{ij}^{[t]} - m] \delta[f_{ij} - k] + K\beta}, \end{aligned} \quad (20.14)$$

which are very similar to the original expressions (equation 20.8) for estimating the parameters given known part labels.

### 20.3.2 Unsupervised object discovery

The preceding model can been used to help analyze the structure of a set of images. Consider fitting this model to an unlabeled data set containing several images each of a number of different object categories. After fitting, each of the  $I$  images is modeled as a mixture of part, and we have an estimate of the mixture weights  $\lambda_i$  for each. We now cluster the images according to the dominant part in this mixtures. For small datasets, it has been shown that this method can separate out different object classes with a high degree of accuracy; this model allows the discovery of object classes in unlabeled datasets.



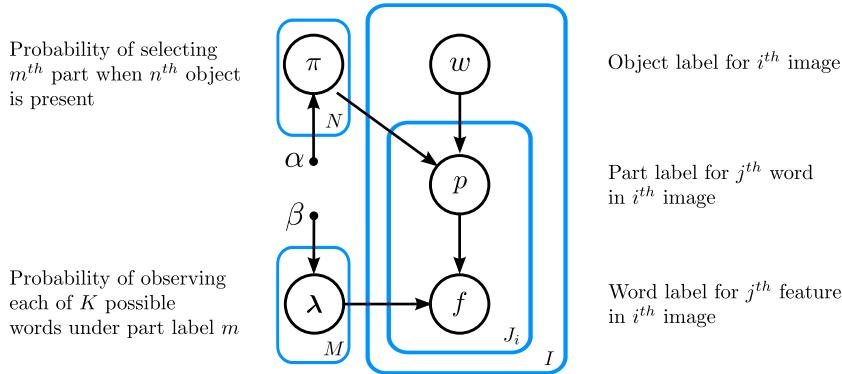
**Figure 20.7** Single author-topic model. The single author topic model is a variant of latent Dirichlet allocation that includes a variable  $w_i \in \{1 \dots N\}$  that represents which of  $N$  possible objects is in the image. It is assumed that the part probabilities  $\{\pi_n\}_{n=1}^N$  are contingent on the particular choice of object. a) Image 1 contains a motorbike and this induces the part probabilities shown in the bottom right-hand corner. The parts are drawn from this probability distribution (color of crosses) and the words (numbers) are drawn based on the parts chosen. b) A second image of a motorbike induces the same part probabilities. c-d) These two images contain a different object and hence have different part probabilities (bottom right).

## 20.4 Single author-topic model

Latent Dirichlet allocation is simply a density model for images containing sets of discrete words. We will now describe an extension to this model that assumes there is a single object in each image, and the identity of this object is characterized by a label  $w_i \in \{1 \dots N\}$ . We now make the assumption that each image of the same object contains the same part probabilities:

$$\begin{aligned} Pr(p_{ij}|w_i = n) &= \text{Cat}_{p_{ij}}[\pi_n] \\ Pr(f_{ij}|p_{ij}) &= \text{Cat}_{f_{ij}}[\lambda_{p_{ij}}]. \end{aligned} \quad (20.15)$$

To complete the model, we add Dirichlet priors to the unknown parameters  $\{\pi_n\}_{n=1}^N$  and  $\{\lambda_m\}_{m=1}^M$ :



**Figure 20.8** Graphical model for the latent author-topic model. The likelihood of the  $j^{th}$  word in the  $i^{th}$  image  $f_{ij}$  being categorized as one word or another depends on which of  $M$  parts it belongs to and this is determined by the associated part label  $p_{ij}$ . The tendency of the part label to take different values is different for each object  $w_i \in \{1 \dots N\}$  is determined by the parameters  $\pi_n$ , where it is assumed there is a single object in each image.

$$\begin{aligned} Pr(\boldsymbol{\pi}_n) &= \text{Dir}_{\boldsymbol{\pi}_n}[\boldsymbol{\alpha}] \\ Pr(\boldsymbol{\lambda}_m) &= \text{Dir}_{\boldsymbol{\lambda}_m}[\boldsymbol{\beta}], \end{aligned} \quad (20.16)$$

where  $\boldsymbol{\alpha} = [\alpha, \alpha, \dots, \alpha]$  and  $\boldsymbol{\beta} = [\beta, \beta, \dots, \beta]$ . For simplicity we will assume that the prior over the object label  $w$  is uniform and not discuss this further. The associated graphical model is illustrated in figure 20.8.

Like the bag of words and latent Dirichlet allocation models, this model was originally used for describing text documents; it assumes that each document was written by one author (each image contains one object) and this determines the relative frequency of topics (of parts). It is a special case of the more general *author-topic* model, which allows multiple authors for each document.

Problem 20.4  
Problem 20.6  
Problem 20.5

## 20.4.1 Learning

Learning proceeds in much the same way as in latent Dirichlet allocation. We are given a set of  $I$  images, each of which has a known object label  $w_i \in \{1 \dots N\}$  and a set of visual words  $\{f_{ij}\}_{j=1}^{J_i}$ , where  $f_{ij} \in \{1 \dots M\}$ . It would be easy to estimate the part probabilities for each object  $\{\pi_n\}_{n=1}^N$  and the word probabilities for each part  $\{\lambda_m\}_{m=1}^M$  if we knew the hidden part labels  $p_{ij}$  associated with each word. As before we take the approach of drawing samples from the posterior distribution over the part labels and using these to estimate the unknown parameters. This posterior is computed via Bayes' rule:

$$Pr(\mathbf{p}|\mathbf{f}, \mathbf{w}) = \frac{Pr(\mathbf{f}|\mathbf{p})Pr(\mathbf{p}|\mathbf{w})}{\sum_{\mathbf{f}} Pr(\mathbf{f}|\mathbf{p})Pr(\mathbf{p}|\mathbf{w})}, \quad (20.17)$$

where  $\mathbf{w} = \{w_i\}_{i=1}^I$  contains all of the object labels.

The likelihood term  $Pr(\mathbf{f}|\mathbf{p})$  is the same as for latent Dirichlet allocation and is given by equation 20.10. The prior term becomes

$$\begin{aligned} Pr(\mathbf{p}|\mathbf{w}) &= \int \prod_{i=1}^I \prod_{j=1}^{J_i} Pr(p_{ij}|w_i, \boldsymbol{\pi}_{1\dots N}) Pr(\boldsymbol{\pi}_{1\dots N}) d\boldsymbol{\pi}_{1\dots N} \\ &= \left( \frac{\Gamma[M\alpha]}{\Gamma[\alpha]^M} \right)^N \prod_{n=1}^N \frac{\prod_{m=1}^M \Gamma \left[ \sum_{i,j} \delta[p_{ij} - m] \delta[w_i - n] + \alpha \right]}{\Gamma \left[ \sum_{i,j,m} \delta[p_{ij} - m] \delta[w_i - n] + M\alpha \right]}. \end{aligned} \quad (20.18)$$

As before, we cannot compute the denominator of Bayes' rule as it involves an intractable summation over all possible words. Hence, we use a Gibbs sampling method in which we repeatedly draw samples  $\mathbf{p}^{[1]} \dots \mathbf{p}^{[T]}$  from each marginal posterior in turn using the relation:

$$Pr(p_{ij} = m | \mathbf{p}_{\setminus ij}, \mathbf{f}, w_i = n) \propto \left( \frac{\sum_{a,b \setminus i,j} \delta[f_{ab} - f_{ij}] \delta[p_{ab} - m] + \beta}{\sum_k \sum_{a,b \setminus i,j} \delta[f_{ab} - k] \delta[p_{ab} - m] + K\beta} \right) \left( \frac{\sum_{a,b \setminus i,j} \delta[p_{ab} - m] \delta[w_i - n] + \alpha}{\sum_m \sum_{a,b \setminus i,j} \delta[p_{ab} - m] \delta[w_i - n] + M\alpha} \right), \quad (20.19)$$

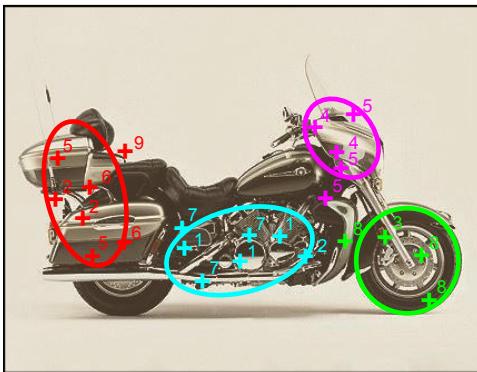
where the notation  $\sum_{a,b \setminus i,j}$  denotes a sum over all valid values of  $a, b$  except for the combination  $i, j$ . This expression has a simple interpretation. The first term is the probability of observing the word  $f_{ij}$  given that part  $p_{ij} = m$ . The second term is the proportion of the time that part  $m$  is present for the  $n^{th}$  object.

Finally, we estimate the unknown parameters using the relations:

$$\begin{aligned} \hat{\pi}_{nm} &= \frac{\sum_{t,i,j} \delta[p_{ij}^{[t]} - m] \delta[w_i - n] + \alpha}{\sum_{t,i,j,m} \delta[p_{ij}^{[t]} - m] \delta[w_i - n] + M\alpha} \\ \hat{\lambda}_{mk} &= \frac{\sum_{t,i,j} \delta[p_{ij}^{[t]} - m] \delta[f_{ij} - k] + \beta}{\sum_{t,i,j,k} \delta[p_{ij}^{[t]} - m] \delta[f_{ij} - k] + K\beta}. \end{aligned} \quad (20.20)$$

### 20.4.2 Inference

In inference, we compute the likelihood of new image data  $\mathbf{f} = \{f_j\}_{j=1}^J$  under each possible object  $w \in \{1 \dots N\}$  using



**Figure 20.9** Constellation model. In the constellation model the object or scene is again described as consisting of set of different parts (colors). A number of words are associated with each part and the word probabilities depend on the part label. However, unlike in the previous models, each part now has a particular range of locations associated with it, which are described as a normal distribution. In this sense it conforms more closely to the normal use of the English word ‘part’.

$$\begin{aligned} Pr(\mathbf{f}|w = n) &= \prod_{j=1}^J \sum_{p_j=1}^M Pr(p_j|w = n) Pr(f_j|p_j) \\ &= \prod_{j=1}^J \sum_{p_j=1}^M \text{Cat}_{p_j}[\boldsymbol{\pi}_n] \text{Cat}_{f_j}[\boldsymbol{\lambda}_{p_j}] \end{aligned} \quad (20.21)$$

We now define suitable priors  $Pr(w)$  over the possible objects and use Bayes’ rule to compute the posterior distribution,

$$Pr(w = n|\mathbf{f}) = \frac{Pr(\mathbf{f}|w = n) Pr(w = n)}{\sum_{n=1}^N Pr(\mathbf{f}|w = n) Pr(w = n)}. \quad (20.22)$$

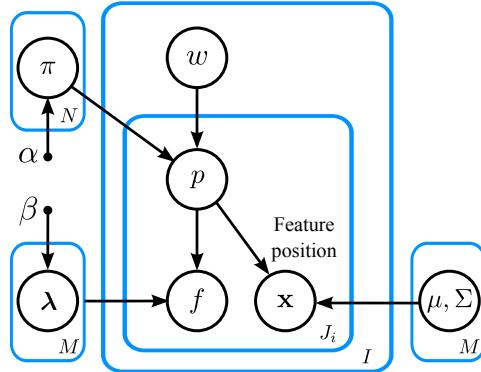
## 20.5 Constellation models

The single author-topic model described earlier is still a very weak description of an object as it contains no spatial information. Constellation models are a general class of model that describe objects in terms of a set of parts and their spatial relations. For example, the pictorial structures model described in section 11.8.3 can be considered a constellation model. Here, we will develop a different type of constellation model that extends the latent Dirichlet allocation model (figure 20.9).

We assume that a part retains the same meaning as before; it is a cluster of co-occurring words. However, each part now induces a spatial distribution over its associated words, which we will model with a 2D normal distribution so that

$$\begin{aligned} Pr(p_{ij}|w_i = n) &= \text{Cat}_{p_{ij}}[\boldsymbol{\pi}_n] \\ Pr(f_{ij}|p_{ij} = m) &= \text{Cat}_{f_{ij}}[\boldsymbol{\lambda}_m] \\ Pr(\mathbf{x}_{ij}|p_{ij} = m) &= \text{Norm}_{\mathbf{x}_{ij}}[\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m], \end{aligned} \quad (20.23)$$

**Figure 20.10** Constellation model. In addition to all of the other variables in the single author topic model (compare to figure 20.8), the position  $\mathbf{x}_{ij}$  of the  $j^{th}$  word in the  $i^{th}$  image is also modeled. This position is contingent on which of the  $M$  parts that the current word is assigned to (determined by the variable  $p_{ij}$ ). When the word is assigned to the  $m^{th}$  part, the position  $\mathbf{x}_{ij}$  is mod as being drawn from a normal distribution with mean and variance  $\mu_m, \Sigma_m$ .



where  $\mathbf{x}_{ij} = [x_{ij}, y_{ij}]^T$  is the two dimensional position of the  $j^{th}$  word in the  $i^{th}$  image. As before we also define Dirichlet priors over the unknown categorical parameters so that

$$\begin{aligned} Pr(\boldsymbol{\pi}_n) &= \text{Dir}_{\boldsymbol{\pi}_n}[\boldsymbol{\alpha}] \\ Pr(\boldsymbol{\lambda}_m) &= \text{Dir}_{\boldsymbol{\lambda}_m}[\boldsymbol{\beta}], \end{aligned} \quad (20.24)$$

where  $\boldsymbol{\alpha} = [\alpha, \alpha, \dots, \alpha]$  and  $\boldsymbol{\beta} = [\beta, \beta, \dots, \beta]$ . The associated graphical model is illustrated in figure 20.10.

This model extends latent Dirichlet allocation to allow it to represent the relative positions of parts of an object or scene. For example, it might learn that words associated with trees usually occur in the center of the image and that those associated with the sky usually occur near the top of the image.

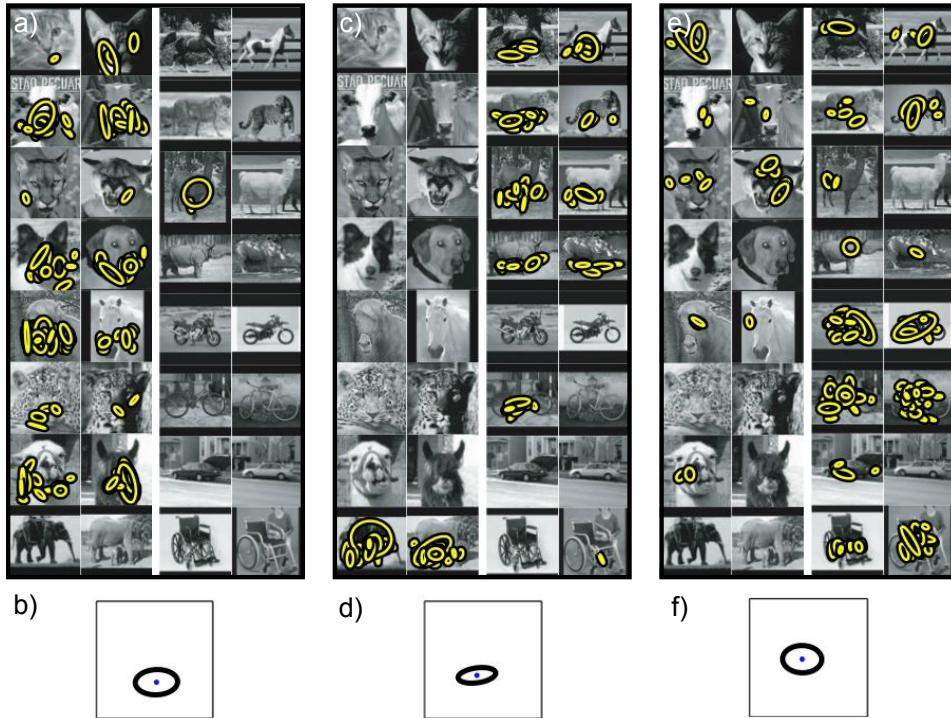
### 20.5.1 Learning

As for latent Dirichlet allocation, the model would be easy to learn if we knew the part assignments  $\mathbf{p} = \{p_{ij}\}_{i=1, j=1}^{I, J_i}$ . By the same logic as before, we hence draw samples from posterior distribution  $Pr(\mathbf{p}|\mathbf{f}, \mathbf{X}, \mathbf{w})$  over the part assignments given the observed word labels  $\mathbf{f} = \{f_{ij}\}_{j=1}^{I, J_i}$ , their associated positions  $\mathbf{X} = \{\mathbf{x}_{ij}\}_{i=1, j=1}^{I, J_i}$ , and the known object labels  $\mathbf{w} = \{w_i\}_{i=1}^I$ . The expression for the posterior is computed via Bayes' rule

$$Pr(\mathbf{p}|\mathbf{f}, \mathbf{X}, \mathbf{w}) = \frac{Pr(\mathbf{f}, \mathbf{X}|\mathbf{p})Pr(\mathbf{p}|\mathbf{w})}{\sum_p Pr(\mathbf{f}, \mathbf{X}|\mathbf{p})Pr(\mathbf{p}|\mathbf{w})}, \quad (20.25)$$

and once again, the terms in the numerator can be computed, but the denominator contains an intractable sum of exponentially many terms. This means that the posterior cannot be computed in closed form, but we can still evaluate the posterior probability for any particular assignment  $\mathbf{p}$  up to an unknown scale factor. This is sufficient to draw samples from the distribution using Gibbs sampling.

The prior probability  $Pr(\mathbf{p}|\mathbf{w})$  of the part assignments is the same as before and is given in equation 20.18. However, the likelihood term  $Pr(\mathbf{f}, \mathbf{X}|\mathbf{p})$  now has



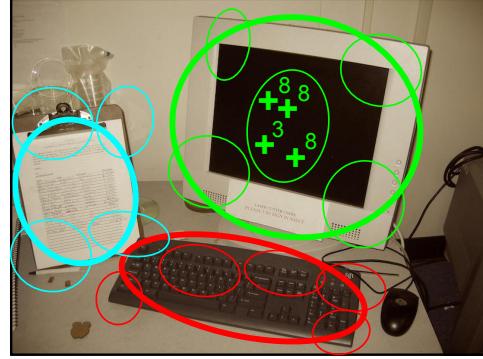
**Figure 20.11** Sharing words in the constellation model. a) Sixteen images from training set (two from each class). Yellow ellipses depict words identified in the image associated with one part of the image (i.e., they are equivalent of the crosses in figure 20.9). It is notable that the words associated with this part mainly belong to the lower part of the faces of the animal images. b) The mean  $\mu$  and variance  $\Sigma$  of this object part. c,d) A second part which seems to correspond to the legs of animals in profile. e,f) A third part contains many words associated with the wheels of objects. Adapted from Sudderth *et al.* (2008). ©2008 IEEE.

an additional component due to the requirement for the word position  $\mathbf{x}_{ij}$  to agree with the normal distribution induced by the part:

$$\begin{aligned}
 & Pr(\mathbf{f}, \mathbf{X} | \mathbf{p}) \tag{20.26} \\
 &= \int \prod_{i=1}^I \prod_{j=1}^{J_i} Pr(f_{ij} | p_{ij}, \boldsymbol{\lambda}_{1\dots M}) Pr(\boldsymbol{\lambda}_{1\dots M}) Pr(\mathbf{x}_{ij} | p_{ij}, \boldsymbol{\mu}_{1\dots M}, \boldsymbol{\Sigma}_{1\dots M}) d\boldsymbol{\lambda}_{1\dots M} \\
 &= \left( \frac{\Gamma[K\beta]}{\Gamma[\beta]^K} \right)^M \prod_{m=1}^M \frac{\prod_{k=1}^K \Gamma \left[ \sum_{i,j} \delta[p_{ij} - m] \delta[f_{ij} - k] + \beta \right]}{\Gamma \left[ \sum_{i,j,k} \delta[p_{ij} - m] \delta[f_{ij} - k] + K\beta \right]} \text{Norm}_{\mathbf{x}_{ij}} [\boldsymbol{\mu}_{p_{ij}}, \boldsymbol{\Sigma}_{p_{ij}}].
 \end{aligned}$$

In Gibbs sampling, we choose one data example  $\{f_{ij}, \mathbf{x}_{ij}\}$  and draw from the

**Figure 20.12** Scene model. Each image consists of a single scene. A scene induces a probability distribution over the presence of different objects (different colors) such as the monitor, piece of paper and keyboard in this scene and their relative positions (thick ellipses). Each object is itself composed of spatially separate parts (thin ellipse). Each part has a number of words associated with it (crosses, shown only for one part for clarity).



posterior distribution assuming that all of the other parts are fixed. An approximate<sup>1</sup> expression to compute the posterior is given by

$$\begin{aligned} Pr(p_{ij} = m | \mathbf{p}_{\setminus ij}, \mathbf{f}, \mathbf{x}_{ij}, w_i = n) &\propto & (20.27) \\ \left( \frac{\sum_{a,b \setminus i,j} \delta[f_{ab} - f_{ij}] \delta[p_{ab} - m] + \beta}{\sum_k \sum_{a,b \setminus i,j} \delta[f_{ab} - k] \delta[p_{ab} - m] + K\beta} \right) \text{Norm}_{\mathbf{x}_{ij}}[\tilde{\boldsymbol{\mu}}_m, \tilde{\boldsymbol{\Sigma}}_m] \\ \left( \frac{\sum_{a,b \setminus i,j} \delta[p_{ab} - m] \delta[w_i - n] + \alpha}{\sum_m \sum_{a,b \setminus i,j} \delta[p_{ab} - m] \delta[w_i - n] + M\alpha} \right), \end{aligned}$$

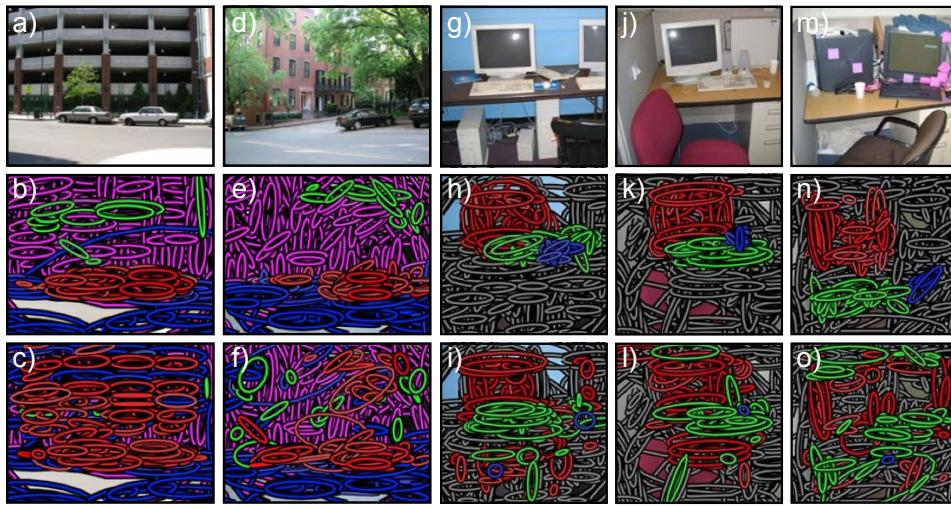
where the notation  $\sum_{a,b \setminus i,j}$  denotes summation over all values of  $\{a, b\}$  except  $i, j$ . The terms  $\tilde{\boldsymbol{\mu}}_m$  and  $\tilde{\boldsymbol{\Sigma}}_m$  are the mean and covariance of all of the word positions associated with the  $m^{th}$  part *ignoring* the contribution of the current position  $\mathbf{x}_{ij}$ .

At the end of the procedure the probabilities are computed using the relations in equation 20.20 and the part locations as

$$\begin{aligned} \hat{\boldsymbol{\mu}}_m &= \frac{\sum_{i,j,t} \mathbf{x}_{ij} \delta[p_{ij}^{[t]} - m]}{\sum_{i,j,t} \delta[p_{ij}^{[t]} - m]} \\ \hat{\boldsymbol{\Sigma}}_m &= \frac{\sum_{i,j,t} (\mathbf{x}_{ij} - \hat{\boldsymbol{\mu}}_m)^T (\mathbf{x}_{ij} - \hat{\boldsymbol{\mu}}_m) \delta[p_{ij}^{[t]} - m]}{\sum_{i,j,t} \delta[p_{ij}^{[t]} - m]}. \end{aligned} \quad (20.28)$$

Example learning results can be seen in figure 20.11. Each part is a spatially localized cluster of words, and these often correspond to real-world objects such as ‘legs’ or ‘wheels.’ The parts are shared between the objects and so there is no need for a different set of parameters to learn the appearance of wheels for bicycles and wheels for motorbikes.

<sup>1</sup>More properly, we should define a prior over the mean and variance of the parts, and marginalize over these parameters as well. This also avoids problems when no features are assigned to a certain part and hence the mean and variance cannot be computed.



**Figure 20.13** Scene recognition. a) Example street image. b) Results of scene parsing model. Each ellipse represents one word (i.e., the equivalent of the crosses in figure 20.12). The ellipse color denotes the object label to which that word is assigned (part labels not shown). c) Results of the bag of words model which makes elementary mistakes such as putting car labels at the top of the image as it has no spatial information. d) Another street scene. e) Interpretation with scene model and f) bag of words model. g-o) Three more office scenes parsed by the scene model and bag of words models. Adapted from Sudderth *et al.* (2008). ©2008 Springer.

## 20.5.2 Inference

In inference, we compute the likelihood of new image data  $\{f_j, \mathbf{x}_j\}_{j=1}^J$  under each possible object  $w \in \{1 \dots N\}$  using

$$\begin{aligned} Pr(\mathbf{f}, \mathbf{X}|w = n) &= \prod_{j=1}^J \sum_{m=1}^M Pr(p_j = m|w = n) Pr(f_j|p_j = m) Pr(\mathbf{x}_j|p_j = m) \\ &= \prod_{j=1}^J \sum_{p_j=1}^M \text{Cat}_{p_j}[\boldsymbol{\pi}_n] \text{Cat}_{f_j}[\boldsymbol{\lambda}_{p_j}] \text{Norm}_{\mathbf{x}_{ij}}[\boldsymbol{\mu}_{p_j}, \boldsymbol{\Sigma}_{p_j}]. \end{aligned} \quad (20.29)$$

We now define suitable priors  $Pr(w)$  over the possible objects and use Bayes' rule to compute the posterior distribution

$$Pr(w = n|\mathbf{f}, \mathbf{X}) = \frac{Pr(\mathbf{f}, \mathbf{X}|w = n) Pr(w = n)}{\sum_{n=1}^N Pr(\mathbf{f}, \mathbf{X}|w = n) Pr(w = n)}. \quad (20.30)$$

## 20.6 Scene models

One limitation of the constellation model is that it assumes that the image contains a single object. However, real images generally contain a number of spatially offset objects. Just as the object determined the probability of the different parts, so the scene determines the relative likelihood of observing different objects (figure 20.12). For example, an office scene might include desks, computers, and chairs, but it is very unlikely to include tigers or icebergs.

To this end we introduce a new set of variables that represent the choice of scene  $\{s_i\}_{i=1}^I \in \{1 \dots C\}$

$$\begin{aligned} Pr(w_{ij}|s_i = c) &= \text{Cat}_{w_{ij}}[\phi_c] \\ Pr(p_{ij}|w_{ij} = n) &= \text{Cat}_{p_{ij}}[\boldsymbol{\pi}_n] \\ Pr(f_{ij}|p_{ij} = m) &= \text{Cat}_{f_{ij}}[\boldsymbol{\lambda}_m] \\ Pr(\mathbf{x}_{ij}|p_{ij} = m, w_{ij} = n) &= \text{Norm}_{\mathbf{x}_{ij}}[\boldsymbol{\mu}_n^{(w)} + \boldsymbol{\mu}_m^{(p)}, \boldsymbol{\Sigma}_n^{(w)} + \boldsymbol{\Sigma}_m^{(p)}]. \end{aligned} \quad (20.31)$$

Each word has an object label  $\{w_{ij}\}_{i=1,j=1}^{I,J_I}$  which denotes which of the  $L$  objects it corresponds to. The scene label  $\{s_i\}$  determines the relative propensity for each object to be present and these probabilities are held in the categorical parameters  $\{\phi_c\}_{c=1}^C$ . Each object type also has a position that is normally distributed with mean and covariance  $\boldsymbol{\mu}_n^{(w)}$  and  $\boldsymbol{\Sigma}_n^{(w)}$ . As before, each object defines a probability distribution over the  $M$  shared parts where the part assignment is held in the label  $p_{ij}$ . Each part has a position that is measured relative to the object position and has mean and covariance  $\boldsymbol{\mu}_m^{(p)}$  and  $\boldsymbol{\Sigma}_m^{(p)}$ , respectively.

**Problem 20.7**

We leave the details of the learning and inference algorithms as an exercise for the reader; the principles are the same as for the constellation model; we generate a series of samples from the posterior over the hidden variables  $w_{ij}$  and  $p_{ij}$  using Gibbs sampling and update the mean and covariances based on the samples.

Figure 20.13 shows several examples of scenes that have been interpreted using a scene model very similar to that described. In each case, the scene is parsed into a number of objects that are likely to co-occur and are in a sensible relative spatial configuration.

## 20.7 Applications

In this chapter we have described a series of generative models for visual words of increasing complexity. Although these models are interesting, it should be emphasized that many applications use only the basic bag of words approach combined with a discriminative classifier. We now describe two representative examples of such systems.



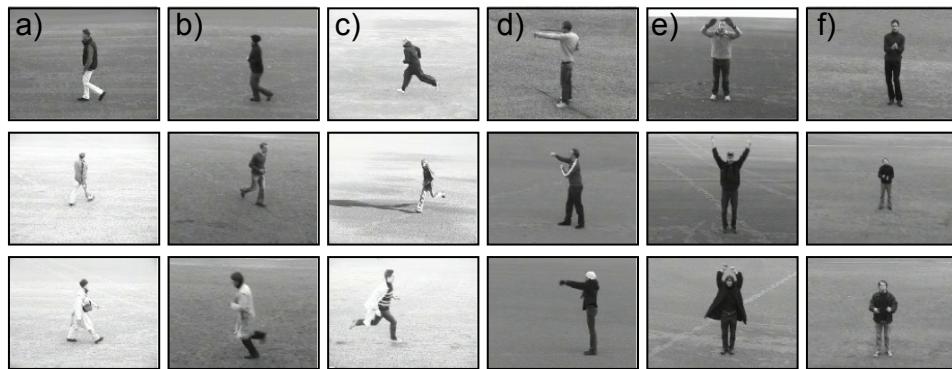
**Figure 20.14** Video Google. a) The user identifies part of one frame of a video by drawing a bounding box around part of the scene. b-i) The system returns a ranked list of frames that contain the same object and identifies where it is in the image (white bounding boxes). The system correctly identifies the leopard-skin patterned hat in a variety of contexts despite changes in scale and position. In h) it mistakes the texture of the vegetation in the background for an instance of the hat.

### 20.7.1 Video Google

Sivic & Zisserman (2003) presented a system based on the bag of words, which can retrieve frames from a movie very efficiently based on a visual query; the user draws a bounding box around the object of interest and the system returns other images that contain the same object (figure 20.14).

The system starts by identifying feature positions in each frame of the video. Unlike conventional bag of words models, these feature positions are tracked through several frames of video and rejected if this cannot be done. The averaged SIFT descriptor over each track is used to represent the image contents in the neighborhood of the feature. These descriptors are then clustered using K-means to create of the order of 6,000-10,000 possible visual words. Each feature is then assigned to one of these words based on the distance to the nearest cluster. Finally, each image or region is characterized by a vector containing the frequencies with which each visual word is found.

When the system receives a query, it compares the vector for the identified region to those for each potential region in the remaining video stream and retrieves



**Figure 20.15** Example images from the KTH database (Schüldt *et al.* 2004). Images in a-f) each show three examples of the six categories of walking, jogging, running, boxing, waving and clapping, respectively. Using the bag of words approach of Laptev *et al.* (2008) these actions can be classified with over 90% accuracy.

those that are closest.

The implementation includes several features that make this process more reliable. First, it discards the top 5% and bottom 10% of words according to their frequency. This eliminates very common words that do not distinguish between frames and words that are very rare and are hence inefficient to search on. Second, it weights the distance measure using the *term-frequency inverse document frequency* scheme: the weight increases if the word is relatively rare in the database (the word is discriminative) and if it is used relatively frequently in this region (it is particularly representative of the region). Finally, the matches are considered more reliable if the spatial arrangement of visual words is similar. The final retrieved results are re-ranked based on their spatial consistency with the query.

The final system reliably returns plausible regions for a feature length movie in less than 0.1 seconds using an inverted file structure to facilitate efficient retrieval.

### 20.7.2 Action recognition

Laptev *et al.* (2008) applied a bag of words approach to action recognition in video sequences. They used a space-time extension of the Harris corner detector to find interest points in the video frames and extracted descriptors at multiple scales around each point. They eliminated detections at boundaries between shots.

