Chapter 14

Recognition

14.1 Object detection . . . 578
  14.1.1 Face detection . . . 578
  14.1.2 Pedestrian detection . . . 585
14.2 Face recognition . . . 588
  14.2.1 Eigenfaces . . . 589
  14.2.2 Active appearance and 3D shape models . . . 596
  14.2.3 *Application*: Personal photo collections . . . 601
14.3 Instance recognition . . . 602
  14.3.1 Geometric alignment . . . 603
  14.3.2 Large databases . . . 604
  14.3.3 *Application*: Location recognition . . . 609
14.4 Category recognition . . . 611
  14.4.1 Bag of words . . . 612
  14.4.2 Part-based models . . . 615
  14.4.3 Recognition with segmentation . . . 620
  14.4.4 *Application*: Intelligent photo editing . . . 621
14.5 Context and scene understanding . . . 625
  14.5.1 Learning and large image collections . . . 627
  14.5.2 *Application*: Image search . . . 630
14.6 Recognition databases and test sets . . . 631
14.7 Additional reading . . . 631
14.8 Exercises . . . 637

DOI 10.1007/978-1-84882-935-0_14, © Springer-Verlag London Limited 2011

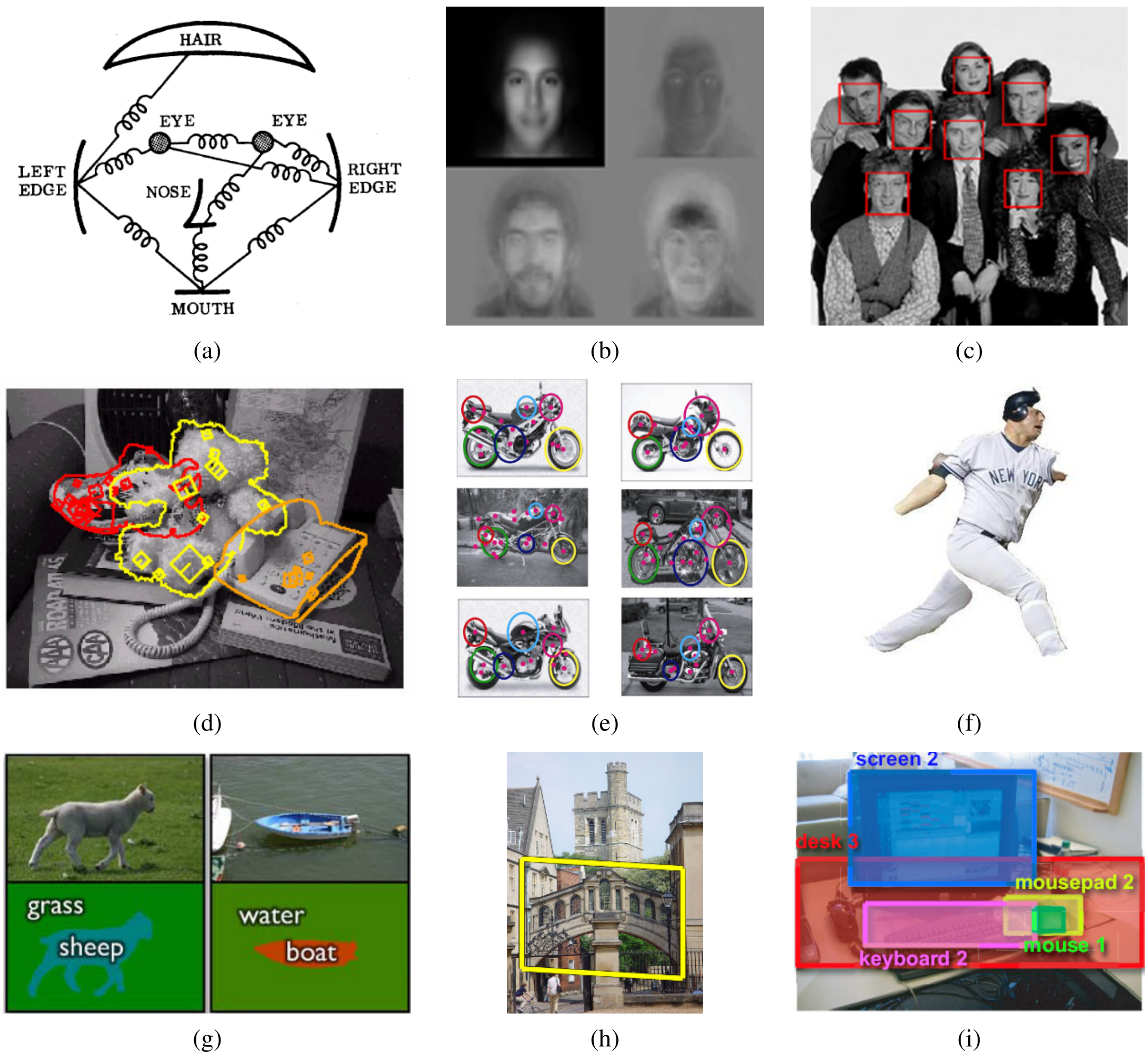


Figure 14.1 Recognition: face recognition with (a) pictorial structures (Fischler and Elschlager 1973) © 1973 IEEE and (b) eigenfaces (Turk and Pentland 1991b); (c) real-time face detection (Viola and Jones 2004) © 2004 Springer; (d) instance (known object) recognition (Lowe 1999) © 1999 IEEE; (e) feature-based recognition (Fergus, Perona, and Zisserman 2007); (f) region-based recognition (Mori, Ren, Efros *et al.* 2004) © 2004 IEEE; (g) simultaneous recognition and segmentation (Shotton, Winn, Rother *et al.* 2009) © 2009 Springer; (h) location recognition (Philbin, Chum, Isard *et al.* 2007) © 2007 IEEE; (i) using context (Russell, Torralba, Liu *et al.* 2007).

Of all the visual tasks we might ask a computer to perform, analyzing a scene and recognizing all of the constituent objects remains the most challenging. While computers excel at accurately reconstructing the 3D shape of a scene from images taken from different views, they cannot name all the objects and animals present in a picture, even at the level of a two-year-old child. There is not even any consensus among researchers on when this level of performance might be achieved.

Why is recognition so hard? The real world is made of a jumble of objects, which all occlude one another and appear in different poses. Furthermore, the variability intrinsic within a class (e.g., dogs), due to complex non-rigid articulation and extreme variations in shape and appearance (e.g., between different breeds), makes it unlikely that we can simply perform exhaustive matching against a database of exemplars.[1]

The recognition problem can be broken down along several axes. For example, if we know what we are looking for, the problem is one of *object detection* (Section 14.1), which involves quickly scanning an image to determine where a match may occur (Figure 14.1c). If we have a specific rigid object we are trying to recognize (*instance recognition*, Section 14.3), we can search for characteristic feature points (Section 4.1) and verify that they align in a geometrically plausible way (Section 14.3.1) (Figure 14.1d).

The most challenging version of recognition is general *category* (or *class*) recognition (Section 14.4), which may involve recognizing instances of extremely varied classes such as animals or furniture. Some techniques rely purely on the presence of features (known as a "bag of words" model—see Section 14.4.1), their relative positions (*part-based models* (Section 14.4.2)), Figure 14.1e, while others involve segmenting the image into semantically meaningful regions (Section 14.4.3) (Figure 14.1f). In many instances, recognition depends heavily on the *context* of surrounding objects and scene elements (Section 14.5). Woven into all of these techniques is the topic of *learning* (Section 14.5.1), since hand-crafting specific object recognizers seems like a futile approach given the complexity of the problem.

Given the extremely rich and complex nature of this topic, this chapter is structured to build from simpler concepts to more complex ones. We begin with a discussion of face and object detection (Section 14.1), where we introduce a number of machine-learning techniques such as boosting, neural networks, and support vector machines. Next, we study face recognition (Section 14.2), which is one of the more widely known applications of recognition. This topic serves as an introduction to subspace (PCA) models and Bayesian approaches to recognition and classification. We then present techniques for instance recognition (Section 14.3), building upon earlier topics in this book, such as feature detection, matching, and geometric alignment (Section 14.3.1). We introduce topics from the information and document retrieval communities, such as frequency vectors, feature quantization, and inverted indices (Section 14.3.2). We also present applications of location recognition (Section 14.3.3).

In the second half of the chapter, we address the most challenging variant of recognition, namely the problem of category recognition (Section 14.4). This includes approaches that use bags of features (Section 14.4.1), parts (Section 14.4.2), and segmentation (Section 14.4.3). We show how such techniques can be used to automate photo editing tasks, such as 3D modeling, scene completion, and creating collages (Section 14.4.4). Next, we discuss the role that context can play in both individual object recognition and more holistic scene under-

[1] However, some recent research suggests that direct image matching may be feasible for large enough databases (Russell, Torralba, Liu *et al.* 2007; Malisiewicz and Efros 2008; Torralba, Freeman, and Fergus 2008).

standing (Section 14.5). We close this chapter with a discussion of databases and test sets for constructing and evaluating recognition systems (Section 14.6).

While there is no comprehensive reference on object recognition, an excellent set of notes can be found in the ICCV 2009 short course (Fei-Fei, Fergus, and Torralba 2009), Antonio Torralba's more comprehensive MIT course (Torralba 2008), and two recent collections of papers (Ponce, Hebert, Schmid *et al.* 2006; Dickinson, Leonardis, Schiele *et al.* 2007) and a survey on object categorization (Pinz 2005). An evaluation of some of the best performing recognition algorithms can be found on the PASCAL Visual Object Classes (VOC) Challenge Web site at http://pascallin.ecs.soton.ac.uk/challenges/VOC/.

14.1 Object detection

If we are given an image to analyze, such as the group portrait in Figure 14.2, we could try to apply a recognition algorithm to every possible sub-window in this image. Such algorithms are likely to be both slow and error-prone. Instead, it is more effective to construct special-purpose *detectors*, whose job it is to rapidly find likely regions where particular objects might occur.

We begin this section with face detectors, which are some of the more successful examples of recognition. For example, such algorithms are built into most of today's digital cameras to enhance auto-focus and into video conferencing systems to control pan-tilt heads. We then look at pedestrian detectors, as an example of more general methods for object detection. Such detectors can be used in automotive safety applications, e.g., detecting pedestrians and other cars from moving vehicles (Leibe, Cornelis, Cornelis *et al.* 2007).

14.1.1 Face detection

Before face recognition can be applied to a general image, the locations and sizes of any faces must first be found (Figures 14.1c and 14.2). In principle, we could apply a face recognition algorithm at every pixel and scale (Moghaddam and Pentland 1997) but such a process would be too slow in practice.

Over the years, a wide variety of fast face detection algorithms have been developed. Yang, Kriegman, and Ahuja (2002) provide a comprehensive survey of earlier work in this field; Yang's ICPR 2004 tutorial[2] and the Torralba (2007) short course provide more recent reviews.[3]

According to the taxonomy of Yang, Kriegman, and Ahuja (2002), face detection techniques can be classified as feature-based, template-based, or appearance-based. Feature-based techniques attempt to find the locations of distinctive image features such as the eyes, nose, and mouth, and then verify whether these features are in a plausible geometrical arrangement. These techniques include some of the early approaches to face recognition (Fischler and Elschlager 1973; Kanade 1977; Yuille 1991), as well as more recent approaches based on modular eigenspaces (Moghaddam and Pentland 1997), local filter jets (Leung, Burl, and Perona 1995; Penev and Atick 1996; Wiskott, Fellous, Krüger *et al.* 1997), support

[2] http://vision.ai.uiuc.edu/mhyang/face-detection-survey.html.

[3] An alternative approach to detecting faces is to look for regions of skin color in the image (Forsyth and Fleck 1999; Jones and Rehg 2001). See Exercise 2.8 for some additional discussion and references.

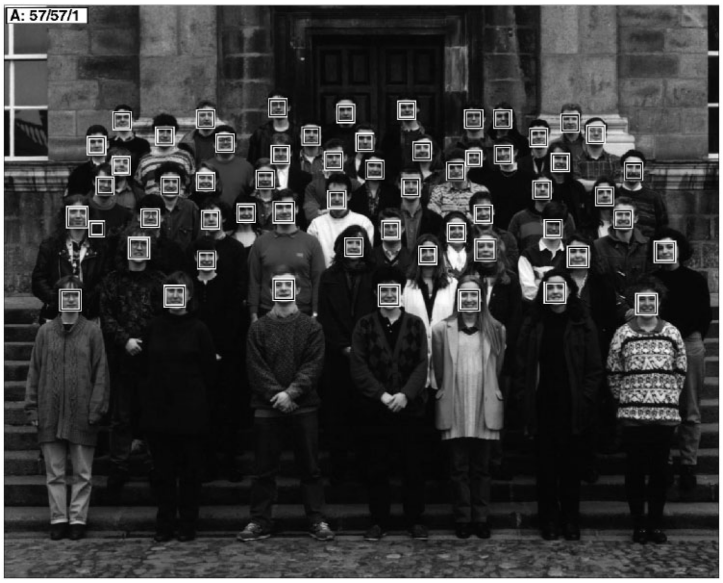


Figure 14.2 Face detection results produced by Rowley, Baluja, and Kanade (1998a) © 1998 IEEE. Can you find the one false positive (a box around a non-face) among the 57 true positive results?

vector machines (Heisele, Ho, Wu *et al.* 2003; Heisele, Serre, and Poggio 2007), and boosting (Schneiderman and Kanade 2004).

Template-based approaches, such as active appearance models (AAMs) (Section 14.2.2), can deal with a wide range of pose and expression variability. Typically, they require good initialization near a real face and are therefore not suitable as fast face detectors.

Appearance-based approaches scan over small overlapping rectangular patches of the image searching for likely face candidates, which can then be refined using a *cascade* of more expensive but selective detection algorithms (Sung and Poggio 1998; Rowley, Baluja, and Kanade 1998a; Romdhani, Torr, Schölkopf *et al.* 2001; Fleuret and Geman 2001; Viola and Jones 2004). In order to deal with scale variation, the image is usually converted into a sub-octave pyramid and a separate scan is performed on each level. Most appearance-based approaches today rely heavily on training classifiers using sets of labeled face and non-face patches.

Sung and Poggio (1998) and Rowley, Baluja, and Kanade (1998a) present two of the earliest appearance-based face detectors and introduce a number of innovations that are widely used in later work by others.

To start with, both systems collect a set of labeled face patches (Figure 14.2) as well as a set of patches taken from images that are known not to contain faces, such as aerial images or vegetation (Figure 14.3b). The collected face images are augmented by artificially mirroring, rotating, scaling, and translating the images by small amounts to make the face detectors less sensitive to such effects (Figure 14.3a).

After an initial set of training images has been collected, some optional pre-processing can be performed, such as subtracting an average gradient (linear function) from the image to compensate for global shading effects and using histogram equalization to compensate for

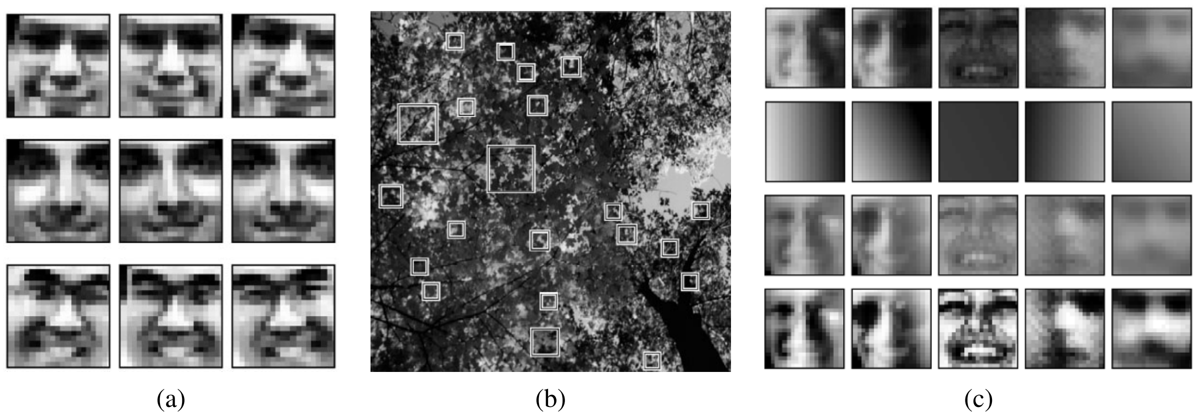


Figure 14.3 Pre-processing stages for face detector training (Rowley, Baluja, and Kanade 1998a) © 1998 IEEE: (a) artificially mirroring, rotating, scaling, and translating training images for greater variability; (b) using images without faces (looking up at a tree) to generate non-face examples; (c) pre-processing the patches by subtracting a best fit linear function (constant gradient) and histogram equalizing.

varying camera contrast (Figure 14.3c).

Clustering and PCA. Once the face and non-face patterns have been pre-processed, Sung and Poggio (1998) cluster each of these datasets into six separate clusters using k-means and then fit PCA subspaces to each of the resulting 12 clusters (Figure 14.4). At detection time, the DIFS and DFFS metrics first developed by Moghaddam and Pentland (1997) (see Figure 14.14 and (14.14)) are used to produce 24 Mahalanobis distance measurements (two per cluster). The resulting 24 measurements are input to a multi-layer perceptron (MLP), which is a neural network with alternating layers of weighted summations and sigmoidal non-linearities trained using the "backpropagation" algorithm (Rumelhart, Hinton, and Williams 1986).

Neural networks. Instead of first clustering the data and computing Mahalanobis distances to the cluster centers, Rowley, Baluja, and Kanade (1998a) apply a neural network (MLP) directly to the 20×20 pixel patches of gray-level intensities, using a variety of differently sized hand-crafted "receptive fields" to capture both large-scale and smaller scale structure (Figure 14.5). The resulting neural network directly outputs the likelihood of a face at the center of every overlapping patch in a multi-resolution pyramid. Since several overlapping patches (in both space and resolution) may fire near a face, an additional merging network is used to merge overlapping detections. The authors also experiment with training several networks and merging their outputs. Figure 14.2 shows a sample result from their face detector.

To make the detector run faster, a separate network operating on 30×30 patches is trained to detect both faces and faces shifted by ± 5 pixels. This network is evaluated at every 10th pixel in the image (horizontally and vertically) and the results of this "coarse" or "sloppy" detector are used to select regions on which to run the slower single-pixel overlap technique. To deal with in-plane rotations of faces, Rowley, Baluja, and Kanade (1998b) train a *router*

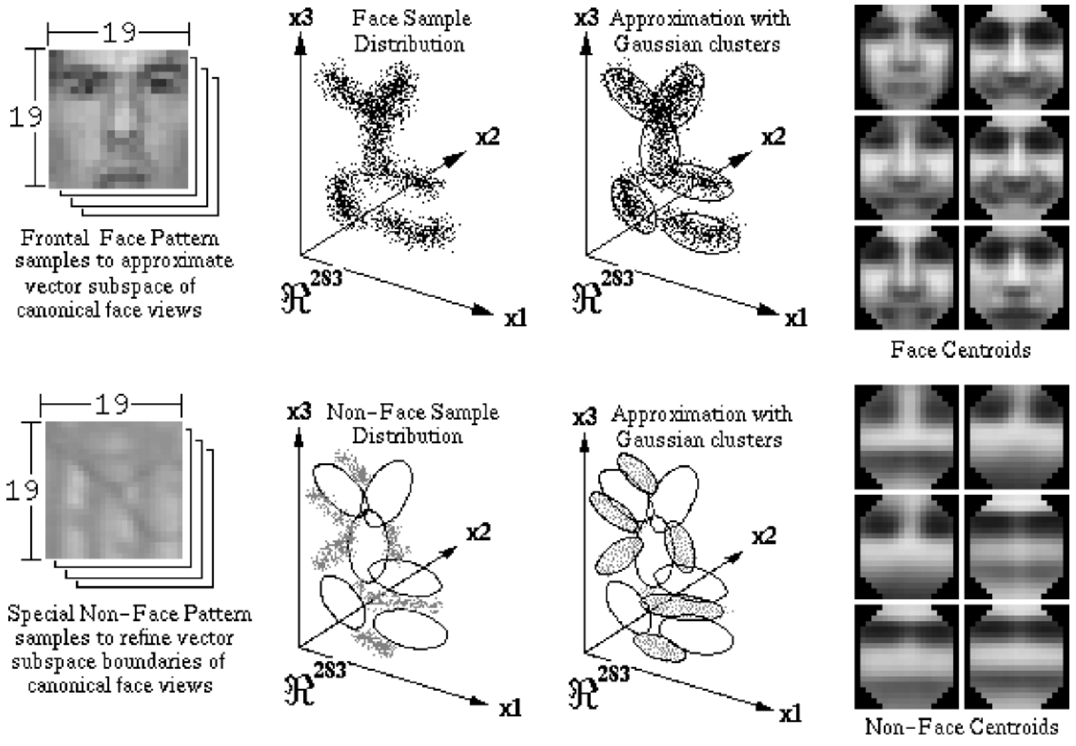

19
19
Frontal Face Pattern samples to approximate vector subspace of canonical face views
x3
Face Sample Distribution
x2
x1
$\Re^{283}$
Approximation with Gaussian clusters
Face Centroids
Special Non-Face Pattern samples to refine vector subspace boundaries of canonical face views
Non-Face Sample Distribution
Non-Face Centroids

Figure 14.4 Learning a mixture of Gaussians model for face detection (Sung and Poggio 1998) © 1998 IEEE. The face and non-face images (19^2-long vectors) are first clustered into six separate clusters (each) using k-means and then analyzed using PCA. The cluster centers are shown in the right-hand columns.

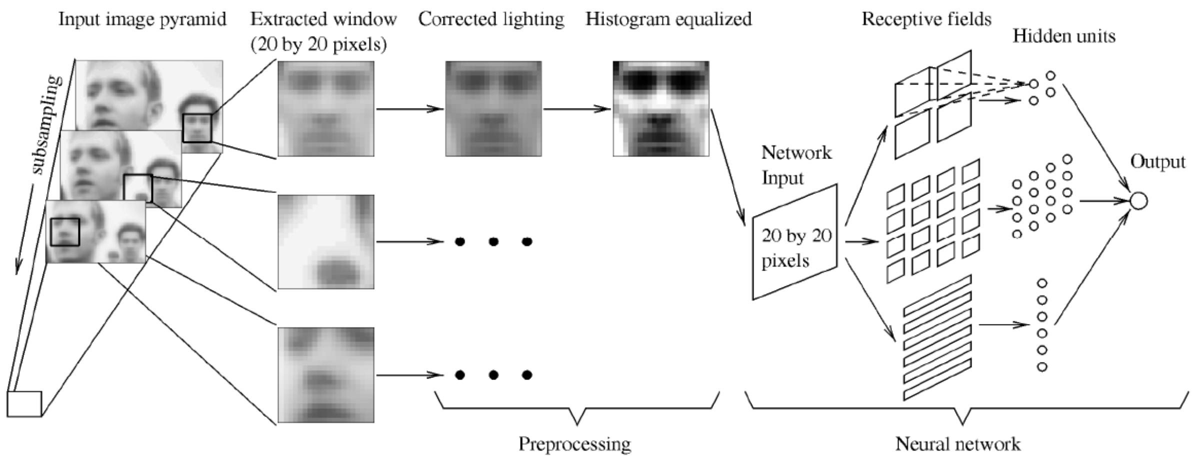


Figure 14.5 A neural network for face detection (Rowley, Baluja, and Kanade 1998a) © 1998 IEEE. Overlapping patches are extracted from different levels of a pyramid and then pre-processed as shown in Figure 14.3b. A three-layer neural network is then used to detect likely face locations.

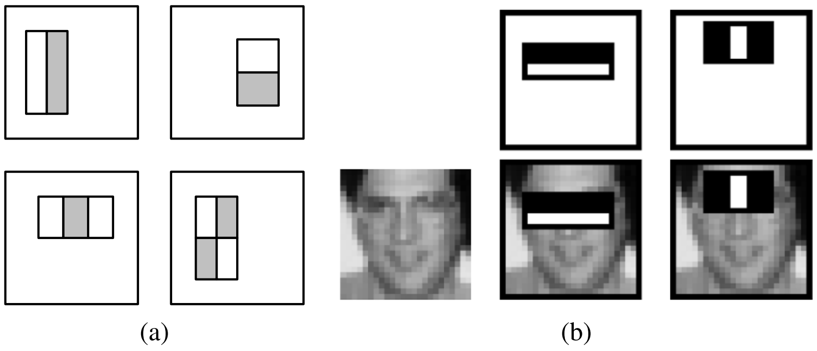


Figure 14.6 Simple features used in boosting-based face detector (Viola and Jones 2004) © 2004 Springer: (a) difference of rectangle feature composed of 2–4 different rectangles (pixels inside the white rectangles are subtracted from the gray ones); (b) the first and second features selected by AdaBoost. The first feature measures the differences in intensity between the eyes and the cheeks, the second one between the eyes and the bridge of the nose.

network to estimate likely rotation angles from input patches and then apply the estimated rotation to each patch before running the result through their upright face detector.

Support vector machines. Instead of using a neural network to classify patches, Osuna, Freund, and Girosi (1997) use a *support vector machine* (SVM) (Hastie, Tibshirani, and Friedman 2001; Schölkopf and Smola 2002; Bishop 2006; Lampert 2008) to classify the same preprocessed patches as Sung and Poggio (1998). An SVM searches for a series of *maximum margin* separating planes in feature space between different classes (in this case, face and non-face patches). In those cases where linear classification boundaries are insufficient, the feature space can be lifted into higher-dimensional features using *kernels* (Hastie, Tibshirani, and Friedman 2001; Schölkopf and Smola 2002; Bishop 2006). SVMs have been used by other researchers for both face detection and face recognition (Heisele, Ho, Wu *et al.* 2003; Heisele, Serre, and Poggio 2007) and are a widely used tool in object recognition in general.

Boosting. Of all the face detectors currently in use, the one introduced by Viola and Jones (2004) is probably the best known and most widely used. Their technique was the first to introduce the concept of *boosting* to the computer vision community, which involves training a series of increasingly discriminating simple classifiers and then blending their outputs (Hastie, Tibshirani, and Friedman 2001; Bishop 2006).

In more detail, boosting involves constructing a *classifier* $h(\boldsymbol{x})$ as a sum of simple *weak learners*,

$$h(\boldsymbol{x}) = \text{sign}\left[\sum_{j=0}^{m-1} \alpha_j h_j(\boldsymbol{x})\right], \tag{14.1}$$

where each of the weak learners $h_j(\boldsymbol{x})$ is an extremely simple function of the input, and hence is not expected to contribute much (in isolation) to the classification performance.

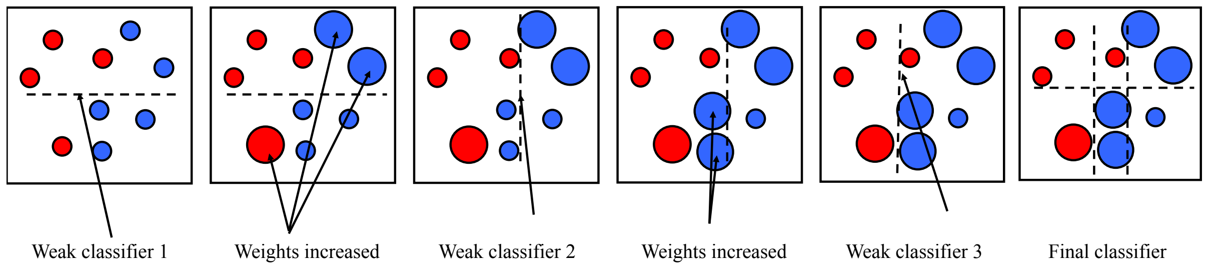


Figure 14.7 Schematic illustration of boosting, courtesy of Svetlana Lazebnik, after original illustrations from Paul Viola and David Lowe. After each weak classifier (decision stump or hyperplane) is selected, data points that are erroneously classified have their weights increased. The final classifier is a linear combination of the simple weak classifiers.