To characterize the local motion and appearance, they computed histogram-based descriptors of the space time-volumes surrounding these feature points. These were either based on the histogram of oriented gradient (HoG) descriptor (section 13.3.3) or based on histograms of local motion. They clustered a subset of 100,000 descriptors from the training data using the K-means algorithm to create



**Figure 20.16** Action recognition in movie database of Laptev *et al.* (2008).  
 a) Example true positives (correct detections), b) true negatives (action correctly identified as being absent), c) false positives (action classified as occurring but didn't) d) false negatives (action classified as not occurring but did). This type of real-world action classification task is still considered very challenging.

4000 clusters and each feature in the test and training data was represented by the index of the nearest cluster center.

They binned these quantized feature indices over a number of different space-time windows. The final decision about the action was based on a *one against all* binary classifier in which each action was separately considered and rated as being present or absent. The kernelized binary classifier combined together information from the two different feature types and the different space-time windows.

Laptev *et al.* (2008) first considered discriminating between six actions from the KTH database (Schüldt *et al.* 2004). This is a relatively simple dataset in which the camera is static and the action occurs against a relatively empty background (figure 20.15). They discriminated between these classes with an average of 91.8% accuracy, with the major confusion being between jogging and running.

They also considered a more complex database containing eight different actions from movie sequences (figure 20.16). It was notable that the performance here relied more on the HoG descriptors than the motion information, suggesting that the local context was providing considerable information (e.g., the action ‘get out of car’ is more likely when a car is present). For this database the performance was much worse, but it was significantly better than chance; action recognition ‘in the wild’ is an open problem in computer vision research.

## Discussion

The models in this chapter treat each image as a set of discrete features. The bag of features model, latent Dirichlet allocation, and single author-topic models do not

explicitly describe the position of objects in the scene. Although they are effective for recognizing objects, they cannot locate them in the image. The constellation model improves this by allowing the parts of object to have spatial relations, and the scene model describes a scene as a collection of displaced parts.

## Notes

**Bag of words models:** Sivic & Zisserman (2003) introduced the term ‘visual words’ and first made the connection with text retrieval. Csurka *et al.* (2004) applied the bag of words methodology to object recognition. A number of other studies then exploited developments in the document search community. For example, Sivic *et al.* (2005) exploited probabilistic latent semantic analysis (Hofmann 1999) and latent Dirichlet allocation (Blei *et al.* 2003) for unsupervised learning of object classes. Sivic *et al.* (2008) extended this work to learn hierarchies of object classes. Li & Perona (2005) constructed a model very similar to the original author-topic model (Rosen-Zvi *et al.* 2004) for learning scene categories. Suderth *et al.* (2005) and Suderth *et al.* (2008) extended the author topic model to contain information about the spatial layout of objects. The constellation and scene models presented in this chapter are somewhat simplified versions of this work. They also extended these models to cope with varying numbers of objects and or parts.

**Applications of bag of words:** Applications of the bag of words method include object recognition (Csurka *et al.* 2004), searching through video (Sivic & Zisserman 2003), scene recognition (Li & Perona 2005), and action recognition (Schüldt *et al.* 2004) and similar approaches have been applied to texture classification (Varma & Zisserman 2004) and labeling facial attributes (Aghajanian *et al.* 2009). Recent progress in object recognition can be reviewed by examining a recent summary of the PASCAL visual object classes challenge (Everingham *et al.* 2010). In the 2007 competition, bag of words approaches with no spatial information at all were still common. Several authors (Nistér & Stewénius 2006; Philbin *et al.* 2007; Jegou *et al.* 2008) have now presented large-scale demonstrations of object instance recognition based on bag of words and this idea has been used in commercial applications such as ‘Google Goggles’.

**Bag of words variants:** Although we have discussed mainly generative models for visual words in this chapter, discriminative approaches generally yield somewhat better performance. Grauman & Darrell (2005) introduced the pyramid match kernel which maps unordered data in a high-dimensional feature space into multi-resolution histograms and computes a weighted histogram intersection in this space. This effectively performs the clustering and feature comparison steps simultaneously. This idea was extended to the spatial domain of the image itself by Lazebnik *et al.* (2006).

**Improving the pipeline:** Yang *et al.* (2007) and Zhang *et al.* (2007) provide quantitative comparisons showing how the various parts of the pipeline (e.g., the matching kernel, interest point detector, clustering method) affect object recognition results.

The focus of recent research has moved on to addressing various weaknesses of the pipeline such as the arbitrariness of the initial vector quantization step and the problem of regular patterns (Chum *et al.* 2007; Philbin *et al.* 2007; Philbin *et al.* 2010; Jégou *et al.* 2009; Mikulík *et al.* 2010; Makadia 2010). The current trend is to increase the problem to realistic sizes and to this end new databases for object recognition (Deng *et al.* 2010) and scene recognition have been released (Xiao *et al.* 2010).

**Action recognition:** There has been a progression in recent years from testing action recognition algorithms in specially captured databases where the subject can easily be separated from the background Schüldt *et al.* (2004), to movie footage Laptev *et al.* (2008), and finally to completely unconstrained footage that may not be professionally shot and may have considerable camera shake. As this progression has taken place, the dominant approach has gradually become to base the system on visual words which capture the context of the scene as well as the action itself (Laptev *et al.* 2008). A comparison between approaches based on visual words and those that used explicit parts,

for action recognition in a still frame is presented by Delaitre *et al.* (2010). Recent work in this area has addressed unsupervised learning of action categories (Niebles *et al.* 2008).

## Problems

**Problem 20.1** The bag of words method in this chapter uses a generative approach to model the frequencies of the visual words. Develop a discriminative approach that models the probability of the object class as a function of the word frequencies.

**Problem 20.2** Prove the relations in equation 20.8, which show how to learn the latent Dirichlet allocation model in the case where we do know the part labels  $\{p_{ij}\}_{i=1,j=1}^{I,J_i}$ .

**Problem 20.3** Show that the likelihood and prior terms in Latent Dirichlet Allocation are given by equations 20.10 and 20.11 respectively.

**Problem 20.4** Li & Perona (2005) developed an alternative model to the single author-topic model in which the hyperparameter  $\alpha$  was different for each value of the object label  $w$ . Modify the graphical model for latent Dirichlet allocation to include this change.

**Problem 20.5** Write out generative equations for the author-topic model in which multiple authors are allowed for each document. Draw the associated graphical model.

**Problem 20.6** In real objects, we might expect visual words  $f$  that are adjacent to one another to take the same part label. How would you modify the author topic model to encourage nearby part labels to be the same. How would the Gibbs sampling procedure for drawing samples from the posterior probability over parts be affected?

**Problem 20.7** Draw a graphical model for the scene model described in section 20.6.

**Problem 20.8** All of the models in this chapter have dealt with classification; we wish to infer a discrete variable representing the state of the world based on discrete observed features  $\{f_j\}$ . Develop a generative model that can be used to infer a continuous variable based on discrete observed features (i.e., a regression model that uses visual words).

# **Part VII**

# **Appendices**



# Appendix A

## Notation

This is a brief guide to the notational conventions used in this text.

### Scalars, vectors and matrices

We denote scalars by either small or capital letters  $a, A, \alpha$ . We denote column vectors by bold small letters  $\mathbf{a}, \boldsymbol{\phi}$ . When we need a row vector we usually present this as the transpose of a column vector  $\mathbf{a}^T, \boldsymbol{\phi}^T$ .

We represent matrices by bold capital letters  $\mathbf{B}, \boldsymbol{\Phi}$ . The  $i^{th}$  row and  $j^{th}$  column of matrix  $\mathbf{A}$  is written as  $a_{ij}$ . The  $j^{th}$  column of matrix  $\mathbf{A}$  is written as  $\mathbf{a}_j$ . When we need to refer to the  $i^{th}$  row of a matrix, we write this as  $\mathbf{a}_{i\bullet}$  where the bullet  $\bullet$  indicates that we are considering all possible values of the column index.

We concatenate two  $D \times 1$  column vectors horizontally as  $\mathbf{a} = [\mathbf{b}, \mathbf{c}]$  to form the  $D \times 2$  matrix  $\mathbf{A}$ . We concatenate two  $D \times 1$  column vectors vertically as  $\mathbf{a} = [\mathbf{b}^T, \mathbf{c}^T]^T$  to form the  $2D \times 1$  vector  $\mathbf{a}$ . Although this notation is cumbersome, it allows us to represent vertical concatenations within a single line of text.

### Variables and parameters

We denote variables with Roman letters  $\mathbf{a}, \mathbf{b}$ . The most common examples are the observed data which is always denoted by  $\mathbf{x}$  and the state of the world which is always denoted by  $\mathbf{w}$ . However, other hidden or latent variables are also represented by Roman letters. We denote parameters of the model by Greek letters  $\mu, \boldsymbol{\Phi}, \sigma^2$ . These are distinguished from variables in that there is usually a single set of parameters that explains the relation between many sets of variables.

### Functions

We write functions as a name, followed by square brackets that contain the arguments of the function. For example,  $\log[x]$  returns the logarithm of the scalar variable  $x$ . Sometimes we will write a function with bullets  $\bullet$  as arguments (e.g.,  $\text{atan2}[\bullet, \bullet]$ ) to focus the interest on the function itself rather than the arguments.

When the function returns one or more vector or matrix arguments, it is written in bold. For example, the function  $\text{aff}[\mathbf{x}, \boldsymbol{\Phi}, \boldsymbol{\tau}]$  applies an affine transformation to the 2D point  $\mathbf{x}$  with parameters  $\boldsymbol{\Phi}, \boldsymbol{\tau}$  and returns a new 2D vector output. When a

function returns multiple outputs, we write this in Matlab notation so  $[\mathbf{U}, \mathbf{L}, \mathbf{V}] = \text{svd}[\mathbf{X}]$  returns the three parts  $\mathbf{U}, \mathbf{L}, \mathbf{V}$  of the singular value decomposition of  $\mathbf{X}$ .

Some functions are used repeatedly throughout the text. These include:

- $\min_x f[x]$ , which returns the minimum possible value of the function  $f[x]$  as we vary  $x$  over its entire valid range,
- $\operatorname{argmin}_x f[x]$ , which returns the value of the argument  $x$  that minimizes  $f[x]$ ,
- $\max_x$  and  $\operatorname{argmax}_x$ , which fulfill the same roles as  $\min_x$  and  $\operatorname{argmin}_x$  but where we are maximizing the function,
- $\operatorname{diag}[\mathbf{A}]$ , which returns a column vector containing the elements on the diagonal of matrix  $\mathbf{A}$ ,
- $\delta[x]$  for continuous  $x$ , which is a Dirac delta function and has the key property  $\int f[x]\delta[x - x_0]dx = f[x_0]$ ,
- $\delta[x]$  for discrete  $x$ , which returns 1 when the argument  $x$  is 0 and returns 0 otherwise, and
- $\operatorname{heaviside}[x]$ , which represents the Heaviside step function. It returns 0 when the argument  $x < 0$  and returns 1 otherwise.

### Probability distributions

We write the probability of a random variable  $x$  as  $Pr(x)$ . We write the joint probability of two variables  $a, b$  as  $Pr(a, b)$  and the conditional probability of  $a$  given  $b$  as  $Pr(a|b)$ . Sometimes, we wish to specify the exact value  $b^*$  that  $a$  is conditioned upon and here we write  $Pr(a|b = b^*)$ . Occasionally, we denote that variables  $a$  and  $b$  are independent by writing  $a \perp\!\!\!\perp b$ . Similarly we indicate that  $a$  and  $b$  are conditionally independent given  $c$  by writing  $a \perp\!\!\!\perp b|c$ .

Probability distributions are written in the style  $Pr(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}, \boldsymbol{\Sigma}]$ . This returns the value of the multivariate normal distribution for data  $\mathbf{x}$  when the distribution has mean  $\boldsymbol{\mu}$  and covariance  $\boldsymbol{\Sigma}$ . In this way, we always distinguish the argument of the distribution (here  $\mathbf{x}$ ) from the parameters (here  $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ ).

### Sets

We denote sets with calligraphic letters  $\mathcal{S}$ . The notation  $\mathcal{S} \subset \mathcal{T}$  indicates that  $\mathcal{S}$  is a subset of  $\mathcal{T}$ . The notation  $x \in \mathcal{S}$  indicates that  $x$  is a member of the set  $\mathcal{S}$ . The notation  $\mathcal{A} = \mathcal{B} \cup \mathcal{C}$  indicates that set  $\mathcal{A}$  is the union of sets  $\mathcal{B}$  and  $\mathcal{C}$ . The notation  $\mathcal{A} = \mathcal{B} \setminus \mathcal{C}$  indicates that set  $\mathcal{A}$  consists of all of the elements of  $\mathcal{B}$  except those that are in  $\mathcal{C}$ .

Often we write out a set explicitly in terms of the elements and for this we use curly brackets so that  $\mathcal{A} = \{x, y, z\}$  indicates that the set  $\mathcal{A}$  contains  $x, y$ , and  $z$  and nothing else. When a set is empty, we write  $\mathcal{A} = \{\}$ . We use the notation  $\{x_i\}_{i=1}^I$  as shorthand to represent the set  $\{x_1, x_2, \dots, x_I\}$ , and we may write the same set in the compact form  $x_{1\dots I}$  if it is part of an equation.

# Appendix B

# Optimization

Throughout this book, we have used iterative nonlinear optimization methods to find the maximum likelihood or MAP parameter estimates. We now provide more details about these methods. It is impossible to do full justice to this topic in the space available; many entire books have been written about nonlinear optimization. Our goal is merely to provide a brief introduction to the main ideas.

## B.1 Problem statement

Continuous nonlinear optimization techniques aim to find the set of parameters  $\hat{\boldsymbol{\theta}}$  that minimize a function  $f[\bullet]$ . In other words, they try to compute,

$$\hat{\boldsymbol{\theta}} = \operatorname{argmin}_{\boldsymbol{\theta}} [f[\boldsymbol{\theta}]], \quad (\text{B.1})$$

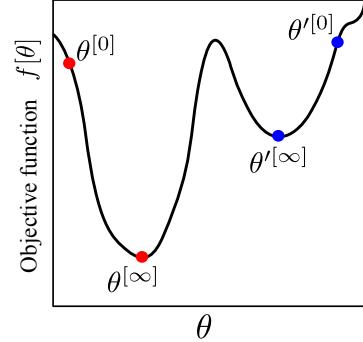
where  $f[\bullet]$  is termed a *cost function* or *objective function*.

Although optimization techniques are usually described in terms of minimizing a function, most optimization problems in this book involve *maximizing* an objective function based on log probability. To turn a maximization problem into a minimization we multiply the objective function by minus one. In other words, instead of maximizing the log probability, we minimize the negative log probability.

### B.1.1 Convexity

The optimization techniques that we consider here are iterative: they start with an estimate  $\boldsymbol{\theta}^{[0]}$  and improve it by finding successive new estimates  $\boldsymbol{\theta}^{[1]}, \boldsymbol{\theta}^{[2]}, \dots, \boldsymbol{\theta}^{[\infty]}$ , each of which is better than the last, until no more improvement can be made. The techniques are purely local in the sense that the decision about where to move next is based on only the properties of the function at the current position. Consequently, these techniques cannot guarantee the correct solution: they may find an estimate  $\boldsymbol{\theta}^{[\infty]}$  from which no local change improves the cost. However, this

**Figure B.1** Local minima. Optimization methods aim to find the minimum of the objective function  $f[\theta]$  with respect to parameters  $\theta$ . Roughly, they work by starting with an initial estimate  $\theta^{[0]}$  and moving iteratively downhill until no more progress can be made (final position represented by  $\theta^{[\infty]}$ ). Unfortunately, it is possible to terminate in a local minimum. For example, if we start at  $\theta^{[0]}$  and move downhill, we wind up in position  $\theta'^{[\infty]}$ .



does not mean there is not a better solution in some distant part of the function that has not yet been explored (figure B.1). In optimization parlance, they can only find *local minima*. One way to mitigate this problem is to start the optimization from a number of different places and choose the final solution with the lowest cost.

In the special case where the function is *convex*, there will only be a single minimum, and we are guaranteed to find it with sufficient iterations (figure B.2). For a 1D function, it is possible to establish the convexity by looking at the second derivative of the function; if this is positive everywhere (i.e., the slope is continuously increasing) then the function is convex and the global minimum can be found. The equivalent test in higher dimensions is to examine the *Hessian matrix* (the matrix of second derivatives of the cost function with respect to the parameters). If this is positive definite everywhere (see appendix C.2.6), then the function is convex and the global minimum will be found. Some of the cost functions in this book are convex, but this is unusual; most optimization problems found in vision do not have this convenient property.

### B.1.2 Overview of approach

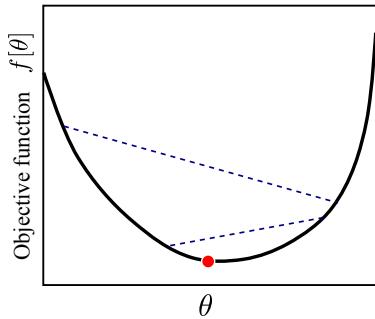
In general the parameters  $\theta$  over which we search are multidimensional. For example, when  $\theta$  has two dimensions, we can think of the function as a two dimensional surface (figure B.3). With this in mind, the principles behind the methods we will discuss are simple. We alternately

- choose a search direction  $s$  based on the local properties of the function, and
- search to find the minimum along the chosen direction. In other words, we seek the distance  $\lambda$  to move such that

$$\hat{\lambda} = \operatorname{argmin}_{\lambda} [f[\theta^{[t]} + \lambda s]], \quad (\text{B.2})$$

and then set  $\theta^{[t+1]} = \theta^{[t]} + \hat{\lambda}s$ . This is termed a *line search*.

We now consider each of these stages in turn.



**Figure B.2** Convex functions. If the function is convex, then the global minimum can be found. A function is convex if no chord (line between two points on the function) intersects the function. The figure shows two example chords (blue dashed lines). The convexity of a function can be established algebraically by considering the matrix of second derivatives. If this is positive definite for all values of  $\theta$ , then the function is convex.

## B.2 Choosing a search direction

We will describe two general methods for choosing a search direction (*steepest descent* and *Newton's method*) and one method which is specialized for least squares problems (the *Gauss-Newton method*). All of these methods rely on computing derivatives of the function with respect to the parameters at the current position. To this end, we are relying on the function being smooth so that the derivatives are well behaved.

For most models, it is easy to find a closed form expression for the derivatives. If this is not the case, then an alternative is to approximate them using finite differences. For example, the first derivative of  $f[\bullet]$  with respect to the  $j^{th}$  element of  $\theta$  can be approximated by

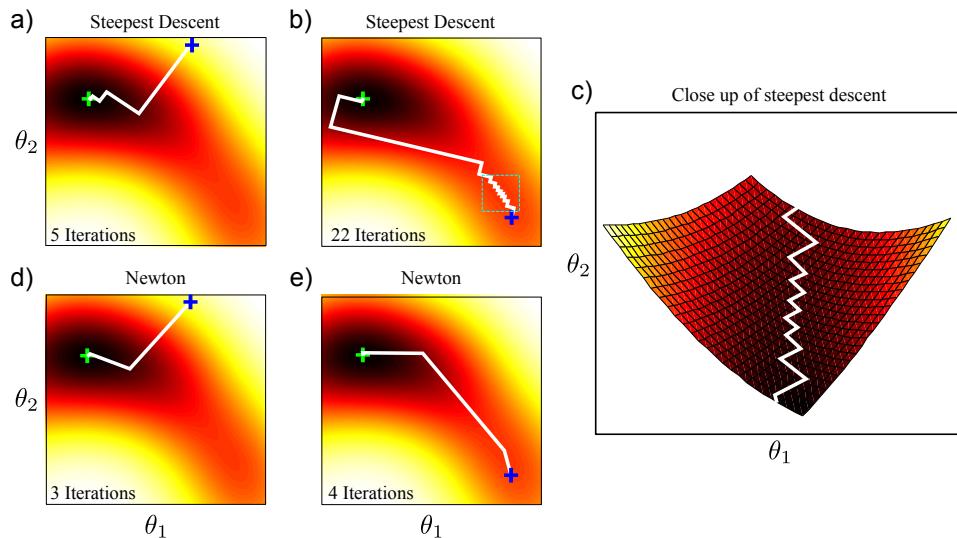
$$\frac{\partial f}{\partial \theta_j} \approx \frac{f[\theta + a\mathbf{e}_j] - f[\theta]}{a}, \quad (\text{B.3})$$

where  $a$  is a small number and  $\mathbf{e}_j$  is the unit vector in the  $j^{th}$  direction. In principle as  $a$  tends to zero, this estimate becomes more accurate. However, in practice the calculation is limited by the floating point precision of the computer, so  $a$  must be chosen with care.

### B.2.1 Steepest descent

An intuitive way to choose the search direction is to measure the gradient and select the direction which moves us downhill fastest. We could move in this direction until the function no longer decreases, then recompute the steepest direction and move again. In this way, we gradually move toward a local minimum of the function (figure B.3a). The algorithm terminates when the gradient is zero and the second derivative is positive, indicating that we are at the minimum point and any further local changes would not result in further improvement. This approach is termed *steepest descent*. More precisely, we choose

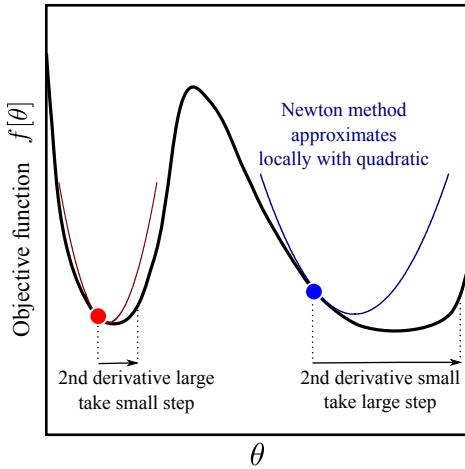
$$\theta^{[t+1]} = \theta^{[t]} - \lambda \left. \frac{\partial f}{\partial \theta} \right|_{\theta^{[t]}}, \quad (\text{B.4})$$



**Figure B.3** Optimization on a two dimensional function (color represents height of function). We wish to find the parameters that minimize the function (green cross). Given an initial starting point  $\theta^0$  (blue cross), we choose a direction and then perform a local search to find the optimal point in that direction. a) One way to chose the direction is steepest descent: at each iteration, we head in the direction where the function changes the fastest. b) When we initialize from a different position, the steepest descent method takes many iterations to converge due to oscillatory behavior. c) Close-up of oscillatory region (see main text). d) Setting the direction using Newton's method results in faster convergence. e) Newton's method does not undergo oscillatory behavior when we initialize from the second position.

where the derivative  $\partial f / \partial \boldsymbol{\theta}$  is the *gradient vector*, which points uphill, and  $\lambda$  is the distance moved downhill in the opposite direction  $-\partial f / \partial \boldsymbol{\theta}$ . The line search procedure (section B.3) selects the value of  $\lambda$ .

Steepest descent sounds like a good idea but can be very inefficient in certain situations (figure B.3b). For example, in a descending valley, it can oscillate ineffectually from one side to the other rather than proceeding straight down the center: the method approaches the bottom of the valley from one side, but overshoots because the valley itself is descending, so the minimum along the search direction is not exactly in the valley center (figure B.3c). When we re-measure the gradient and perform a second line search, we overshoot in the other direction. This is not an unusual situation: it is guaranteed that the gradient at the new point will be perpendicular to the previous one, so the only way to avoid this oscillation is to hit the valley at exactly right angles.



**Figure B.4** Use of second derivatives. The gradient at the red and blue points is the same, but the magnitude of the second derivative is larger at the red point than the blue point: the gradient is changing faster at the red point than the blue point. The distance we move should be moderated by the second derivative: if the gradient is changing fast, then the minimum may be nearby and we should move a small distance. If it is changing slowly, then it is safe to move further. Newton's method takes into account the second derivative: it uses a Taylor expansion to create a quadratic approximation to the function and then moves toward the minimum.

## B.2.2 Newton's method

Newton's method is an improved approach that also exploits the second derivatives at the current point: it considers both the gradient of the function and how that gradient is changing.

To motivate the use of second derivatives, consider a one-dimensional function (figure B.4). If the magnitude of the second derivative is low, then the gradient is changing slowly. Consequently, it will probably take a while before it completely flattens out and becomes a minimum, and so it is safe to move a long distance. Conversely, if the magnitude of the second derivative is high, then things are changing rapidly, and we should move only a small distance.

Now consider the same argument in two dimensions. Imagine we are at a point where the gradient is identical in both dimensions. For steepest descent, we would move equally in both dimensions. However, if the magnitude of the second derivative in the first direction is much greater than that in the second, we would nonetheless wish to move further in the second direction.

To see how to exploit the second derivatives algebraically, consider a truncated Taylor expansion around the current estimate  $\boldsymbol{\theta}^{[t]}$ :

$$f[\boldsymbol{\theta}] \approx f[\boldsymbol{\theta}^{[t]}] + (\boldsymbol{\theta} - \boldsymbol{\theta}^{[t]})^T \frac{\partial f}{\partial \boldsymbol{\theta}} \Big|_{\boldsymbol{\theta}^{[t]}} + \frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\theta}^{[t]})^T \frac{\partial^2 f}{\partial \boldsymbol{\theta}^2} \Big|_{\boldsymbol{\theta}^{[t]}} (\boldsymbol{\theta} - \boldsymbol{\theta}^{[t]}), \quad (\text{B.5})$$

where  $\boldsymbol{\theta}$  is a  $D \times 1$  variable, the first derivative vector is of size  $D \times 1$ , and the Hessian matrix of second derivatives is  $D \times D$ . To find the local extrema, we now take derivatives with respect to  $\boldsymbol{\theta}$  and set the result to zero

$$\frac{\partial f}{\partial \boldsymbol{\theta}} \approx \frac{\partial f}{\partial \boldsymbol{\theta}} \Big|_{\boldsymbol{\theta}^{[t]}} + \frac{\partial^2 f}{\partial \boldsymbol{\theta}^2} \Big|_{\boldsymbol{\theta}^{[t]}} (\boldsymbol{\theta} - \boldsymbol{\theta}^{[t]}) = 0. \quad (\text{B.6})$$

By re-arranging this equation, we get an expression for the minimum  $\hat{\boldsymbol{\theta}}$ ,

$$\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}^{[t]} - \left( \frac{\partial^2 f}{\partial \boldsymbol{\theta}^2} \right)^{-1} \frac{\partial f}{\partial \boldsymbol{\theta}}, \quad (\text{B.7})$$

where the derivatives are still taken at  $\boldsymbol{\theta}^{[t]}$ , but we have stopped writing this for clarity. In practice we would implement Newton's method as a series of iterations

$$\boldsymbol{\theta}^{[t+1]} = \boldsymbol{\theta}^{[t]} - \lambda \left( \frac{\partial^2 f}{\partial \boldsymbol{\theta}^2} \right)^{-1} \frac{\partial f}{\partial \boldsymbol{\theta}}, \quad (\text{B.8})$$

where the  $\lambda$  is the step size. This can be set to one, or we can find the optimal value using line search.

One interpretation of Newton's method is that we have locally approximated the function as a quadratic. On each iteration, we move toward its extremum (or move exactly to it if we fix  $\lambda = 1$ ). Note that we are assuming that we are close enough to the correct solution that the nearby extremum *is* a minimum and not a saddle point or maximum. In particular, if the Hessian is not positive definite then a direction that is not downhill may be chosen. In this sense Newton's method is not as robust as steepest descent.

Subject to this limitation, Newton's method converges in fewer iterations than steepest descent (figure B.3d-e). However, it requires more computation per iteration as we have to invert the  $D \times D$  Hessian matrix at each step. Choosing this method usually implies that we can write the Hessian in closed form; approximating the Hessian from finite derivatives requires many function evaluations and so is potentially very costly.

### B.2.3 Gauss-Newton method

Cost functions in computer vision often take the special form of a least squares problem

$$f[\boldsymbol{\theta}] = \sum_{i=1}^I (\mathbf{x}_i - \mathbf{g}[\mathbf{w}_i, \boldsymbol{\theta}])^T (\mathbf{x}_i - \mathbf{g}[\mathbf{w}_i, \boldsymbol{\theta}]), \quad (\text{B.9})$$

where  $\mathbf{g}[\bullet, \bullet]$  is a function that transfers the variables  $\{\mathbf{w}_i\}$  into the space of the variables  $\{\mathbf{x}_i\}$ , and is parameterized by  $\boldsymbol{\theta}$ . In other words, we seek the values of  $\boldsymbol{\theta}$  that most closely map  $\{\mathbf{w}_i\}$  to  $\{\mathbf{x}_i\}$  in a least squares sense. This cost function is a special case of the more general form  $f[\boldsymbol{\theta}] = \mathbf{z}^T \mathbf{z}$ , where

$$\mathbf{z} = \begin{bmatrix} \mathbf{x}_1 - \mathbf{g}[\mathbf{w}_1, \boldsymbol{\theta}] \\ \mathbf{x}_2 - \mathbf{g}[\mathbf{w}_2, \boldsymbol{\theta}] \\ \vdots \\ \mathbf{x}_I - \mathbf{g}[\mathbf{w}_I, \boldsymbol{\theta}] \end{bmatrix}. \quad (\text{B.10})$$

The *Gauss-Newton method* is an optimization technique that is used to solve least squares problems of the form

$$\hat{\boldsymbol{\theta}} = \operatorname{argmin} [f[\boldsymbol{\theta}]] \quad \text{where } f[\boldsymbol{\theta}] = \mathbf{z}[\boldsymbol{\theta}]^T \mathbf{z}[\boldsymbol{\theta}]. \quad (\text{B.11})$$

To minimize this objective function, we approximate the term  $\mathbf{z}[\boldsymbol{\theta}]$  with a Taylor series expansion around the current estimate  $\boldsymbol{\theta}^{[t]}$  of the parameters:

$$\mathbf{z}[\boldsymbol{\theta}] \approx \mathbf{z}[\boldsymbol{\theta}^{[t]}] + \mathbf{J}(\boldsymbol{\theta} - \boldsymbol{\theta}^{[t]}), \quad (\text{B.12})$$

where  $\mathbf{J}$  is the Jacobian matrix. The entry  $j_{mn}$  at the  $m^{th}$  row and the  $n^{th}$  column of  $\mathbf{J}$  contains the derivative of the  $m^{th}$  element of  $\mathbf{z}$  with respect to the  $n^{th}$  parameter so that

$$j_{mn} = \frac{\partial z_m}{\partial \theta_n}. \quad (\text{B.13})$$

Now we substitute the approximation for  $\mathbf{z}[\boldsymbol{\theta}]$  into the original cost function  $f[\boldsymbol{\theta}] = \mathbf{z}^T \mathbf{z}$  to yield

$$\begin{aligned} f[\boldsymbol{\theta}] &\approx (\mathbf{z}[\boldsymbol{\theta}^{[t]}] + \mathbf{J}(\boldsymbol{\theta} - \boldsymbol{\theta}^{[t]}))^T (\mathbf{z}[\boldsymbol{\theta}^{[t]}] + \mathbf{J}(\boldsymbol{\theta} - \boldsymbol{\theta}^{[t]})) \\ &= \mathbf{z}[\boldsymbol{\theta}^{[t]}]^T \mathbf{z}[\boldsymbol{\theta}^{[t]}] + 2(\boldsymbol{\theta} - \boldsymbol{\theta}^{[t]})^T \mathbf{J}^T \mathbf{z}[\boldsymbol{\theta}^{[t]}] + (\boldsymbol{\theta} - \boldsymbol{\theta}^{[t]})^T \mathbf{J}^T \mathbf{J}(\boldsymbol{\theta} - \boldsymbol{\theta}^{[t]}). \end{aligned} \quad (\text{B.14})$$

Finally, we take derivatives of this expression with respect to the parameters  $\boldsymbol{\theta}$  and equate to zero to get the relation

$$\frac{\partial f}{\partial \boldsymbol{\theta}} \approx 2\mathbf{J}^T \mathbf{z}[\boldsymbol{\theta}^{[t]}] + 2\mathbf{J}^T \mathbf{J}(\boldsymbol{\theta} - \boldsymbol{\theta}^{[t]}) = 0. \quad (\text{B.15})$$

Re-arranging, we get the update rule:

$$\boldsymbol{\theta} = \boldsymbol{\theta}^{[t]} - (\mathbf{J}^T \mathbf{J})^{-1} \mathbf{J}^T \mathbf{z}[\boldsymbol{\theta}^{[t]}]. \quad (\text{B.16})$$

We can rewrite this by noting that

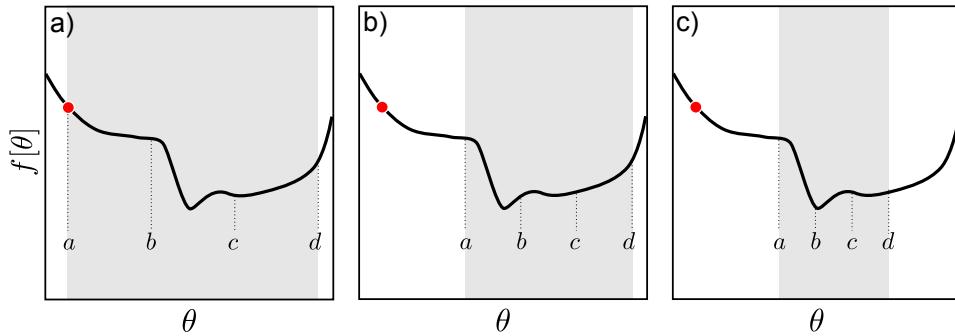
$$\left. \frac{\partial f}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta}^{[t]}} = \left. \frac{\partial \mathbf{z}^T \mathbf{z}}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta}^{[t]}} = 2\mathbf{J}^T \mathbf{z}[\boldsymbol{\theta}^{[t]}], \quad (\text{B.17})$$

to give the final Gauss-Newton update

$$\boldsymbol{\theta}^{[t+1]} = \boldsymbol{\theta}^{[t]} - \lambda (\mathbf{J}^T \mathbf{J})^{-1} \frac{\partial f}{\partial \boldsymbol{\theta}}, \quad (\text{B.18})$$

where the derivative is taken at  $\boldsymbol{\theta}^{[t]}$  and  $\lambda$  is the step size.