In most variants of boosting, the weak learners are threshold functions,

$$h_j(\boldsymbol{x}) = a_j[f_j < \theta_j] + b_j[f_j \geq \theta_j] = \begin{cases} a_j & \text{if } f_j < \theta_j \\ b_j & \text{otherwise,} \end{cases} \tag{14.2}$$

which are also known as *decision stumps* (basically, the simplest possible version of *decision trees*). In most cases, it is also traditional (and simpler) to set a_j and b_j to ± 1, i.e., $a_j = -s_j$, $b_j = +s_j$, so that only the feature f_j, the threshold value θ_j, and the polarity of the threshold $s_j \in \pm 1$ need to be selected.[4]

In many applications of boosting, the features are simply coordinate axes x_k, i.e., the boosting algorithm selects one of the input vector components as the best one to threshold. In Viola and Jones' face detector, the features are differences of rectangular regions in the input patch, as shown in Figure 14.6. The advantage of using these features is that, while they are more discriminating than single pixels, they are extremely fast to compute once a summed area table has been pre-computed, as described in Section 3.2.3 (3.31–3.32). Essentially, for the cost of an $O(N)$ pre-computation phase (where N is the number of pixels in the image), subsequent differences of rectangles can be computed in $4r$ additions or subtractions, where $r \in \{2, 3, 4\}$ is the number of rectangles in the feature.

The key to the success of boosting is the method for incrementally selecting the weak learners and for re-weighting the training examples after each stage (Figure 14.7). The AdaBoost (Adaptive Boosting) algorithm (Hastie, Tibshirani, and Friedman 2001; Bishop 2006) does this by re-weighting each sample as a function of whether it is correctly classified at each stage, and using the stage-wise average classification error to determine the final weightings α_j among the weak classifiers, as described in Algorithm 14.1. While the resulting classifier is extremely fast in practice, the training time can be quite slow (in the order of weeks), because of the large number of feature (difference of rectangle) hypotheses that need to be examined at each stage.

To further increase the speed of the detector, it is possible to create a *cascade* of classifiers, where each classifier uses a small number of tests (say, a two-term AdaBoost classifier) to reject a large fraction of non-faces while trying to pass through all potential face candidates

[4]Some variants, such as that of Viola and Jones (2004), use $(a_j, b_j) \in [0, 1]$ and adjust the learning algorithm accordingly.

1. Input the positive and negative training examples along with their labels $\{(\boldsymbol{x}_i, y_i)\}$, where $y_i = 1$ for positive (face) examples and $y_i = -1$ for negative examples.

2. Initialize all the weights to $w_{i,1} \leftarrow \frac{1}{N}$, where N is the number of training examples. (Viola and Jones (2004) use a separate N_1 and N_2 for positive and negative examples.)

3. For each training stage $j = 1 \ldots M$:

   (a) Renormalize the weights so that they sum up to 1 (divide them by their sum).

   (b) Select the best classifier $h_j(\boldsymbol{x}; f_j, \theta_j, s_j)$ by finding the one that minimizes the weighted classification error

$$
\begin{aligned}
e_j &= \sum_{i=0}^{N-1} w_{i,j} e_{i,j}, && (14.3)\\
e_{i,j} &= 1 - \delta(y_i, h_j(\boldsymbol{x}_i; f_j, \theta_j, s_j)). && (14.4)
\end{aligned}
$$

   For any given f_j function, the optimal values of (θ_j, s_j) can be found in linear time using a variant of weighted median computation (Exercise 14.2).

   (c) Compute the modified error rate β_j and classifier weight α_j,

$$
\beta_j = \frac{e_j}{1 - e_j} \quad \text{and} \quad \alpha_j = -\log \beta_j. \qquad (14.5)
$$

   (d) Update the weights according to the classification errors $e_{i,j}$

$$
w_{i,j+1} \leftarrow w_{i,j} \beta_j^{1-e_{i,j}}, \qquad (14.6)
$$

   i.e., downweight the training samples that were correctly classified in proportion to the overall classification error.

4. Set the final classifier to

$$
h(\boldsymbol{x}) = \text{sign}\left[\sum_{j=0}^{m-1} \alpha_j h_j(\boldsymbol{x})\right]. \qquad (14.7)
$$

Algorithm 14.1 The AdaBoost training algorithm, adapted from Hastie, Tibshirani, and Friedman (2001), Viola and Jones (2004), and Bishop (2006).

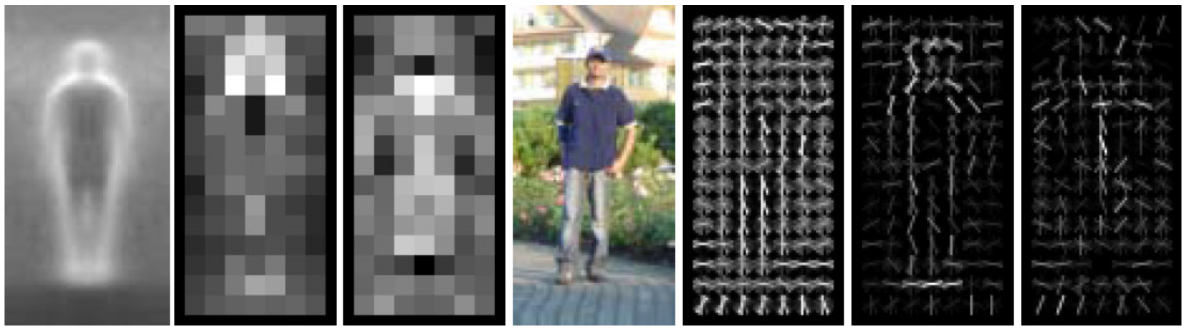


Figure 14.8 Pedestrian detection using histograms of oriented gradients (Dalal and Triggs 2005) © 2005 IEEE: (a) the average gradient image over the training examples; (b) each "pixel" shows the maximum positive SVM weight in the block centered on the pixel; (c) likewise, for the negative SVM weights; (d) a test image; (e) the computed R-HOG (rectangular histogram of gradients) descriptor; (f) the R-HOG descriptor weighted by the positive SVM weights; (g) the R-HOG descriptor weighted by the negative SVM weights.

(Fleuret and Geman 2001; Viola and Jones 2004). An even faster algorithm for performing cascade learning has recently been developed by Brubaker, Wu, Sun *et al.* (2008).

14.1.2 Pedestrian detection

While a lot of the research on object detection has focused on faces, the detection of other objects, such as pedestrians and cars, has also received widespread attention (Gavrila and Philomin 1999; Gavrila 1999; Papageorgiou and Poggio 2000; Mohan, Papageorgiou, and Poggio 2001; Schneiderman and Kanade 2004). Some of these techniques maintain the same focus as face detection on speed and efficiency. Others, however, focus instead on accuracy, viewing detection as a more challenging variant of generic class recognition (Section 14.4) in which the locations and extents of objects are to be determined as accurately as possible. (See, for example, the PASCAL VOC detection challenge, http://pascallin.ecs.soton.ac.uk/challenges/VOC/.)

An example of a well-known pedestrian detector is the algorithm developed by Dalal and Triggs (2005), who use a set of overlapping *histogram of oriented gradients* (HOG) descriptors fed into a support vector machine (Figure 14.8). Each HOG has cells to accumulate magnitude-weighted votes for gradients at particular orientations, just as in the scale invariant feature transform (SIFT) developed by Lowe (2004), which we discussed in Section 4.1.2 and Figure 4.18. Unlike SIFT, however, which is only evaluated at interest point locations, HOGs are evaluated on a regular overlapping grid and their descriptor magnitudes are normalized using an even coarser grid; they are only computed at a single scale and a fixed orientation. In order to capture the subtle variations in orientation around a person's outline, a large number of orientation bins is used and no smoothing is performed in the central difference gradient computation—see the work of Dalal and Triggs (2005) for more implementation details. Figure 14.8d shows a sample input image, while Figure 14.8e shows the associated HOG descriptors.

Once the descriptors have been computed, a support vector machine (SVM) is trained

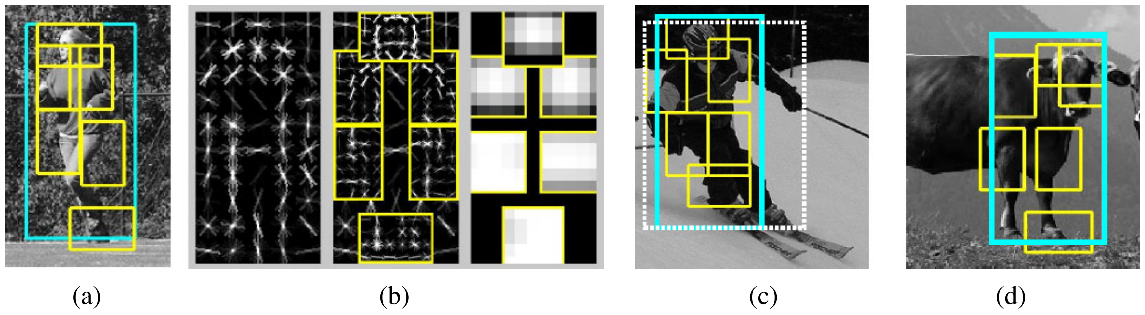


Figure 14.9 Part-based object detection (Felzenszwalb, McAllester, and Ramanan 2008) © 2008 IEEE: (a) An input photograph and its associated person (blue) and part (yellow) detection results. (b) The detection model is defined by a coarse template, several higher resolution part templates, and a spatial model for the location of each part. (c) True positive detection of a skier and (d) false positive detection of a cow (labeled as a person).

on the resulting high-dimensional continuous descriptor vectors. Figures 14.8b–c show a diagram of the (most) positive and negative SVM weights in each block, while Figures 14.8f–g show the corresponding weighted HOG responses for the central input image. As you can see, there are a fair number of positive responses around the head, torso, and feet of the person, and relatively few negative responses (mainly around the middle and the neck of the sweater).

The fields of pedestrian and general object detection have continued to evolve rapidly over the last decade (Belongie, Malik, and Puzicha 2002; Mikolajczyk, Schmid, and Zisserman 2004; Leibe, Seemann, and Schiele 2005; Opelt, Pinz, and Zisserman 2006; Torralba 2007; Andriluka, Roth, and Schiele 2009, 2010; Dollàr, Belongie, and Perona 2010). Munder and Gavrila (2006) compare a number of pedestrian detectors and conclude that those based on local receptive fields and SVMs perform the best, with a boosting-based approach coming close. Maji, Berg, and Malik (2008) improve on the best of these results using non-overlapping multi-resolution HOG descriptors and a histogram intersection kernel SVM based on a spatial pyramid match kernel from Lazebnik, Schmid, and Ponce (2006).

When detectors for several different classes are being constructed simultaneously, Torralba, Murphy, and Freeman (2007) show that sharing features and weak learners between detectors yields better performance, both in terms of faster computation times and fewer training examples. To find the features and decision stumps that work best in a shared manner, they introduce a novel *joint boosting* algorithm that optimizes, at each stage, a summed expected exponential loss function using the "gentleboost" algorithm of Friedman, Hastie, and Tibshirani (2000).

In more recent work, Felzenszwalb, McAllester, and Ramanan (2008) extend the histogram of oriented gradients person detector to incorporate flexible parts models (Section 14.4.2). Each part is trained and detected on HOGs evaluated at two pyramid levels below the overall object model and the locations of the parts relative to the parent node (the overall bounding box) are also learned and used during recognition (Figure 14.9b). To compensate for inaccuracies or inconsistencies in the training example bounding boxes (dashed white lines in Figure 14.9c), the "true" location of the parent (blue) bounding box is considered a latent (hidden) variable and is inferred during both training and recognition. Since the locations

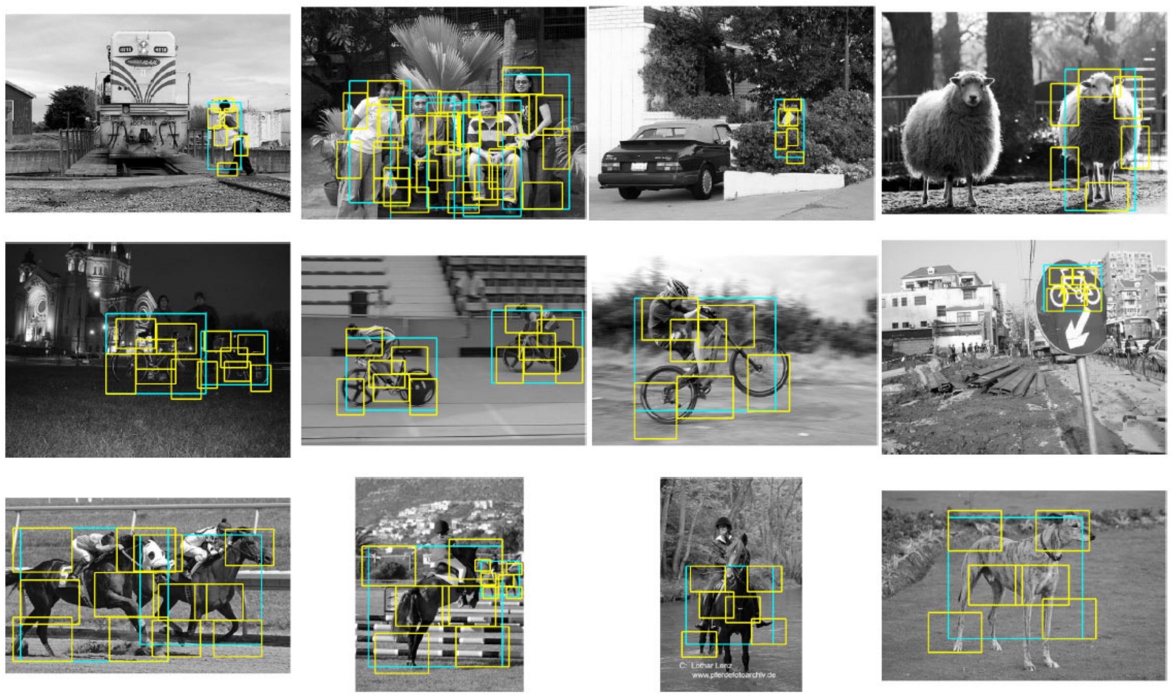


Figure 14.10 Part-based object detection results for people, bicycles, and horses (Felzenszwalb, McAllester, and Ramanan 2008) © 2008 IEEE. The first three columns show correct detections, while the rightmost column shows false positives.

of the parts are also latent, the system can be trained in a semi-supervised fashion, without needing part labels in the training data. An extension to this system (Felzenszwalb, Girshick, McAllester *et al.* 2010), which includes among its improvements a simple contextual model, was among the two best object detection systems in the 2008 Visual Object Classes detection challenge. Other recent improvements to part-based person detection and pose estimation include the work by Andriluka, Roth, and Schiele (2009) and Kumar, Zisserman, and H.S.Torr (2009).