Comparing with the Newton update (equation B.8), we see that we can consider this update as approximating the Hessian matrix as  $\mathbf{H} \approx \mathbf{J}^T \mathbf{J}$ . It provides better results than gradient descent without ever computing second derivatives. Moreover, the term  $\mathbf{J}^T \mathbf{J}$  is normally positive definite resulting in increased stability.



**Figure B.5** Line search over region  $[a, d]$  using bracketing approach. Gray region indicates current search region. a) We define two points  $b, c$  that are interior to the search region and evaluate the function at these points. Here  $f[b] > f[c]$  so we eliminate the range  $[a, b]$ . b) We evaluate two points  $[b, c]$  interior to the new range and compare their values. This time we find that  $f[b] < f[c]$  so we eliminate the range  $[c, d]$ . c) We repeat this process until the minimum is closely bracketed.

### B.2.4 Other methods

There are numerous other methods for choosing the optimization direction. Many of these involve approximating the Hessian in some way with the goal of either ensuring that a downhill direction is always chosen or reducing the computational burden. For example, if computation of the Hessian is prohibitive, a practical approach is to approximate it with its own diagonal. This usually provides a better direction than steepest descent.

*Quasi-Newton methods* such as the *Broyden Fletcher Goldfarb Shanno (BFGS)* method approximate the Hessian with information gathered by analyzing successive gradient vectors. The *Levenberg-Marquardt* algorithm interpolates between the Gauss-Newton algorithm and steepest descent with the aim of producing a method that requires few iterations and is also robust. *Damped Newton* and *trust-region methods* also attempt to improve the robustness of Newton's method. The nonlinear *conjugate gradient algorithm* is another valuable method when only first derivatives are available.

## B.3 Line search

Having chosen a sensible direction using steepest descent, Newton's method or some other approach, we must now decide how far to move: we need an efficient method to find the minimum of the function in the chosen direction. Line search methods start by determining the range over which to search for the minimum.

This is usually guided by the magnitude of the second derivative along the line, which provides information about the likely search range (see figure B.4).

There are many heuristics to find the minimum, but we will discuss only the direct search method (figure B.5). Consider searching over the region  $[a, d]$ . We compute the function at two internal points  $b$  and  $c$  where  $a < b < c < d$ . If  $f[b] < f[c]$ , we eliminate the range  $[c, d]$  and search over the new region  $[a, c]$  at the next iteration. Conversely, if  $f[b] > f[c]$ , we eliminate the range  $[a, b]$  and search over a new region  $[b, d]$ .

This method is applied iteratively until the minimum is closely bracketed. It is typically not worth exactly locating the minimum; the line search direction is rarely optimal and so the minimum of the line search is usually far from the overall minimum of the function. Once the remaining interval is sufficiently small, an estimate of the minimum position can be computed by making a parabolic fit to the three points that remain after eliminating one region or the other and selecting the position of the minimum of this parabola.

## B.4 Reparameterization

Often in vision problems, we must find the best parameters  $\theta$  subject to one or more constraints. Typical examples include optimizing variances  $\sigma^2$  which must be positive, covariance matrices, which must be positive definite, and matrices that represent geometric rotations, which must be orthogonal. The general topic of constrained optimization is beyond the scope of this volume, but we briefly describe a trick that can be used to convert constrained optimization problems into unconstrained ones, which can be solved using the techniques already described.

The idea of reparameterization is to represent the parameters  $\theta$  in terms of a new set of parameters  $\phi$ , which do not have any constraints on them, so that

$$\theta = g[\phi], \quad (\text{B.19})$$

where  $g[\bullet]$  is a carefully chosen function.

Then we optimize with respect to the new unconstrained parameters  $\phi$ . The objective function becomes  $f[g[\phi]]$  and the derivatives are computed using the chain rule so that the first derivative would be

$$\frac{\partial f}{\partial \phi} = \sum_{k=1}^K \frac{\partial f}{\partial \theta_k} \frac{\partial \theta_k}{\partial \phi}. \quad (\text{B.20})$$

where  $\theta_k$  is the  $k^{th}$  element of  $\theta$ . This strategy is easier to understand with some concrete examples.

### Parameters that must be positive

When we optimize a variance parameter  $\theta = \sigma^2$  we must ensure that the final answer is positive. To this end, we use the relation:

$$\theta = \exp[\phi], \quad (\text{B.21})$$

and now optimize with respect to the new scalar parameter  $\phi$ . Alternatively, we can use the square relation:

$$\theta = \phi^2, \quad (\text{B.22})$$

and again optimize with respect to the parameter  $\phi$ .

#### Parameters that must lie between 0 and 1

To ensure that a scalar parameter  $\theta$  lies between zero and one, we use the logistic sigmoid function:

$$\theta = \frac{1}{1 + \exp[-\phi]}, \quad (\text{B.23})$$

and optimize with respect to the new scalar parameter  $\phi$ .

#### Parameters that must be positive and sum to one

To ensure that the elements of a  $K \times 1$  multivariable parameter  $\theta$  sum to one and are all positive we use the softmax function:

$$\theta_k = \frac{\exp[\phi_k]}{\sum_{j=1}^K \exp[\phi_j]}, \quad (\text{B.24})$$

and optimize with respect to the new  $K \times 1$  variable  $\phi$ .

#### 3D rotation matrices

A  $3 \times 3$  rotation matrix contains three independent quantities spread throughout its nine entries. A number of nonlinear constraints exist between the entries: the norm of each column and row must be one, each column is perpendicular to the other columns, each row is perpendicular to the other rows, and the determinant is one.

One way to enforce these constraints is to re-parameterize the rotation matrix as a *quaternion* and optimize with respect to this new representation. A quaternion  $\mathbf{q}$  is a 4D quantity  $\mathbf{q} = [q_0, q_1, q_2, q_3]$ . Mathematically speaking, they are a four dimensional extension of complex numbers, but the relevance for vision is that they can be used to represent 3D rotations. We use the relation:

$$\Theta = \frac{1}{q_0^2 + q_1^2 + q_2^2 + q_3^2} \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}. \quad (\text{B.25})$$

Although the quaternion contains four numbers, only the ratios of those numbers are important (giving 3 degrees of freedom): each element of equation B.25

consists of squared terms, which are normalized by the squared amplitude constant, and so any constant that multiplies the elements of  $\mathbf{q}$  is canceled out when we convert back to a rotation matrix.

Now we optimize with respect to the quaternion  $\mathbf{q}$ . The derivatives with respect to the  $k^{th}$  element of  $\mathbf{q}$  can be computed as

$$\frac{\partial f}{\partial q_k} = \sum_{i=1}^3 \sum_{j=1}^3 \frac{\partial f}{\partial \Theta_{ij}} \frac{\partial \Theta_{ij}}{\partial q_k}. \quad (\text{B.26})$$

The quaternion optimization is stable as long as we do not approach the singularity at  $\mathbf{q} = \mathbf{0}$ . One way to achieve this is to periodically re-normalize the quaternion to length 1 during the optimization procedure.

### Positive definite matrices

When we optimize over a  $K \times K$  covariance matrix  $\Theta = \Sigma$ , we must ensure that the result is positive definite. A simple way to do this is to use the relation:

$$\Theta = \Phi \Phi^T, \quad (\text{B.27})$$

where  $\Phi$  is an arbitrary  $K \times K$  matrix.



# Appendix C

## Linear algebra

### C.1 Vectors

A vector is a geometric entity in  $D$  dimensional space that has both a direction and a magnitude. It is represented by a  $D \times 1$  array of numbers. In this book, we write vectors as bold, small, Roman or Greek letters (e.g.,  $\mathbf{a}, \phi$ ). The transpose  $\mathbf{a}^T$  of vector  $\mathbf{a}$  is a  $1 \times D$  array of numbers where the order of the numbers is retained.

#### C.1.1 Dot product

The *dot product* or *scalar product* between two vectors  $\mathbf{a}$  and  $\mathbf{b}$  is defined as

$$c = \mathbf{a}^T \mathbf{b} = \sum_{d=1}^D a_d b_d, \quad (\text{C.1})$$

where  $\mathbf{a}^T$  is the transpose of  $\mathbf{a}$  (i.e.,  $\mathbf{a}$  converted to a row vector) and the returned value  $c$  is a scalar. Two vectors are said to be *orthogonal* if the dot product between them is zero.

#### C.1.2 Norm of a vector

The *magnitude* or *norm* of a vector is the square root of the sum of the square of the  $D$  elements so that,

$$\text{norm}[\mathbf{a}] = |\mathbf{a}| = \left( \sum_{d=1}^D a_d^2 \right)^{1/2} = (\mathbf{a}^T \mathbf{a})^{1/2}. \quad (\text{C.2})$$

### C.1.3 Cross product

The *cross product* or *vector product* is specialized to three dimensions. The operation  $\mathbf{c} = \mathbf{a} \times \mathbf{b}$  is equivalent to the matrix multiplication (section C.2.1):

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}, \quad (\text{C.3})$$

or for short

$$\mathbf{c} = \mathbf{A}_\times \mathbf{b}, \quad (\text{C.4})$$

where  $\mathbf{A}_\times$  is the  $3 \times 3$  matrix from equation C.3 that implements the cross product.

It is easily shown that the result  $\mathbf{c}$  of the cross product is orthogonal to both  $\mathbf{a}$  and  $\mathbf{b}$ . In other words,

$$\mathbf{a}^T(\mathbf{a} \times \mathbf{b}) = \mathbf{b}^T(\mathbf{a} \times \mathbf{b}) = 0. \quad (\text{C.5})$$

## C.2 Matrices

Matrices are used extensively throughout the book and are written as bold, capital, Roman or Greek letters (e.g.,  $\mathbf{A}, \Phi$ ). We categorize matrices as *landscape* (more columns than rows), *square* (the same number of columns and rows) or *portrait* (more rows than columns). They are always indexed by row first and then column, so  $a_{ij}$  denotes the element of matrix  $\mathbf{A}$  at the  $i^{th}$  row and the  $j^{th}$  column.

A *diagonal matrix* is a square matrix with zeros everywhere except on the diagonal (i.e., elements  $a_{ii}$ ) where the elements may take any value. An important special case of a diagonal matrix is the *identity matrix*  $\mathbf{I}$ . This has zeros everywhere except for the diagonal, where all the elements are ones.

### C.2.1 Matrix multiplication

To take the matrix product  $\mathbf{C} = \mathbf{AB}$ , we compute the elements of  $\mathbf{C}$  as

$$c_{ij} = \sum_{k=1}^K a_{ik} b_{kj}. \quad (\text{C.6})$$

This can only be done when the number of columns in  $\mathbf{A}$  equals the number of rows in  $\mathbf{B}$ . Matrix multiplication is associative so that  $\mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C} = \mathbf{ABC}$ . However it is not commutative so that in general  $\mathbf{AB} \neq \mathbf{BA}$ .

### C.2.2 Transpose

The transpose of a matrix  $\mathbf{A}$  is written as  $\mathbf{A}^T$  and is formed by reflecting it around the principal diagonal, so that the  $k^{th}$  column becomes the  $k^{th}$  row and vice-versa. If we take the transpose of a matrix product  $\mathbf{AB}$ , then we take the transpose of the original matrices but reverse the order so that

$$(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T. \quad (\text{C.7})$$

### C.2.3 Inverse

A square matrix  $\mathbf{A}$  may or may not have an inverse  $\mathbf{A}^{-1}$  such that  $\mathbf{A}^{-1}\mathbf{A} = \mathbf{AA}^{-1} = \mathbf{I}$ . If a matrix does not have an inverse, it is called *singular*.

Diagonal matrices are particularly easy to invert: the inverse is also a diagonal matrix, with each diagonal value  $d_{ii}$  replaced by  $1/d_{ii}$ . Hence, any diagonal matrix that has non-zero values on the diagonal is invertible. It follows that the inverse of the identity matrix is the identity matrix itself.

If we take the inverse of a matrix product  $\mathbf{AB}$ , then we can equivalently take the inverse of each matrix individually, and reverse the order of multiplication

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1} \mathbf{A}^{-1}. \quad (\text{C.8})$$

### C.2.4 Determinant and trace

Every square matrix  $\mathbf{A}$  has a scalar associated with it called the determinant, denoted by  $|\mathbf{A}|$  or  $\det[\mathbf{A}]$ . It is (loosely) related to the scaling applied by the matrix. As a rule of thumb, matrices where the magnitude of the determinant is small tend to make vectors smaller upon multiplication and matrices where the magnitude of the determinant is large tend to make them larger. If a matrix is singular, the determinant will be zero and there will be at least one direction in space that is mapped to the origin when the matrix is applied. For a diagonal matrix, the determinant is the product of the diagonal values. It follows that the determinant of the identity matrix is 1. Determinants of matrix expressions can be computed using the following rules:

$$|\mathbf{A}^T| = |\mathbf{A}| \quad (\text{C.9})$$

$$|\mathbf{AB}| = |\mathbf{A}||\mathbf{B}| \quad (\text{C.10})$$

$$|\mathbf{A}^{-1}| = 1/|\mathbf{A}|. \quad (\text{C.11})$$

The trace of a matrix is a second number associated with a square matrix  $\mathbf{A}$ . It is the sum of the diagonal values (the matrix itself need not be diagonal). The traces of compound terms are bound by the following rules:

$$\text{tr}[\mathbf{A}^T] = \text{tr}[\mathbf{A}] \quad (\text{C.12})$$

$$\text{tr}[\mathbf{AB}] = \text{tr}[\mathbf{BA}] \quad (\text{C.13})$$

$$\text{tr}[\mathbf{A} + \mathbf{B}] = \text{tr}[\mathbf{A}] + \text{tr}[\mathbf{B}] \quad (\text{C.14})$$

$$\text{tr}[\mathbf{ABC}] = \text{tr}[\mathbf{BCA}] = \text{tr}[\mathbf{CAB}], \quad (\text{C.15})$$

where in the last relation, the trace is invariant for cyclic permutations only, so that in general  $\text{tr}[\mathbf{ABC}] \neq \text{tr}[\mathbf{BAC}]$ .

### C.2.5 Orthogonal and rotation matrices

An important class of square matrix is the orthogonal matrix. Orthogonal matrices have the following special properties:

1. Each column has norm one, and each row has norm one.
2. Each column is orthogonal to every other column, and each row is orthogonal to every other row.

The inverse of an orthogonal matrix  $\Omega$  is its own transpose, so  $\Omega^T\Omega = \Omega^{-1}\Omega = \mathbf{I}$ ; orthogonal matrices are easy to invert! When this class of matrix pre-multiplies a vector, the effect is to rotate it around the origin and possibly reflect it.

Rotation matrices are a sub-class of orthogonal matrices that have the additional property that the determinant is one. As the name suggests, when this class of matrix pre-multiplies a vector, the effect is to rotate it around the origin with no reflection.

### C.2.6 Positive definite matrices

A  $D \times D$  real symmetric matrix  $\mathbf{A}$  is *positive definite* if  $\mathbf{x}^T \mathbf{Ax} > 0$  for all non-zero vectors  $\mathbf{x}$ . Every positive definite matrix is invertible and its inverse is also positive definite. The determinant and trace of a symmetric positive definite matrix are always positive. The covariance matrix  $\Sigma$  of a normal distribution is always positive definite.

### C.2.7 Null space of a matrix

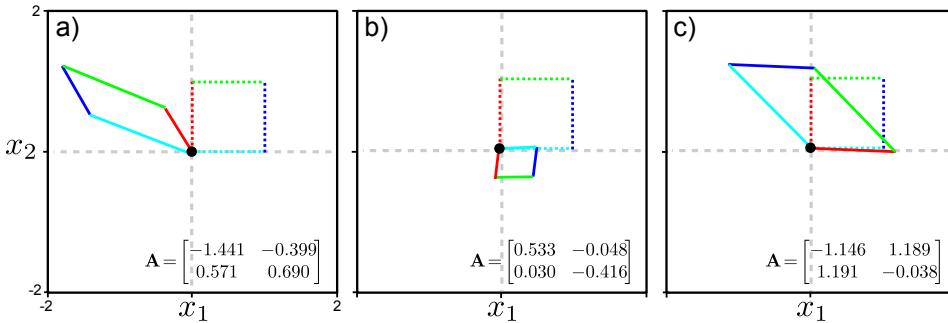
The right null space of a matrix  $\mathbf{A}$  consists of the set of vectors  $\mathbf{x}$  for which

$$\mathbf{Ax} = \mathbf{0}. \quad (\text{C.16})$$

Similarly, the left null space of a matrix  $\mathbf{A}$  consists of the set of vectors  $\mathbf{x}$  for which

$$\mathbf{x}^T \mathbf{A} = \mathbf{0}^T. \quad (\text{C.17})$$

A square matrix only has a non-trivial null space (i.e., not just  $\mathbf{x} = \mathbf{0}$ ) if the matrix is singular (non-invertible) and hence the determinant is zero.



**Figure C.1** Effect of applying three linear transformations to a unit square. Dashed square is before transformation. Solid square is after. The origin is always mapped to the origin. Co-linear points remain co-linear. Parallel lines remain parallel. The linear transformation encompasses shears, reflections, rotations and scalings.

## C.3 Tensors

We will occasionally have need for  $D > 2$  dimensional quantities that we shall refer to as  $D$ -dimensional tensors. For our purposes a matrix can be thought of as the special case of a 2 dimensional tensor, and a vector as the special case of a 1D tensor.

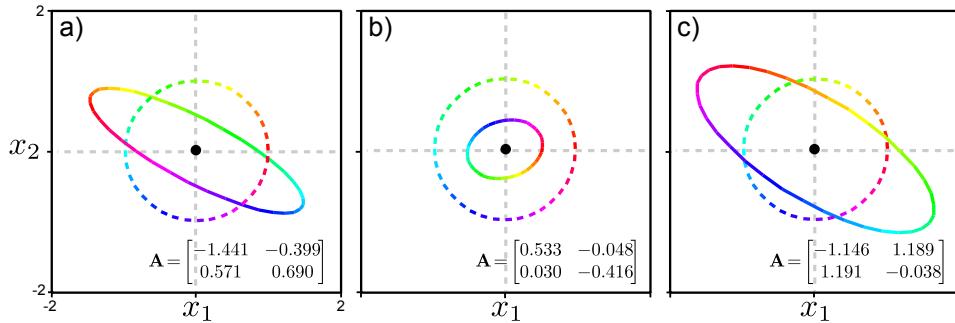
The idea of taking matrix products generalizes to higher dimensions and is denoted using the special notion  $\times_n$  where  $n$  is the dimension over which we take the product. For example, the  $l^{th}$  element  $f_l$  of the tensor product  $\mathbf{f} = \mathbf{A} \times_2 \mathbf{b} \times_3 \mathbf{c}$  is given by

$$f_l = \sum_m \sum_n A_{lmn} b_m c_n, \quad (\text{C.18})$$

where  $l$ ,  $m$  and  $n$  index the 3D tensor  $\mathbf{A}$ , and  $\mathbf{b}$  and  $\mathbf{c}$  are vectors.

## C.4 Linear transformations

When we pre-multiply a vector by a matrix this is called a linear transformation. Figure C.1 shows the results of applying several different 2D linear transformations (randomly chosen  $2 \times 2$  matrices) to the 2D vectors that represent the points of the unit square. We can deduce several things from this figure. First, the point  $(0,0)$  at the origin is always mapped back onto itself. Second, collinear points remain collinear. Third, parallel lines are always mapped to parallel lines. Viewed as a geometric transformation, pre-multiplication by a matrix can account for shearing, scaling, reflection and rotation around the origin.



**Figure C.2** Effect of applying three linear transformations to a circle. Dashed circle is before transformation. Solid ellipse is after. After the transformation, the circle is mapped to an ellipse. This demonstrates that there is one special direction that is expanded the most (becomes the major axis of the ellipse), and one special direction that is expanded the least (becomes the minor axis of the ellipse).

A different perspective on linear transforms comes from applying different transformations to points on the unit circle (figure C.2). In each case, the circle is transformed to an ellipse. The ellipse can be characterized by its major axis (most elongated axis) and its minor axis (most compact axis), which are perpendicular to one another. This tells us something interesting: in general, there is a special direction in space (position on the original circle) that gets stretched the most (or compressed the least) by the transformation. Likewise there is a second direction that gets stretched the least or compressed the most.

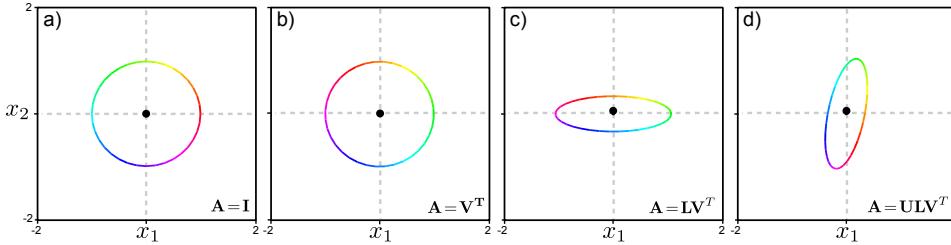
## C.5 Singular value decomposition

The singular value decomposition (SVD) is a factorization of a  $M \times N$  matrix  $\mathbf{A}$ , such that

$$\mathbf{A} = \mathbf{U}\mathbf{L}\mathbf{V}^T, \quad (\text{C.19})$$

where  $\mathbf{U}$  is a  $M \times M$  orthogonal matrix,  $\mathbf{L}$  is a  $M \times N$  diagonal matrix, and  $\mathbf{V}$  is a  $N \times N$  orthogonal matrix. It is always possible to compute this factorization, although a description of how to do so is beyond the scope of this book.

The best way to get the flavor of the SVD is to consider some examples. First let us consider a square matrix:



**Figure C.3** Cumulative effect of SVD components for matrix  $\mathbf{A}_3$ . a) Original object. b) Applying matrix  $\mathbf{V}^T$  rotates and reflects the object around the origin. c) Subsequently applying  $\mathbf{L}$  causes stretching/compression along the coordinate axes. d) Finally, applying matrix  $\mathbf{U}$  rotates and reflects this distorted structure.

$$\begin{aligned}\mathbf{A}_1 &= \begin{bmatrix} 0.183 & 0.307 & 0.261 \\ -1.029 & 0.135 & -0.941 \\ 0.949 & 0.515 & -0.162 \end{bmatrix} = \mathbf{U}\mathbf{L}\mathbf{V}^T \quad (\text{C.20}) \\ &= \begin{bmatrix} -0.204 & -0.061 & -0.977 \\ 0.832 & -0.535 & -0.140 \\ -0.514 & -0.842 & 0.160 \end{bmatrix} \begin{bmatrix} 1.590 & 0 & 0 \\ 0 & 0.856 & 0 \\ 0 & 0 & 0.303 \end{bmatrix} \begin{bmatrix} -0.870 & -0.302 & 0.389 \\ -0.135 & -0.613 & -0.778 \\ -0.474 & 0.729 & -0.492 \end{bmatrix}.\end{aligned}$$

Notice that by convention, the non-negative values on the principal diagonal of  $\mathbf{L}$  decrease monotonically as we move from top-left to bottom-right. These are known as the *singular values*.

Now consider the singular value decomposition of a portrait matrix:

$$\begin{aligned}\mathbf{A}_2 &= \begin{bmatrix} 0.537 & 0.862 \\ 1.839 & 0.318 \\ -2.258 & -1.307 \end{bmatrix} = \mathbf{U}\mathbf{L}\mathbf{V}^T \quad (\text{C.21}) \\ &= \begin{bmatrix} -0.263 & 0.698 & 0.665 \\ -0.545 & -0.676 & 0.493 \\ 0.795 & -0.233 & 0.559 \end{bmatrix} \begin{bmatrix} 3.273 & 0 \\ 0 & 0.76 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} -0.898 & -0.440 \\ -0.440 & 0.898 \end{bmatrix}.\end{aligned}$$

For this rectangular matrix, the orthogonal matrices  $\mathbf{U}$  and  $\mathbf{V}$  are different sizes and the diagonal matrix  $\mathbf{L}$  is the same size as the original matrix. The singular values are still found on the diagonal, but the number is determined by the smallest dimension. In other words, if the original matrix was  $M \times N$  then there will be  $\min[M, N]$  singular values.

To further understand the SVD, let us consider a third example:

$$\mathbf{A}_3 = \begin{bmatrix} -0.147 & 0.357 \\ -0.668 & 0.811 \end{bmatrix} = \begin{bmatrix} 0.189 & 0.981 \\ 0.981 & -0.189 \end{bmatrix} \begin{bmatrix} 1.068 & 0 \\ 0 & 0.335 \end{bmatrix} \begin{bmatrix} -0.587 & 0.8091 \\ 0.809 & 0.587 \end{bmatrix}. \quad (\text{C.22})$$

Figure C.3 illustrates the cumulative effect of the transformations in the decomposition  $\mathbf{A}_3 = \mathbf{ULV}^T$ . The matrix  $\mathbf{V}^T$  rotates and reflects the original points. The matrix  $\mathbf{L}$  scales the result differently along each dimension. In this case it is stretched along the first dimension and shrunk along the second. Finally, the matrix  $\mathbf{U}$  rotates the result.

Figure C.4 provides a second perspective on this process. Each pair of panels depicts what happens when we modify a different part of the SVD but keep the remaining parts the same. When we change  $\mathbf{V}$ , the shape of the final ellipse is the same, but the mapping from original directions to points on the ellipse changes (observe the color change along the major axis). When we modify the first element of  $\mathbf{L}$ , the length of the major axis changes. When we change the other non-zero element of  $\mathbf{L}$ , the length of the minor axis changes. When we change the matrix  $\mathbf{U}$ , the orientation of the ellipse changes.

### C.5.1 Analyzing the singular values

We can learn a lot about a matrix by looking at the singular values. We saw in the previous section, that as we decrease the smallest singular value the minor axis of the ellipse becomes progressively smaller. When it actually becomes zero, both sides of the unit circle collapse into one another (as do points from circles of all radii). Now there is a many-to-one mapping from the original points to the transformed ones, and the matrix is no longer invertible. In general, a matrix is only invertible if all of the singular values are non-zero.

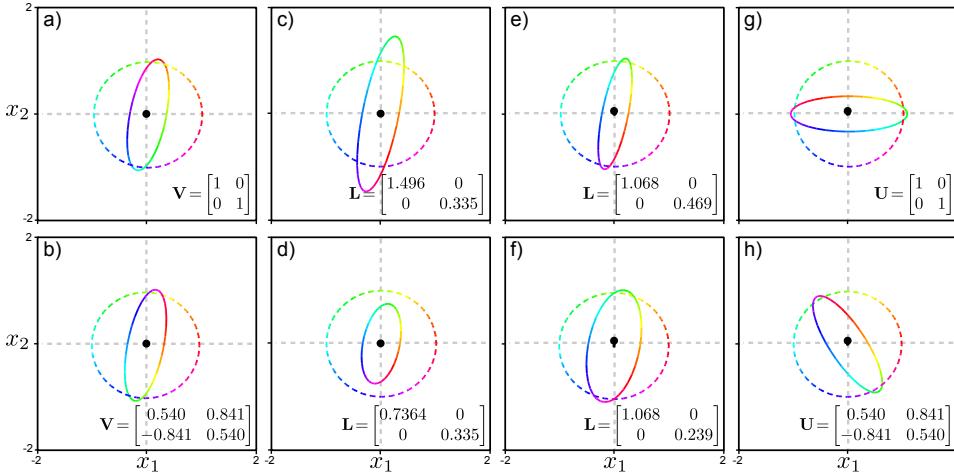
The number of non-zero singular values is called the rank of the matrix. The ratio of the smallest to the largest singular values is known as the condition number: it is roughly a measure of how ‘invertible’ the matrix is. As it becomes close to zero, our ability to invert the matrix decreases.

The singular values scale the different axes of the ellipse by different amounts (figure C.3c). Hence, the area of a unit circle is changed by a factor that is equal to the product of the singular values. In fact this scaling factor is the determinant (section C.2.4). When the matrix is singular, at least one of the singular values is zero and hence the determinant is also zero. The right null space consists of all of the vectors that can be reached by taking a weighted sum of those columns of  $\mathbf{V}$  whose corresponding singular values are zero. Similarly, the left null space consists of all of the vectors that can be reached by taking a weighted sum of those columns of  $\mathbf{U}$  whose corresponding singular values are zero.

Orthogonal matrices only rotate and reflect points, and rotation matrices just rotate them. In either case, there is no change in area to the unit circle: all the singular values are one for these matrices and the determinant is also one.

### C.5.2 Inverse of a matrix

We can also see what happens when we invert a square matrix in terms of the singular value decomposition. Using the rule  $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$ , we have



**Figure C.4** Manipulating different parts of the SVD of  $\mathbf{A}_3$ . a-b) Changing matrix  $\mathbf{V}$  does not affect the final ellipse, but changes which directions (colors) are mapped to the minor and major axes. c-d) Changing the first diagonal element of  $\mathbf{L}$  changes the length of the major axis of the ellipse. e-f) Changing the second diagonal element of  $\mathbf{L}$  changes the length of the minor axis. g-h) Changing  $\mathbf{U}$  affects the final orientation of the ellipse.

$$\mathbf{A}^{-1} = (\mathbf{U}\mathbf{L}\mathbf{V}^T)^{-1} = (\mathbf{V}^T)^{-1}\mathbf{L}^{-1}\mathbf{U}^{-1} = \mathbf{V}\mathbf{L}^{-1}\mathbf{U}^T, \quad (\text{C.23})$$

where we have used the fact that  $\mathbf{U}$  and  $\mathbf{V}$  are orthogonal matrices so  $\mathbf{U}^{-1} = \mathbf{U}^T$  and  $\mathbf{V}^{-1} = \mathbf{V}^T$ . The matrix  $\mathbf{L}$  is diagonal so  $\mathbf{L}^{-1}$  will also be diagonal with new non-zero entries that are the reciprocal of the original values. This also shows that the matrix is not invertible when any of the singular values are zero: we cannot take the reciprocal of 0.

Expressed in this way, the inverse has the opposite geometric effect to that of the original matrix: if we consider the effect on the transformed ellipse in figure C.3d, it first rotates by  $\mathbf{U}^T$  so its major and minor axis are aligned with the coordinate axes (figure C.3c). Then it scales these axes (using the elements of  $\mathbf{L}^{-1}$ ), so that the ellipse becomes a circle (figure C.3b). Finally, it rotates the result by  $\mathbf{V}$  to get back to the original position (figure C.3a).

## C.6 Matrix calculus

We are often called upon to take derivatives of compound matrix expressions. The derivative of a function  $f[\mathbf{a}]$  that takes a vector as its argument and returns a scalar is a vector  $\mathbf{b}$  with elements

$$b_i = \frac{\partial f}{\partial a_i} \quad (\text{C.24})$$

The derivative of a function  $f[\mathbf{A}]$  that returns a scalar, with respect to an  $M \times N$  matrix  $\mathbf{A}$  will be a  $M \times N$  matrix  $\mathbf{B}$  with elements

$$b_{ij} = \frac{\partial f}{\partial a_{ij}}. \quad (\text{C.25})$$

The derivative of a function  $\mathbf{f}[\mathbf{a}]$  that returns a vector with respect to vector  $\mathbf{a}$  is a matrix  $\mathbf{B}$  with elements

$$b_{ij} = \frac{\partial f_i}{\partial a_j}. \quad (\text{C.26})$$

where  $f_i$  is the  $i^{th}$  element of the vector returned by the function  $\mathbf{f}[\mathbf{a}]$ .

We now provide several commonly used results for reference.

1. Derivative of linear function:

$$\frac{\partial \mathbf{x}^T \mathbf{a}}{\partial \mathbf{x}} = \mathbf{a} \quad (\text{C.27})$$

$$\frac{\partial \mathbf{a}^T \mathbf{x}}{\partial \mathbf{x}} = \mathbf{a} \quad (\text{C.28})$$

$$\frac{\partial \mathbf{a}^T \mathbf{X} \mathbf{b}}{\partial \mathbf{X}} = \mathbf{a} \mathbf{b}^T \quad (\text{C.29})$$

$$\frac{\partial \mathbf{a}^T \mathbf{X}^T \mathbf{b}}{\partial \mathbf{X}} = \mathbf{b} \mathbf{a}^T. \quad (\text{C.30})$$

2. Derivative of quadratic function:

$$\frac{\partial \mathbf{b}^T \mathbf{X}^T \mathbf{X} \mathbf{c}}{\partial \mathbf{X}} = \mathbf{X}(\mathbf{b} \mathbf{c}^T + \mathbf{c} \mathbf{b}^T) \quad (\text{C.31})$$

$$\frac{\partial (\mathbf{Bx} + \mathbf{b})^T \mathbf{C}(\mathbf{Dx} + \mathbf{d})}{\partial \mathbf{x}} = \mathbf{B}^T \mathbf{C}(\mathbf{Dx} + \mathbf{d}) + \mathbf{D}^T \mathbf{C}^T (\mathbf{Bx} + \mathbf{b}) \quad (\text{C.32})$$

$$\frac{\partial \mathbf{x}^T \mathbf{Bx}}{\partial \mathbf{x}} = (\mathbf{B} + \mathbf{B}^T) \mathbf{x} \quad (\text{C.33})$$

$$\frac{\partial \mathbf{b}^T \mathbf{X}^T \mathbf{D} \mathbf{X} \mathbf{c}}{\partial \mathbf{X}} = \mathbf{D}^T \mathbf{X} \mathbf{b} \mathbf{c}^T + \mathbf{D} \mathbf{X} \mathbf{c} \mathbf{b}^T \quad (\text{C.34})$$

$$\frac{\partial (\mathbf{Xb} + \mathbf{c})^T \mathbf{D} (\mathbf{Xb} + \mathbf{c})}{\partial \mathbf{X}} = (\mathbf{D} + \mathbf{D}^T) (\mathbf{Xb} + \mathbf{c}) \mathbf{b}^T. \quad (\text{C.35})$$

3. Derivative of determinant:

$$\frac{\partial \det[\mathbf{Y}]}{\partial x} = \det[\mathbf{Y}] \text{tr} \left[ \mathbf{Y}^{-1} \frac{\partial \mathbf{Y}}{\partial x} \right], \quad (\text{C.36})$$

which leads to the relation

$$\frac{\partial \det[\mathbf{Y}]}{\partial \mathbf{Y}} = \det[\mathbf{Y}] \mathbf{Y}^{-T}. \quad (\text{C.37})$$

4. Derivative of log determinant:

$$\frac{\partial \log[\det[\mathbf{Y}]]}{\partial \mathbf{Y}} = \mathbf{Y}^{-T}. \quad (\text{C.38})$$

5. Derivative of inverse:

$$\frac{\partial \mathbf{Y}^{-1}}{\partial x} = -\mathbf{Y}^{-1} \frac{\partial \mathbf{Y}}{\partial x} \mathbf{Y}^{-1}. \quad (\text{C.39})$$

6. Derivative of trace:

$$\frac{\partial \text{tr}[\mathbf{F}[\mathbf{X}]]}{\partial \mathbf{X}} = \left( \frac{\partial \mathbf{F}[\mathbf{X}]}{\partial \mathbf{X}} \right)^T. \quad (\text{C.40})$$

More information about matrix calculus can be found in Petersen *et al.* (2006).

## C.7 Common problems

In this section, we discuss several standard linear algebra problems that are found repeatedly in computer vision.

### C.7.1 Least squares problems

Many inference and learning tasks in computer vision result in least squares problems. The most frequent context is when we use maximum likelihood methods with the normal distribution. The least squares problem may be formulated in a number of ways. We may be asked to find the vector  $\mathbf{x}$  that solves the system

$$\mathbf{Ax} = \mathbf{b} \quad (\text{C.41})$$

in a least squares sense. Alternatively, we may be given  $i$  of smaller sets of equations of the form

$$\mathbf{A}_i \mathbf{x} = \mathbf{b}_i, \quad (\text{C.42})$$

and again asked to solve for  $\mathbf{x}$ . In this latter case, we form the compound matrix  $\mathbf{A} = [\mathbf{A}_1^T, \mathbf{A}_2^T \dots \mathbf{A}_I^T]^T$  and compound vector  $\mathbf{b} = [\mathbf{b}_1^T, \mathbf{b}_2^T \dots \mathbf{b}_I^T]^T$ , and the problem is the same as in equation C.41.

We may equivalently see the same problem in an explicit least-squares form,

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \left[ (\mathbf{Ax} - \mathbf{b})^T (\mathbf{Ax} - \mathbf{b}) \right]. \quad (\text{C.43})$$

Finally, we may be presented the problem as a sum of smaller terms

$$\hat{\mathbf{x}} = \operatorname{argmin}_{\mathbf{x}} \left[ \sum_{i=1}^I (\mathbf{A}_i \mathbf{x} - \mathbf{b}_i)^T (\mathbf{A}_i \mathbf{x} - \mathbf{b}_i) \right], \quad (\text{C.44})$$

in which case we form compound matrices  $\mathbf{A}$  and  $\mathbf{b}$ , which changes the problem back to that in equation C.43.

To make progress, we multiply out the terms in equation C.43

$$\begin{aligned} \hat{\mathbf{x}} &= \operatorname{argmin}_{\mathbf{x}} [(\mathbf{Ax} - \mathbf{b})^T (\mathbf{Ax} - \mathbf{b})] \\ &= \operatorname{argmin}_{\mathbf{x}} [\mathbf{x}^T \mathbf{A}^T \mathbf{Ax} - \mathbf{b}^T \mathbf{Ax} - \mathbf{x}^T \mathbf{A}^T \mathbf{b} + \mathbf{b}^T \mathbf{b}] \\ &= \operatorname{argmin}_{\mathbf{x}} [\mathbf{x}^T \mathbf{A}^T \mathbf{Ax} - 2\mathbf{x}^T \mathbf{A}^T \mathbf{b} + \mathbf{b}^T \mathbf{b}], \end{aligned} \quad (\text{C.45})$$

where we have combined two terms in the last line by noting that they are both the same: they are transposes of one another, but they are also scalars, so they equal their own transpose. Now we take the derivative with respect to  $\mathbf{x}$  and equate the result to zero to give

$$2\mathbf{A}^T \mathbf{Ax} - 2\mathbf{A}^T \mathbf{b} = 0, \quad (\text{C.46})$$

which we can re-arrange to give the standard least squares result

$$\mathbf{x} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}. \quad (\text{C.47})$$

This result can only be computed if there are at least as many rows in  $\mathbf{A}$  as there are unknown values in  $\mathbf{x}$  (i.e., if the matrix  $\mathbf{A}$  is square or portrait). Otherwise, the matrix  $\mathbf{A}^T \mathbf{A}$  will be singular. For implementations in Matlab, it is better to make use of the backslash operator ‘\’ rather than explicitly implement equation C.47.

### C.7.2 Principal direction / minimum direction

We define the principal and minimal directions as

$$\begin{aligned} \hat{\mathbf{b}} &= \operatorname{argmax}_{\mathbf{b}} [\mathbf{Ab}] \quad \text{subject to } |\mathbf{b}| = 1 \\ \hat{\mathbf{b}} &= \operatorname{argmin}_{\mathbf{b}} [\mathbf{Ab}] \quad \text{subject to } |\mathbf{b}| = 1, \end{aligned} \quad (\text{C.48})$$

respectively. This problem has exactly the geometric form of figure C.2. The constraint that  $|\mathbf{b}| = 1$  means that  $\mathbf{b}$  has to lie on the circle (or sphere or hypersphere in higher dimensions). In the principal direction problem, we are hence seeking the direction that is mapped to the major axis of the resulting ellipse/ellipsoid. In the minimum direction problem, we seek the direction that is mapped to the minor axis of the ellipsoid.

We saw in figure C.4 that it is the matrix  $\mathbf{V}$  from the singular value decomposition of  $\mathbf{A}$  that controls which direction is mapped to the different axes of the ellipsoid. To solve the principal direction problem, we hence compute the SVD  $\mathbf{A} = \mathbf{U}\mathbf{L}\mathbf{V}^T$ , and set  $\mathbf{b}$  to be the first column of  $\mathbf{V}$ . To solve the minimum direction problem, we set  $\mathbf{b}$  to be the last column of  $\mathbf{V}$ .

### C.7.3 Orthogonal Procrustes problem

The orthogonal Procrustes problem is to find the closest linear mapping  $\Omega$  between one set of vectors  $\mathbf{A}$  and another  $\mathbf{B}$  such that  $\Omega$  is an orthogonal matrix. In layman's terms, we seek the best Euclidean rotation (possibly including mirroring) that maps points  $\mathbf{A}$  to points  $\mathbf{B}$ .

$$\hat{\Omega} = \underset{\Omega}{\operatorname{argmin}} [|\Omega\mathbf{A} - \mathbf{B}|_F], \quad (\text{C.49})$$

where  $|\bullet|_F$  denotes the Frobenius norm of a matrix – the sum of the square of all of the elements. To make progress, we recall that the trace of a matrix is the sum of its diagonal entries and so  $|\mathbf{X}|_F = \operatorname{tr}[\mathbf{X}^T \mathbf{X}]$  which gives the new criterion

$$\begin{aligned} \hat{\Omega} &= \underset{\Omega}{\operatorname{argmin}} \left[ \operatorname{tr}[\mathbf{A}^T \mathbf{A}] + \operatorname{tr}[\mathbf{B}^T \mathbf{B}] - 2\operatorname{tr}[\mathbf{A}^T \Omega^T \mathbf{B}] \right] \\ &= \underset{\Omega}{\operatorname{argmax}} \left[ \operatorname{tr}[\mathbf{A}^T \Omega^T \mathbf{B}] \right] \\ &= \underset{\Omega}{\operatorname{argmax}} \left[ \operatorname{tr}[\Omega^T \mathbf{B} \mathbf{A}^T] \right], \end{aligned} \quad (\text{C.50})$$

where we have used relation C.15 between the last two lines. We now compute the SVD  $\mathbf{B}\mathbf{A}^T = \mathbf{U}\mathbf{L}\mathbf{V}^T$  to get the criterion

$$\begin{aligned} \hat{\Omega} &= \underset{\Omega}{\operatorname{argmax}} \left[ \operatorname{tr}[\Omega^T \mathbf{U} \mathbf{L} \mathbf{V}^T] \right] \\ &= \underset{\Omega}{\operatorname{argmax}} \left[ \operatorname{tr}[\mathbf{V}^T \Omega^T \mathbf{U} \mathbf{L}] \right], \end{aligned} \quad (\text{C.51})$$

and notice that

$$\operatorname{tr}[\mathbf{V}^T \Omega^T \mathbf{U} \mathbf{L}] = \operatorname{tr}[\mathbf{Z} \mathbf{L}] = \sum_{i=1}^I z_{ii} l_{ii}, \quad (\text{C.52})$$

where we defined  $\mathbf{Z} = \mathbf{V}^T \Omega^T \mathbf{U}$ , and used the fact that the  $\mathbf{L}$  is a diagonal matrix, so each entry scales the diagonal of  $\mathbf{Z}$  on multiplication.

We note that the matrix  $\mathbf{Z}$  is orthogonal (it is the product of three orthogonal matrices). Hence, every value on the diagonal of the orthogonal matrix  $\mathbf{Z}$  must be less than or equal to one (the norms of each column are exactly one), and so we maximize the criterion in equation C.52 by choosing  $\mathbf{Z} = \mathbf{I}$  when the diagonal

values are equal to one. To achieve this, we set  $\Omega^T = \mathbf{V}\mathbf{U}^T$  so that the overall solution is

$$\hat{\Omega} = \mathbf{U}\mathbf{V}^T. \quad (\text{C.53})$$

A special case of this problem is to find the closest orthogonal matrix  $\Omega$  to a given square matrix  $\mathbf{B}$  in a least squares sense. In other words, we seek to optimize

$$\hat{\Omega} = \underset{\Omega}{\operatorname{argmin}} [|\Omega - \mathbf{B}|_F]. \quad (\text{C.54})$$

This is clearly equivalent to optimizing the criterion in equation C.49 but with  $\mathbf{A} = \mathbf{I}$ . It follows that the solution can be found by computing the singular value decomposition  $\mathbf{B} = \mathbf{U}\mathbf{L}\mathbf{V}^T$  and setting  $\Omega = \mathbf{U}\mathbf{V}^T$ .

## C.8 Tricks for inverting large matrices

Inversion of a  $D \times D$  matrix has a complexity of  $\mathcal{O}(D^3)$ . In practice, this means it is difficult to invert matrices whose dimension is larger than a few thousand. Fortunately, matrices are often highly structured, and we can exploit that structure using a number of tricks to speed up the process.

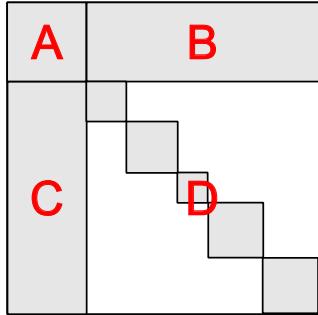
### C.8.1 Diagonal and block-diagonal matrices

Diagonal matrices can be inverted by forming a new diagonal matrix, where the values on the diagonal are reciprocal of the original values. Block diagonal matrices are matrices of the form

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2 & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{A}_N \end{bmatrix}. \quad (\text{C.55})$$

The inverse of a block-diagonal matrix can be computed by taking the inverse of each block separately so that

$$\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{A}_1^{-1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2^{-1} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{A}_N^{-1} \end{bmatrix}. \quad (\text{C.56})$$



**Figure C.5** Inversion relation #1. Gray regions indicate parts of matrix with non-zero values, white regions represent zeros. This relation is suited to the case where the matrix can be divided into four sub-matrices  $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$  and the bottom right block is easy to invert (e.g., diagonal, block diagonal or structured in another way that means that inversion is efficient). After applying this relation, the remaining inverse is the size of sub-matrix  $\mathbf{A}$ .

### C.8.2 Inversion relation #1: Schur complement identity

The inverse of a matrix with sub-blocks  $\mathbf{A}, \mathbf{B}, \mathbf{C}$ , and  $\mathbf{D}$  in the top-left, top right, bottom-left, and bottom right positions, respectively, can easily be shown to be

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} (\mathbf{A} - \mathbf{BD}^{-1}\mathbf{C})^{-1} & -(\mathbf{A} - \mathbf{BD}^{-1}\mathbf{C})^{-1}\mathbf{BD}^{-1} \\ -\mathbf{D}^{-1}\mathbf{C}(\mathbf{A} - \mathbf{BD}^{-1}\mathbf{C})^{-1} & \mathbf{D}^{-1} + \mathbf{D}^{-1}\mathbf{C}(\mathbf{A} - \mathbf{BD}^{-1}\mathbf{C})^{-1}\mathbf{BD}^{-1} \end{bmatrix}, \quad (\text{C.57})$$

by multiplying the original matrix with the right hand side and showing that the result is the identity matrix.

This result is extremely useful when the matrix  $\mathbf{D}$  is diagonal or block-diagonal (figure C.5). In this circumstance,  $\mathbf{D}^{-1}$  is fast to compute, and the remaining inverse quantity  $(\mathbf{A} - \mathbf{BD}^{-1}\mathbf{C})^{-1}$  is much smaller and easier to invert than the original matrix. The quantity  $\mathbf{A} - \mathbf{BD}^{-1}\mathbf{C}$  is known as the Schur complement.

### C.8.3 Inversion relation #2

Consider the  $d \times d$  matrix  $\mathbf{A}$ , the  $k \times k$  matrix  $\mathbf{C}$  and the  $k \times d$  matrix  $\mathbf{B}$  where  $\mathbf{A}$  and  $\mathbf{C}$  are symmetric, positive definite matrices. The following equality holds:

$$(\mathbf{A}^{-1} + \mathbf{B}^T \mathbf{C}^{-1} \mathbf{B})^{-1} \mathbf{B}^T \mathbf{C}^{-1} = \mathbf{A} \mathbf{B}^T (\mathbf{B} \mathbf{A} \mathbf{B}^T + \mathbf{C})^{-1}. \quad (\text{C.58})$$

**Proof:**

$$\begin{aligned} \mathbf{B}^T \mathbf{C}^{-1} \mathbf{B} \mathbf{A} \mathbf{B}^T + \mathbf{B}^T &= \mathbf{B}^T + \mathbf{B}^T \mathbf{C}^{-1} \mathbf{B} \mathbf{A} \mathbf{B}^T \\ \mathbf{B}^T \mathbf{C}^{-1} (\mathbf{B} \mathbf{A} \mathbf{B}^T + \mathbf{C}) &= (\mathbf{A}^{-1} + \mathbf{B}^T \mathbf{C}^{-1} \mathbf{B}) \mathbf{A} \mathbf{B}^T. \end{aligned} \quad (\text{C.59})$$

Taking the inverse of both sides we get

$$(\mathbf{A}^{-1} + \mathbf{B}^T \mathbf{C}^{-1} \mathbf{B})^{-1} \mathbf{B}^T \mathbf{C}^{-1} = \mathbf{A} \mathbf{B}^T (\mathbf{B} \mathbf{A} \mathbf{B}^T + \mathbf{C})^{-1}, \quad (\text{C.60})$$

as required.

This relation is very useful when  $\mathbf{B}$  is a landscape matrix with many more columns  $C$  than rows  $R$ . On the left-hand side, the term that we must invert is of size  $C \times C$ , which might be very costly. However, on the right-hand side, the inversion is only of size  $R \times R$ , which might be considerably more efficient.

#### C.8.4 Inversion relation #3: Sherman-Morrison-Woodbury

Consider the  $d \times d$  matrix  $\mathbf{A}$ , the  $k \times k$  matrix  $\mathbf{C}$  and the  $k \times d$  matrix  $\mathbf{B}$  where  $\mathbf{A}$  and  $\mathbf{C}$  are symmetric, positive definite matrices. The following equality holds:

$$(\mathbf{A}^{-1} + \mathbf{B}^T \mathbf{C}^{-1} \mathbf{B})^{-1} = \mathbf{A} - \mathbf{A} \mathbf{B}^T (\mathbf{B} \mathbf{B}^T + \mathbf{C})^{-1} \mathbf{B} \mathbf{A}. \quad (\text{C.61})$$

This is sometimes known as the *matrix inversion lemma*.

**Proof:**

$$\begin{aligned} & (\mathbf{A}^{-1} + \mathbf{B}^T \mathbf{C}^{-1} \mathbf{B})^{-1} \\ &= (\mathbf{A}^{-1} + \mathbf{B}^T \mathbf{C}^{-1} \mathbf{B})^{-1} (\mathbf{I} + \mathbf{B}^T \mathbf{C}^{-1} \mathbf{B} \mathbf{A} - \mathbf{B}^T \mathbf{C}^{-1} \mathbf{B} \mathbf{A}) \\ &= (\mathbf{A}^{-1} + \mathbf{B}^T \mathbf{C}^{-1} \mathbf{B})^{-1} ((\mathbf{A}^{-1} + \mathbf{B}^T \mathbf{C}^{-1} \mathbf{B}) \mathbf{A} - \mathbf{B}^T \mathbf{C}^{-1} \mathbf{B} \mathbf{A}) \\ &= \mathbf{A} - (\mathbf{A}^{-1} + \mathbf{B}^T \mathbf{C}^{-1} \mathbf{B})^{-1} \mathbf{B}^T \mathbf{C}^{-1} \mathbf{B} \mathbf{A}. \end{aligned} \quad (\text{C.62})$$

Now, applying inversion relation #2 to the term in brackets:

$$\begin{aligned} & (\mathbf{A}^{-1} + \mathbf{B}^T \mathbf{C}^{-1} \mathbf{B})^{-1} = \mathbf{A} - (\mathbf{A}^{-1} + \mathbf{B}^T \mathbf{C}^{-1} \mathbf{B})^{-1} \mathbf{B}^T \mathbf{C}^{-1} \mathbf{B} \mathbf{A} \\ &= \mathbf{A} - \mathbf{A} \mathbf{B}^T (\mathbf{B} \mathbf{B}^T + \mathbf{C})^{-1} \mathbf{B} \mathbf{A}, \end{aligned} \quad (\text{C.63})$$

as required.

#### C.8.5 Matrix determinant lemma

The matrices that we need to invert are often the covariances in the normal distribution. When this is the case we sometimes also need to compute the determinant of the same matrix and this may be prohibitively expensive. Fortunately, there is a direct analogy of the matrix inversion lemma for determinants which can reduce the cost when the covariance is structured.

Consider the  $d \times d$  matrix  $\mathbf{A}$ , the  $k \times k$  matrix  $\mathbf{C}$  and the  $k \times d$  matrix  $\mathbf{B}$  where  $\mathbf{A}$  and  $\mathbf{C}$  are symmetric, positive definite covariance matrices. The following equality holds:

$$|\mathbf{A}^{-1} + \mathbf{B}^T \mathbf{C}^{-1} \mathbf{B}| = |\mathbf{I} + \mathbf{B} \mathbf{A} \mathbf{B}^T| |\mathbf{C}|^{-1} |\mathbf{A}|^{-1}. \quad (\text{C.64})$$

# Bibliography

- AESCHLIMAN, C., PARK, J., & KAK, A. C. (2010) A novel parameter estimation algorithm for the multivariate t-distribution and its application to computer vision. In *European Conference on Computer Vision*, pp. 594–607. [134](#), [140](#)
- AGARWAL, A., & TRIGGS, B. (2006) Recovering 3D human pose from monocular images. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **28** (1): 44–48. [144](#), [166](#), [169](#)
- AGARWAL, S., SNAVELY, N., SEITZ, S. M., & SZELISKI, R. (2010) Bundle adjustment in the large. In *European Conference on Computer Vision*, pp. 29–42. [453](#)
- AGARWAL, S., SNAVELY, N., SIMON, I., SEITZ, S. M., & SZELISKI, R. (2009) Building Rome in a day. In *IEEE International Conference on Computer Vision*, pp. 72–79. [453](#)
- AGARWALA, A., DONTCHEVA, M., AGRAWALA, M., DRUCKER, S. M., COLBURN, A., CURLESS, B., SALESIN, D., & COHEN, M. F. (2004) Interactive digital photomontage. *ACM Transactions on Graphics* **23** (3): 294–302. [316](#)
- AGHAJANIAN, J., WARRELL, J., PRINCE, S. J. D., LI, P., ROHN, J. L., & BAUM, B. (2009) Patch-based within-object classification. In *IEEE International Conference on Computer Vision*, pp. 1125–1132. [595](#)
- AHONEN, T., HADID, A., & PIETIKÄINEN, M. (2004) Face recognition with local binary patterns. In *European Conference on Computer Vision*, pp. 469–481. [534](#)
- ALAHARI, K., KOHLI, P., & TORR, P. H. S. (2008) Reduce, reuse & recycle: Efficiently solving multi-label MRFs. In *IEEE Computer Vision & Pattern Recognition*. [316](#)
- ALLEN, B., CURLESS, B., & POPOVIC, Z. (2003) The space of human body shapes: reconstruction and parameterization from range scans. *ACM Transactions on Graphics* **22** (3): 587–594. [500](#)
- ALOIMONOS, J. Y. (1990) Perspective approximations. *Image and Vision Computing* **8** (3): 177–192. [385](#)
- AMBERG, B., BLAKE, A., & VETTER, T. (2009) On compositional image alignment, with an application to active appearance models. In *IEEE Computer Vision & Pattern Recognition*, pp. 1714–1721. [499](#)
- AMINI, A., WEYMOUTH, T., & JAIN, R. (1990) Using dynamic programming for solving variational problems in vision. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **12** (9): 855–867. [275](#), [499](#)
- AMIT, Y., & GEMAN, D. (1997) Shape quantization and recognition with randomized trees. *Neural Computation* **9** (7): 1545–1588. [209](#)
- AMIT, Y., & KONG, A. (1996) Graphical templates for model registration. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **18** (3): 225–236. [274](#)
- ANDRILUKA, M., ROTH, S., & SCHIELE, B. (2009) Pictorial structures revisited: people detection and articulated pose estimation. In *IEEE Computer Vision & Pattern Recognition*. [274](#)
- ANGUELOV, D., SRINIVASAN, P., KOLLER, D., THRUN, S., RODGERS, J., & DAVIS, J. (2005) SCAPE: shape completion and animation of people. *ACM Transactions on Graphics* **24** (3): 408–416. [496](#), [497](#), [498](#), [500](#)
- ARULAMPALAM, M., MASKELL, S., GORDON, N., & CLAPP, T. (2002) A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing* **50** (2): 174–188. [567](#)
- AVIDAN, S., & SHAMIR, A. (2007) Seam carving for content-aware image resizing. *ACM Transactions on Graphics* **26** (3): 10. [274](#)

- AYACHE, N. (1991) *Artificial vision for mobile robots: stereo vision and multisensory perception*. MIT Press. 567
- BABALOLA, K., COOTES, T., TWINING, C., PETROVIC, V., & TAYLOR, C. (2008) 3D brain segmentation using active appearance models and local regressors. In *Medical Image Computing and Computer-Assisted Intervention 2008*, ed. by D. Metaxas, L. Axel, G. Fichtinger, & G. Székely, volume 5241 of *Lecture Notes in Computer Science*, pp. 401–408. Springer Berlin / Heidelberg. 481
- BAILEY, T., & DURRANT-WHYTE, H. (2006) Simultaneous localization and mapping (SLAM): Part II. *Robotics & Automation Magazine, IEEE* **13** (3): 108–117. 568
- BAKER, H. H., & BINFORD, T. O. (1981) Depth from edge and intensity-based stereo. In *International Joint Conference on Artificial Intelligence*, pp. 631–636. 274
- BALLAN, L., & CORTELAZZO, G. M. (2008) Marker-less motion capture of skinned models in a four camera set-up using optical flow and silhouettes. In *3D Data Processing, Visualization and Transmission*. 385
- BALUJA, S., & ROWLEY, H. A. (2003) Boosting sex identification performance. *International Journal of Computer Vision* **71** (1): 111–119. 210
- BARBER, D. (2012) *Bayesian Reasoning and Machine Learning*. Cambridge University Press. 226, 239, 275
- BARTLETT, M. S., LADES, H. M., & SEJNOWSKI, T. J. (1998) Independent component representations for face recognition. In *Proceedings of the SPIE Symposium on Electronic Imaging: Science and Technology: Conference on Human Vision and Electronic Imaging III*, pp. 528–539. 533
- BASRI, R., COSTA, L., GEIGER, D., & JACOBS, D. (1998) Determining the similarity of deformable shapes. *Vision Research* **38** (15–16): 2365–2385. 274
- BATLLE, J., MOUADDIB, E., & SALVI, J. (1998) Recent progress in coded structured light as a technique to solve the correspondence problem: a survey. *Pattern Recognition* **31** (7): 963–982. 385
- BAUMGART, B. G. (1974) *Geometric modeling for computer vision*. Stanford University PhD dissertation. 385
- BAY, H., ESS, A., TUYTELAARS, T., & GOOL, L. J. V. (2008) Speeded-up robust features (SURF). *Computer Vision and Image Understanding* **110** (3): 346–359. 352
- BEARDSLEY, P. A., REID, I. D., ZISSERMAN, A., & MURRAY, D. W. (1995) Active visual navigation using non-metric structure. In *IEEE International Conference on Computer Vision*, pp. 58–65. 567
- BEKIOS-CALFA, J., BUENAPOSADA, J. M., & BAUMELA, L. (2011) Revisiting linear discriminant techniques in gender recognition. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **33** (4): 858–864. 209
- BELHUMEUR, P. N., HESPANHA, J. P., & KRIEGMANN, D. J. (1997) Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **19** (7): 711–720. 533
- BELKIN, M., & NIYOGI, P. (2001) Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems*, pp. 585–591. 352
- BELONGIE, S., CARSON, C., GREENSPAN, H., & MALIK, J. (1998) Color- and texture-based image segmentation using EM and its application to content based image retrieval. In *IEEE International Conference on Computer Vision*, pp. 675–682. 140
- BELONGIE, S., MALIK, J., & PUZICHA, J. (2002) Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **24** (4): 509–522. 352
- BENGIO, Y., & DELALLEAU, O. (2009) Justifying and generalizing contrastive divergence. *Neural Computation* **21** (6): 1601–1621. 239
- BERGTHOLDT, M., CREMERS, D., & SCHÖRR, C. (2005) Variational segmentation with shape priors. In *Handbook of Mathematical Models in Computer Vision*, ed. by Y. C. N. Paragios & O. Faugeras, 131–144. 499
- BEYMER, D., & KONOLIGE, K. (1999) Real-time tracking of multiple people using continuous detection. In *IEEE Frame Rate Workshop*. 567
- BIRCHFIELD, S., & TOMASI, C. (1998) Depth discontinuities by pixel-to-pixel stereo. In *IEEE International Conference on Computer Vision*, pp. 1073–1080. 274
- BISHOP, C. M. (2006) *Pattern Recognition and Machine Learning*. Springer, 2nd edition. 19, 33, 45, 77, 97, 159, 209, 239, 275

- BLAKE, A. (2006) Visual tracking: a short research roadmap. In *Handbook of Mathematical Models in Computer Vision*, ed. by N. Paragios & Y. C. and O. Faugeras, pp. 293–307. Springer. [567](#)
- BLAKE, A., CURWEN, R. W., & ZISSERMAN, A. (1993) A framework for spatiotemporal control in the tracking of visual contours. *International Journal of Computer Vision* **11** (2): 127–145. [567](#)
- BLAKE, A., & ISARD, M. (1996) The CONDENSATION algorithm – conditional density propagation and applications to visual tracking. In *Advances in Neural Information Processing Systems*, pp. 361–367. [567](#)
- BLAKE, A., & ISARD, M. (1998) *Active Contours*. Springer. [499](#), [538](#), [565](#), [566](#), [567](#)
- BLAKE, A., ISARD, M., & REYNARD, D. (1995) Learning to track the visual motion of contours. *Artificial Intelligence* **78** (1–2): 179–212. [567](#)
- BLAKE, A., KOHLI, P., & ROTHER, C. eds. (2011) *Advances in Markov Random Fields for Vision and Image Processing*. MIT Press. [316](#)
- BLANZ, V., BASSO, C., POGGIO, T., & VETTER, T. (2003) Reanimating faces in images and video. *Comput. Graph. Forum* **22** (3): 641–650. [500](#)
- BLANZ, V., GROTH, P., PHILLIPS, P. J., & VETTER, T. (2005) Face recognition based on frontal views generated from non-frontal images. In *IEEE Computer Vision & Pattern Recognition*, pp. 454–461. [495](#), [500](#), [533](#)
- BLANZ, V., & VETTER, T. (1999) A morphable model for the synthesis of 3D faces. In *SIGGRAPH*, pp. 187–194. [494](#), [495](#), [500](#)
- BLANZ, V., & VETTER, T. (2003) Face recognition based on fitting a 3D morphable model. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **25** (9): 1063–1074. [494](#), [495](#), [500](#)
- BLEI, D. M., NG, A. Y., & JORDAN, M. I. (2003) Latent Dirichlet allocation. *Journal of Machine Learning Research* **3**: 993–1022. [595](#)
- BLEYER, M., & CHAMBON, S. (2010) Does color really help in dense stereo matching? In *International Symposium on 3D Data Processing, Visualization and Transmission*. [318](#)
- BOR WANG, S., QUATTINI, A., MORENCY, L. P., DEMIRDJIAN, D., & DARRELL, T. (2006) Hidden conditional random fields for gesture recognition. In *IEEE Computer Vision & Pattern Recognition*, pp. 1521–1527. [274](#)
- BOSCH, A., ZISSERMAN, A., & MUÑOZ, X. (2007) Image classification using random forests and ferns. In *IEEE International Conference on Computer Vision*. [209](#)
- BOUWMANS, T., BAF, F. E., & VACHON, B. (2010) Statistical background modeling for foreground detection: A survey. In *Handbook of Pattern Recognition and Computer Vision*, ed. by C. H. Chen, L. F. Pau, & P. S. P. Wang, pp. 181–199. World Scientific Publishing. [97](#)
- BOYKOV, Y., & FUNKA LEA, G. (2006) Graph cuts and efficient N-D image segmentation. *International Journal of Computer Vision* **70** (2): 109–131. [317](#)
- BOYKOV, Y., & JOLLY, M.-P. (2001) Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *IEEE International Conference on Computer Vision*, pp. 105–112. [317](#)
- BOYKOV, Y., & KOLMOGOROV, V. (2004) An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **26** (9): 1124–1137. [316](#)
- BOYKOV, Y., & VEKSLER, O. (2006) Graph cuts in vision and graphics: theories and applications. In *Handbook of Mathematical Models in Computer Vision*, ed. by Y. C. N. Paragios & O. Faugeras, pp. 79–96. Springer. [316](#)
- BOYKOV, Y., VEKSLER, O., & ZABIH, R. (1999) Fast approximate energy minimization via graph cuts. In *IEEE International Conference on Computer Vision*, pp. 377–384. [307](#), [308](#)
- BOYKOV, Y., VEKSLER, O., & ZABIH, R. (2001) Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **23** (11): 1222–1239. [316](#)
- BRAND, J., & MASON, J. (2000) A comparative assessment of three approaches to pixel-level skin-detection. In *International Conference on Pattern Recognition*, pp. 1056–1059. [97](#)
- BRAND, M. (2002) Charting a manifold. In *Advances in Neural Information Processing Systems*, pp. 961–968. [352](#)
- BREGLER, C., & MALIK, J. (1998) Tracking people with twists and exponential maps. In *IEEE Computer Vision & Pattern Recognition*, pp. 8–15. [500](#)