An even more accurate estimate of a person's pose and location is presented by Rogez, Rihan, Ramalingam *et al.* (2008), who compute both the phase of a person in a walk cycle and the locations of individual joints, using random forests built on top of HOGs (Figure 14.11). Since their system produces full 3D pose information, it is closer in its application domain to 3D person trackers (Sidenbladh, Black, and Fleet 2000; Andriluka, Roth, and Schiele 2010), which we discussed in Section 12.6.4.

One final note on person and object detection. When video sequences are available, the additional information present in the optic flow and motion discontinuities can greatly aid in the detection task, as discussed by Efros, Berg, Mori *et al.* (2003), Viola, Jones, and Snow (2003), and Dalal, Triggs, and Schmid (2006).

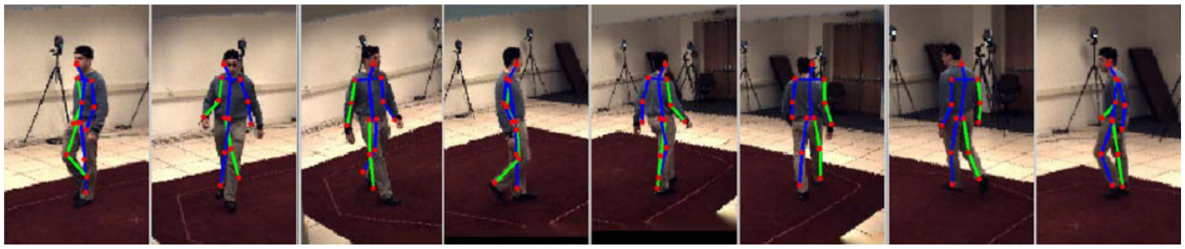


Figure 14.11 Pose detection using random forests (Rogez, Rihan, Ramalingam *et al.* 2008) © 2008 IEEE. The estimated pose (state of the kinematic model) is drawn over each input frame.

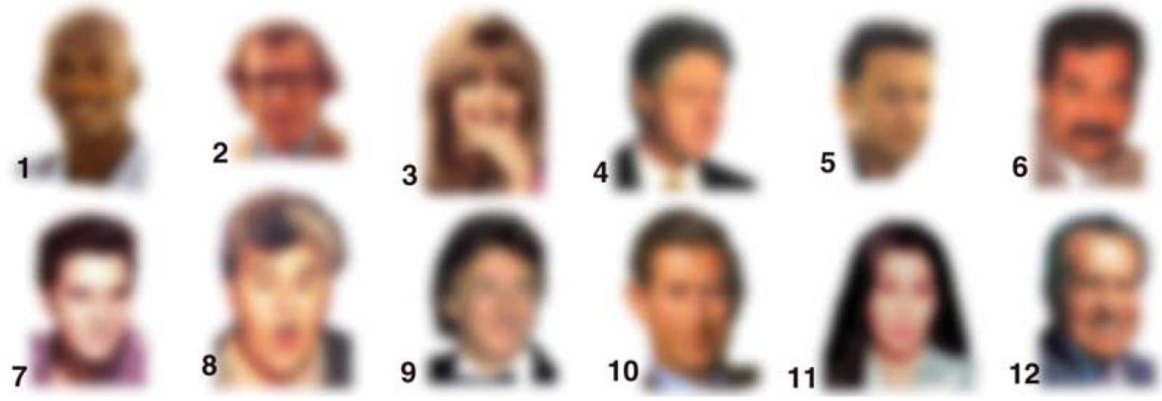


Figure 14.12 Humans can recognize low-resolution faces of familiar people (Sinha, Balas, Ostrovsky *et al.* 2006) © 2006 IEEE.

14.2 Face recognition

Among the various recognition tasks that computers might be asked to perform, face recognition is the one where they have arguably had the most success.[5] While computers cannot pick out suspects from thousands of people streaming in front of video cameras (even people cannot readily distinguish between similar people with whom they are not familiar (O'Toole, Jiang, Roark *et al.* 2006; O'Toole, Phillips, Jiang *et al.* 2009)), their ability to distinguish among a small number of family members and friends has found its way into consumer-level photo applications, such as Picasa and iPhoto. Face recognition can also be used in a variety of additional applications, including human–computer interaction (HCI), identity verification (Kirovski, Jojic, and Jancke 2004), desktop login, parental controls, and patient monitoring (Zhao, Chellappa, Phillips *et al.* 2003).

Today's face recognizers work best when they are given full frontal images of faces under relatively uniform illumination conditions, although databases that include large amounts of pose and lighting variation have been collected (Phillips, Moon, Rizvi *et al.* 2000; Sim,

[5]Instance recognition, i.e., the re-recognition of known objects such as locations or planar objects, is the other most successful application of general image recognition. In the general domain of *biometrics*, i.e., identity recognition, specialized images such as irises and fingerprints perform even better (Jain, Bolle, and Pankanti 1999; Pankanti, Bolle, and Jain 2000; Daugman 2004).

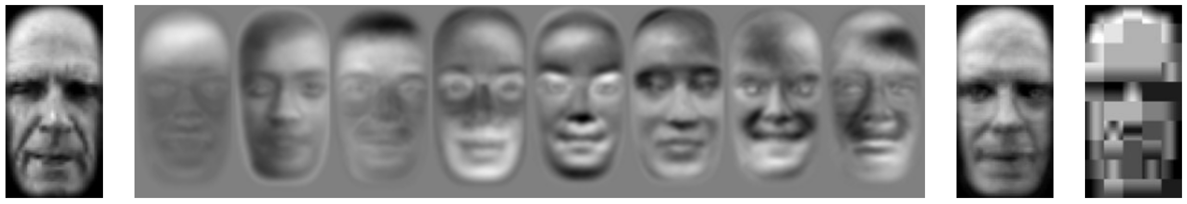

(a) (b) (c) (d)

Figure 14.13 Face modeling and compression using eigenfaces (Moghaddam and Pentland 1997) © 1997 IEEE: (a) input image; (b) the first eight eigenfaces; (c) image reconstructed by projecting onto this basis and compressing the image to 85 bytes; (d) image reconstructed using JPEG (530 bytes).

Baker, and Bsat 2003; Gross, Shi, and Cohn 2005; Huang, Ramesh, Berg *et al.* 2007; Phillips, Scruggs, O'Toole *et al.* 2010). (See Table 14.1 in Section 14.6 for more details.)

Some of the earliest approaches to face recognition involved finding the locations of distinctive image features, such as the eyes, nose, and mouth, and measuring the distances between these feature locations (Fischler and Elschlager 1973; Kanade 1977; Yuille 1991). More recent approaches rely on comparing gray-level images projected onto lower dimensional subspaces called *eigenfaces* (Section 14.2.1) and jointly modeling shape and appearance variations (while discounting pose variations) using *active appearance models* (Section 14.2.2).

Descriptions of additional face recognition techniques can be found in a number of surveys and books on this topic (Chellappa, Wilson, and Sirohey 1995; Zhao, Chellappa, Phillips *et al.* 2003; Li and Jain 2005) as well as the Face Recognition Web site.[6] The survey on face recognition by humans by Sinha, Balas, Ostrovsky *et al.* (2006) is also well worth reading; it includes a number of surprising results, such as humans' ability to recognize low-resolution images of familiar faces (Figure 14.12) and the importance of eyebrows in recognition.

14.2.1 Eigenfaces

Eigenfaces rely on the observation first made by Kirby and Sirovich (1990) that an arbitrary face image $\boldsymbol{x}$ can be compressed and reconstructed by starting with a mean image $\boldsymbol{m}$ (Figure 14.1b) and adding a small number of scaled signed images $\boldsymbol{u}_i$,[7]

$$\tilde{\boldsymbol{x}} = \boldsymbol{m} + \sum_{i=0}^{M-1} a_i \boldsymbol{u}_i, \tag{14.8}$$

where the signed basis images (Figure 14.13b) can be derived from an ensemble of training images using *principal component analysis* (also known as *eigenvalue analysis* or the *Karhunen–Loève transform*). Turk and Pentland (1991a) recognized that the coefficients a_i in the eigenface expansion could themselves be used to construct a fast image matching algorithm.

[6] http://www.face-rec.org/.

[7] In previous chapters, we used I to indicate images; in this chapter, we use the more abstract quantities $\boldsymbol{x}$ and $\boldsymbol{u}$ to indicate collections of pixels in an image turned into a vector.

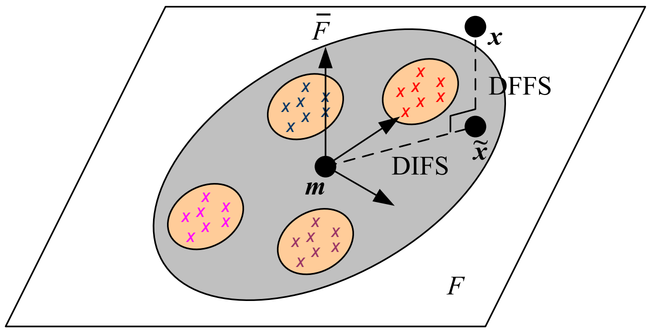


Figure 14.14 Projection onto the linear subspace spanned by the eigenface images (Moghaddam and Pentland 1997) © 1997 IEEE. The distance from face space (DFFS) is the orthogonal distance to the plane, while the distance in face space (DIFS) is the distance along the plane from the mean image. Both distances can be turned into Mahalanobis distances and given probabilistic interpretations.

In more detail, let us start with a collection of *training images* $\{\boldsymbol{x}_j\}$, from which we can compute the mean image $\boldsymbol{m}$ and a *scatter* or *covariance* matrix

$$\boldsymbol{C} = \frac{1}{N}\sum_{j=0}^{N-1}(\boldsymbol{x}_j - \boldsymbol{m})(\boldsymbol{x}_j - \boldsymbol{m})^T. \tag{14.9}$$

We can apply the eigenvalue decomposition (A.6) to represent this matrix as

$$\boldsymbol{C} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T = \sum_{i=0}^{N-1}\lambda_i \boldsymbol{u}_i\boldsymbol{u}_i^T, \tag{14.10}$$

where the λ_i are the eigenvalues of $\boldsymbol{C}$ and the $\boldsymbol{u}_i$ are the *eigenvectors*. For general images, Kirby and Sirovich (1990) call these vectors *eigenpictures*; for faces, Turk and Pentland (1991a) call them *eigenfaces* (Figure 14.13b).[8]

Two important properties of the eigenvalue decomposition are that the optimal (best approximation) coefficients a_i for any new image $\boldsymbol{x}$ can be computed as

$$a_i = (\boldsymbol{x} - \boldsymbol{m}) \cdot \boldsymbol{u}_i, \tag{14.11}$$

and that, assuming the eigenvalues $\{\lambda_i\}$ are sorted in decreasing order, truncating the approximation given in (14.8) at any point M gives the best possible approximation (least error) between $\tilde{\boldsymbol{x}}$ and $\boldsymbol{x}$. Figure 14.13c shows the resulting approximation corresponding to Figure 14.13a and shows how much better it is at compressing a face image than JPEG.

Truncating the eigenface decomposition of a face image (14.8) after M components is equivalent to projecting the image onto a linear subspace F, which we can call the *face space* (Figure 14.14). Because the eigenvectors (eigenfaces) are orthogonal and of unit norm, the

[8] In actual practice, the full $P \times P$ scatter matrix (14.9) is never computed. Instead, a smaller $N \times N$ matrix consisting of the inner products between all the signed deviations $(\boldsymbol{x}_i - \boldsymbol{m})$ is accumulated instead. See Appendix A.1.2 (A.13–A.14) for details.

distance of a projected face $\tilde{\boldsymbol{x}}$ to the mean face $\boldsymbol{m}$ can be written as

$$\text{DIFS} = \|\tilde{\boldsymbol{x}} - \boldsymbol{m}\| = \sqrt{\sum_{i=0}^{M-1} a_i^2}, \tag{14.12}$$

where DIFS stands for *distance in face space* (Moghaddam and Pentland 1997). The remaining distance between the original image $\boldsymbol{x}$ and its projection onto face space $\tilde{\boldsymbol{x}}$, i.e., the *distance from face space* (DFFS), can be computed directly in pixel space and represents the "faceness" of a particular image.[9] It is also possible to measure the distance between two different faces in face space as

$$\text{DIFS}(\boldsymbol{x}, \boldsymbol{y}) = \|\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{y}}\| = \sqrt{\sum_{i=0}^{M-1} (a_i - b_i)^2}, \tag{14.13}$$

where the $b_i = (\boldsymbol{y} - \boldsymbol{m}) \cdot \boldsymbol{u}_i$ are the eigenface coefficients corresponding to $\boldsymbol{y}$.