- BREIMAN, L. (2001) Random forests. *Machine Learning* **45**: 5–32. 209
- BRISHNAPURAM, B., FIGUEIREDO, M., CARIN, L., & HARTEMINK, A. (2005) Sparse multinomial logistic regression: fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **27** (6): 957–968. 209
- BROIDA, T. J., & CHELLAPPA, R. (1986) Estimation of object motion parameters from noisy images. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **8** (1): 90–99. 567
- BROWN, M., HUA, G., & WINDER, S. A. J. (2011) Discriminative learning of local image descriptors. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **33** (1): 43–57. 352
- BROWN, M., & LOWE, D. G. (2005) Unsupervised 3D object recognition and reconstruction in unordered datasets. In *3D Digital Imaging and Modeling*, pp. 56–63. 453
- BROWN, M., & LOWE, D. G. (2007) Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision* **74** (1): 59–73. 421
- BROWN, M. Z., BURSCHKA, D., & HAGER, G. D. (2003) Advances in computational stereo. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **25** (8): 993–1008. 318
- BRUBAKER, M. A., FLEET, D. J., & HERTZMANN, A. (2010) Physics-based person tracking using the anthropomorphic walker. *International Journal of Computer Vision* **87** (1–2): 140–155. 500
- BRUNELLI, R., & POGGIO, T. (1993) Face recognition: Features versus templates. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **15** (10): 1042–1052. 533
- BUCHANAN, A. M., & FITZGIBBON, A. W. (2005) Damped Newton algorithms for matrix factorization with missing data. In *IEEE Computer Vision & Pattern Recognition*, pp. 316–322. 453
- BURGESS, C. J. C. (2010) Dimension reduction: a guided tour. *Foundations and Trends in Machine Learning* **2** (4): 275–365. 352
- BYRÖD, M., & ÅSTRÖM, K. (2010) Conjugate gradient bundle adjustment. In *European Conference on Computer Vision*, pp. 114–127. 453
- CAI, D., HE, X., HU, Y., HAN, J., & HUANG, T. S. (2007) Learning a spatially smooth subspace for face recognition. In *IEEE Computer Vision & Pattern Recognition*. 533
- CANNY, J. (1986) A computational approach to edge detection. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **8** (6): 679–698. 352
- CARREIRA-PERPIÑÁN, M. Á., & HINTON, G. E. (2005) On contrastive divergence learning. In *Artificial Intelligence and Statistics*, volume 2005, p. 17. 239
- CASELLES, V., KIMMEL, R., & SAPIRO, G. (1997) Geodesic active contours. *International Journal of Computer Vision* **22** (1): 61–79. 499
- CHELLAPPA, R., SINHA, P., & PHILLIPS, P. J. (2010) Face recognition by computers and humans. *IEEE Computer* **43** (2): 46–55. 533
- CHEN, L.-F., LIAO, H.-Y. M., KO, M.-T., LIN, J.-C., & YU, G.-J. (2000) A new LDA-based face recognition system which can solve the small sample size problem. *Pattern Recognition* **33** (10): 1713–1726. 533
- CHEN, R., & LIU, J. S. (2000) Mixture Kalman filters. *Journal of the Royal Statistical Society B* **62** (3): 493–508. 567
- CHEN, S. E. (1995) QuickTime VR: an image-based approach to virtual environment navigation. In *SIGGRAPH*, pp. 29–38. 421
- CHEUNG, G. K. M., BAKER, S., & KANADE, T. (2004) Shape-from-silhouette across time part I: Theory and algorithms. *International Journal of Computer Vision* **62** (3): 221–247. 385
- CHIA, A. Y. S., RAHARDJA, S., RAJAN, D., & LEUNG, M. K. H. (2010) Object recognition by discriminative combinations of line segments and ellipses. In *IEEE Computer Vision & Pattern Recognition*, pp. 2225–2232. 499
- CHITTAJALLU, D. R., SHAH, S. K., & KAKADIRIS, I. A. (2010) A shape-driven MRF model for the segmentation of organs in medical images. In *IEEE Computer Vision & Pattern Recognition*, pp. 3233–3240. 317
- CHO, Y., LEE, J., & NEUMANN, U. (1998) A multi-ring color fiducial system and an intensity-invariant detection method for scalable fiducial-tracking augmented reality. In *International Workshop on Augmented Reality*, pp. 147–165. 420
- CHOI, S., KIM, T., & YU, W. (2009) Performance evaluation of RANSAC family. In *British Machine Vision Conference*. 420

- CHUM, O., PHILBIN, J., SIVIC, J., ISARD, M., & ZISSERMAN, A. (2007) Total recall: Automatic query expansion with a generative feature model for object retrieval. In *IEEE International Conference on Computer Vision*, pp. 1–8. [595](#)
- CHUM, O., WERNER, T., & MATAS, J. (2005) Two-view geometry estimation unaffected by a dominant plane. In *IEEE Computer Vision & Pattern Recognition*, pp. 772–779. [420](#)
- CLAUS, D., & FITZGIBBON, A. W. (2005) A rational function lens distortion model for general cameras. In *IEEE Computer Vision & Pattern Recognition*, pp. 213–219. [385](#)
- COHEN, L. (1991) On active contour models and balloons. *CGVIP: Image Understanding* **53** (2): 211–218. [499](#)
- COOTES, T. F., EDWARDS, G. J., & TAYLOR, C. J. (2001) Active appearance models. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **23** (6): 681–685. [499](#)
- COOTES, T. F., & TAYLOR, C. J. (1997) A mixture model for representing shape variation. In *British Machine Vision Conference*. [499](#)
- COOTES, T. F., TAYLOR, C. J., COOPER, D. H., & GRAHAM, J. (1995) Active shape models – their training and application. *Computer Vision & Image Understanding* **61** (1): 38–59. [499](#)
- CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., & STEIN, C. (2001) *Introduction to Algorithms*. MIT Press, 2nd edition. [270](#), [316](#)
- COUGHLAN, J., YUIILLE, A., ENGLISH, C., & SNOW, D. (2000) Efficient deformable template detection and localization without user interaction. *Computer Vision & Image Understanding* **78** (3): 303–319. [274](#)
- CRISTIANINI, M., & SHAWE-TAYLOR, J. (2000) *An Introduction to Support Vector Machines*. Cambridge University Press. [209](#)
- CSURKA, G., DANCE, C., FAN, L., WILLIAMOWSKI, J., & BRAY, C. (2004) Visual categorization with bags of keypoints. In *ECCV International Workshop on Statistical Learning in Computer Vision*. [209](#), [574](#), [575](#), [595](#)
- CUZZOLIN, F. (2006) Using bilinear models for view-invariant action and identity recognition. In *IEEE Computer Vision & Pattern Recognition*, pp. 1701–1708. [534](#)
- DALAL, N., & TRIGGS, B. (2005) Histograms of oriented gradients for human detection. In *IEEE Computer Vision & Pattern Recognition*, pp. 886–893. [352](#)
- DAVISON, A. J., REID, I. D., MOLTON, N., & STASSE, O. (2007) MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **29** (6): 1052–1067. [564](#), [565](#), [567](#)
- DE AGUIAR, E., STOLL, C., THEOBALT, C., AHMED, N., SEIDEL, H.-P., & THRUN, S. (2008) Performance capture from sparse multi-view video. *ACM Transactions on Graphics* **27** (3). [385](#)
- DE LA GORCE, M., PARAGIOS, N., & FLEET, D. J. (2008) Model-based hand tracking with texture, shading and self-occlusions. In *IEEE Computer Vision & Pattern Recognition*.
- DE LA TORRE, F. (2011) A least-squares framework for component analysis. *IEEE Transactions on Pattern Analysis & Machine Intelligence* . [352](#), [533](#)
- DE RIDDER, D., & FRANC, V. (2003) Robust subspace mixture models using t-distributions. In *British Machine Vision Conference*, pp. 319–328. [140](#)
- DELAITRE, V., LAPTEV, I., & SIVIC, J. (2010) Recognizing human actions in still images: a study of bag-of-features and part-based representations. In *British Machine Vision Conference*. [596](#)
- DELONG, A., & BOYKOV, Y. (2009) Globally optimal segmentation of multi-region objects. In *IEEE International Conference on Computer Vision*, pp. 285–292. [317](#)
- DEMPSTER, A. P., LAIRD, M. N., & RUBIN, D. B. (1977) Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B* **39** (1): 1–38. [140](#)
- DENG, J., BERG, A., LI, K., & FEI-FEI, L. (2010) What does classifying more than 10,000 image categories tell us? *European Conference on Computer Vision* pp. 71–84. [595](#)
- DENG, Y., & LIN, X. (2006) A fast line segment based stereo algorithm using tree dynamic programming. In *European Conference on Computer Vision*, pp. 201–212. [274](#)
- DEUTSCHER, J., BLAKE, A., & REID, I. D. (2000) Articulated body motion capture by annealed particle filtering. In *IEEE Computer Vision & Pattern Recognition*, pp. 2126–2133. [500](#)

- DEVERNAY, F., & FAUGERAS, O. D. (2001) Straight lines have to be straight. *Machine Vision Applications* **13** (1): 14–24. [385](#)
- DOLLÁR, P., TU, Z., & BELONGIE, S. (2006) Supervised learning of edges and object boundaries. In *IEEE Computer Vision & Pattern Recognition*, pp. 1964–1971. [209](#), [352](#)
- DOMKE, J., KARAPURKAR, A., & ALOIMONOS, Y. (2008) Who killed the directed model? In *IEEE Computer Vision & Pattern Recognition*. [317](#)
- DUDA, R. O., HART, P. E., & STORK, D. G. (2001) *Pattern Classification*. John Wiley and Sons, 2nd edition. [97](#)
- DURRANT-WHYTE, H., & BAILEY, T. (2006) Simultaneous localization and mapping (SLAM): part I. *Robotics & Automation Magazine, IEEE* **13** (2): 99–110. [568](#)
- DURRANT-WHYTE, H., RYE, D., & NEBOT, E. (1996) Localisation of automatic guided vehicles. In *International Symposium on Robotics Research*, pp. 613–625. [567](#)
- DURRANT-WHYTE, H. F. (1988) Uncertain geometry in robotics. *IEEE Transactions on Robot Automation* **4** (1): 23–31. [567](#)
- EDWARDS, G. J., TAYLOR, C. J., & COOTES, T. F. (1998) Interpreting face images using active appearance models. In *IEEE International Conference on Automatic Face & Gesture Recognition*, pp. 300–305. [533](#)
- EFROS, A. A., & FREEMAN, W. T. (2001) Image quilting for texture synthesis and transfer. In *SIGGRAPH*, pp. 341–346. [311](#), [312](#), [317](#)
- EFROS, A. A., & LEUNG, T. K. (1999) Texture synthesis by non-parametric sampling. In *IEEE International Conference on Computer Vision*, pp. 1033–1038. [317](#)
- EICHNER, M., & FERRARI, V. (2009) Better appearance models for pictorial structures. In *British Machine Vision Conference*. [274](#)
- ELDER, J. H. (1999) Are edges incomplete? *International Journal of Computer Vision* **34** (2–3): 97–122. [336](#), [352](#)
- ELGAMMAL, A. (2011) Figure-ground segmentation – pixel-based. In *Guide to visual analysis of humans: looking at people*, ed. by T. Moeslund, A. Hilton, Krüger, & L. Sigal. Springer. [97](#)
- ELGAMMAL, A., HARWOOD, D., & DAVIS, L. (2000) Non-parametric model for background subtraction. In *European Conference on Computer Vision*, pp. 751–767. [97](#)
- ELGAMMAL, A. M., & LEE, C.-S. (2004) Separating style and content on a nonlinear manifold. In *IEEE Computer Vision & Pattern Recognition*, pp. 478–485. [534](#)
- ENGELS, C., STEWÉNIUS, H., & NISTÉR, D. (2006) Bundle adjustment rules. *Photogrammetric Computer Vision*. [453](#)
- EVERINGHAM, M., SIVIC, J., & ZISSERMAN, A. (2006) Hello! My name is ... Buffy – automatic naming of characters in TV video. In *British Machine Vision Conference*, pp. 889–908. [274](#), [533](#)
- EVERINGHAM, M., VAN GOOL, L., WILLIAMS, C., WINN, J., & ZISSERMAN, A. (2010) The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision* **88** (2): 303–338. [595](#)
- FAUGERAS, O. (1993) *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press. [385](#)
- FAUGERAS, O., LUONG, Q., & PAPADOPOULO, T. (2001) *The Geometry of Multiple Images*. MIT PRESS. [385](#), [453](#)
- FAUGERAS, O. D. (1992) What can be seen in three dimensions with an uncalibrated stereo rig. In *European Conference on Computer Vision*, pp. 563–578. [453](#)
- FAUGERAS, O. D., & KERIVEN, R. (1998) Variational principles, surface evolution, PDEs, level set methods, and the stereo problem. *IEEE Transactions on Image Processing* **7** (3): 336–344. [454](#)
- FAUGERAS, O. D., LUONG, Q.-T., & MAYBANK, S. J. (1992) Camera self-calibration: Theory and experiments. In *European Conference on Computer Vision*, pp. 321–334. [453](#)
- FELZENZWALB, P., & ZABIH, R. (2011) Dynamic programming and graph algorithms in computer vision. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **33** (4): 721–740. [272](#), [274](#), [316](#)
- FELZENZWALB, P. F., GIRSHICK, R. B., MCALLESTER, D. A., & RAMANAN, D. (2010) Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **32** (9): 1627–1645. [274](#)
- FELZENZWALB, P. F., & HUTTENLOCHER, D. P. (2005) Pictorial structures for object recognition. *International Journal of*

- Computer Vision* **61** (1): 55–79. 270, 271, 272, 274, 500
- FELZENZWALB, P. F., & VEKSLER, O. (2010) Tiered scene labeling with dynamic programming. In *IEEE Computer Vision & Pattern Recognition*. 274, 317
- FERENCZ, A., LEARNED-MILLER, E. G., & MALIK, J. (2008) Learning to locate informative features for visual identification. *International Journal of Computer Vision* **77** (1–3): 3–24. 534
- FISCHLER, M., & BOLLES, R. (1981) Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Communications of the ACM* **24** (6): 381–395. 420
- FISCHLER, M. A., & ERSCHLAGER, R. A. (1973) The representation and matching of pictorial structures. *IEEE Transactions on Computers* **22** (1): 67–92. 274
- FITZGIBBON, A. W., & ZISSEMAN, A. (1998) Automatic camera recovery for closed or open image sequences. In *European Conference on Computer Vision*, pp. 311–326. 453
- FORD, L., & FULKERSON, D. (1962) *Flows in Networks*. Princeton University Press. 316
- FORSSÉN, P.-E., & LOWE, D. G. (2007) Shape descriptors for maximally stable extremal regions. In *IEEE International Conference on Computer Vision*, pp. 1–8. 352
- FÖRSTNER, W. (1986) A feature-based correspondence algorithm for image matching. *International Archives of Photogrammetry and Remote Sensing* **26** (3): 150–166. 352
- FORSYTH, D. A., ARIKAN, O., IKEMOTO, L., O'BRIEN, J., & RAMANAN, D. (2006) Computational studies of human motion: Part 1, Tracking and motion synthesis. *Foundations and Trends in Computer Graphics and Computer Vision* **1** (2/2): 77–254. 500
- FRAHM, J.-M., GEORGEL, P. F., GALLUP, D., JOHNSON, T., RAGURAM, R., WU, C., JEN, Y.-H., DUNN, E., CLIPP, B., & LAZEBNIK, S. (2010) Building Rome on a cloudless day. In *European Conference on Computer Vision*, pp. 368–381. 450
- FRAHM, J.-M., & POLLEFEYS, M. (2006) RANSAC for (quasi-)degenerate data (QDEGSAC). In *IEEE Computer Vision & Pattern Recognition*, pp. 453–460. 420
- FRANCO, J.-S., & BOYER, E. (2005) Fusion of multi-view silhouette cues using a space occupancy grid. In *IEEE International Conference on Computer Vision*, pp. 1747–1753. 385
- FREEMAN, W. T., PASZTOR, E. C., & CARMICHAEL, O. T. (2000) Learning low-level vision. *International Journal of Computer Vision* **40**: 25–47. 275, 311, 316
- FREIFELD, O., WEISS, A., ZUFFI, S., & BLACK, M. J. (2010) Contour people: A parameterized model of 2D articulated human shape. In *IEEE Computer Vision & Pattern Recognition*, pp. 639–646. 499
- FREIMAN, M., KRONMAN, A., ESSES, S. J., JOSKOWICZ, L., & SOSNA, J. (2010) Non-parametric iterative model constraint graph min-cut for automatic kidney segmentation. In *Medical Image Computing and Computer-Assisted Intervention*, pp. 73–80. 317
- FREUND, Y., & SCHAPIRA, R. (1996) Experiments with a new boosting algorithm. In *International conference on machine learning*, pp. 148–156. 209
- FREUND, Y., & SCHAPIRA, R. E. (1995) A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory: Eurocolt '95*, pp. 23–37. 209
- FREY, B., KSCHISCHANG, F., LOELIGER, H., & WIBERG, N. (1997) Factor graphs and algorithms. In *Allerton Conference on Communication, Control and Computing*. 275
- FREY, B. J., & JOJIC, N. (1999a) Estimating mixture models of images and inferring spatial transformations using the EM algorithm. In *IEEE Computer Vision & Pattern Recognition*, pp. 416–422. 140
- FREY, B. J., & JOJIC, N. (1999b) Transformed component analysis: joint estimation of spatial transformations and image components. In *IEEE Computer Vision & Pattern Recognition*, pp. 1190–1196,. 140
- FRIEDMAN, J., HASTIE, T., & TIBSHIRANI, R. (2000) Additive logistic regression: a statistical view of boosting. *Annals of Statistics* **28** (2): 337–407. 209
- FRIEDMAN, J. H. (1999) Greedy function approximation: A gradient boosting machine. Technical report, Department of Statistics, Stanford University. 209
- FRIEDMAN, N., & RUSSELL, S. J. (1997) Image segmentation in video sequences: a probabilistic approach. In *Uncertainty in Artificial Intelligence*, pp. 175–181. 97

- FUA, P., & LECLERC, Y. G. (1995) Object-centered surface reconstruction: Combining multi-image stereo and shading. *International Journal of Computer Vision* **16** (1): 35–56. [454](#)
- FUSIELLO, A., TRUCCO, E., & VERRI, A. (2000) A compact algorithm for rectification of stereo pairs. *Mach. Vis. Appl.* **12** (1): 16–22. [453](#)
- GAO, X.-S., HOU, X., TANG, J., & CHENG, H.-F. (2003) Complete solution classification for the perspective-three-point problem. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **25** (8): 930–943. [385](#)
- GAVRILA, D., & DAVIS, L. S. (1996) 3-D model-based tracking of humans in action: a multi-view approach. In *IEEE Computer Vision & Pattern Recognition*, pp. 73–80. [500](#)
- GEIGER, B., LADENDORF, B., & YUILLE, A. (1992) Occlusions and binocular stereo. In *European Conference on Computer Vision*, pp. 425–433. [274](#)
- GEIGER, D., GUPTA, A., COSTA, L. A., & VLONTZOS, J. (1995) Dynamic-programming for detecting, tracking and matching deformable contours. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **17** (3): 294–302. [275](#)
- GELB, A. (1974) *Applied Optimal Estimation*. MIT Press. [567](#)
- GELMAN, A., CARLIN, J. B., STERN, H. S., & RUBIN, D. B. (2004) *Bayesian Data Analysis*. Chapman and Hall / CRC. [45](#), [65](#)
- GEMAN, S., & GEMAN, D. (1984) Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **6** (6): 721–741. [316](#)
- GEYER, C., & DANILIDIS, K. (2001) Catadioptric projective geometry. *International Journal of Computer Vision* **45** (3): 223–243. [385](#)
- GHAHRAMANI, Z. (2001) An introduction to hidden Markov models and Bayesian networks. In *Hidden Markov models: applications in computer vision*, ed. by B. H. Juang, pp. 9–42. World Scientific Publishing. [274](#)
- GHAHRAMANI, Z., & HINTON, G. (1996a) Parameter estimation for linear dynamical systems. Technical Report CRG-TR-96-2, Department of Computer Science, University of Toronto. [567](#)
- GHAHRAMANI, Z., & HINTON, G. (1996b) Switching state-space models. Technical Report CRG-TR-96-3, Department of Computer Science, University of Toronto. [567](#)
- GHAHRAMANI, Z., & HINTON, G. E. (1996c) The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, University of Toronto. [140](#)
- GOLDBERG, A., & TARJAN, R. (1988) A new approach to the maximum flow problem. *Journal of the Association for Computing Machinery* **35** (4): 921–940. [316](#)
- GOLOMB, B. A., LAWRENCE, D. T., & SEJNOWSKI, T. (1990) SEXNET: a neural network identifies sex from human faces. In *Advances in Neural Information Processing Systems*, pp. 572–579. [209](#)
- GONG, M., & YANG, Y. H. (2005) Near real-time reliable stereo matching using programmable graphics hardware. In *IEEE Computer Vision & Pattern Recognition*, pp. 924–931. [274](#)
- GONZALEZ, R. C., & WOODS, R. E. (2002) *Digital Image Processing*. Prentice Hall, 2nd edition. [352](#)
- GOWER, J. C., & DIJKSTERHUIS, G. B. (2004) *Procrustes Problems*. Oxford University Press. [420](#)
- GRADY, L. (2006) Random walks for image segmentation. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **28** (11): 1768–1783. [317](#)
- GRAUMAN, K., & DARRELL, T. (2005) The pyramid match kernel: Discriminative classification with sets of image features. In *IEEE International Conference on Computer Vision*, pp. 1458–1465. [595](#)
- GRAUMAN, K., SHAKHNAROVICH, G., & DARRELL, T. (2003) A Bayesian approach to image-based visual hull reconstruction. In *IEEE Computer Vision & Pattern Recognition*, pp. 187–194. [385](#)
- GREIG, D. M., PORTEOUS, B. T., & SEHEULT, A. H. (1989) Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B* **51** (2): 271–279. [316](#)
- GRIMES, D. B., SHON, A. P., & RAO, R. P. N. (2003) Probabilistic bilinear models for appearance-based vision. In *IEEE International Conference on Computer Vision*, pp. 1478–1485. [534](#)
- GROSS, R., MATTHEWS, I., & BAKER, S. (2002) Eigen light-fields and face recognition across pose. In *FGR*, pp. 3–9. [533](#)

- GUESEBROEK, J. M., BUGHOUTS, G. J., & SMEULDERS, A. W. M. (2005) A.W.M.: The Amsterdam library of object images. *International Journal of Computer Vision* **61** (1): 103–112. 134
- GUILLAUMIN, M., VERBEEK, J. J., & SCHMID, C. (2009) Is that you? metric learning approaches for face identification. In *IEEE International Conference on Computer Vision*, pp. 498–505. 534
- GUIVANT, J. E., & NEBOT, E. M. (2001) Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE Transactions on Robotics* **17** (3): 242–257. 567
- HANDA, A., CHLI, M., STRASDAT, H., & DAVISON, A. J. (2010) Scalable active matching. In *IEEE Computer Vision & Pattern Recognition*, pp. 1546–1553. 567
- HARRIS, C. (1992) Tracking with rigid objects. In *Active Vision*, ed. by A. Blake & A. L. Yuille, pp. 59–73. 420
- HARRIS, C., & STEPHENS, M. J. (1988) A combined corner and edge detector. In *Alvey Vision Conference*, pp. 147–152. 352, 453
- HARRIS, C. G., & PIKE, J. M. (1987) 3D positional integration from image sequences. In *Alvey Vision Conference*, pp. 233–236. 567
- HARTLEY, R. I. (1992) Estimation of relative camera positions for uncalibrated cameras. In *European Conference on Computer Vision*, pp. 579–587. 453
- HARTLEY, R. I. (1994) Projective reconstruction from line correspondence. In *IEEE Computer Vision & Pattern Recognition*, pp. 579–587. 453
- HARTLEY, R. I. (1997) In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **19** (6): 580–593. 434, 453
- HARTLEY, R. I., & GUPTA, R. (1994) Linear pushbroom cameras. In *European Conference on Computer Vision*, pp. 555–566. 385
- HARTLEY, R. I., & ZISSERMAN, A. (2004) *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, 2nd edition. 19, 369, 385, 406, 420, 431, 440, 453
- HE, X., YAN, S., HU, Y., NIYOGI, P., & ZHANG, H. (2005) Face recognition using Laplacianfaces. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **27** (3): 328–340. 533
- HE, X., ZEMEL, R. S., & CARREIRA-PERPIÑÁN, M. Á. (2004) Multiscale conditional random fields for image labeling. In *IEEE Computer Vision & Pattern Recognition*, pp. 695–702. 209, 210
- HE, X., ZEMEL, R. S., & RAY, D. (2006) Learning and incorporating top-down cues in image segmentation. In *European Conference on Computer Vision*, pp. 338–351. 210
- HEAP, T., & HOGG, D. (1996) Towards 3D hand tracking using a deformable model. In *IEEE International Conference on Automatic Face & Gesture Recognition*, pp. 140–145. 500
- HEEGER, D., & BERGEN, J. (1995) Pyramid-based texture analysis/synthesis. In *Computer graphics and interactive techniques*, pp. 229–238. ACM. 317
- HEESS, N., WILLIAMS, C. K. I., & HINTON, G. E. (2009) Learning generative texture models with extended fields of experts. In *British Machine Vision Conference*. 317
- HERNÁNDEZ, C., & SCHMITT, F. (2004) Silhouette and stereo fusion for 3D object modeling. *Computer Vision & Image Understanding* **96** (3): 367–392. 454
- HINTON, G. E. (2002) Training products of experts by minimizing contrastive divergence. *Neural Computation* **14** (8): 1771–1800. 239
- HIRSCHMÜLLER, H. (2005) Accurate and efficient stereo processing by semi-global matching and mutual information. In *IEEE Computer Vision & Pattern Recognition*, pp. 807–814. 318
- HIRSCHMÜLLER, H., & SCHARSTEIN, D. (2009) Evaluation of stereo matching costs on images with radiometric differences. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **31** (9): 1582–1599. 318
- HOFMANN, T. (1999) Probabilistic latent semantic analysis. In *Uncertainty in artificial intelligence*, pp. 289–296. 595
- HOGG, D. (1983) Model-based vision: a program to see a walking person. *Image and Vision Computing* **1** (1): 5–20. 500
- HOIEM, D., EFROS, A., & HEBERT, M. (2005) Automatic photo pop-up. *ACM Transactions on Graphics (SIGGRAPH)* **24** (3): 577–584. 206
- HOIEM, D., EFROS, A. A., & HEBERT, M. (2007) Recovering surface layout from an image. *International Journal of Computer Vision* **75** (1): 151–172. 205, 206, 209

- HORN, B. K. P. (1990) Relative orientation. *International Journal of Computer Vision* **4** (1): 59–78. [453](#)
- HORN, E., & KIRYATI, N. (1999) Toward optimal structured light patterns. *Image and vision computing* **17** (2): 87–97. [385](#)
- HORPRASET, T., HARWOOD, D., & DAVIS, L. S. (2000) A robust background subtraction and shadow detection. In *Asian Conference on Computer Vision*, pp. 983–988. [97](#)
- HSU, R. L., ABDEL-MOTTALEB, M., & JAIN, A. K. (2002) Face detection in color images. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **24** (5): 696–707. [97](#)
- HUANG, C., AI, H., LI, Y., & LAO, S. (2007a) High-performance rotation invariant multi-view face detection. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **29** (4): 671–686. [210](#)
- HUANG, G. B., RAMESH, M., BERG, T., & LEARNED-MILLER, E. (2007b) Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report Technical Report 07-49, University of Massachusetts, Amherst. [529, 534](#)
- HUANG, T. S., & FAUGERAS, O. D. (1989) Some properties of the E matrix in two-view motion estimation. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **11** (12): 1310–1312. [453](#)
- HUANG, Y., LIU, Q., & METAXAS, D. N. (2011) A component-based framework for generalized face alignment. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* **41** (1): 287–298. [491, 499](#)
- HUBER, P. J. (2009) *Robust Statistics*. John Wiley and Sons, 2nd edition. [420](#)
- HUMAYUN, A., MAC AODHA, O., & BROSTOW, G. J. (2011) Learning to find occlusion regions. In *CVPR*. [209](#)
- IOFFE, S. (2006) Probabilistic linear discriminant analysis. In *European Conference on Computer Vision*, pp. 531–542. [533](#)
- ISACK, H., & BOYKOV, Y. (2012) Energy-based geometric multi-model fitting. *International Journal of Computer Vision* **97** (2): 123–147. [316, 416, 420](#)
- ISARD, M., & BLAKE, A. (1998) A mixed-state CONDENSATION tracker with automatic model-switching. In *IEEE International Conference on Computer Vision*, pp. 107–112. [567](#)
- ISHIKAWA, H. (2003) Exact optimization for Markov random fields with convex priors. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **25** (10): 1333–1336. [316](#)
- ISHIKAWA, H. (2009) Higher order clique reduction in binary graph cut. In *IEEE Computer Vision & Pattern Recognition*. [317](#)
- JAZWINSKI, A. H. (1970) *Stochastic Processes and Filtering Theory*. Academic Press. [567](#)
- JÉGOU, H., DOUZE, M., & SCHMID, C. (2008) Recent advances in large scale image search. In *Emerging Trends in Visual Computing*, pp. 305–326. [595](#)
- JÉGOU, H., DOUZE, M., & SCHMID, C. (2009) Packing bag-of-features. In *IEEE International Conference on Computer Vision*, pp. 2357–2364. [595](#)
- JEONG, Y., NISTÉR, D., STEEDLY, D., SZELISKI, R., & KWON, I.-S. (2010) Pushing the envelope of modern methods for bundle adjustment. In *IEEE Computer Vision & Pattern Recognition*, pp. 1474–1481. [453](#)
- JIANG, H., & MARTIN, D. R. (2008) Global pose estimation using non-tree models. In *IEEE Computer Vision & Pattern Recognition*. [274](#)
- JOJIC, N., & FREY, B. J. (2001) Learning flexible sprites in video layers. In *IEEE Computer Vision & Pattern Recognition*, pp. 199–206. [140](#)
- JOJIC, N., FREY, B. J., & KANNAN, A. (2003) Epitomic analysis of appearance and shape. In *IEEE International Conference on Computer Vision*, pp. 34–41. [141](#)
- JONES, E., & SOATTO, S. (2005) Layered active appearance models. In *IEEE International Conference on Computer Vision*, pp. 1097–1102. [499](#)
- JONES, M. J., & REHG, J. M. (2002) Statistical color models with application to skin detection. *International Journal of Computer Vision* **46** (1): 81–96. [97, 140](#)
- JORDAN, M. I. (2004) Graphical models. *Statistical science* **19** (1): 140–155. [239](#)
- JORDAN, M. I., & JACOBS, R. A. (1994) Hierarchical mixtures of experts and the EM algorithm. *Neural Computation* **6** (2): 181–214. [211](#)
- JUAN, O., & BOYKOV, Y. (2006) Active graph cuts. In *IEEE Computer Vision & Pattern Recognition*, pp. 1023–1029. [316](#)