Computing such distances in Euclidean vector space, however, does not exploit the additional information that the eigenvalue decomposition of our covariance matrix (14.10) provides. If we interpret the covariance matrix $\boldsymbol{C}$ as the covariance of a multi-variate Gaussian (Appendix B.1.1),[10] we can turn the DIFS into a log likelihood by computing the *Mahalanobis distance*

$$\text{DIFS}' = \|\tilde{\boldsymbol{x}} - \boldsymbol{m}\|_{\boldsymbol{C}^{-1}} = \sqrt{\sum_{i=0}^{M-1} a_i^2/\lambda_i^2}. \tag{14.14}$$

Instead of measuring the squared distance along each principal component in face space F, the Mahalanobis distance measures the *ratio* between the squared distance and the corresponding *variance* $\sigma_i^2 = \lambda_i$ and then sums these squared ratios (per-component log-likelihoods). An alternative way to implement this is to pre-scale each eigenvector by the inverse square root of its corresponding eigenvalue,

$$\hat{\boldsymbol{U}} = \boldsymbol{U}\boldsymbol{\Lambda}^{-1/2}. \tag{14.15}$$

This *whitening* transformation then means that Euclidean distances in feature (face) space now correspond directly to log likelihoods (Moghaddam, Jebara, and Pentland 2000). (This same whitening approach can also be used in feature-based matching algorithms, as discussed in Section 4.1.3.)

If the distribution in eigenface space is very elongated, the Mahalanobis distance properly scales the components to come up with a sensible (probabilistic) distance from the mean. A similar analysis can be performed for computing a sensible difference from face space (DFFS) (Moghaddam and Pentland 1997) and the two terms can be combined to produce an estimate of the likelihood of being a true face, which can be useful in doing face detection (Section 14.1.1). More detailed explanations of probabilistic and Bayesian PCA can be found in textbooks on statistical learning (Hastie, Tibshirani, and Friedman 2001; Bishop 2006), which also discuss techniques for selecting the optimum number of components M to use in modeling a distribution.

[9] This can be used to form a simple face detector, as mentioned in Section 14.1.1.

[10] The ellipse shown in Figure 14.14 denotes an equi-probability contour of this multi-variate Gaussian.

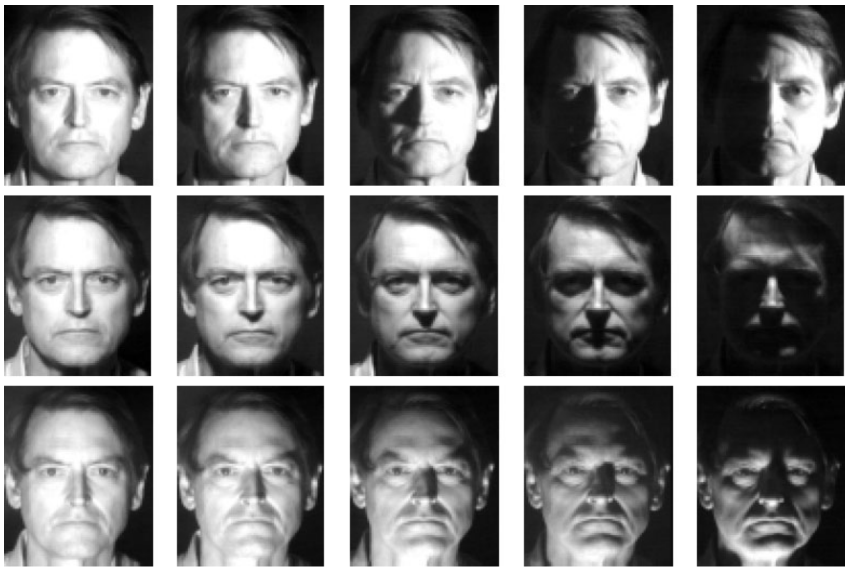


Figure 14.15 Images from the Harvard database used by Belhumeur, Hespanha, and Kriegman (1997) © 1997 IEEE. Note the wide range of illumination variation, which can be more dramatic than inter-personal variations.

One of the biggest advantages of using eigenfaces is that they reduce the comparison of a new face image $\boldsymbol{x}$ to a prototype (training) face image $\boldsymbol{x}_k$ (one of the colored xs in Figure 14.14) from a P-dimensional difference in pixel space to an M-dimensional difference in face space,

$$\|\boldsymbol{x} - \boldsymbol{x}_k\| = \|\boldsymbol{a} - \boldsymbol{a}_k\|, \tag{14.16}$$

where $\boldsymbol{a} = \boldsymbol{U}^T(\boldsymbol{x} - \boldsymbol{m})$ (14.11) involves computing a dot product between the signed difference-from-mean image $(\boldsymbol{x} - \boldsymbol{m})$ and each of the eigenfaces $\boldsymbol{u}_i$. Once again, however, this Euclidean distance ignores the fact that we have more information about face likelihoods available in the distribution of training images.

Consider the set of images of one person taken under a wide range of illuminations shown in Figure 14.15. As you can see, the *intrapersonal* variability within these images is much greater than the typical *extrapersonal* variability between any two people taken under the same illumination. Regular PCA analysis fails to distinguish between these two sources of variability and may, in fact, devote most of its principal components to modeling the intrapersonal variability.

If we are going to approximate faces by a linear subspace, it is more useful to have a space that *discriminates* between different classes (people) and is less sensitive to within-class variations (Belhumeur, Hespanha, and Kriegman 1997). Consider the three classes shown as different colors in Figure 14.16. As you can see, the distributions within a class (indicated by the tilted colored axes) are elongated and tilted with respect to the main face space PCA, which is aligned with the black x and y axes. We can compute the total *within-class* scatter matrix as

$$\boldsymbol{S}_{\mathrm{W}} = \sum_{k=0}^{K-1} \boldsymbol{S}_k = \sum_{k=0}^{K-1} \sum_{i \in C_k} (\boldsymbol{x}_i - \boldsymbol{m}_k)(\boldsymbol{x}_i - \boldsymbol{m}_k)^T, \tag{14.17}$$

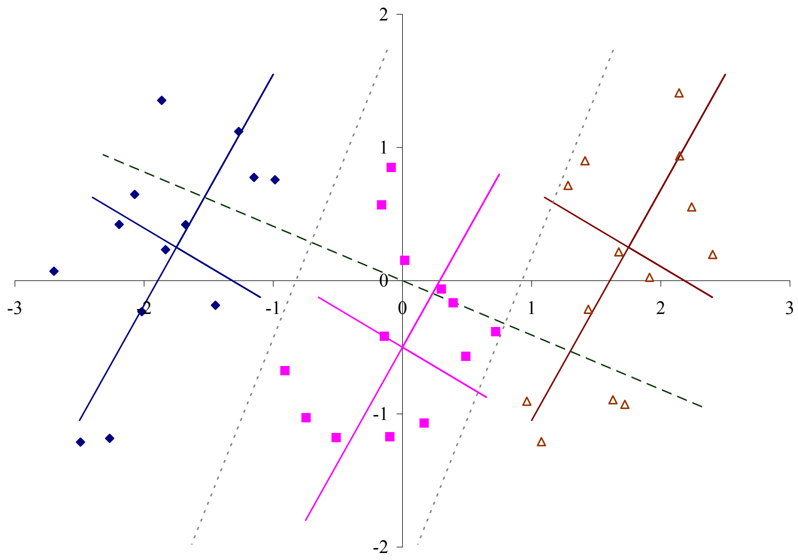


Figure 14.16 Simple example of Fisher linear discriminant analysis. The samples come from three different classes, shown in different colors along with their principal axes, which are scaled to $2\sigma_i$. (The intersections of the tilted axes are the class means $\boldsymbol{m}_k$.) The dashed line is the (dominant) Fisher linear discriminant direction and the dotted lines are the linear discriminants between the classes. Note how the discriminant direction is a blend between the principal directions of the between-class and within-class scatter matrices.

where $\boldsymbol{m}_k$ is the mean of class k and $\boldsymbol{S}_k$ is its within-class scatter matrix.[11] Similarly, we can compute the *between-class* scatter as

$$\boldsymbol{S}_\mathrm{B} = \sum_{k=0}^{K-1} N_k(\boldsymbol{m}_k - \boldsymbol{m})(\boldsymbol{m}_k - \boldsymbol{m})^T, \tag{14.18}$$

where N_k are the number of exemplars in each class and $\boldsymbol{m}$ is the overall mean. For the three distributions shown in Figure 14.16, we have

$$\boldsymbol{S}_\mathrm{W} = 3N \begin{bmatrix} 0.246 & 0.183 \\ 0.183 & 0.457 \end{bmatrix} \quad \text{and} \quad \boldsymbol{S}_\mathrm{B} = N \begin{bmatrix} 6.125 & 0 \\ 0 & 0.375 \end{bmatrix}, \tag{14.19}$$

where $N = N_k = 13$ is the number of samples in each class.

To compute the most discriminating direction, *Fisher's linear discriminant* (FLD) (Belhumeur, Hespanha, and Kriegman 1997; Hastie, Tibshirani, and Friedman 2001; Bishop 2006), which is also known as *linear discriminant analysis* (LDA), selects the direction $\boldsymbol{u}$ that results in the largest ratio between the projected between-class and within-class variations

$$\boldsymbol{u}^* = \arg\max_{\boldsymbol{u}} \frac{\boldsymbol{u}^T \boldsymbol{S}_\mathrm{B} \boldsymbol{u}}{\boldsymbol{u}^T \boldsymbol{S}_\mathrm{W} \boldsymbol{u}}, \tag{14.20}$$

[11] To be consistent with Belhumeur, Hespanha, and Kriegman (1997), we use $\boldsymbol{S}_\mathrm{W}$ and $\boldsymbol{S}_\mathrm{B}$ to denote the scatter matrices, even though we use $\boldsymbol{C}$ elsewhere (14.9).

which is equivalent to finding the eigenvector corresponding to the largest eigenvalue of the generalized eigenvalue problem

$$\boldsymbol{S}_\mathrm{B}\boldsymbol{u} = \lambda \boldsymbol{S}_\mathrm{W}\boldsymbol{u} \quad \text{or} \quad \lambda\boldsymbol{u} = \boldsymbol{S}_\mathrm{W}^{-1}\boldsymbol{S}_\mathrm{B}\boldsymbol{u}. \tag{14.21}$$

For the problem shown in Figure 14.16,

$$\boldsymbol{S}_\mathrm{W}^{-1}\boldsymbol{S}_\mathrm{B} = \begin{bmatrix} 11.796 & -0.289 \\ -4.715 & 0.3889 \end{bmatrix} \quad \text{and} \quad \boldsymbol{u} = \begin{bmatrix} 0.926 \\ -0.379 \end{bmatrix} \tag{14.22}$$

As you can see, using this direction results in a better separation between the classes than using the dominant PCA direction, which is the horizontal axis. In their paper, Belhumeur, Hespanha, and Kriegman (1997) show that Fisherfaces significantly outperform the original eigenfaces algorithm, especially when faces have large amounts of illumination variation, as in Figure 14.15.

An alternative for modeling within-class (intrapersonal) and between-class (extrapersonal) variations is to model each distribution separately and then use Bayesian techniques to find the closest exemplar (Moghaddam, Jebara, and Pentland 2000). Instead of computing the mean for each class and then the within-class and between-class distributions, consider evaluating the difference images

$$\boldsymbol{\Delta}_{ij} = \boldsymbol{x}_i - \boldsymbol{x}_j \tag{14.23}$$

between all pairs of training images $(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The differences between pairs that are in the same class (the same person) are used to estimate the intrapersonal covariance matrix $\boldsymbol{\Sigma}_I$, while differences between different people are used to estimate the extrapersonal covariance $\boldsymbol{\Sigma}_E$.[12] The principal components (eigenfaces) corresponding to these two classes are shown in Figure 14.17.

At recognition time, we can compute the distance $\boldsymbol{\Delta}_i$ between a new face $\boldsymbol{x}$ and a stored training image $\boldsymbol{x}_i$ and evaluate its intrapersonal likelihood as

$$p_I(\boldsymbol{\Delta}_i) = p_\mathcal{N}(\boldsymbol{\Delta}_i; \boldsymbol{\Sigma}_I) = \frac{1}{|2\pi\boldsymbol{\Sigma}_I|^{1/2}} \exp -\|\boldsymbol{\Delta}_i\|^2_{\boldsymbol{\Sigma}_I^{-1}}, \tag{14.24}$$

where $p_\mathcal{N}$ is a normal (Gaussian) distribution with covariance $\boldsymbol{\Sigma}_I$ and

$$|2\pi\boldsymbol{\Sigma}_I|^{1/2} = (2\pi)^{M/2}\prod_{j=1}^{M}\lambda_j^{1/2} \tag{14.25}$$

is its volume. The Mahalanobis distance

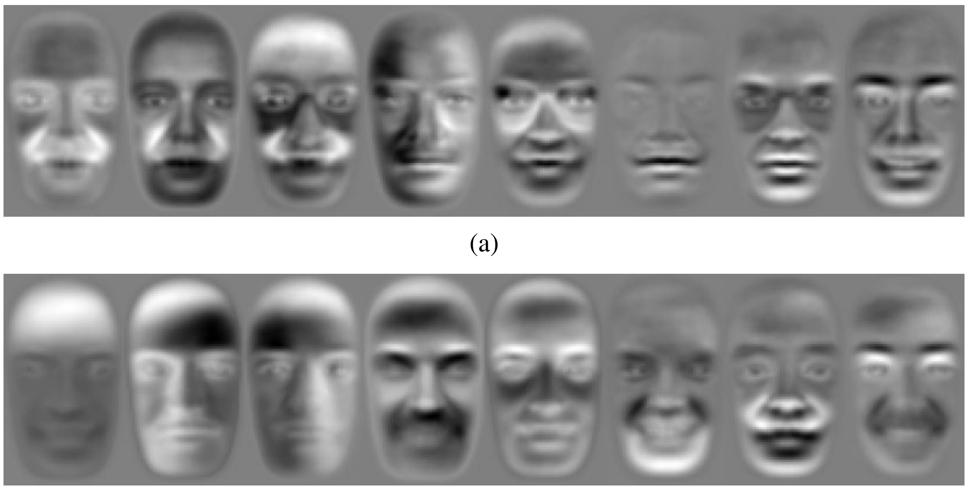
$$\|\boldsymbol{\Delta}_i\|^2_{\boldsymbol{\Sigma}_I^{-1}} = \boldsymbol{\Delta}_i^T\boldsymbol{\Sigma}_I^{-1}\boldsymbol{\Delta}_i = \|\boldsymbol{a}^I - \boldsymbol{a}_i^I\|^2 \tag{14.26}$$

can be computed more efficiently by first projecting the new image $\boldsymbol{x}$ into the whitened intrapersonal face space (14.15)

$$\boldsymbol{a}^I = \hat{\boldsymbol{U}}^I\boldsymbol{x} \tag{14.27}$$

and then computing a Euclidean distance to the training image vector $\boldsymbol{a}_i^I$, which can be precomputed offline. The extrapersonal likelihood $p_E(\boldsymbol{\Delta}_i)$ can be computed in a similar fashion.