- JULIER, S., & UHLMANN, J. (1997) A new extension of the Kalman filter to nonlinear systems. In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, volume 3, p. 26. [567](#)
- KADIR, T., & BRADY, M. (2001) Saliency, scale and image description. *International Journal of Computer Vision* **45** (2): 83–105. [352](#)
- KAKUMANU, P., MAKROGIANNIS, S., & BOURBAKIS, N. G. (2007) A survey of skin-colour modeling and detection methods. *Pattern Recognition* **40** (3): 1106–1122. [97](#)
- KALMAN, R. E. (1960) A new approach to linear filtering and prediction problems. *Journal of Basic Engineering* **82** (1): 35–45. [567](#)
- KALMAN, R. E., & BUCY, R. S. (1961) New results in linear filtering and prediction theory. *Transactions of the American Society for Mechanical Engineering D* **83** (1): 95–108. [567](#)
- KANADE, T., RANDER, P., & NARAYANAN, P. J. (1997) Virtualized reality: Constructing virtual worlds from real scenes. *IEEE MultiMedia* **4** (1): 34–47. [385](#)
- KASS, M., WITKIN, A., & TERZOPOLOUS, D. (1987) Snakes: Active contour models. *International Journal of Computer Vision* **1** (4): 321–331. [275, 499](#)
- KATO, H., & BILLINGHURST, M. (1999) Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In *International Workshop on Augmented Reality*, pp. 85–94. [420](#)
- KATO, H., BILLINGHURST, M., POUPYREV, I., IMAMOTO, K., & TACHIBANA, K. (2000) Virtual object manipulation on a table-top AR environment. In *International Symposium on Augmented Reality*, pp. 111–119. [420](#)
- KENDALL, D. G. (1984) Shape manifolds, Procrustean metrics, and complex projective spaces. *Bulletin of the London Mathematical Society* **16** (2): 81–121. [462](#)
- KHAN, Z., & DELLAERT, F. (2004) Robust generative subspace modelling: the subspace t-distribution. Technical Report GIT-GVU-04-11, Georgia Institute of Technology. [140](#)
- KIM, J. C., LEE, K. M., CHOI, B., & LEE, S. U. (2005) A dense stereo matching using two pass dynamic programming with generalized control points. In *IEEE Computer Vision & Pattern Recognition*, pp. 1075–1082. [274](#)
- KLEIN, G., & MURRAY, D. (2007) Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*. [567](#)
- KOHLI, P., KUMAR, M. P., & TORR, P. H. S. (2009a) P3 & beyond: Move making algorithms for solving higher order functions. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **31** (9): 1645–1656. [317](#)
- KOHLI, P., LADICKY, L., & TORR, P. H. S. (2009b) Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision* **82** (3): 302–324. [317](#)
- KOHLI, P., & TORR, P. H. S. (2005) Efficiently solving dynamic Markov random fields using graph cuts. In *IEEE International Conference on Computer Vision*, pp. 922–929. [316](#)
- KOLEV, K., & CREMERS, D. (2008) Integration of multiview stereo and silhouettes via convex functionals on convex domains. In *European Conference on Computer Vision*, pp. 752–765. [454](#)
- KOLLER, D., & FRIEDMAN, N. (2009) *Probabilistic graphical models*. MIT Press. [220, 226, 239, 275](#)
- KOLLER, D., KLINKER, G., ROSE, E., BREEN, D., WHITAKER, R., & TUCERYAN, M. (1997) Real-time vision-based camera tracking for augmented reality applications. In *ACM Symposium on Virtual Reality Software and Technology*, pp. 87–94. [420](#)
- KOLMOGOROV, V. (2006) Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **28** (10): 1568–1583. [317](#)
- KOLMOGOROV, V., CRIMINISI, A., BLAKE, A., CROSS, G., & ROTHER, C. (2006) Probabilistic fusion of stereo with color and contrast for bi-layer segmentation. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **28** (9): 1480–1492. [316](#)
- KOLMOGOROV, V., & ROTHER, C. (2007) Minimizing non-submodular graph functions with graph-cuts – a review. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **29** (7): 1274–1279. [316, 317](#)
- KOLMOGOROV, V., & ZABIH, R. (2001) Computing visual correspondence with occlusions via graph cuts. In *IEEE International Conference on Computer Vision*, pp. 508–515. [308, 316, 317](#)

- KOLMOGOROV, V., & ZABIH, R. (2002) Multi-camera scene reconstruction via graph cuts. In *European Conference on Computer Vision*, pp. 82–96. [316](#)
- KOLMOGOROV, V., & ZABIH, R. (2004) What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis & Machine Intelligence* **26** (2): 147–159. [316](#)
- KOMODAKIS, N., TZIRITAS, G., & 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 & Image Understanding* **112** (1): 14–29. [316](#)
- KOTZ, S., & NADARAJAH, S. (2004) *Multivariate distributions and their applications*. Cambridge University Press. [140](#)
- KSCHISCHANG, F. R., FREY, B., & LOELIGER, H. A. (2001) Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory* **47** (2): 498–519. [275](#)
- KUMAR, M. P., TORR, P., & ZISSERMAN, A. (2004) Extending pictorial structures for object recognition. In *British Machine Vision Conference*. [274](#)
- KUMAR, M. P., TORR, P. H. S., & ZISSERMAN, A. (2005) OBJ CUT. In *IEEE Computer Vision & Pattern Recognition*, pp. 18–25. [317](#)
- KUMAR, M. P., VEKSLER, O., & TORR, P. H. S. (2011) Improved moves for truncated convex models. *Journal of Machine Learning Research* **12**: 31–67. [316](#)
- KUMAR, N., BELHUMEUR, P., & NAYAR, S. K. (2008) Face tracer: A search engine for large collections of images with faces. In *European Conference on Computer Vision*. [209, 210](#)
- KUMAR, N., BERG, A. C., BELHUMEUR, P. N., & NAYAR, S. K. (2009) Attribute and simile classifiers for face verification. In *IEEE International Conference on Computer Vision*, pp. 365–372. [534](#)
- KUMAR, S., & HEBERT, M. (2003) Discriminative random fields: A discriminative framework for contextual interaction in classification. In *IEEE International Conference on Computer Vision*, pp. 1150–1159. [316](#)
- KUTULAKOS, K. N., & SEITZ, S. M. (2000) A theory of shape by space carving. *International Journal of Computer Vision* **38** (3): 199–218. [454](#)
- KWATRA, V., SCHÖDL, A., ESSA, I., TURK, G., & BOICK, A. (2003) Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics (SIGGRAPH 2003)* **22** (3): 277–286. [316, 317](#)
- LANITIS, A., TAYLOR, C. J., & COOTES, T. F. (1997) Automatic interpretation and coding of face images using flexible models. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **19** (7): 743–756. [499](#)
- LAPTEV, I., MARSZAŁEK, M., SCHMID, C., & ROZENFELD, B. (2008) Learning realistic human actions from movies. In *CVPR*. [592, 593, 595](#)
- LAURENTINI, A. (1994) The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **16** (2): 150–162. [385](#)
- LAWRENCE, N. D. (2004) Probabilistic non-linear principal component analysis with Gaussian process latent variable models. Technical Report CS-04-08, University of Sheffield. [140, 352](#)
- LAWRENCE, N. D. (2005) Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of Machine Learning Research* **6**: 1783–1816. [499](#)
- LAZEBNIK, S., SCHMID, C., & PONCE, J. (2006) Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *IEEE Computer Vision & Pattern Recognition*, pp. 2169–2178. [595](#)
- LEE, J., MOGHADDAM, B., PFISTER, H., & MACHIRAJU, R. (2005) A bilinear illumination model for robust face recognition. In *IEEE International Conference on Computer Vision*, pp. 1177–1184. [534](#)
- LEMPITSKY, V., BLAKE, A., & ROTHER, C. (2008) Image segmentation by branch-and-mincut. In *European Conference on Computer Vision*, pp. 15–29. [317](#)
- LEMPITSKY, V., ROTHER, C., ROTH, S., & BLAKE, A. (2010) Fusion moves for Markov random field optimization. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **32** (8): 1392–1405. [316](#)
- LEONARD, J. J., & FEDER, H. J. S. (2000) A computational efficient method for large-scale concurrent mapping and localisation. In *International Symposium on Robotics Research*, pp. 169–176. [567](#)

- LEORDEANU, M., HEBERT, M., & SUKTHANKAR, R. (2007) Beyond local appearance: Category recognition from pairwise interactions of simple features. In *IEEE Computer Vision & Pattern Recognition*. 499
- LEOTTA, M. J., & MUNDY, J. L. (2011) Vehicle surveillance with a generic, adaptive, 3D vehicle model. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **33** (7): 1457–1469. 500
- LEPETIT, V., & FUA, P. (2005) Monocular model-based 3D tracking of rigid objects: A survey. *Foundations and Trends in Computer Graphics and Vision* **1** (1): 1–89. 420
- LEPETIT, V., & FUA, P. (2006) Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **28** (9): 1465–1479. 420
- LEPETIT, V., LAGGER, P., & FUA, P. (2005) Randomized trees for real-time keypoint recognition. In *IEEE Computer Vision & Pattern Recognition*, pp. 775–781. 209, 417
- LEPETIT, V., MORENO-NOGUER, F., & FUA, P. (2009) EPnP: an accurate  $O(n)$  solution to the PnP problem. *International Journal of Computer Vision* **81** (2): 155–166. 385
- LEVENTON, M. E., GRIMSON, W. E. L., & FAUGERAS, O. D. (2000) Statistical shape influence in geodesic active contours. In *IEEE Computer Vision & Pattern Recognition*, pp. 1316–1323. 499
- LEVIN, A., LISCHINSKI, D., & WEISS, Y. (2004) Colorization using optimization. *ACM Transactions on Graphics* **23** (3): 689–694. 316
- LHUIILLIER, M., & QUAN, L. (2002) Match propagation for image-based modeling and rendering. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **24** (8): 1140–1146. 318
- LI, F.-F., & PERONA, P. (2005) A Bayesian hierarchical model for learning natural scene categories. In *IEEE Computer Vision & Pattern Recognition*, pp. 524–531. 595, 596
- LI, J., & ALLINSON, N. M. (2008) A comprehensive review of current local features for computer vision. *Neurocomputing* **71** (10–12): 1771–1787. 352
- LI, P., FU, Y., MOHAMMED, U., ELDER, J., & PRINCE, S. J. D. (2011) Probabilistic models for inference about identity. *IEEE Transactions on Pattern Analysis & Machine Intelligence* . 516, 528, 529, 533, 534
- LI, P., WARRELL, J., AGHAJANIAN, J., & PRINCE, S. (2010) Context-based additive logistic model for facial keypoint localization. In *British Machine Vision Conference*, pp. 1–11. 533
- LI, S. Z. (2010) *Markov Random Field Modeling in Image Analysis*. Springer, 3rd edition. 316
- LI, S. Z., & JAIN, A. K. eds. (2005) *Handbook of face recognition*. Springer. 533
- LI, S. Z., & ZHANG, Z. (2004) Floatboost learning and statistical face detection. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **26** (9): 1112–1123. 210
- LI, S. Z., ZHANG, Z. Q., SHUM, H. Y., & ZHANG, H. J. (2003) Floatboost learning for classification. In *Advances in Neural Information Processing Systems*, pp. 993–1000. 209
- LI, S. Z., ZHUANG, Z., BLAKE, A., ZHANG, H., & SHUM, H. (2002) Statistical learning of multi-view face detection. In *European Conference on Computer Vision*, pp. 67–82. 210
- LI, Y., DU, Y., & LIN, X. (2005) Kernel-based multifactor analysis for image synthesis and recognition. In *IEEE International Conference on Computer Vision*, pp. 114–119. 534
- LI, Y., SUN, J., TANG, C.-K., & SHUM, H.-Y. (2004) Lazy snapping. *ACM Transactions on Graphics* **23** (3): 303–308. 317
- LIENHART, R., KURANOV, A., & PISAREVSKY, V. (2003) Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *Deutsche Arbeitsgemeinschaft für Mustererkennung*, pp. 297–304. 210
- LIU, C., & RUBIN, D. B. (1995) ML estimation of the t distribution using EM and its extensions ECM and ECME. *Statistica Sinica* **5** (1): 19–39. 140
- LIU, C., & SHUM, H. Y. (2003) Kullback-Leibler boosting. In *IEEE Computer Vision & Pattern Recognition*, pp. 407–411. 209
- LIU, C., & WECHSLER, H. (1998) Probabilistic reasoning models for face recognition. In *IEEE Computer Vision & Pattern Recognition*, pp. 827–832. 533
- LIU, J., SUN, J., & SHUM, H.-Y. (2009) Paint selection. *ACM Transactions on Graphics* **28** (3). 317

- LONGUET-HIGGINS, H. C. (1981) A computer algorithm for reconstructing a scene from two projections. *Nature* **293**: 133–135. [453](#)
- LOOP, C. T., & ZHANG, Z. (1999) Computing rectifying homographies for stereo vision. In *IEEE Computer Vision & Pattern Recognition*, pp. 1125–1131. [453](#)
- LOURAKIS, M. I. A., & ARGYROS, A. A. (2009) SBA: A software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software* **36** (1). [453](#)
- LOWE, D. G. (2004) Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* **60** (2): 91–110. [352](#), [453](#), [534](#)
- LOXAM, J., & DRUMMOND, T. (2008) Student mixture filter for robust, real-time visual tracking. In *European Conference on Computer Vision*, pp. 372–385. [140](#)
- LU, S., METAXAS, D. N., SAMARAS, D., & OLIENSIUS, J. (2003) Using multiple cues for hand tracking and model refinement. In *IEEE Computer Vision & Pattern Recognition*, pp. 443–450. [500](#)
- LUCAS, B. D., & KANADE, T. (1981) An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, pp. 647–679. [453](#)
- LUCEY, S., & CHEN, T. (2006) Learning patch dependencies for improved pose mismatched face verification. In *IEEE Computer Vision & Pattern Recognition*, pp. 909–915. [533](#)
- MA, Y., DERKSEN, H., HONG, W., & WRIGHT, J. (2007) Segmentation of multivariate mixed data via lossy data coding and compression. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **29** (9): 1546–1562. [140](#)
- MA, Y., SOATTO, S., & KOSECKÁ, J. (2004) *An Invitation to 3-D Vision*. Springer. [385](#), [453](#)
- MAC AODHA, O., BROSTOW, G. J., & POLLEFEYS, M. (2010) Segmenting video into classes of algorithm-suitability. In *IEEE Computer Vision & Pattern Recognition*. [209](#)
- MACKAY, D. J. (2003) *Information theory, learning and inference algorithms*. Cambridge University Press. [65](#)
- MAKADIA, A. (2010) Feature tracking for wide-baseline image retrieval. In *European Conference on Computer Vision*, pp. 310–323. [595](#)
- MÄKINEN, E., & RAISAMO, R. (2008a) Evaluation of gender classification methods with automatically detected and aligned faces. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **30** (3): 541–547. [210](#)
- MÄKINEN, E., & RAISAMO, R. (2008b) An experimental comparison of gender classification methods. *Pattern recognition methods* **29** (10): 1544–1556. [210](#)
- MALCOLM, J. G., RATHI, Y., & TANNENBAUM, A. (2007) Graph cut segmentation with nonlinear shape priors. In *IEEE International Conference on Image Processing*, pp. 365–368. [317](#)
- MALLADI, R., SETHIAN, J. A., & VEMURI, B. C. (1994) Evolutionary fronts for topology-independent shape modeling and recovery. In *European Conference on Computer Vision*, pp. 3–13. [499](#)
- MATAS, J., CHUM, O., URBAN, M., & PAJDLA, T. (2002) Robust wide baseline stereo from maximally stable extremal regions. In *British Machine Vision Conference*. [352](#)
- MATTHEWS, I., & BAKER, S. (2004) Active appearance models revisited. *International Journal of Computer Vision* **60** (2): 135–164. [499](#)
- MATTHEWS, I., COOTES, T. F., BANGHAM, J. A., COX, S., & HARVEY, R. (2002) Extraction of visual features for lipreading. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **24** (2): 198–213. [499](#)
- MATTHEWS, I., XIAO, J., & 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. [496](#), [499](#), [500](#)
- MATTHIES, L., KANADE, T., & SZELISKI, R. (1989) Kalman filter-based algorithms for estimating depth from image sequences. *International Journal of Computer Vision* **3** (3): 209–238. [567](#)
- MATUSIK, W., BUEHLER, C., RASKAR, R., GORTLER, S. J., & McMILLAN, L. (2000) Image-based visual hulls. In *SIGGRAPH*, pp. 369–374. [385](#)
- MAYBANK, S. J. (1998) *Theory of reconstruction from image motion*. Springer-Verlag. [453](#)

- MAYBECK, P. S. (1990) The Kalman filter: an introduction to concepts. In *Autonomous Robot Vehicles*, ed. by I. J. Cox & G. T. Wilfong, pp. 194–204. Springer-Verlag. [567](#)
- McLACHLAN, G. J., & KRISHNAN, T. (2008) *The EM algorithm and extensions*. Wiley, 2nd edition. [140](#)
- MEI, C., SIBLEY, G., CUMMINS, M., NEWMAN, P., & REID, I. (2009) A constant time efficient stereo SLAM system. In *British Machine Vision Conference*. [567](#)
- MEIR, R., & MÄTSCH, G. (2003) An introduction to boosting and leveraging. In *Advanced Lectures on Machine Learning*, ed. by S. Mendelson & A. Smola, pp. 119–184. [209](#)
- MESMER, K., MATAS, J., KITTLER, J., LUETTIN, J., & MAITRE, G. (1999) XM2VTS: The extended M2VTS database. In *Conference on Audio and Video-base Biometric Personal Verification*, pp. 72–77. [534](#)
- MIKOŁAJCZYK, K., & SCHMID, C. (2002) An affine invariant interest point detector. In *European Conference on Computer Vision*, pp. 128–142. [352](#)
- MIKOŁAJCZYK, K., & SCHMID, C. (2004) Scale & affine invariant interest point detectors. *International Journal of Computer Vision* **60** (1): 63–86. [352](#)
- MIKOŁAJCZYK, K., & SCHMID, C. (2005) A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **27** (10): 1615–1630. [352](#)
- MIKOŁAJCZYK, K., TUYTELAARS, T., SCHMID, C., ZISSERMAN, A., MATAS, J., SCHAFFALITZKY, F., KADIR, T., & GOOL, L. J. V. (2005) A comparison of affine region detectors. *International Journal of Computer Vision* **65** (1-2): 43–72. [352](#)
- MIKULÍK, A., PERDOCH, M., CHUM, O., & MATAS, J. (2010) Learning a fine vocabulary. In *European Conference on Computer Vision*, pp. 1–14. [595](#)
- MINKA, T. (2002) Bayesian inference in dynamic models: an overview. Technical report, Carnegie Mellon University. [567](#)
- MOESLUND, T. B., HILTON, A., & KRÜGER, V. (2006) A survey of advances in vision-based human motion capture and analysis. *Computer Vision & Image Understanding* **104** (2–3): 90–126. [500](#)
- MOGHADDAM, B., JEbara, T., & PENTLAND, A. (2000) Bayesian face recognition. *Pattern Recognition* **33** (11): 1771–1782. [533](#), [535](#)
- MOGHADDAM, B., & PENTLAND, A. (1997) Probabilistic visual learning for object representation. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **19** (7): 696–710. [140](#), [533](#)
- MOGHADDAM, B., & YANG, M. H. (2002) Learning gender with support faces. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **24** (5): 707–711. [209](#)
- MOHAMMED, U., PRINCE, S. J. D., & KAUTZ, J. (2009) Visio-lization: generating novel facial images. *ACM Transactions on Graphics (SIGGRAPH)* **28** (3). [314](#), [315](#)
- MONI, M. A., & ALI, A. B. M. S. (2009) HMM based hand gesture recognition: a review on techniques and approaches. In *Int. Conf. on Computer Science and Information Technology*, pp. 433–437. [274](#)
- MONTEMERLO, M., THRUN, S., KOLLER, D., & WEGBREIT, B. (2002) FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *AAAI/IAAI*, pp. 593–598. [567](#)
- MONTEMERLO, M., THRUN, S., KOLLER, D., & WEGBREIT, B. (2003) FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *International Joint Conference on Uncertainty in Artificial Intelligence*, pp. 1151–1156. [567](#)
- MOONS, T. (1998) A guided tour through multi-view relations. In *SMILE*, ed. by R. Koch & L. J. V. Gool, volume 1506 of *Lecture Notes in Computer Science*, pp. 304–346. Springer. [453](#)
- MOONS, T., VAN GOOL, L. J., & VERGAUWEN, M. (2009) 3D reconstruction from multiple images: Part 1 – principles. *Foundations and Trends in Computer Graphics and Vision* **4** (4): 287–404. [453](#)
- MOORE, A. P., PRINCE, S. J. D., & WARRELL, J. (2010) "Lattice Cut" – constructing superpixels using layer constraints. In *IEEE Computer Vision & Pattern Recognition*, pp. 2117–2124. [316](#), [317](#)
- MOORE, A. P., PRINCE, S. J. D., WARRELL, J., MOHAMMED, U., & JONES, G. (2008) Superpixel lattices. In *IEEE Computer Vision & Pattern Recognition*. [274](#)
- MOOSMANN, F., NOWAK, E., & JURIE, F. (2008) Randomized clustering forests for image classification. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **30** (9): 1632–1646. [209](#)

- MOOSMANN, F., TRIGGS, B., & JURIE, F. (2006) Fast discriminative visual codebooks using randomized clustering forests. In *Advances in Neural Information Processing Systems*, pp. 985–992. [209](#)
- MORAVEC, H. (1983) The Stanford cart and the CMU rover. *Proceedings of the IEEE* **71** (7): 872–884. [352](#)
- MORI, G., REN, X., EFROS, A. A., & MALIK, J. (2004) Recovering human body configurations: Combining segmentation and recognition. In *IEEE Computer Vision & Pattern Recognition*, pp. 326–333. [500](#)
- MUNDY, J., & ZISSERMAN, A. (1992) *Geometric invariance in computer vision*. MIT Press. [385](#)
- MURASE, H., & NAYAR, S. K. (1995) Visual learning and recognition of 3D objects from appearance. *International Journal of Computer Vision* **14** (1): 5–24. [140](#)
- MURINO, V., CASTELLANI, U., ETRARI, E., & FUSIELLO, A. (2002) Registration of very time-distant aerial images. In *IEEE International Conference on Image Processing*, pp. 989–992. [420](#)
- MURPHY, K., WEISS, Y., & JORDAN, M. (1999) Loopy belief propagation for approximate inference: an empirical study. In *Uncertainty in Artificial Intelligence*, pp. 467–475. [275](#)
- MURPHY, K. P. (1998) Switching Kalman filters. Technical report. [567](#)
- MÍCUŠÍK, B., & PAJDLA, T. (2003) Estimation of omnidirectional camera model from epipolar geometry. In *IEEE Computer Vision & Pattern Recognition*, pp. 485–490. [385](#)
- NADARAJAH, S., & KOTZ, S. (2008) Estimation methods for the multivariate t distribution. *Acta Applicandae Mathematicae: An International Survey Journal on Applying Mathematics and Mathematical Applications* **102** (1): 99–118. [140](#)
- NAVARATNAM, R., FITZGIBBON, A. W., & CIPPOLA, R. (2007) The joint manifold model for semi-supervised multi-valued regression. In *IEEE International Conference on Computer Vision*, pp. 1–8. [169](#)
- NEAL, R., & HINTON, G. (1999) A view of the EM algorithm that justifies incremental, sparse and other variants. In *Learning in Graphical Models*, ed. by M. I. Jordan. MIT Press. [140](#)
- NEWCOMBE, R. A., & DAVISON, A. J. (2010) Live dense reconstruction with a single moving camera. In *IEEE Computer Vision & Pattern Recognition*, pp. 1498–1505. [453](#)
- NEWCOMBE, R. A., LOVEGROVE, S., & DAVISON, A. J. (2011) DTAM: Dense tracking and mapping in real-time. In *IEEE International Conference on Computer Vision*. [567](#)
- NEWMAN, P., & HO, K. L. (2005) SLAM – loop closing with visually salient features. In *IEEE International Conference on Robotics and Automation*. [568](#)
- NGUYEN, H. V., & BAI, L. (2010) Cosine similarity metric learning for face verification. In *Asian Conference on Computer Vision*, pp. 709–720. [534](#)
- NIEBLES, J. C., WANG, H., & LI, F.-F. (2008) Unsupervised learning of human action categories using spatial-temporal words. *International Journal of Computer Vision* **79** (3): 299–318. [596](#)
- NISTÉR, D. (2004) An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **26** (6): 756–777. [453](#)
- NISTÉR, D., NARODITSKY, O., & BERGEN, J. R. (2004) Visual odometry. In *IEEE Computer Vision & Pattern Recognition*, pp. 652–659. [567](#)
- NISTÉR, D., & STEWÉNIUS, H. (2006) Scalable recognition with a vocabulary tree. In *IEEE Computer Vision & Pattern Recognition*, pp. 2161–2168. [595](#)
- NIXON, M., & AGUADO, A. S. (2008) *Feature extraction and image processing*. Academic Press, 2nd edition. [19](#), [352](#)
- NOWAK, E., & JURIE, F. (2007) Learning visual similarity measures for comparing never seen objects. In *IEEE Computer Vision & Pattern Recognition*. [534](#)
- O' GORMAN, L., SAMMON, M. J., & SEUL, M. (2008) *Practical Algorithms for Image Analysis*. Cambridge University Press, 2nd edition. [352](#)
- OH, S. M., REHG, J. M., BALCH, T. R., & DELLAERT, F. (2005) Learning and inference in parametric switching linear dynamical systems. In *IEEE International Conference on Computer Vision*, pp. 1161–1168. [567](#)
- OHTA, Y., & KANADE, T. (1985) Stereo by intra- and inter-scanline search using dynamic programming. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **7** (2): 139–154. [269](#), [274](#)

- OJALA, T., PIETIKÄINEN, M., & MÄENPÄÄ, T. (2002) Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **24** (7): 971–987. [352](#), [534](#)
- OLIVER, N., ROSARIO, B., & PENTLAND, A. (2000) A Bayesian computer vision system for modeling human interactions. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **22** (8): 831–843. [97](#), [274](#)
- OPELT, A., PINZ, A., & ZISSERMAN, A. (2006) A boundary-fragment-model for object detection. In *European Conference on Computer Vision*, pp. 575–588. [499](#)
- OSUNA, E., FREUND, R., & GIROSI, F. (1997) Training support vector machines: an application to face detection. In *IEEE Computer Vision & Pattern Recognition*, pp. 746–751. [210](#)
- ÖZUYŞAL, M., CALONDER, M., LEPETIT, V., & FUĆ, P. (2010) Fast keypoint recognition using random ferns. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **32** (3): 448–461. [420](#)
- PAPOURIS, A. (1991) *Probability, Random Variables and Stochastic Processes*. McGraw Hill, 3rd edition. [33](#)
- PEARL, J. (1988) *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann. [275](#)
- PEEL, D., & McLACHLAN, G. (2000) Robust mixture modelling using the t distribution. *Statistics and Computing* **10** (4): 339–348. [140](#)
- PERLIBAKAS, V. (2004) Distance measures for PCA-based face recognition. *Pattern Recognition Letters* **25** (6): 711–724. [533](#)
- PETERSEN, K. B., PEDERSEN, M. S., LARSEN, J., STRIMMER, K., CHRISTIANSEN, L., HANSEN, K., HE, L., THIBAUT, L., BARO, M., HATTINGER, S., SIMA, V., & THE, W. (2006) The matrix cookbook. Technical report. [623](#)
- PHAM, M., & CHAN, T. (2007a) Fast training and selection of Haar features using statistics in boosting-based face detection. In *IEEE International Conference on Computer Vision*. [209](#), [210](#)
- PHAM, M., & CHAN, T. (2007b) Online learning asymmetric boosted classifiers for object detection. In *IEEE Computer Vision & Pattern Recognition*. [210](#)
- PHILBIN, J., CHUM, O., ISARD, M., SIVIC, J., & ZISSERMAN, A. (2007) Object retrieval with large vocabularies and fast spatial matching. In *IEEE Computer Vision & Pattern Recognition*. [595](#)
- PHILBIN, J., ISARD, M., SIVIC, J., & ZISSERMAN, A. (2010) Descriptor learning for efficient retrieval. In *European Conference on Computer Vision*, pp. 677–691. [352](#), [595](#)
- PHILLIPS, P. J., MOON, H., RIZVI, S. A., & RAUSS, P. J. (2000) The FERET evaluation methodology for face-recognition algorithms. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **22** (10): 1090–1104. [534](#)
- PHUNG, S., BOUZERDOUM, A., & CHAI, D. (2005) Skin segmentation using color pixel classification: analysis and comparison. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **27** (1): 147–154. [97](#)
- PICCARDI, M. (2004) Background subtraction techniques: a review. In *IEEE Int. Conf. Systems, Man and Cybernetics*, pp. 3099–3105. [97](#)
- PINTO, N., & COX, D. (2011) Beyond simple features: A large-scale feature search approach to unconstrained face recognition. In *IEEE International Conference on Automatic Face & Gesture Recognition*. [534](#)
- POLLEFEYS, M. (2002) Visual 3D modeling from images. *On-line tutorial*: <http://www.cs.unc.edu/marc/tutorial>. [453](#)
- POLLEFEYS, M., KOCH, R., & VAN GOOL, L. J. (1999a) Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters. *International Journal of Computer Vision* **32** (1): 7–25. [449](#)
- POLLEFEYS, M., KOCH, R., & VAN GOOL, L. J. (1999b) A simple and efficient rectification method for general motion. In *IEEE International Conference on Computer Vision*, pp. 496–501. [443](#), [453](#)
- POLLEFEYS, M., & VAN GOOL, L. J. (2002) Visual modelling: from images to images. *Journal of Visualization and Computer Animation* **13** (4): 199–209. [447](#), [448](#)
- POLLEFEYS, M., VAN GOOL, L. J., VERGAUWEN, M., VERBIEST, F., CORNELIS, K., TOPS, J., & KOCH, R. (2004) Visual modeling with a hand-held camera. *International Journal of Computer Vision* **59** (3): 207–232. [449](#), [453](#)

- PONS, J.-P., KERIVEN, R., & FAUGERAS, O. D. (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. [454](#)
- POPPE, R. (2007) Vision-based human motion analysis: An overview. *Computer Vision and Image Understanding* **108** (1–2): 4–18. [500](#), [567](#)
- PORTILLA, J., & SIMONCELLI, E. (2000) A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision* **40** (1): 49–70. [317](#)
- PRATT, W. H. (2007) *Digital Image Processing*. Wiley Interscience, 3rd edition. [352](#)
- PRINCE, S., CHEOK, A. D., FARBIZ, F., WILLIAMSON, T., JOHNSON, N., BILLINGHURST, M., & KATO, H. (2002) 3D live: Real time captured content for mixed reality. In *International Symposium on Mixed and Augmented Reality*, pp. 317–324. [382](#), [383](#), [384](#), [385](#)
- PRINCE, S. J. D., & AGHAJANIAN, J. (2009) Gender classification in uncontrolled settings using additive logistic models. In *IEEE International Conference on Image Processing*, pp. 2557–2560. [201](#), [210](#)
- PRINCE, S. J. D., ELDER, J. H., WARRELL, J., & FELISBERTI, F. M. (2008) Tied factor analysis for face recognition across large pose differences. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **30** (6): 970–984. [522](#), [524](#), [533](#), [534](#)
- PRINZIE, A., & VAN DEN POEL, D. (2008) Random forests for multiclass classification: random multinomial logit. *Expert Systems with Applications* **35** (3): 1721–1732. [209](#)
- PRITCH, Y., KAV-VENAKI, E., & PELEG, S. (2009) Shift-map image editing. In *IEEE International Conference on Computer Vision*, pp. 151–158. [308](#), [309](#), [310](#), [316](#)
- QUAN, L., & LAN, Z.-D. (1999) Linear n-point camera pose determination. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **21** (8): 774–780. [385](#)
- RABINER, L. (1989) A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* **77** (2): 257–286. [274](#)
- RAE, R., & RITTER, H. (1998) Recognition of human head orientation based on artificial neural networks. *IEEE Transactions on Neural Networks* **9** (2): 257–265. [169](#)
- RAGURAM, R., FRAHM, J.-M., & POLLEFEYS, M. (2008) A comparative analysis of RANSAC techniques leading to adaptive real-time random sample consensus. In *European Conference on Computer Vision*, pp. 500–513. [420](#)
- RAMANAN, D., FORSYTH, D. A., & ZISSERMAN, A. (2008) Tracking people by learning their appearance. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **30** (1): 65–81. [274](#)
- RANGANATHAN, A. (2009) Semantic scene segmentation using random multinomial logit. In *British Machine Vision Conference*. [210](#)
- RANGANATHAN, A., & YANG, M. (2008) Online sparse matrix Gaussian process regression and vision applications. In *European Conference on Computer Vision*, pp. 468–482. [169](#)
- RAPHAEL, C. (2001) Course-to-fine dynamic programming. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **23** (12): 1379–1390. [274](#)
- RASMUSSEN, C. E., & WILLIAMS, C. K. I. (2006) *Gaussian Processes for Machine Learning*. MIT Press. [169](#), [209](#)
- REHG, J., & KANADE, T. (1994) Visual tracking of high DOF articulated structures: an application to human hand tracking. In *European Conference on Computer Vision*, pp. 35–46. [500](#)
- REHG, J. M., & KANADE, T. (1995) Model-based tracking of self-occluding articulated objects. In *IEEE International Conference on Computer Vision*, pp. 612–617. [500](#)
- REHG, J. M., MORRIS, D. D., & KANADE, T. (2003) Ambiguities in visual tracking of articulated objects using two- and three-dimensional models. *I. J. Robotic Res.* **22** (6): 393–418. [500](#)
- REKIMOTO, J. (1998) MATRIX: A realtime object identification and registration method for augmented reality. In *Asia Pacific Computer Human Interaction*, pp. 63–69. [420](#)
- REN, X., BERG, A. C., & MALIK, J. (2005) Recovering human body configurations using pairwise constraints between parts. In *IEEE International Conference on Computer Vision*, pp. 824–831. [274](#)
- RIGOLL, G., KOSMALA, A., & EICKELER, S. (1998) High performance real-time gesture recognition using hidden Markov