[12] Note that the difference distributions are zero mean because for every $\boldsymbol{\Delta}_{ij}$ there corresponds a negative $\boldsymbol{\Delta}_{ji}$.


(a)

(b)

Figure 14.17 "Dual" eigenfaces (Moghaddam, Jebara, and Pentland 2000) © 2000 Elsevier: (a) intrapersonal and (b) extrapersonal.

Once the intrapersonal and extrapersonal likelihoods have been computed, we can compute the Bayesian likelihood of a new image $\boldsymbol{x}$ matching a training image $\boldsymbol{x}_i$ as

$$p(\boldsymbol{\Delta}_i) = \frac{p_I(\boldsymbol{\Delta}_i)l_I}{p_I(\boldsymbol{\Delta}_i)l_I + p_E(\boldsymbol{\Delta}_i)l_E}, \tag{14.28}$$

where l_I and l_E are the prior probabilities of two images being in the same or in different classes (Moghaddam, Jebara, and Pentland 2000). A simpler approach, which does not require the evaluation of extrapersonal probabilities, is to simply choose the training image with the highest likelihood $p_I(\boldsymbol{\Delta}_i)$. In this case, nearest neighbor search techniques in the space spanned by the precomputed $\{\boldsymbol{a}_i^I\}$ vectors could be used to speed up finding the best match.[13]

Another way to improve the performance of eigenface-based approaches is to break up the image into separate regions such as the eyes, nose, and mouth (Figure 14.18) and to match each of these *modular eigenspaces* independently (Moghaddam and Pentland 1997; Heisele, Ho, Wu *et al.* 2003; Heisele, Serre, and Poggio 2007). The advantage of such a modular approach is that it can tolerate a wider range of viewpoints, because each part can move relative to the others. It also supports a larger variety of combinations, e.g., we can model one person as having a narrow nose and bushy eyebrows, without requiring the eigenfaces to span all possible combinations of nose, mouth, and eyebrows. (If you remember the cardboard children's books where you can select different top and bottom faces, or Mr. Potato Head, you get the idea.)

Another approach to dealing with large variability in appearance is to create *view-based* (view-specific) eigenspaces, as shown in Figure 14.19 (Moghaddam and Pentland 1997). We can think of these view-based eigenspaces as local descriptors that select different axes depending on which part of the face space you are in. Note that such approaches, however,

[13] Note that while the covariance matrices $\boldsymbol{\Sigma}_I$ and $\boldsymbol{\Sigma}_E$ are computed by looking at differences between *all* pairs of images, the run-time evaluation selects the *nearest* image to determine the facial identity. Whether this is statistically correct is explored in Exercise 14.4.

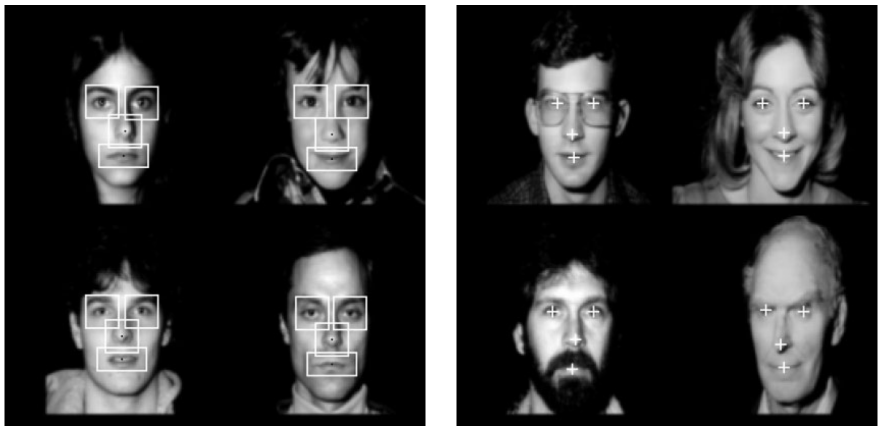

(a) (b)

Figure 14.18 Modular eigenspace for face recognition (Moghaddam and Pentland 1997) © 1997 IEEE. (a) By detecting separate features in the faces (eyes, nose, mouth), separate eigenspaces can be estimated for each one. (b) The relative positions of each feature can be detected at recognition time, thus allowing for more flexibility in viewpoint and expression.

potentially require large amounts of training data, i.e., pictures of every person in every possible pose or expression. This is in contrast to the shape and appearance models we study in Section 14.2.2, which can learn deformations across all individuals.

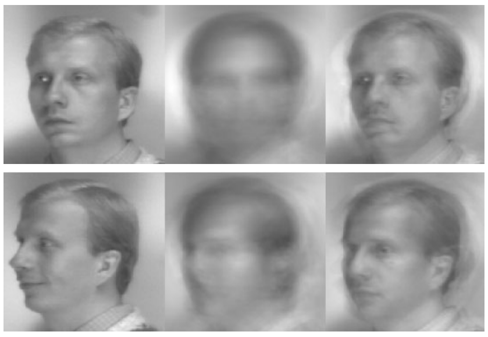
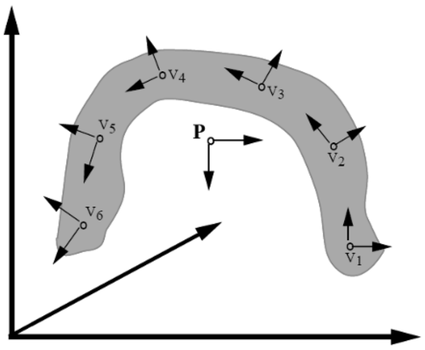
It is also possible to generalize the bilinear factorization implicit in PCA and SVD approaches to multilinear (tensor) formulations that can model several interacting factors simultaneously (Vasilescu and Terzopoulos 2007). These ideas are related to currently active topics in machine learning such as *subspace learning* (Cai, He, Hu *et al.* 2007), *local distance functions* (Frome, Singer, Sha *et al.* 2007), and *metric learning* (Ramanan and Baker 2009). Learning approaches play an increasingly important role in face recognition, e.g., in the work of Sivic, Everingham, and Zisserman (2009) and Guillaumin, Verbeek, and Schmid (2009).

14.2.2 Active appearance and 3D shape models

The need to use modular or view-based eigenspaces for face recognition is symptomatic of a more general observation, i.e., that facial appearance and identifiability depend as much on *shape* as they do on color or texture (which is what eigenfaces capture). Furthermore, when dealing with 3D head rotations, the *pose* of a person's head should be discounted when performing recognition.

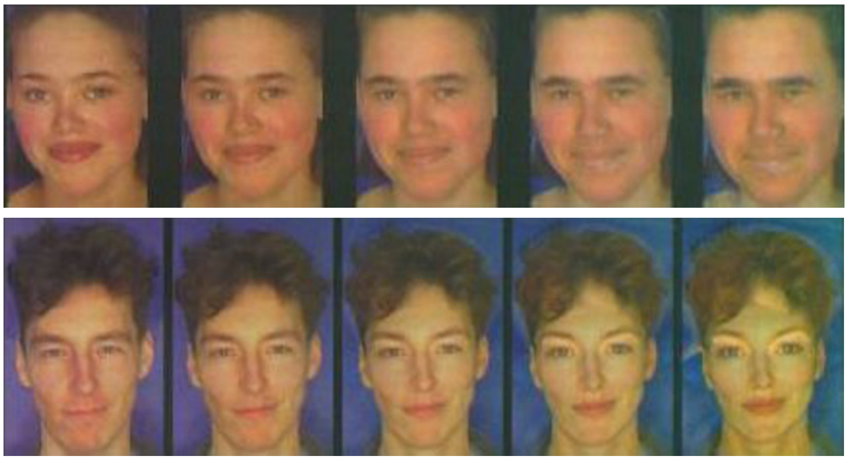
In fact, the earliest face recognition systems, such as those by Fischler and Elschlager (1973), Kanade (1977), and Yuille (1991), found distinctive feature points on facial images and performed recognition on the basis of their relative positions or distances. Newer techniques such as *local feature analysis* (Penev and Atick 1996) and *elastic bunch graph matching* (Wiskott, Fellous, Krüger *et al.* 1997) combine local filter responses (jets) at distinctive feature locations together with shape models to perform recognition.

A visually compelling example of why both shape and texture are important is the work of Rowland and Perrett (1995), who manually traced the contours of facial features and then

(a) (b)

Figure 14.19 View-based eigenspace (Moghaddam and Pentland 1997) © 1997 IEEE. (a) Comparison between a regular (parametric) eigenspace reconstruction (middle column) and a view-based eigenspace reconstruction (right column) corresponding to the input image (left column). The top row is from a training image, the bottom row is from the test set. (b) A schematic representation of the two approaches, showing how each view computes its own local basis representation.



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

Figure 14.20 Manipulating facial appearance through shape and color (Rowland and Perrett 1995) © 1995 IEEE. By adding or subtracting gender-specific shape and color characteristics to (b) an input image, different amounts of gender variation can be induced. The amounts added (from the mean) are: (a) +50% (gender enhancement), (c) -50% (near "androgyny"), (d) -100% (gender switched), and (e) -150% (opposite gender attributes enhanced).

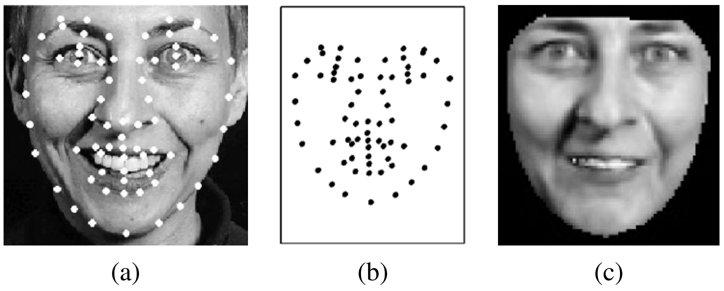


Figure 14.21 Active Appearance Models (Cootes, Edwards, and Taylor 2001) © 2001 IEEE: (a) input image with registered feature points; (b) the feature points (shape vector $\boldsymbol{s}$); (c) the shape-free appearance image (texture vector $\boldsymbol{t}$).

used these contours to normalize (warp) each image to a canonical shape. After analyzing both the shape and color images for deviations from the mean, they were able to associate certain shape and color deformations with personal characteristics such as age and gender (Figure 14.20). Their work demonstrates that both shape and color have an important influence on the perception of such characteristics.

Around the same time, researchers in computer vision were beginning to use simultaneous shape deformations and texture interpolation to model the variability in facial appearance caused by identity or expression (Beymer 1996; Vetter and Poggio 1997), developing techniques such as Active Shape Models (Lanitis, Taylor, and Cootes 1997), 3D Morphable Models (Blanz and Vetter 1999), and Elastic Bunch Graph Matching (Wiskott, Fellous, Krüger *et al.* 1997).[14]

Of all these techniques, the *active appearance models* (AAMs) of Cootes, Edwards, and Taylor (2001) are among the most widely used for face recognition and tracking. Like other shape and texture models, an AAM models both the variation in the shape of an image $\boldsymbol{s}$, which is normally encoded by the location of key feature points on the image (Figure 14.21b), as well as the variation in texture $\boldsymbol{t}$, which is normalized to a canonical shape before being analyzed (Figure 14.21c).[15]

Both shape and texture are represented as deviations from a mean shape $\bar{\boldsymbol{s}}$ and texture $\bar{\boldsymbol{t}}$,

$$\boldsymbol{s} = \bar{\boldsymbol{s}} + \boldsymbol{U}_s \boldsymbol{a} \tag{14.29}$$

$$\boldsymbol{t} = \bar{\boldsymbol{t}} + \boldsymbol{U}_t \boldsymbol{a}, \tag{14.30}$$

where the eigenvectors in $\boldsymbol{U}_s$ and $\boldsymbol{U}_t$ have been pre-scaled (whitened) so that unit vectors in $\boldsymbol{a}$ represent one standard deviation of variation observed in the training data. In addition to these principal deformations, the shape parameters are transformed by a global similarity to match the location, size, and orientation of a given face. Similarly, the texture image contains a scale and offset to best match novel illumination conditions.

As you can see, the same appearance parameters $\boldsymbol{a}$ in (14.29–14.30) simultaneously control both the shape and texture deformations from the mean, which makes sense if we believe

[14] We have already seen the application of PCA to 3D head and face modeling and animation in Section 12.6.3.

[15] When only the shape variation is being captured, such models are called *active shape models* (ASMs) (Cootes, Cooper, Taylor *et al.* 1995; Davies, Twining, and Taylor 2008). These were already discussed in Section 5.1.1 (5.13–5.17).