- models. In *Int. Workshop on Gesture and Sign language in Human-Computer Interaction*. 274
- ROGEZ, G., RIHAN, J., RAMALINGAM, S., ORRITE, C., & TORR, P. (2006) Randomized trees for human pose detection. In *Advances in Neural Information Processing Systems*, pp. 985–992. 209
- ROMDHANI, S., CONG, S., & PSARROU, A. (1999) A multi-view non-linear active shape model using kernel PCA. In *British Machine Vision Conference*. 499
- ROSALES, R., & SCLAROFF, S. (1999) 3D trajectory recovery for tracking multiple objects and trajectory guided recognition of actions. In *IEEE Computer Vision & Pattern Recognition*, pp. 2117–2123. 563, 567
- ROSEN-ZVI, M., GRIFFITHS, T. L., STEYVERS, M., & SMYTH, P. (2004) The author-topic model for authors and documents. In *Uncertainty in Artificial Intelligence*, pp. 487–494. 595
- ROSENBLATT, F. (1958) The Perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review* **65** (6): 386–408. 209
- ROSTEN, E., & DRUMMOND, T. (2006) Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pp. 430–443. 352
- ROTH, S., & BLACK, M. J. (2009) Fields of experts. *International Journal of Computer Vision* **82** (2): 205–229. 140, 317
- ROTHER, C., BORDEAUX, L., HAMADI, Y., & BLAKE, A. (2006) Autocollage. *ACM Transactions on Graphics* **25** (3): 847–852. 316
- ROTHER, C., KOHLI, P., FENG, W., & JIA, J. (2009) Minimizing sparse higher order energy functions of discrete variables. In *IEEE Computer Vision & Pattern Recognition*, pp. 1382–1389. 317
- ROTHER, C., KOLMOGOROV, V., & BLAKE, A. (2004) Grabcut – interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (SIGGRAPH 2004)* **23** (3): 309–314. 316, 317
- ROTHER, C., KOLMOGOROV, V., LEMPITSKY, V. S., & SZUMMER, M. (2007) Optimizing binary MRFs via extended roof duality. In *IEEE Computer Vision & Pattern Recognition*. 316
- ROTHER, C., KUMAR, S., KOLMOGOROV, V., & BLAKE, A. (2005) Digital tapestry. In *IEEE Computer Vision & Pattern Recognition*, pp. 589–586. 305, 306, 316
- ROTHWELL, C. A., ZISSERMAN, A., FORSYTH, D. A., & MUNDY, J. L. (1995) Planar object recognition using projective shape representation. *International Journal of Computer Vision* **16** (1): 57–99. 422
- ROUSSEEUW, P. J. (1984) Least median of squares regression. *Journal of the American Statistical Association* **79** (388): 871–880. 420
- ROUSSON, M., & PARAGIOS, N. (2002) Shape priors for level set representations. In *European Conference on Computer Vision*, pp. 78–92. 499
- ROWEIS, S., & SAUL, L. (2000) Nonlinear dimensionality reduction by locally linear embedding. *Science* **290** (5500): 2323–2326. 352
- ROWEIS, S. T., & GHAHRAMANI, Z. (1999) A unifying review of linear Gaussian models. *Neural Computation* **11** (2): 305–345. 567
- ROWEIS, S. T., & GHAHRAMANI, Z. (2001) Learning nonlinear dynamical systems using the expectation-maximization algorithm. In *Kalman Filtering and Neural Networks*, ed. by S. Haykin, pp. 175–220. Wiley. 567
- RUBIN, D., & THAYER, D. (1982) EM algorithms for ML factor analysis. *Psychometrika* **47** (1): 69–76. 140, 499
- RUMELHART, D. E., HINTON, G. E., & WILLIAMS, R. (1986) Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*, ed. by D. Rumelhart, J. McClelland, & The PDP Research Group, pp. 318–362. 209
- SALMEN, J., SCHLIPSING, M., EDELBRUNNER, J., HEGEMANN, S., & LÜKE, S. (2009) Real-time stereo vision: Making more out of dynamic programming. In *Proceedings of the 13th International Conference on Computer Analysis of Images and Patterns*, CAIP '09, pp. 1096–1103. Springer-Verlag. 274
- SALVI, J., PAGS, J., & BATLLÉ, J. (2004) Pattern codification strategies in structured light systems. *Pattern Recognition* **37** (4): 827 – 849. Agent Based Computer Vision. 385
- SCHAFFALITZKY, F., & ZISSERMAN, A. (2002) Multi-view matching for unordered image sets, or “How do I organize my holiday snaps?”. In *European Conference on Computer Vision*, pp. 414–431. 352

- SCHAPIRE, R., & SINGER, Y. (1998) Improved boosting algorithms using confidence-rated predictions. In *Conference on Computational Learning Theory*, pp. 80–91. [209](#)
- SCHARSTEIN, D., & SZELISKI, R. (2002) A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision* **47** (1): 7–42. [318](#)
- SCHARSTEIN, D., & SZELISKI, R. (2003) High-accuracy depth maps using structured light. In *IEEE Computer Vision & Pattern Recognition*, pp. 194–202. [378](#), [379](#), [380](#), [385](#)
- SCHLESINGER, D., & FLACH, B. (2006) Transforming an arbitrary minsum problem into a binary one. Technical Report TUD-FI06-01, Dresden University of Technology. [316](#)
- SCHMUGGE, S. J., JAYARAM, S., SHIN, M., & TSAP, L. (2007) Objective evaluation of approaches to skin detection using roc analysis. *Computer Vision & Image Understanding* **108** (1–2): 41–51. [97](#)
- SCHNEIDERMAN, H., & KANADE, T. (2000) A statistical method for 3D object detection applied to faces and cards. In *IEEE International Conference on Computer Vision*, pp. 746–751. [210](#)
- SCHÖLKOPF, B., SMOLA, A. J., & MÜLLER, K.-R. (1997) Kernel principal component analysis. In *International Conf. on Artificial Neural Networks*, pp. 583–588. [352](#)
- SCHÖLKOPF, B., SMOLA, A. J., & MÜLLER, K.-R. (1998) Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation* **10** (5): 1299–1319. [499](#)
- SCHÜLDT, C., LAPTEV, I., & CAPUTO, B. (2004) Recognizing human actions: A local SVM approach. In *International Conference on Pattern Recognition*, pp. 32–36. [592](#), [593](#), [595](#)
- SEITZ, S. M., CURLESS, B., DIEBEL, J., SCHARSTEIN, D., & SZELISKI, R. (2006) A comparison and evaluation of multi-view stereo reconstruction algorithms. In *IEEE Computer Vision & Pattern Recognition*, pp. 519–528. [454](#)
- SEO, H., & MAGNENAT-THALMANN, N. (2003) An automatic modeling of human bodies from sizing parameters. In *SI3D*, pp. 19–26. [500](#)
- SFIKAS, G., NIKOU, C., & GALATSANOS, N. (2007) Robust image segmentation with mixtures of Student's t-distributions. In *IEEE International Conference on Image Processing*, pp. 273–276. [135](#), [136](#), [140](#)
- SHA'ASHUA, A., & ULLMAN, S. (1988) Structural saliency: the detection of globally salient structures using a locally connected network. In *IEEE International Conference on Computer Vision*, pp. 321–327. [274](#)
- SHAKHNAROVICH, G., VIOLA, P. A., & DARRELL, T. (2003) Fast pose estimation with parameter-sensitive hashing. In *IEEE International Conference on Computer Vision*, pp. 750–759. [500](#)
- SHAN, C. (in press) Learning local binary patterns for gender classification on real-world face images. *Pattern Recognition Letters*. [210](#)
- SHEPHERD, B. (1983) An appraisal of a decision tree approach to image classification. In *International Joint Conferences on Artificial Intelligence*, pp. 473–475. [209](#)
- SHI, J., & TOMASI, C. (1994) Good features to track. In *IEEE Computer Vision & Pattern Recognition*, pp. 311–326. [453](#)
- SHOTTON, J., BLAKE, A., & CIPOLLA, R. (2008a) Multiscale categorical object recognition using contour fragments. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **30** (7): 1270–1281. [499](#)
- SHOTTON, J., FITZGIBBON, A. W., COOK, M., SHARP, T., FINOCCIO, M., MOORE, R., KIPMAN, A., & BLAKE, A. (2011) Real-time human pose recognition in parts from single depth images. In *IEEE Computer Vision & Pattern Recognition*. [207](#), [209](#)
- SHOTTON, J., JOHNSON, M., & CIPOLLA, R. (2008b) Semantic texton forests for image categorization and segmentation. In *IEEE Computer Vision & Pattern Recognition*. [210](#)
- SHOTTON, J., WINN, J., ROTHER, C., & CRIMINSKI, A. (2009) Textonboost for image understanding: multi-class object recognition and segmentation by jointly modeling texture, layout and context. *International Journal of Computer Vision* **81** (1): 2–23. [203](#), [204](#), [205](#), [209](#), [210](#), [316](#), [335](#)
- SHUM, H.-Y., & SZELISKI, R. (2000) Construction of panoramic image mosaics with global and local alignment. *International Journal of Computer Vision* **36** (2): 101–130. [421](#)

- SHUMWAY, R. H., & STOFFER, D. S. (1991) Dynamic linear models with switching. *Journal of the American Statistical Association* **86** (415): 763–769. [567](#)
- SHUMWAY, R. H., & STOFFER, D. S. (2007) An approach to time series smoothing and forecasting using the EM algorithm. *J. Time Series Analysis* **3** (4): 253–264. [567](#)
- SIDENBLADH, H., BLACK, M. J., & FLEET, D. J. (2000) Stochastic tracking of 3D human figures using 2D image motion. In *European Conference on Computer Vision*, pp. 702–718. [500](#)
- SIGAL, L., BALAN, A. O., & 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. [500](#)
- SIGAL, L., & BLACK, M. J. (2006) Measure locally, reason globally: Occlusion-sensitive articulated pose estimation. In *IEEE Computer Vision & Pattern Recognition*, pp. 2041–2048. [274](#)
- SIM, R., ELINAS, P., GRIFFIN, M., & LITTLE, J. (2005) Vision-based SLAM using the Rao-Blackwellised particle filter. In *IJCAI Workshop on Reasoning with Uncertainty in Robotics*, pp. 9–16. [567](#)
- SIMON, G., & BERGER, M.-O. (2002) Pose estimation for planar structures. *IEEE Computer Graphics and Applications* **22** (6): 46–53. [420](#)
- SIMON, G., FITZGIBBON, A. W., & ZISSEMAN, A. (2000) Markerless tracking using planar structures in the scene. In *International Symposium on Mixed and Augmented Reality*, pp. 120–128. [420](#)
- SINHA, S. N., MORDOHAI, P., & POLLEFEYS, M. (2007) Multi-view stereo via graph cuts on the dual of an adaptive tetrahedral mesh. In *IEEE International Conference on Computer Vision*, pp. 1–8. [454](#)
- SINHA, S. N., & POLLEFEYS, M. (2005) Multi-view reconstruction using photo-consistency and exact silhouette constraints: A maximum-flow formulation. In *IEEE International Conference on Computer Vision*, pp. 349–356. [454](#)
- SIVIC, J., RUSSELL, B. C., EFROS, A. A., ZISSERMAN, A., & FREEMAN, W. T. (2005) Discovering objects and their localization in images. In *IEEE International Conference on Computer Vision*, pp. 370–377. [595](#)
- SIVIC, J., RUSSELL, B. C., ZISSERMAN, A., FREEMAN, W. T., & EFROS, A. A. (2008) Unsupervised discovery of visual object class hierarchies. In *IEEE Computer Vision & Pattern Recognition*. [595](#)
- SIVIC, J., & ZISSERMAN, A. (2003) Video google: A text retrieval approach to object matching in videos. In *ICCV*, pp. 1470–1477. [591](#), [595](#)
- SIZINTSEV, M., KUTHIRUMMAL, S., SAWHNEY, H., CHAUDHRY, A., SAMARASEKERA, S., & KUMAR, R. (2010) GPU accelerated real-time stereo for augmented reality. In *International Symposium 3D Data Processing, Visualization and Transmission*. [444](#)
- SIZINTSEV, M., & WILDES, R. P. (2010) Coarse-to-fine stereo vision with accurate 3D boundaries. *Image and Vision Computing* **28** (3): 352–366. [318](#)
- SKRYPNYK, I., & LOWE, D. G. (2004) Scene modelling, recognition and tracking with invariant image features. In *International Symposium on Mixed and Augmented Reality*, pp. 110–119. [420](#)
- SMITH, R., & CHEESEMAN, P. (1987) On the representation of spatial uncertainty. *International Journal of Robotics Research* **5** (4): 56–68. [567](#)
- SMITH, R., SELF, M., & CHEESEMAN, P. (1990) Estimating uncertain spatial relationships in robotics. In *Autonomous Robot Vehicles*, ed. by I. J. Cox & G. T. Wilton, pp. 167–193. Springer. [567](#)
- SMITH, S. M., & BRADY, J. M. (1997) SU-SAN - a new approach to low level image processing. *International Journal of Computer Vision* **23** (1): 45–78. [352](#)
- SNAVELY, N., GARG, R., SEITZ, S. M., & SZELISKI, R. (2008) Finding paths through the world's photos. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2008)* **27** (3): 11–21. [450](#)
- SNAVELY, N., SEITZ, S. M., & SZELISKI, R. (2006) Photo tourism: Exploring photo collections in 3D. In *SIGGRAPH Conference Proceedings*, pp. 835–846. ACM Press. [449](#)
- STARCK, J., MAKI, A., NOBUHURA, S., HILTON, A., & MASTUYAMA, T. (2009) The multiple-camera 3-D production studio. *IEEE Transactions on Circuits and Systems for Video Technology* **19** (6): 856–869. [385](#)
- STARNER, T., WEAVER, J., & PENTLAND, A. (1998) A wearable computer based amer-

- ican sign language recognizer. In *Assistive Technology and Artificial Intelligence*, Lecture Notes in Computer Science, Volume 1458, pp. 84–96. 267, 274
- STATE, A., HIROTA, G., CHEN, D. T., GARRETT, W. F., & LIVINGSTON, M. A. (1996) Superior augmented reality registration by integrating landmark tracking and magnetic tracking. In *SIGGRAPH*, pp. 429–438. 420
- STAUFFER, C., & GRIMSON, E. (1999) Adaptive background classification using time-based co-occurrences. In *IEEE Computer Vision & Pattern Recognition*, pp. 246–252. 97, 140
- STEGMANN, M. B. (2002) Analysis and segmentation of face images using point annotations and linear subspace techniques. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby. 483, 484, 499
- STENGER, B., MENDONÇA, P. R. S., & CIPOLLA, R. (2001a) Model-based 3D tracking of an articulated hand. In *IEEE Computer Vision & Pattern Recognition*, pp. 310–315. 492, 500
- STENGER, B., MENDONÇA, P. R. S., & CIPOLLA, R. (2001b) Model-based hand tracking using an unscented Kalman filter. In *British Machine Vision Conference*. 567
- STEWÉNIUS, H., ENGELS, C., & NISTÉR, D. (2006) Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing* **60**: 284–294. 453
- STRASDAT, H., MONTIEL, J. M. M., & DAVIDSON, A. J. (2010) Real-time monocular SLAM: Why filter? In *IEEE International Conference on Robotics and Automation*, pp. 2657–2664. 567
- STURM, P. F. (2000) Algorithms for plane-based pose estimation. In *IEEE Computer Vision & Pattern Recognition*, pp. 1706–1711. 420
- STURM, P. F., & MAYBANK, S. J. (1999) On plane-based camera calibration: A general algorithm, singularities, applications. In *IEEE Computer Vision & Pattern Recognition*, pp. 1432–1437. 420
- STURM, P. F., RAMALINGAM, S., TARDIF, J.-P., GASPARINI, S., & BARRETO, J. (2011) Camera models and fundamental concepts used in geometric computer vision. *Foundations and Trends in Computer Graphics and Vision* **6** (1-2): 1–183. 385
- STURM, P. F., & TRIGGS, B. (1996) A factorization based algorithm for multi-image projective structure and motion. In *European Conference on Computer Vision*, pp. 709–720. 453
- SUDDERTH, E. B., TORRALBA, A., FREEMAN, W. T., & WILLSKY, A. S. (2005) Learning hierarchical models of scenes, objects, and parts. In *IEEE International Conference on Computer Vision*, pp. 1331–1338. 595
- SUDDERTH, E. B., TORRALBA, A., FREEMAN, W. T., & WILLSKY, A. S. (2008) Describing visual scenes using transformed objects and parts. *International Journal of Computer Vision* **77** (1-3): 291–330. 587, 589, 595
- SUGIMOTO, A. (2000) A linear algorithm for computing the homography for conics in correspondence. *Journal of Mathematical Imaging and Vision* **13** (2): 115–130. 420
- SUN, J., ZHANG, W., TANG, X., & SHUM, H. Y. (2006) Background cut. In *European Conference on Computer Vision*, pp. 628–641. 97
- SUN, J., ZHENG, N., & SHUM, H. Y. (2003) Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **25** (7): 787–800. 275, 317
- SUTHERLAND, I. E. (1963) Sketchpad: a man-machine graphical communications system. Technical Report Technical Report 296, MIT Lincoln Laboratories. 420
- SUTTON, C., & McCALLUM, A. (2011) An introduction to conditional random fields. *Foundations and Trends in Machine Learning*. 316
- SZELISKI, R. (1996) Video mosaics for virtual environments. *IEEE Computer Graphics and Applications* **16** (2): 22–30. 421
- SZELISKI, R. (2006) Image alignment and stitching: A tutorial. *Foundations and Trends in Computer Graphics and Vision* **2** (1). 421
- SZELISKI, R. (2010) *Computer vision: algorithms and applications*. Springer. 17, 19, 318, 421
- SZELISKI, R., & SHUM, H.-Y. (1997) Creating full view panoramic image mosaics and environment maps. In *SIGGRAPH*, pp. 251–258. 421

- SZELISKI, R., ZABIH, R., SCHARSTEIN, D., VEKSLER, O., KOLMOGOROV, V., AGARWALA, A., TAPPEN, M., & ROTHER, C. (2008) A comparative study of energy minimization methods for Markov random fields. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **30** (6): 1068–1080. [317](#)
- TAIGMAN, Y., WOLF, L., & HASSNER, T. (2009) Multiple one-shots for utilizing class label information. In *British Machine Vision Conference*. [534](#)
- TAPPEN, M. F., & FREEMAN, W. T. (2003) Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters. In *IEEE International Conference on Computer Vision*, pp. 900–907. [317](#)
- TARLOW, D., GIVONI, I., ZEMEL, R., & FREY, B. (2011) Graph cuts is a max-product algorithm. In *Uncertainty in Artificial Intelligence*. [316](#)
- TENENBAUM, J., SILVA, V., & LANGFORD, J. (2000) A global geometric framework for nonlinear dimensionality reduction. *Science* **290** (5500): 2319–2315. [352](#)
- TENENBAUM, J. B., & FREEMAN, W. T. (2000) Separating style and content with bilinear models. *Neural Computation* **12** (6): 1247–1283. [527, 534](#)
- TERZOPOLOUS, D., & SZELISKI, R. (1992) Tracking with Kalman snakes. In *Active Vision*, ed. by A. Blake & A. Y. Yuille, pp. 3–29. [567](#)
- THAYANANTHAN, A., NAVATNAM, R., STENGER, B., TORR, P., & CIPOLLA, R. (2006) Multivariate relevance vector machines for tracking. In *European Conference on Computer Vision*, pp. 124–138. [169](#)
- THEOBALT, C., AHMED, N., LENSCHE, H. P. A., MAGNOR, M. A., & SEIDEL, H.-P. (2007) Seeing people in different light—joint shape, motion, and reflectance capture. *IEEE Transactions on Visualization and Computer Graphics* **13** (4): 663–674. [385](#)
- TIPPING, M., & BISHOP, C. M. (1999) Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B* **61** (3): 611–622. [140](#)
- TIPPING, M. E. (2001) Sparse Bayesian learning and the relevance vector machine. *J. Machine Learning Research* **1**: 211–244. [159, 169, 209, 499](#)
- TOMASI, C., & KANADE, T. (1991) Detection and tracking of point features. Technical Report CMU-SC-91-132, Carnegie Mellon. [453](#)
- TOMASI, C., & KANADE, T. (1992) Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision* **9** (2): 137–154. [453](#)
- TOMBARI, F., MATTOCCIA, S., DI STEFANO, L., & ADDIMANDA, E. (2008) Classification and evaluation of cost aggregation methods for stereo correspondence. In *IEEE Computer Vision & Pattern Recognition*. [318](#)
- TORR, P. (1998) Geometric motion segmentation and model selection. *Philosophical Transactions of the Royal Society A* **356** (1740): 1321–1340. [420](#)
- TORR, P. H. S., & CRIMINISI, A. (2004) Dense stereo using pivoted dynamic programming. *Image and vision computing* **22** (10): 795–806. [274](#)
- TORR, P. H. S., & ZISSERMAN, A. (2000) MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision & Image Understanding* **78** (1): 138–156. [420](#)
- TORRALBA, A., MURPHY, K., & FREEMAN, W. T. (2007) Sharing visual features for multiclass and multi-view object detection. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **29** (5): 854–869. [203, 209](#)
- TREIBITZ, T., SCHECHNER, Y. Y., & SINGH, H. (2008) Flat refractive geometry. In *IEEE Computer Vision & Pattern Recognition*. [385](#)
- TRIGGS, B., MC LAUCHLAN, P. F., HARTLEY, R. I., & FITZGIBBON, A. W. (1999) Bundle adjustment – a modern synthesis. In *Workshop on Vision Algorithms*, pp. 298–372. [453](#)
- TSAI, R. (1987) A versatile cameras calibration technique for high accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *Journal of Robotics and Automation* **3** (4): 323–344. [385](#)
- TURK, M., & PENTLAND, A. P. (2001) Face recognition using eigenfaces. In *IEEE Computer Vision & Pattern Recognition*, pp. 586–591. [533](#)
- TUYTELAARS, T., & MIKOŁAJCZYK, K. (2007) Local invariant feature detectors: A survey. *Foundations and Trends in Computer Graphics and Vision* **3** (3): 177–280. [352](#)

- URTASUN, R., FLEET, D. J., & FUA, P. (2006) 3D people tracking with Gaussian process dynamical models. In *IEEE Computer Vision & Pattern Recognition*, pp. 238–245. 169
- VAPNIK, V. (1995) *The Nature of Statistical Learning Theory*. Springer Verlag. 209
- VARMA, M., & ZISSERMAN, A. (2004) Unifying statistical texture classification frameworks. *Image and Vision Computing* **22** (14): 1175–1183. 595
- VASILESCU, M. A. O., & TERZOPoulos, D. (2002) Multilinear analysis of image ensembles: Tensorfaces. In *European Conference on Computer Vision*, pp. 447–460. 534
- VASILESCU, M. A. O., & TERZOPoulos, D. (2004) Tensortextures: multilinear image-based rendering. *ACM Transactions on Graphics* **23** (3): 336–342. 530
- VASWANI, N., CHOWDHURY, A. K. R., & CHELLAPPA, R. (2003) Activity recognition using the dynamics of the configuration of interacting objects. In *IEEE Computer Vision & Pattern Recognition*, pp. 633–642. 567
- VEKSLER, O. (2005) Stereo correspondence by dynamic programming on a tree. In *IEEE Computer Vision & Pattern Recognition*, pp. 384–390. 269, 270, 274
- VEKSLER, O. (2008) Star shape prior for graph-cut image segmentation. In *European Conference on Computer Vision*, pp. 454–467. 317
- VEKSLER, O., BOYKOV, Y., & MEHRANI, P. (2010) Superpixels and supervoxels in an energy optimization framework. In *European Conference on Computer Vision*, pp. 211–224. 316
- VEZHNEVETS, V., SAZONOV, V., & ANDREEVA, A. (2003) A survey on pixel-based skin color detection techniques. In *Graphicon*, pp. 85–92. 97
- VICENTE, S., KOLMOGOROV, V., & ROTHER, C. (2008) Graph cut based image segmentation with connectivity priors. In *IEEE Computer Vision & Pattern Recognition*, pp. 1–8. 317
- VINCENT, E., & LAGANIÈRE, R. (2001) Detecting planar homographies in an image pair. In *Image and Signal Processing Analysis*, pp. 182–187. 420
- VIOLA, P., & JONES, M. (2002) Fast and robust classification using asymmetric adaboost and a detector cascade. In *Advances in Neural Information Processing Systems*, pp. 1311–1318. 209, 210
- VIOLA, P. A., & JONES, M. J. (2004) Robust real-time face detection. *International Journal of Computer Vision* **57** (2): 137–154. 202, 203, 209, 210
- VIOLA, P. A., JONES, M. J., & SNOW, D. (2005) Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision* **63** (2): 153–161. 203, 204
- VLASIC, D., BARAN, I., MATUSIK, W., & POPOVIC, J. (2008) Articulated mesh animation from multi-view silhouettes. *ACM Transactions on Graphics* **27** (3). 385
- VOGIATZIS, G., ESTEBAN, C. H., TORR, P. H. S., & CIPOLLA, R. (2007) Multiview stereo via volumetric graph-cuts and occlusion robust photo-consistency. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **29** (12): 2241–2246. 316, 450, 451, 454
- VUYLSTEKE, P., & OOSTERLINCK, A. (1990) Range image acquisition with a single binary-encoded light pattern. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **12** (2): 148–164. 385
- WAGNER, D., REITMAYR, G., MULLONI, A., DRUMMOND, T., & SCHMALSTIEG, D. (2008) Pose tracking from natural features on mobile phones. In *International Symposium on Mixed and Augmented Reality*, pp. 125–134. 420
- WAINRIGHT, M., JAakkola, T., & WILLSKY, A. (2005) MAP estimation via agreement on trees: message passing and linear programming. *IEEE Transactions on Information Theory* **5** (11): 3697–3717. 317
- WANG, J. M., FLEET, D. J., & HERTZMANN, A. (2007) Multifactor Gaussian process models for style-content separation. In *ICML*, pp. 975–982. 531, 532, 534
- WANG, J. M., FLEET, D. J., & HERTZMANN, A. (2008) Gaussian process dynamical models for human motion. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **30** (2): 283–298. 567
- WANG, X., & TANG, X. (2003) Bayesian face recognition using Gabor features. In *Proceedings of the 2003 ACM SIGMM workshop on Biometrics methods and applications*, pp. 70–73. ACM. 533, 534
- WANG, X., & TANG, X. (2004a) Dual-space linear discriminant analysis for face recognition. In *IEEE Computer Vision & Pattern Recognition*, pp. 564–569. 533

- WANG, X., & TANG, X. (2004b) A unified framework for subspace face recognition. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **26** (9): 1222–1228. [533](#)
- WEI, L.-Y., & LEVOY, M. (2000) Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH*, pp. 479–488. [317](#)
- WEISS, Y., & FREEMAN, W. (2001) On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory* **47** (2): 723–735. [275](#), [317](#)
- WEISS, Y., YANOVER, C., & MELTZER, T. (2011) Linear programming and variants of belief propagation. In *Advances in Markov Random Fields*, ed. by A. Blake, P. Kohli, & C. Rother. MIT Press. [317](#)
- WILBUR, R. B., & KAK, A. C. (2006) Purdue RVL-SLLL american sign language database. Technical Report TR-06-12, Purdue University, School of Electrical and Computer Engineering. [244](#)
- WILLIAMS, C., & BARBER, D. (1998) Bayesian classification with Gaussian priors. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **20** (2): 1342–1351. [209](#)
- WILLIAMS, D., & SHAH, M. (1992) A fast algorithm for active contours and curvature estimation. *CVGIP: Image understanding* **55** (1): 14–26. [499](#)
- WILLIAMS, O. M. C., BLAKE, A., & CIPOLLA, R. (2005) Sparse Bayesian learning for efficient tracking. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **27** (8): 1292–1304. [167](#), [169](#)
- WILLIAMS, O. M. C., BLAKE, A., & CIPOLLA, R. (2006) Sparse and semi-supervised visual mapping with the S3P. In *IEEE Computer Vision & Pattern Recognition*, pp. 230–237. [169](#)
- WISKOTT, L., FELLOUS, J.-M., KRÜGER, N., & VON DER MALSBURG, C. (1997) Face recognition by elastic bunch graph matching. In *IEEE International Conference on Image Processing*, pp. 129–132. [533](#)
- WOLF, L., HASSNER, T., & TAIGMAN, Y. (2009) The one-shot similarity kernel. In *IEEE International Conference on Computer Vision*, pp. 897–902. [534](#)
- WOODFORD, O., TORR, P. H. S., REID, I., & FITZGIBBON, A. W. (2009) Global stereo reconstruction under second-order smoothness priors. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **31** (12): 2115–2128. [316](#)
- WREN, C. R., AAZARBAYEJANI, A., DARRELL, T., & PENTLAND, A. P. (1997) Pfnder: real-time tracking of the human body. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **19** (7): 780–785. [97](#)
- WRIGHT, J., YANG, A. Y., GANESH, A., SASTRY, S. S., & MA, Y. (2009) Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **31** (2): 210–227. [534](#)
- WU, B., AI, H., HUANG, C., & LAO, S. (2007) Fast rotation invariant multi-view face detection based on real adaboost. In *IEEE Workshop on Face and Gesture Recognition*, pp. 79–84. [210](#)
- WU, C., AGARWAL, S., CURLESS, B., & SEITZ, S. (2011) Multicore bundle adjustment. In *IEEE Computer Vision & Pattern Recognition*, pp. 3057–3064. [453](#)
- WU, Y., LIN, J. Y., & HUANG, T. S. (2001) Capturing natural hand articulation. In *IEEE International Conference on Computer Vision*, pp. 426–432. [500](#)
- XIAO, J., HAYS, J., EHINGER, K. A., OLIVA, A., & TORRALBA, A. (2010) Sun database: Large-scale scene recognition from abbey to zoo. In *IEEE Computer Vision & Pattern Recognition*, pp. 3485–3492. [595](#)
- XU, C., & PRINCE, J. L. (1998) Snakes, shapes, and gradient vector flow. *IEEE Transactions on Image Processing* **7** (3): 359–369. [499](#)
- YANG, J., JIANG, Y.-G., HAUPTMANN, A. G., & NGO, C.-W. (2007) Evaluating bag-of-visual-words representations in scene classification. In *Multimedia Information Retrieval*, pp. 197–206. [595](#)
- YANG, M.-H. (2002) Kernel eigenfaces vs. kernel fisherfaces: Face recognition using kernel methods. In *IEEE International Conference on Automatic Face & Gesture Recognition*, pp. 215–220. [533](#)
- YILMAZ, A., JAVED, O., & SHAH, M. (2006) Object tracking: A survey. *ACM Computing Surveys (CSUR)* **38** (4): 1–45. [567](#)
- YIN, P., CRIMINISI, A., WINN, J., & ESSA, I. (2007) Tree based classifiers for bilayer video segmentation. In *IEEE Computer Vision & Pattern Recognition*. [209](#)
- YOON, K.-J., & KWEON, I.-S. (2006) Adaptive support-weight approach for correspondence search. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **28** (4): 650–656. [318](#)

- ZHANG, C., & ZHANG, Z. (2010) A survey of recent advances in face detection. Technical Report MSR-TR-2010-66, Microsoft Research, Redmond. [210](#)
- ZHANG, J., MARSZALEK, M., LAZEBNIK, S., & 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. [595](#)
- ZHANG, X., & GAO, Y. (2009) Face recognition across pose: A review. *Pattern Recognition* **42** (11): 2876–2896. [533](#)
- ZHANG, Z. (2000) A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **22** (11): 1330–1334. [420](#)
- ZHAO, J., & JIANG, Q. (2006) Probabilistic PCA for t distributions. *Neurocomputing* **69** (16–18): 2217–2226. [140](#)
- ZHAO, W.-Y., CHELLAPPA, R., PHILLIPS, P. J., & ROSENFELD, A. (2003) Face recogni-  
tion: A literature survey. *ACM Comput. Surv.* **35** (4): 399–458. [533](#)
- ZHOU, S., CHELLAPPA, R., & MOGHADDAM, B. (2004) Visual tracking and recognition using appearance-adaptive models in particle filters. *IEEE Transactions on Image Processing*, **13** (11): 1491–1506. [567](#)
- ZHOU, S. K., & CHELLAPPA, R. (2004) Probabilistic identity characterization for face recognition. In *IEEE Computer Vision & Pattern Recognition*, pp. 805–812. [533](#)
- ZHU, X., YANG, J., & WAIBEL, A. (2000) Segmenting hands of arbitrary colour. In *IEEE International Conference on Automatic Face & Gesture Recognition*, pp. 446–453. [97](#)
- ZITNICK, C. L., & KANADE, T. (2000) A cooperative algorithm for stereo matching and occlusion detection. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **22** (7): 675–684. [318](#)