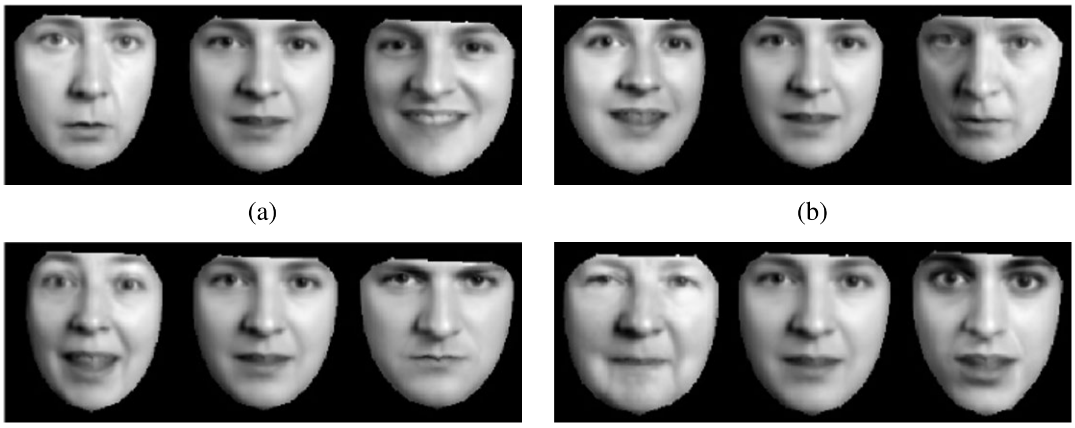


Figure 14.22 Principal modes of variation in active appearance models (Cootes, Edwards, and Taylor 2001) © 2001 IEEE. The four images show the effects of simultaneously changing the first four modes of variation in both shape and texture by $\pm\sigma$ from the mean. You can clearly see how the shape of the face and the shading are simultaneously affected.

them to be correlated. Figure 14.22 shows how moving three standard deviations along each of the first four principal directions ends up changing several correlated factors in a person's appearance, including expression, gender, age, and identity.

In order to fit an active appearance model to a novel image, Cootes, Edwards, and Taylor (2001) pre-compute a set of "difference decomposition" images, using an approach related to other fast techniques for incremental tracking, such as those we discussed in Sections 4.1.4, 8.1.3, and 8.2 (Gleicher 1997; Hager and Belhumeur 1998), which often *learn* a discriminative mapping between matching errors and incremental displacements (Avidan 2001; Jurie and Dhome 2002; Liu, Chen, and Kumar 2003; Sclaroff and Isidoro 2003; Romdhani and Vetter 2003; Williams, Blake, and Cipolla 2003).

In more detail, Cootes, Edwards, and Taylor (2001) compute the derivatives of a set of training images with respect to each of the parameters in $\boldsymbol{a}$ using finite differences and then compute a set of *displacement weight* images

$$\boldsymbol{W} = \left[\frac{\partial \boldsymbol{x}^T}{\partial \boldsymbol{a}} \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{a}} \right]^{-1} \frac{\partial \boldsymbol{x}^T}{\partial \boldsymbol{a}}, \qquad (14.31)$$

which can be multiplied by the current error residual to produce an update step in the parameters, $\delta\boldsymbol{a} = -\boldsymbol{W}\boldsymbol{r}$. Matthews and Baker (2004) use their *inverse compositional method*, which they first developed for parametric optical flow (8.64–8.65), to further speed up active appearance model fitting and tracking. Examples of AAMs being fitted to two input images are shown in Figure 14.23.

Although active appearance models are primarily designed to accurately capture the variability in appearance and deformation that are characteristic of faces, they can be adapted to face recognition by computing an identity subspace that separates variation in identity from other sources of variability such as lighting, pose, and expression (Costen, Cootes, Edwards *et al.* 1999). The basic idea, which is modeled after similar work in eigenfaces (Belhumeur,

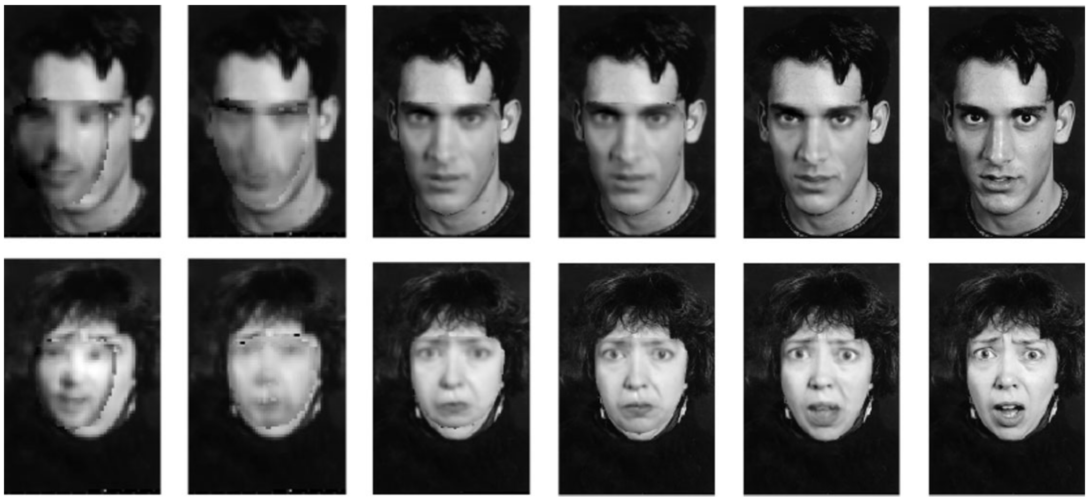


Figure 14.23 Multiresolution model fitting (search) in active appearance models (Cootes, Edwards, and Taylor 2001) © 2001 IEEE. The columns show the initial model, the results after 3, 8, and 11 iterations, and the final convergence. The rightmost column shows the input image.

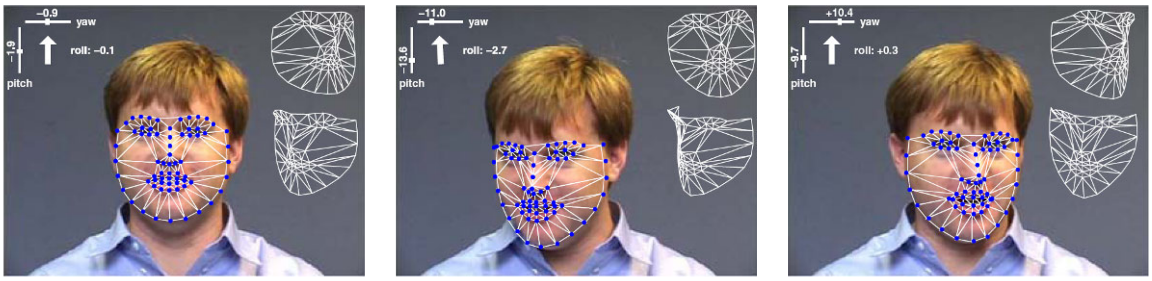
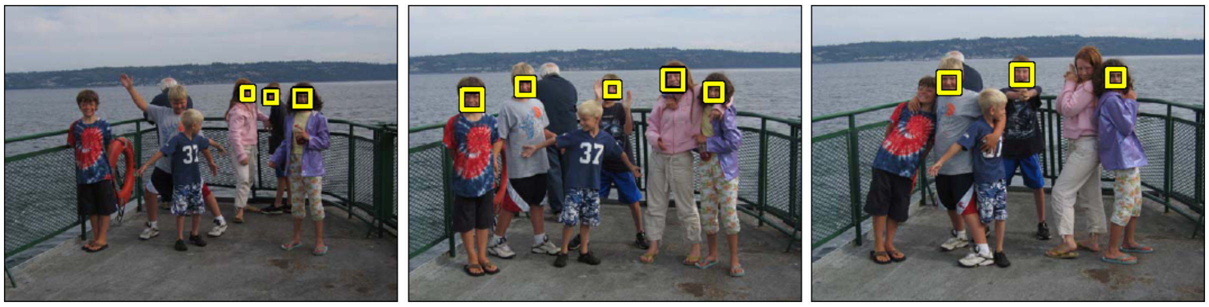


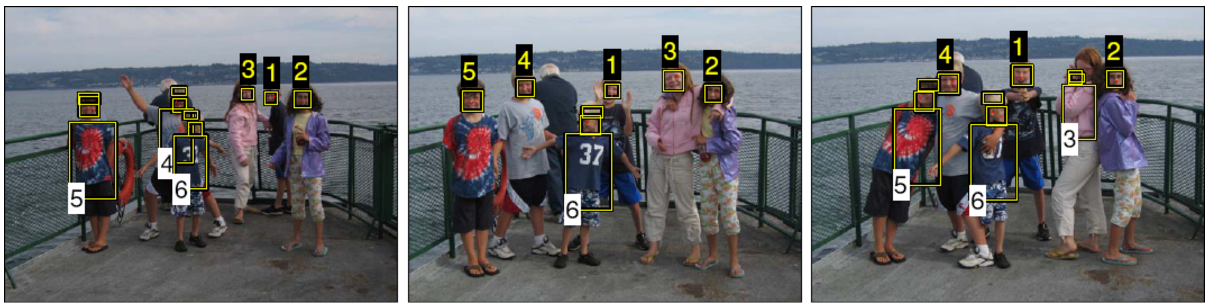
Figure 14.24 Head tracking with 3D AAMs (Matthews, Xiao, and Baker 2007) © 2007 Springer. Each image shows a video frame along with the estimate yaw, pitch, and roll parameters and the fitted 3D deformable mesh.

Hespanha, and Kriegman 1997; Moghaddam, Jebara, and Pentland 2000), is to compute separate statistics for intrapersonal and extrapersonal variation and then find discriminating directions in these subspaces. While AAMs have sometimes been used directly for recognition (Blanz and Vetter 2003), their main use in the context of recognition is to align faces into a canonical pose (Liang, Xiao, Wen *et al.* 2008) so that more traditional methods of face recognition (Penev and Atick 1996; Wiskott, Fellous, Krüger *et al.* 1997; Ahonen, Hadid, and Pietikäinen 2006; Zhao and Pietikäinen 2007; Cao, Yin, Tang *et al.* 2010) can be used. AAMs (or, actually, their simpler version, Active Shape Models (ASMs)) can also be used to align face images to perform automated morphing (Zanella and Fuentes 2004).

Active appearance models continue to be an active research area, with enhancements to deal with illumination and viewpoint variation (Gross, Baker, Matthews *et al.* 2005) as well as occlusions (Gross, Matthews, and Baker 2006). One of the most significant extensions is to construct 3D models of shape (Matthews, Xiao, and Baker 2007), which are much better at capturing and explaining the full variability of facial appearance across wide changes in pose.



(a)



(b)

Figure 14.25 Person detection and re-recognition using a combined face, hair, and torso model (Sivic, Zitnick, and Szeliski 2006) © 2006 Springer. (a) Using face detection alone, several of the heads are missed. (b) The combined face and clothing model successfully re-finds all the people.

Such models can be constructed either from monocular video sequences (Matthews, Xiao, and Baker 2007), as shown in Figure 14.24, or from multi-view video sequences (Ramnath, Koterba, Xiao *et al.* 2008), which provide even greater reliability and accuracy in reconstruction and tracking. (For a recent review of progress in head pose estimation, please see the survey paper by Murphy-Chutorian and Trivedi (2009).)

14.2.3 *Application*: Personal photo collections

In addition to digital cameras automatically finding faces to aid in auto-focusing and video cameras finding faces in video conferencing to center on the speaker (either mechanically or digitally), face detection has found its way into most consumer-level photo organization packages, such as iPhoto, Picasa, and Windows Live Photo Gallery. Finding faces and allowing users to tag them makes it easier to find photos of selected people at a later date or to automatically share them with friends. In fact, the ability to tag friends in photos is one of the more popular features on Facebook.

Sometimes, however, faces can be hard to find and recognize, especially if they are small, turned away from the camera, or otherwise occluded. In such cases, combining face recognition with person detection and clothes recognition can be very effective, as illustrated in Figure 14.25 (Sivic, Zitnick, and Szeliski 2006). Combining person recognition with other kinds of context, such as location recognition (Section 14.3.3) or activity or event recognition, can also help boost performance (Lin, Kapoor, Hua *et al.* 2010).

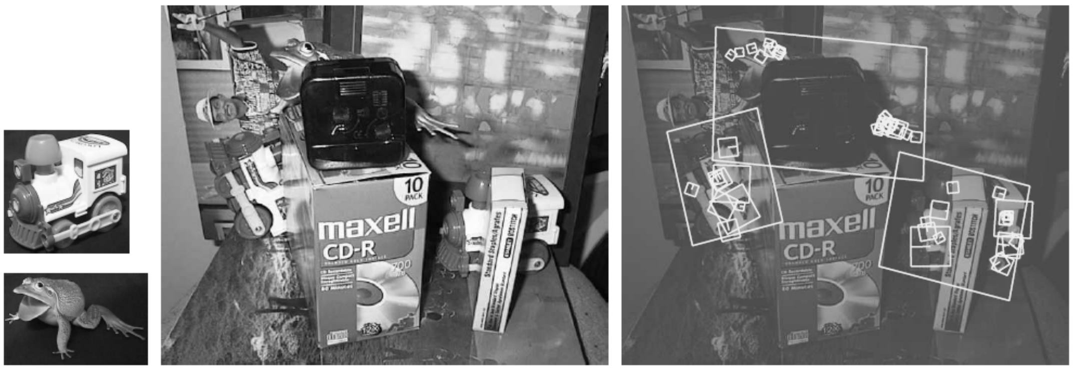


Figure 14.26 Recognizing objects in a cluttered scene (Lowe 2004) © 2004 Springer. Two of the training images in the database are shown on the left. They are matched to the cluttered scene in the middle using SIFT features, shown as small squares in the right image. The affine warp of each recognized database image onto the scene is shown as a larger parallelogram in the right image.

14.3 Instance recognition

General object recognition falls into two broad categories, namely *instance recognition* and *class recognition*. The former involves re-recognizing a known 2D or 3D rigid object, potentially being viewed from a novel viewpoint, against a cluttered background, and with partial occlusions. The latter, which is also known as *category-level* or *generic* object recognition (Ponce, Hebert, Schmid *et al.* 2006), is the much more challenging problem of recognizing any instance of a particular general class such as “cat”, “car”, or “bicycle”.