# Index

- 3D body model, 496–497  
3D morphable model, 494–496, 500  
3D reconstruction, 359, 369–370, 376–377, 453  
    from structured light, 378–380  
    pipeline, 447–449  
    volumetric graph cuts, 450–452
- action recognition, 592–593, 595  
activation, 172  
active appearance model, 482–487, 499  
active contour model, 463–468, 499  
active shape model, 462, 471–482, 499  
    3D, 482  
adaboost, 200, 202, 209, *see also* boosting  
affine transformation, 393–394  
    learning, 399–400  
alignment of shapes, 472–473  
alpha-beta swap, 316, 319  
alpha-expansion algorithm, 298–300, 316  
ancestral sampling, 232  
application  
    3D reconstruction, 359, 447–453  
    action recognition, 592–593, 595  
    animation synthesis, 531–532  
    augmented reality tracking, 390, 416–417, 420  
    background subtraction, 94, 97, 305  
    body pose estimation, 143, 166, 207–208, 271–272  
    body tracking, 500  
    changing face pose, 137  
    contour tracking, 537, 565  
    denoising, 279, 282–284, 291, 300  
    depth from structured light, 378–380  
    face detection, 101, 133–134, 140, 202–203, 210  
    face recognition, 136, 503, 510–514, 517, 528–530, 533  
    face synthesis, 314–315  
    finding facial features, 271, 274  
    fitting 3D body model, 496–497  
    fitting 3D shape model, 494–496  
    gender classification, 171, 201, 209  
    gesture tracking, 243, 267  
    image retargeting, 308–309  
    interactive segmentation, 305–307, 317  
    multi-view reconstruction, 450–453  
    object recognition, 134–135, 140, 571–592
- panorama, 417, 420  
pedestrian detection, 202–203  
pedestrian tracking, 563–564  
Photo-tourism, 449–450  
scene recognition, 571, 590  
segmentation, 135–136, 272–273, 463–468  
semantic segmentation, 203–205, 210  
shape from silhouette, 380–383  
sign language interpretation, 243, 246, 267  
skin detection, 93–94, 97  
SLAM, 564, 567  
stereo vision, 267–270, 274, 307–308, 317  
super-resolution, 310–311  
surface layout recovery, 205–206  
TensorTextures, 530–531  
texture synthesis, 311–314, 317  
tracking head position, 167  
Video Google, 591–592
- approximate inference, 231  
AR Toolkit, 420  
argmax function, 600  
argmin function, 600  
articulated models, 492–493  
    pictorial structures, 271  
asymmetric bilinear model, 518–524  
augmented reality, 390, 416–417, 420  
augmenting paths algorithm, 286, 316  
author-topic model, 582–585, 595  
auto-calibration, 444
- back propagation, 200  
background subtraction, 94, 97, 305  
bag of words, 344, 573–576, 595  
Baum-Welch algorithm, 263  
Bayes' Rule, 30  
Bayesian approach to fitting, 50–51  
Bayesian belief propagation, 257–262, 275  
    loopy, 266  
    sum-product algorithm, 258–259  
Bayesian linear regression, 147–150  
Bayesian logistic regression, 176–180  
Bayesian model selection, 66, 511  
Bayesian network, 219–222  
    comparison to undirected, 225  
    learning, 234–235  
    sampling, 232  
Bayesian nonlinear regression, 153  
belief propagation, 257–262, 275

- loopy, 266
- sum-product algorithm, 258–259
- Bernoulli distribution, 35–37
  - conjugate prior, 42, 46
  - relation to binomial, 45
- beta distribution, 35, 37–38
- between-individual variation, 507
- BFGS, 608
- bilateral filter, 353
- bilinear model, 534
  - asymmetric, 518–524
  - symmetric, 524–528
- binary classification, 88–91, 171–196
- binomial distribution, 45
- bivariate distribution, 69
- block diagonal matrix, 626
- blurring, 328
- body pose estimation, 143, 166, 207–208, 271–272
- body tracking, 500
- boosting, 209
  - adaboost, 202
  - jointboost, 203
  - logitboost, 193–194
- bottom-up approach, 461
- branching logistic regression, 194–196
- Brownian motion, 548
- Broyden Fletcher Goldfarb Shanno, 608
- bundle adjustment, 445–448, 453
- calibration
  - from 3D object, 368, 375–376
  - from a plane, 405–406, 420
- calibration target, 375, 405
  - 3D, 368
  - planar, 405
- camera
  - geometry, 385
  - orthographic, 387
  - other camera models, 385
  - orthographic, 455
  - parameters, 365
  - pinhole, 359–366
    - in Cartesian coordinates, 364–365
    - in homogeneous coordinates, 372–373
  - projective, 359
  - weak perspective, 387
- camera calibration
  - from 3D object, 368, 375–376
  - from a plane, 405–406, 420
- Canny edge detector, 336–338, 464
- canonical correlation analysis, 519
- capacity, 285
- cascade structured classifier, 203
- categorical distribution, 35, 38
  - Bayesian fitting, 62–63
  - conjugate prior, 42, 46
  - fitting, 60–63
  - MAP fitting, 61
- ML fitting, 60–61
- relation to multinomial, 45
- chain model, 243–251, 254–262, 537
  - directed, 244–245
  - learning, 262
  - MAP inference, 246
  - marginal posterior inference, 254
  - sum product algorithm in, 259
  - undirected, 245
- changing face pose, 137
- Chapman–Kolmogorov equation, 539
- checkerboard, 405
- class conditional density function, 91, 102
- classification, 83, 171–201, 209
  - adaboost, 200
  - applications of, 201–209
  - Bayesian logistic regression, 176–180
  - binary, 88–91, 171–196
  - boosting, 193–194
  - cascade structure, 203
  - dual logistic regression, 183–185
  - fern, 199
  - gender, 201
  - kernel logistic regression, 185–186
  - logistic regression, 89, 171–175
  - multi-class, 197–198
  - multi-layer perceptron, 200
  - non-probabilistic models, 200–201
  - nonlinear logistic regression, 181
  - one-against-all, 197
  - random classification tree, 198–200
  - random forest, 200
  - relevance vector, 186–190
  - support vector machine, 200
  - tree, 194–196, 209
  - weak classifier, 194
- clique, 223, 281
  - maximal, 224
- closed set face identification, 512
- clustering, 113, 349–351
- coarse-to-fine approach, 309, 480
- collinearity, 395, *see* homography
- color model, 93, 95, 305
- combining variables, 265
- concave function, 176
- condensation algorithm, 558–562
  - for tracking contour, 565
- condition number, 620
- conditional independence, 217
  - in a directed model, 220
  - in an undirected model, 224
- conditional probability distribution, 28
  - of multivariate normal, 73
- conditional random field, 316
  - 1D, 263
  - 2D, 300–302
- conic, 387, 421, 462
- conjugacy, 42
  - Bernoulli/beta, 46

- categorical/Dirichlet distribution, 46
- normal/normal inverse Wishart, 47
- normal/normal-scaled inverse gamma, 47
- self-conjugacy of normal, 75
- conjugate gradient method, 608
- constellation model, 585–588
- constraint edge, 294, 316, 319
- continuous random variable, 25
- contour model, 463–468
- contour tracking, 537, 565
- contrastive divergence, 236–237
  - persistent, 237
- convex function, 176, 602
- convex potentials, 296, 297
- corner detection, 336, 339–341, 352
  - Harris corner detector, 339
  - SIFT, 339–341
- cost function, 601
- covariance, 32
- covariance matrix, 41, 69
  - diagonal, 69
  - full, 69
  - spherical, 69
- CRF, 316
  - 1D, 263
  - 2D, 300–302
- cross product, 614
- cross-ratio, 422
- cut on a graph, 285
  - cost, 285
  - minimum, 286
- damped Newton, 608
- data association, 470, 560
- decision boundary, 98, 172
- Delaunay triangulation, 443
- delta function, 600
- denoising, 279, 282–284
  - binary, 291
  - multi-label, 297, 300
- dense stereo vision, 267–270, 307–308, 443
- depth from structured light, 378–380
- derivative filter, 329
- descriptor, 341–345, 352
  - bag of words, 344–345
  - histogram, 341–342
  - HOG, 343–344
  - SIFT, 342–343
- determinant of matrix, 615
- diagonal covariance matrix, 69
- diagonal matrix, 614
  - inverting, 626
- dictionary of visual words, 344, 571
- difference of Gaussians, 331
- digits
  - modeling, 138
- dimensionality reduction, 345–352
  - dual PCA, 349
  - K-means, 349–351
- PCA, 348
- direct linear transformation algorithm, 400–402
- direct search method, 609
- directed graphical model, 219–222
  - chain, 244–245
  - comparison to undirected, 225
  - establishing conditional independence relations in, 220
  - for grids, 304
  - learning, 234–235
  - Markov blanket, 220
  - sampling, 232
- Dirichlet distribution, 35, 39
- discrete random variable, 25
- discriminative model, 84–85
  - classification, 171–201
  - regression, 143–169
- disparity, 268, 307
- displacement expert, 167
- distance transform, 464
- distribution
  - Bernoulli, 35–37
  - beta, 35, 37–38
  - binomial, 45
  - categorical, 35, 38
  - conjugate, 42
  - Dirichlet, 35, 39
  - gamma, 117
  - multinomial, 45
  - multivariate normal, 41–42, 69–76
  - normal, 35
  - normal inverse Wishart, 35, 42
  - normal-scaled inverse gamma, 35, 40
  - probability, 35–45
  - t-distribution, 115–120
  - univariate normal, 40
- DLT algorithm, 400–402
- dolly zoom, 385
- domain of a random variable, 35
- dot product, 613
- dual
  - linear regression, 161–163
  - logistic regression, 183–185
  - parameterization, 161, 184
  - PCA, 349
- dynamic programming, 248, 274
  - for stereo vision, 274
  - in a chain, 248–251
  - in a loop, 278
  - in a tree, 251–254
- E-step, 106, 128, 130–132
- edge detection, 336, 352
  - Canny, 336–338, 464
- edge filter, 329
- eight-point algorithm, 433–435
- EKF, 550–554
- EM algorithm, 106–108, 127–132
  - E-step, 107, 128, 130–132

- for factor analyzer, 124–126
- for mixture of Gaussians, 110–115
- for t distribution, 117–120
- lower bound, 129
- M-step, 107, 132
- empirical max-marginals, 231
- energy minimization, 223
- epipolar constraint, 424
- epipolar geometry, 424
- epipolar line, 424
  - computing, 429
- epipole, 425–426
  - computing, 429
- essential matrix, 427–429, 453
  - decomposition, 430–431
  - properties, 428–429
- estimating parameters, 49–65
- Euclidean transformation, 389–392
  - learning, 398
- evidence, 30, 65
  - framework, 66
- expectation, 31–32
- expectation maximization, 106–108, 127–132, 140
  - E-step, 107, 128, 130–132
  - for factor analyzer, 124–126
  - for mixture of Gaussians, 110–115
  - for t-distribution, 117–120
  - lower bound, 129
  - M-step, 107, 132
- expectation step, 106, 107, 128, 130–132
- expert, 195
- exponential family, 45
- extended Kalman filter, 550–554
- exterior orientation problem, 373, 385
  - 3D scene, 367, 373–375
  - planar scene, 403–405
- extrinsic parameters, 365
  - estimation, 385
  - learning
    - 3D scene, 367, 373–375
    - planar scene, 403–405
- face
  - clustering, 512, 529
  - detection, 101, 102, 133–134, 140, 202–203, 210
  - recognition, 136, 503, 517, 528–530, 533
    - across pose, 137
    - as model comparison, 510–514
    - closed set identification, 512
    - open set identification, 512
  - synthesis, 314–315
  - verification, 503
- face model
  - 3D morphable, 494–496
- facial features
  - aligning, 533
  - finding, 271, 274
- factor analysis, 120, 140, 503
  - as a marginalization, 122
  - learning, 124–126
  - mixture of factor analyzers, 126
  - probability density function, 121
- factor graph, 240, 257, 275
- factorization, 445
  - of a probability distribution, 219, 223
  - Tomasi-Kanade, 453, 455
- feature, 453
  - tracking, 453
- feature descriptor, 341–345
  - bag of words, 344–345
  - histogram, 341–342
  - HOG, 343–344
  - SIFT, 342–343
- feature detector, 336
  - Canny edge detector, 336–338
  - Harris corner detector, 339
  - SIFT detector, 339–341
- fern, 199
- field of view, 363
- filter, 327
  - bilateral, 353
  - derivative, 329
  - different of Gaussian, 331
  - edge, 329
  - Gabor, 331
  - Haar, 331
  - Laplacian, 329
  - Laplacian of Gaussian, 329
  - Prewitt, 329
  - Sobel, 329
- fitting probability models, 49–65
- fixed interval smoothing, 547–548
- fixed lag smoothing, 546–547
- flow
  - optical, 308
  - through graph, 285
- focal length, 360
  - parameter, 362
- forest, 200, 207
- forward-backward algorithm, 255–257
- Frobenius norm, 625
- frustum, 416
- full covariance matrix, 69
- fundamental matrix, 432, 453
  - decomposition, 441
  - estimation, 432–435
  - relation to essential matrix, 432
- Gabor energy, 331
- Gabor filter, 331
- gallery face, 512
- gamma distribution, 117
- gamma function, 37
- gating function, 195
- Gauss-Newton method, 606–607
- Gaussian distribution, *see* normal distribution

- Gaussian Markov random field, 318  
Gaussian process  
classification, 186  
latent variable model, 487–491, 518  
multi-factor, 531–532  
regression, 156, 169  
gender classification, 171, 201, 209  
generalized Procrustes analysis, 472–473  
generative model, 84, 85  
comparison to discriminative model, 91  
geodesic distance, 307  
geometric invariants, 422  
geometric transformation model, 389–415  
2D, 389–396  
application, 415–418  
learning, 396  
gesture tracking, 243, 267  
Gibbs distribution, 223, 280  
Gibbs sampling, 233  
GPLVM, 487–491, 518  
multi-factor, 531–532  
GrabCut, 305–307  
gradient vector, 175, 604  
graph cuts, 284–300, 316  
alpha-expansion, 298–300  
applications of, 304–309  
binary variables, 286–291  
efficient reuse of solution, 316  
multi-label, 293–300  
reparameterization, 290–291  
volumetric, 450–452  
graphical model, 217–239  
applications in computer vision, 227  
chain, 243, 537  
directed, 219–222  
learning, 234–235  
sampling, 232  
directed vs. undirected, 225  
factor graph, 240  
grid-based, 263  
plate notation, 222  
tree, 243  
undirected, 223  
learning, 235–238  
sampling, 233  
Gray codes, 380  
grid-based model, 263, 279–318  
applications, 316  
directed, 304  
  
Haar-like filter, 203, 331  
hand model, 492, 493, 500  
Harris corner detector, 339  
head position  
tracking, 167  
Heaviside step function, 181, 193, 600  
Hessian matrix, 175, 602  
hidden layer, 200  
hidden Markov model, 227, 245, 246, 267, 274  
hidden variable, 104, 105  
representing transformations, 138  
higher order cliques, 303, 317  
Hinton diagram, 25  
histogram equalization, 326  
histogram of oriented gradients, 343–344, 352  
histogram, RGB, 341  
HMM, 227, 245, 246, 274  
HOG descriptor, 343–344, 352  
homogeneous coordinates, 371  
homography, 395–396  
learning, 400–402  
properties, 407–409  
human part identification, 207–208  
human performance capture, 382, 385  
human pose estimation, 271–272  
hyperparameter, 36  
hysteresis thresholding, 338  
  
ICP, 470  
ideal point, 370  
identity, 503  
identity / style model, 503  
asymmetric bilinear, 518–524  
multi-factor GPLVM, 531–532  
multi-linear, 528  
nonlinear, 517–518  
PLDA, 514–517  
subspace identity model, 506–514  
symmetric bilinear, 524–528  
identity matrix, 614  
image denoising, 279, 282–284  
binary, 291  
multi-label, 300  
image descriptor, 352  
image plane, 359, 360  
image processing, 325–345, 352  
image quilting, 311–314  
image retargeting, 308–309  
image structure tensor, 339  
importance sampling, 562  
incremental fitting  
of logistic regression, 190–193  
independence, 31  
conditional, 217  
inference, 84  
algorithm, 84  
empirical max-marginals, 231  
in graphical models with loops, 265  
MAP solution, 230  
marginal posterior distribution, 230  
maximum marginals, 231  
sampling from posterior, 231  
innovation, 543  
integral image, 332  
intensity normalization, 325  
interactive segmentation, 305–307, 317  
interest point detection, 336, 352  
Harris corner detector, 339

- SIFT, 339–341
- intersection of two lines, 387
- intrinsic matrix, 365
- intrinsic parameters, 365
  - learning
    - from 3D object, 368, 375–376
    - from a plane, 405–406
  - invariant
    - geometric, 422
  - inverse of a matrix, 615, 620–621
    - computing for large matrices, 626
- Ishikawa construction, 316
- iterated extended Kalman filter, 552
- iterative closest point, 470
- Jensen’s inequality, 130
- joint probability, 26
- jointboost, 203
- junction tree algorithm, 265, 266
- K-means algorithm, 113, 349–351
- Kalman filter, 229, 540–548
  - temporal and measurement models, 548
  - derivation, 541
  - extended, 550–554
    - iterated extended, 552
    - recursions, 543–544
    - smoothing, 546–548
    - unscented, 554–558
  - Kalman gain, 542
  - Kalman smoothing, 546–548
  - kernel function, 155–156, 185
  - kernel logistic regression, 185–186
  - kernel PCA, 349, 352
  - kernel trick, 155
  - Kinect, 207
  - kinematic chain, 492
  - Kullback-Leibler divergence, 131
  - landmark point, 463
  - landscape matrix, 614
  - Laplace approximation, 178, 179, 188
  - Laplacian filter, 329
  - Laplacian of Gaussian filter, 329
  - latent Dirichlet allocation, 576–581, 595
    - learning, 578–581
  - latent variable, 104, 105
  - LDA (latent Dirichlet allocation), 576–581, 595
    - learning, 578–581
  - LDA (linear discriminant analysis), 533
  - learning, 49, 84
    - Bayesian approach, 50–51
    - in chains and trees, 262
    - in directed models, 234–235
    - in undirected models, 235–238
    - least squares, 54
    - maximum a posteriori, 50
    - maximum likelihood, 49
    - learning algorithm, 84
  - least median of squares regression, 420
  - least squares, 54
    - solving least squares problems, 623
  - likelihood, 30
  - line, 387, 421
    - epipolar, 424
    - joining two points, 387
  - line search, 608
  - linear algebra, 613–628
    - common problems, 623–626
  - linear discriminant analysis, 533
  - linear regression, 143–145
    - Bayesian approach, 147–150
    - limitations of, 145–146
  - linear subspace, 121
  - linear transform, 617
  - local binary pattern, 333, 352
  - local maximum / minimum, 176, 602
  - log likelihood, 53
  - logistic classification tree, 194–196
  - logistic regression, 89, 171–175
    - Bayesian approach, 176–180
    - branching, 194–196
    - dual, 183–185
    - kernel, 185–186
    - multi-class, 197–198
    - nonlinear, 181
  - logistic sigmoid function, 171
  - logitboost, 193–194
  - loopy belief propagation, 266
    - applications, 275
  - M-estimator, 420
  - M-step, 106, 129, 132
  - magnitude of vector, 613
  - manifold, 346
  - MAP estimation, 50
  - marginal distribution, 27
    - of multivariate normal, 72
  - marginal posterior distribution, 230
  - marginalization, 27
  - Markov assumption, 244, 537
  - Markov blanket, 220, 224
    - in a directed model, 220
    - in an undirected model, 224
  - Markov chain Monte Carlo, 233
  - Markov network
    - learning, 235–238
    - sampling, 233
  - Markov random field, 224, 227, 279, 280, 316
    - Gaussian, 318
    - applications, 304–309, 316
    - higher order, 303, 317
    - pairwise, 281
  - Markov tree, 227
  - matrix, 614
    - block diagonal, 626
    - calculus, 621–623
    - condition number, 620

- determinant, 615
- diagonal, 614
- Frobenious norm, 625
- identity, 614
- inverse, 615, 620–621
- inverting large, 626
- landscape, 614
- multiplication, 614
- null space, 616, 620
- orthogonal, 616
- portrait, 614
- positive definite, 616
- rank, 620
- rotation, 616
- singular, 615
- square, 614
- trace, 615
- transpose, 615
- matrix determinant lemma, 628
- matrix inversion lemma, 148, 628
- max flow, 285
  - algorithms, 316
  - augmenting paths algorithm, 286
- max function, 600
- maximal clique, 224
- maximization step, 106, 107, 129, 132
- maximum a posteriori estimation, 50
- maximum likelihood estimation, 49
- maximum marginals, 231
- MCMC, 233
- measurement incorporation step, 539
- measurement model, 537
- Mercer's theorem, 155
- min cut, 285
- min function, 600
- minimum direction problem, 374, 624
- mixture model
  - mixture of experts, 211
  - mixture of factor analyzers, 126, 140
  - mixture of Gaussians, 108–115, 140
  - mixture of PLDAs, 518
  - mixture of t-distributions, 126, 140
  - robust, 126
- ML estimation, 49
- model, 84
  - discriminative, 84–85
  - generative, 84, 85
- model comparison, 66
- model selection, 511
- moment, 31–32
  - about mean, 32
  - about zero, 32
- MonoSLAM, 564
- morphable model, 494–496
- mosaic, 417, 420
- MRF, 224, 227, 279, 280, 316
  - applications, 304–309, 316
  - Gaussian, 318
  - higher order, 303, 317
- pairwise, 281
- multi-class classification, 197–198
  - multi-class logistic regression, 197–198
  - random classification tree, 198–200
- multi-factor GPLVM, 531–532
- multi-factor model, 528
- multi-layer perceptron, 200, 209
- multi-linear model, 528, 534
- multi-view geometry, 453
- multi-view reconstruction, 369–370, 376–377, 443, 450–453
- multinomial distribution, 45
- multiple view geometry, 423
- multivariate normal distribution, 35, 69–76
- naïve Bayes, 94
- neural network, 200
- Newton method, 176, 605–606
- non-convex potentials, 297
- non-stationary model, 545
- nonlinear identity model, 517–518
- nonlinear logistic regression, 181
- nonlinear optimization, 601–611
  - BFGS, 608
  - Broyden Fletcher Goldfarb Shanno, 608
  - conjugate gradient method, 608
  - Gauss-Newton method, 606–607
  - line search, 608
  - Newton method, 605–606
  - over positive definite matrices, 611
  - over rotation matrices, 610
  - quasi-Newton methods, 608
  - reparameterization, 609
  - steepest descent, 603–604
  - trust-region methods, 608
- nonlinear regression, 150
  - Bayesian, 153
- norm of vector, 613
- normal distribution, 35, 69–76
  - Bayesian fitting, 56
  - change of variable, 75
  - conditional distribution, 73
  - covariance decomposition, 71
  - MAP fitting, 54
  - marginal distribution, 72
  - ML fitting, 51
  - multivariate, 35, 41–42
  - product of two normals, 74, 78
  - self-conjugacy, 75
  - transformation of variable, 72
  - univariate, 35, 40
- normal inverse Wishart distribution, 35, 42
- normal-scaled inverse gamma distribution, 35, 40
- normalized camera, 361
- normalized image coordinates, 373
- null space, 616, 620
- object recognition, 134–135, 140, 571–592

- unsupervised, 581
- objective function, 601
- offset parameter, 363
- one-against-all classifier, 197
- open-set face identification, 512
- optical axis, 360
- optical center, 359
- optical flow, 308
- optimization, 601–611
  - BFGS, 608
  - Broyden Fletcher Goldfarb Shanno, 608
  - conjugate gradient method, 608
  - Gauss-Newton method, 606–607
  - line search, 608
  - Newton method, 605–606
  - over positive definite matrix, 611
  - over rotation matrix, 610
  - quasi-Newton methods, 608
  - reparameterization, 609
  - steepest descent, 603–604
  - trust-region methods, 608
- orthogonal matrix, 616
- orthogonal Procrustes problem, 374, 398, 625
- orthogonal vectors, 613
- orthographic camera, 387, 455
- outlier, 115, 411
- pairwise MRF, 281
- pairwise term, 248, 284
- panorama, 417, 420
- parametric contour model, 463–468
- part of object, 577
- particle filtering, 558–562
- partition function, 223
- PCA, 348
  - dual, 349
  - kernel, 349, 352
  - probabilistic, 122, 476–479
- PDF, 25
- PEarl algorithm, 415, 420
- pedestrian detection, 202–203
- pedestrian tracking, 563–564
- per-pixel image processing, 325
- persistent contrastive divergence, 237
- perspective projection, 361
- perspective-n-point problem, 367, 385
- Phong shading model, 494
- Photo-tourism, 449–450
- photoreceptor spacing, 362
- pictorial structure, 270, 274
- pinhole, 359
- pinhole camera, 359–366, 423
  - in Cartesian coordinates, 364–365
  - in homogeneous coordinates, 372–373
- plate, 222
- PLDA, 514–517
- PnP problem, 367, 385
- point distribution model, 471–482
- point estimate, 50
- point operator, 325
- polar rectification, 443
- portrait matrix, 614
- pose estimation, 420
- positive definite matrix, 616
  - optimization over, 611
- posterior distribution, 30
- potential function, 223
- potentials
  - convex, 296
  - non-convex, 297
- Potts model, 298, 319
- PPCA, 476–479
  - learning parameters, 477
- prediction step, 539
- predictive distribution, 49
- preprocessing, 133, 325–351
- Prewitt operators, 329
- principal component analysis, 348
  - dual PCA, 349
  - probabilistic, 122, 476–479
- principal direction problem, 624
- principal point, 360
- prior, 30
- probabilistic latent semantic analysis, 595
- probabilistic linear discriminant analysis, 514–517
- probabilistic principal component analysis, 122, 140, 476–479
  - learning parameters, 477
- probability
  - conditional, 28
  - joint, 26
  - marginal, 27
- probability density function, 25
- probability distribution, 35–45
  - fitting, 49–65
- probe face, 512
- Procrustes analysis
  - generalized, 472–473
- Procrustes problem, 374, 625
- product of experts, 223
- projective camera, 359
- projective pinhole camera, 359
- projective reconstruction, 377
- projective transformation, 395–396
  - fitting, 400–402
  - properties, 407–409
- propose, expand and re-learn, 415, 420
- prototype vector, 349
- pruning graphical models, 265, 270
- quadri-focal tensor, 444
- quadric, 492
  - truncated, 493
- Quasi-Newton methods, 608
- quaternion, 610
- radial basis function, 151, 191

- radial distortion, 365  
random classification tree, 198–200  
random forest, 200  
random sample consensus, 411–413, 420  
  sequential, 413–414  
random variable, 25  
  continuous, 25  
  discrete, 25  
  domain of, 35  
rank of matrix, 620  
RANSAC, 411–413, 420  
  sequential, 413–414  
Rao-Blackwellization, 562  
reconstruction, 359, 369–370, 376–377  
  from structured light, 378–380  
  multi-view, 443, 453  
  projective, 377  
  two view, 435  
reconstruction error, 346  
reconstruction pipeline, 447–449, 453  
rectification, 267, 439, 453  
  planar, 439  
  polar, 443  
region descriptor, 341–344  
  bag of words, 344–345  
  histogram, 341–342  
  HOG, 343–344  
  SIFT, 342–343  
regression, 83, 143–169  
  Bayesian linear, 147–150  
  dual, 161–163  
  Gaussian process, 156, 169  
  linear, 86, 143–145  
    limitations of, 145–146  
  nonlinear, 150  
  nonlinear, Bayesian, 153  
  polynomial, 150  
  relevance vector, 163–165  
  sparse, 157  
  to multivariate data, 165  
relative orientation, 431, 453  
relevance vector  
  classification, 186–190  
  regression, 163–165  
reparameterization  
  for optimization, 609  
  in graph cuts, 290–291  
  multi-label case, 296  
reprojection error, 424  
resection-intersection, 446  
responsibility, 111  
robust density modeling, 115–120  
robust learning, 410, 420  
  PEaRL, 415  
  RANSAC, 411–413  
  sequential RANSAC, 413–414  
robust mixture model, 126  
robust subspace model, 126  
rotation matrix, 616  
  optimization over, 610  
  rotation of camera, 408  
sampling  
  ancestral, 232  
  directed models, 232  
  Gibbs, 233  
  undirected models, 233  
sampling from posterior, 231  
scalar product, 613  
scale invariant feature transform, 339–343  
SCAPE, 496–497  
scene model, 590  
scene recognition, 571  
Schur complement, 627  
segmentation, 135–136, 140, 272, 461, 463–468  
  supervised, 305–307  
semantic segmentation, 203–205, 210  
sequential RANSAC, 413–414  
seven point algorithm, 436  
shape, 461  
  alignment, 472–473  
  definition, 462  
  statistical model, 471–482  
shape and appearance models, 482–487  
shape context descriptor, 166, 345  
shape from silhouette, 380–383, 385  
shape model  
  3D, 482  
  articulated, 492–493  
  non-Gaussian, 487–491  
  subspace, 475  
shape template, 468, 469  
Sherman-Morrison-Woodbury relation, 148, 628  
shift map image editing, 308–309  
SIFT, 416  
  descriptor, 342–343, 352  
  detector, 339–341  
sign language interpretation, 243, 246, 267  
silhouette  
  shape from, 380–383  
similarity transformation, 392  
  learning, 399  
simultaneous localization and mapping, 564, 567  
single author-topic model, 582–585  
singular matrix, 615  
singular value decomposition, 618–620  
singular values, 619  
skew (camera parameter), 364  
skew (moment), 32  
skin detection, 93–94, 97  
SLAM, 564, 567  
smoothing, 546–548  
  fixed interval, 547–548  
  fixed lag, 546–547  
snake, 272–273, 275, 463–468, 499  
Sobel operator, 329  
softmax function, 197  
sparse classification model, 186–190

- sparse linear regression, 157  
 sparse stereo vision, 359  
 sparsity, 157, 164, 187, 190  
 spherical covariance matrix, 69  
 square matrix, 614  
 squared reprojection error, 424  
 statistical shape model, 471–482  
 steepest descent, 603–604  
 step function, 193  
 stereo reconstruction, 369–370, 376–377  
 stereo vision, 267–270, 274, 307–308, 317  
     dense, 267–270  
     dynamic programming, 274  
     graph cuts formulation, 307–308  
     sparse, 359  
 strong classifier, 194  
 structure from motion, 423, 444  
 structured light, 379, 385  
 Student t-distribution, 115–120  
 style, 503  
 style / identity model, 503  
     asymmetric bilinear, 518–524  
     multi-factor GPLVM, 531–532  
     multi-linear, 528  
     nonlinear, 517–518  
     PLDA, 514–517  
     subspace identity model, 506–514  
     symmetric bilinear, 524–528  
 style translation, 524  
 submodularity, 291, 296  
     multi-label case, 296  
 subspace, 121  
 subspace identity model, 506–514  
 subspace model, 120, 140, 475, 499, 503  
     bilinear asymmetric, 518–524  
     bilinear symmetric, 524–528  
     dual PCA, 349  
     factor analysis, 503  
     for face recognition, 533  
     multi-factor GPLVM, 531–532  
     multi-linear model, 528  
     PLDA, 514–517  
     principal component analysis, 348  
     subspace identity model, 506–514  
     subspace shape model, 475  
 sum-product algorithm, 257–259, 275  
     for chain model, 259  
     for tree model, 262  
 super-resolution, 310–311  
 superpixel, 205  
 supervised segmentation, 305–307  
 support vector machine, 200, 209  
 surface layout recovery, 205–206  
 SVD, 618–620  
 SVM, 200  
 symmetric bilinear model, 524–528  
 symmetric epipolar distance, 433  
 t-distribution, 115–120, 140  
     mixture of, 126  
     multivariate, 116  
     univariate, 116  
 t-test, 66  
 temporal model, 537–568  
 tensor, 617  
     multiplication, 617  
 TensorTextures, 530–531  
 texton, 204, 334  
 textonboost, 203–205  
 texture synthesis, 311–314, 317  
 tied factor analysis, 519  
 Tomasi-Kanade factorization, 445, 453, 455  
 top-down approach, 461  
 topic, 576  
 trace of matrix, 615  
 tracking, 537–568  
     pedestrian, 563–564  
     applications, 567  
     condensation algorithm, 558–562  
     displacement expert, 167  
     features, 453  
     for augmented reality, 416–417  
     head position, 167  
     particle filtering, 558–562  
     through clutter, 565  
 transformation, 389–415, 420  
     2D, 389–396  
     affine, 393–394  
     application, 415–418  
     between images, 407  
     Euclidean, 389–392  
     homography, 395–396  
     indexed by hidden variable, 138  
     inference, 401  
     inverting, 401  
     learning, 396  
         affine, 399–400  
         Euclidean, 398  
         homography, 400–402  
         projective, 400–402  
         similarity, 399  
     linear, 617  
     projective, 395–396  
     robust learning, 410  
     similarity, 392  
 transpose, 615  
 tree model, 243  
     learning, 262  
     MAP inference, 251–254  
     marginal posterior inference, 262  
 tri-focal tensor, 444  
 triangulation, 370  
 truncating potentials, 300  
 trust-region methods, 608  
 two-view geometry, 424  
 UKF, 554–558  
 unary term, 284, 304

undirected graphical model, 223  
chain, 245  
conditional independence relations in, 224  
learning, 235–238  
Markov blanket, 224  
sampling, 233  
univariate normal distribution, 35  
unscented Kalman filter, 554–558  
unsupervised object discovery, 581  
  
variable elimination, 255  
variance, 32  
vector, 613  
    norm, 613  
    product, 614  
Vertigo, 385  
Video Google, 591–592  
virtual image, 359  
visual hull, 381  
visual word, 344, 571  
Viterbi algorithm, 248–251  
volumetric graph cuts, 450–452  
  
weak classifier, 194  
weak perspective camera, 387  
whitening, 325  
whitening transform, 77  
within-individual variation, 507, 514  
Woodbury inversion identity, 148, 628  
word, 344, 571  
world state, 83