Over the years, many different algorithms have been developed for instance recognition. Mundy (2006) surveys earlier approaches, which focused on extracting lines, contours, or 3D surfaces from images and matching them to known 3D object models. Another popular approach was to acquire images from a large set of viewpoints and illuminations and to represent them using an eigenspace decomposition (Murase and Nayar 1995). More recent approaches (Lowe 2004; Rothganger, Lazebnik, Schmid *et al.* 2006; Ferrari, Tuytelaars, and Van Gool 2006b; Gordon and Lowe 2006; Obdržálek and Matas 2006; Sivic and Zisserman 2009) tend to use viewpoint-invariant 2D features, such as those we saw in Section 4.1.2. After extracting informative sparse 2D features from both the new image and the images in the database, image features are matched against the object database, using one of the sparse feature matching strategies described in Section 4.1.3. Whenever a sufficient number of matches have been found, they are verified by finding a geometric transformation that aligns the two sets of features (Figure 14.26).

Below, we describe some of the techniques that have been proposed for representing the geometric relationships between such features (Section 14.3.1). We also discuss how to make the feature matching process more efficient using ideas from text and information retrieval (Section 14.3.2).

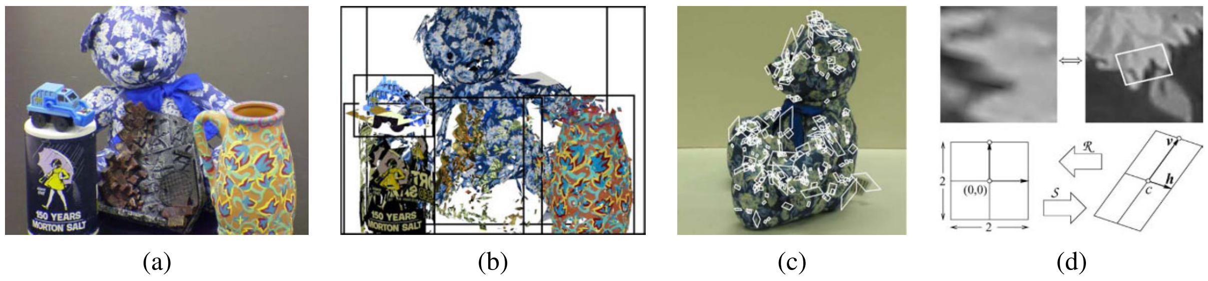


Figure 14.27 3D object recognition with affine regions (Rothganger, Lazebnik, Schmid *et al.* 2006) © 2006 Springer: (a) sample input image; (b) five of the recognized (reprojected) objects along with their bounding boxes; (c) a few of the local affine regions; (d) local affine region (patch) reprojected into a canonical (square) frame, along with its geometric affine transformations.

14.3.1 Geometric alignment

To recognize one or more instances of some known objects, such as those shown in the left column of Figure 14.26, the recognition system first extracts a set of interest points in each database image and stores the associated descriptors (and original positions) in an indexing structure such as a search tree (Section 4.1.3). At recognition time, features are extracted from the new image and compared against the stored object features. Whenever a sufficient number of matching features (say, three or more) are found for a given object, the system then invokes a *match verification* stage, whose job is to determine whether the spatial arrangement of matching features is consistent with those in the database image.

Because images can be highly cluttered and similar features may belong to several objects, the original set of feature matches can have a large number of outliers. For this reason, Lowe (2004) suggests using a Hough transform (Section 4.3.2) to accumulate votes for likely geometric transformations. In his system, he uses an affine transformation between the database object and the collection of scene features, which works well for objects that are mostly planar, or where at least several corresponding features share a quasi-planar geometry.[16]

Since SIFT features carry with them their own location, scale, and orientation, Lowe uses a four-dimensional similarity transformation as the original Hough binning structure, i.e., each bin denotes a particular location for the object center, scale, and in-plane rotation. Each matching feature votes for the nearest 2^4 bins and peaks in the transform are then selected for a more careful affine motion fit. Figure 14.26 (right image) shows three instances of the two objects on the left that were recognized by the system. Obdržálek and Matas (2006) generalize Lowe's approach to use feature descriptors with full local affine frames and evaluate their approach on a number of object recognition databases.

Another system that uses local affine frames is the one developed by Rothganger, Lazebnik, Schmid *et al.* (2006). In their system, the affine region detector of Mikolajczyk and Schmid (2004) is used to rectify local image patches (Figure 14.27d), from which both a SIFT descriptor and a 10×10 UV color histogram are computed and used for matching and recognition. Corresponding patches in different views of the same object, along with

[16] When a larger number of features is available, a full fundamental matrix can be used (Brown and Lowe 2002; Gordon and Lowe 2006). When image stitching is being performed (Brown and Lowe 2007), the motion models discussed in Section 9.1 can be used instead.

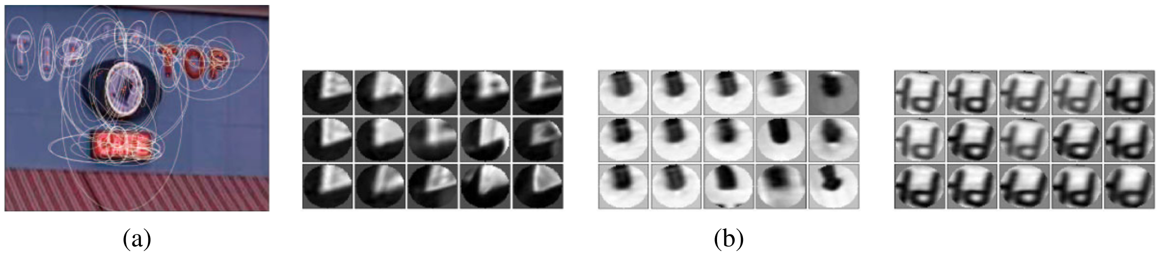


Figure 14.28 Visual words obtained from elliptical normalized affine regions (Sivic and Zisserman 2009) © 2009 IEEE. (a) Affine covariant regions are extracted from each frame and clustered into visual words using k-means clustering on SIFT descriptors with a learned Mahalanobis distance. (b) The central patch in each grid shows the query and the surrounding patches show the nearest neighbors.

their local affine deformations, are used to compute a 3D affine model for the object using an extension of the factorization algorithm of Section 7.3, which can then be upgraded to a Euclidean reconstruction (Tomasi and Kanade 1992).

At recognition time, local Euclidean neighborhood constraints are used to filter potential matches, in a manner analogous to the affine geometric constraints used by Lowe (2004) and Obdržálek and Matas (2006). Figure 14.27 shows the results of recognizing five objects in a cluttered scene using this approach.

While feature-based approaches are normally used to detect and localize known objects in scenes, it is also possible to get pixel-level segmentations of the scene based on such matches. Ferrari, Tuytelaars, and Van Gool (2006b) describe such a system for simultaneously recognizing objects and segmenting scenes, while Kannala, Rahtu, Brandt *et al.* (2008) extend this approach to non-rigid deformations. Section 14.4.3 re-visits this topic of joint recognition and segmentation in the context of generic class (category) recognition.

14.3.2 Large databases

As the number of objects in the database starts to grow large (say, millions of objects or video frames being searched), the time it takes to match a new image against each database image can become prohibitive. Instead of comparing the images one at a time, techniques are needed to quickly narrow down the search to a few likely images, which can then be compared using a more detailed and conservative verification stage.

The problem of quickly finding partial matches between documents is one of the central problems in *information retrieval* (IR) (Baeza-Yates and Ribeiro-Neto 1999; Manning, Raghavan, and Schütze 2008). The basic approach in fast document retrieval algorithms is to pre-compute an *inverted index* between individual words and the documents (or Web pages or news stories) where they occur. More precisely, the *frequency* of occurrence of particular words in a document is used to quickly find documents that match a particular query.

Sivic and Zisserman (2009) were the first to adapt IR techniques to visual search. In their Video Google system, affine invariant features are first detected in all the video frames they are indexing using both *shape adapted* regions around Harris feature points (Schaffalitzky and Zisserman 2002; Mikolajczyk and Schmid 2004) and maximally stable extremal regions (Matas, Chum, Urban *et al.* 2004), (Section 4.1.1), as shown in Figure 14.28a. Next, 128-

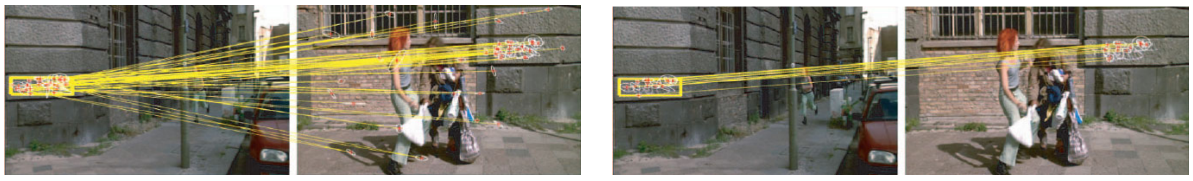


Figure 14.29 Matching based on visual words (Sivic and Zisserman 2009) © 2009 IEEE. (a) Features in the query region on the left are matched to corresponding features in a highly ranked video frame. (b) Results after removing the stop words and filtering the results using spatial consistency.

dimensional SIFT descriptors are computed from each normalized region (i.e., the patches shown in Figure 14.28b). Then, an average covariance matrix for these descriptors is estimated by accumulating statistics for features tracked from frame to frame. The feature descriptor covariance $\mathbf{\Sigma}$ is then used to define a Mahalanobis distance between feature descriptors,

$$d(\boldsymbol{x}_0, \boldsymbol{x}_1) = \|\boldsymbol{x}_0 - \boldsymbol{x}_1\|_{\mathbf{\Sigma}^{-1}} = \sqrt{(\boldsymbol{x}_0 - \boldsymbol{x}_1)^T \mathbf{\Sigma}^{-1} (\boldsymbol{x}_0 - \boldsymbol{x}_1)}. \tag{14.32}$$

In practice, feature descriptors are *whitened* by pre-multiplying them by $\mathbf{\Sigma}^{-1/2}$ so that Euclidean distances can be used.[17]

In order to apply fast information retrieval techniques to images, the high-dimensional feature descriptors that occur in each image must first be mapped into discrete *visual words*. Sivic and Zisserman (2003) perform this mapping using k-means clustering, while some of newer methods discussed below (Nistér and Stewénius 2006; Philbin, Chum, Isard *et al.* 2007) use alternative techniques, such as vocabulary trees or randomized forests. To keep the clustering time manageable, only a few hundred video frames are used to learn the cluster centers, which still involves estimating several thousand clusters from about 300,000 descriptors. At visual query time, each feature in a new query region (e.g., Figure 14.28a, which is a cropped region from a larger video frame) is mapped to its corresponding visual word. To keep very common patterns from contaminating the results, a *stop list* of the most common visual words is created and such words are dropped from further consideration.

Once a query image or region has been mapped into its constituent visual words, likely matching images or video frames must then be retrieved from the database. Information retrieval systems do this by matching word distributions (*term frequencies*) n_{id}/n_d between the query and target documents, where n_{id} is how many times word i occurs in document d, and n_d is the total number of words in document d. In order to downweight words that occur frequently and to focus the search on rarer (and hence, more informative) terms, an *inverse document frequency* weighting $\log N/N_i$ is applied, where N_i is the number of documents containing word i, and N is the total number of documents in the database. The combination of these two factors results in the *term frequency-inverse document frequency (tf-idf)* measure,

$$t_i = \frac{n_{id}}{n_d} \log \frac{N}{N_i}. \tag{14.33}$$

[17] Note that the computation of feature covariances from matched feature points is much more sensible than simply performing a PCA on the descriptor space (Winder and Brown 2007). This corresponds roughly to the *within-class* scatter matrix (14.17) we studied in Section 14.2.1.

1. **Vocabulary construction (off-line)**
   (a) Extract affine covariant regions from each database image.
   (b) Compute descriptors and optionally whiten them to make Euclidean distances meaningful (Sivic and Zisserman 2009).
   (c) Cluster the descriptors into visual words, either using k-means (Sivic and Zisserman 2009), hierarchical clustering (Nistér and Stewénius 2006), or randomized k-d trees (Philbin, Chum, Isard *et al.* 2007).
   (d) Decide which words are too common and put them in the stop list.
2. **Database construction (off-line)**
   (a) Compute term frequencies for the visual word in each image, document frequencies for each word, and normalized *tf-idf* vectors for each document.
   (b) Compute inverted indices from visual words to images (with word counts).
3. **Image retrieval (on-line)**
   (a) Extract regions, descriptors, and visual words, and compute a *tf-idf* vector for the query image or region.
   (b) Retrieve the top image candidates, either by exhaustively comparing sparse *tf-idf* vectors (Sivic and Zisserman 2009) or by using inverted indices to examine only a subset of the images (Nistér and Stewénius 2006).
   (c) Optionally re-rank or verify all the candidate matches, using either spatial consistency (Sivic and Zisserman 2009) or an affine (or simpler) transformation model (Philbin, Chum, Isard *et al.* 2007).
   (d) Optionally expand the answer set by re-submitting highly ranked matches as new queries (Chum, Philbin, Sivic *et al.* 2007).

Algorithm 14.2 Image retrieval using visual words (Sivic and Zisserman 2009; Nistér and Stewénius 2006; Philbin, Chum, Isard *et al.* 2007; Chum, Philbin, Sivic *et al.* 2007; Philbin, Chum, Sivic *et al.* 2008).

At match time, each document (or query region) is represented by its *tf-idf* vector,

$$\boldsymbol{t} = (t_1, \ldots, t_i, \ldots t_m). \tag{14.34}$$

The similarity between two documents is measured by the dot product between their corresponding normalized vectors $\hat{\boldsymbol{t}} = \boldsymbol{t}/\|\boldsymbol{t}\|$, which means that their dissimilarity is proportional to their Euclidean distance. In their journal paper, Sivic and Zisserman (2009) compare this simple metric to a dozen other metrics and conclude that it performs just about as well as more complicated metrics. Because the number of non-zero t_i terms in a typical query or document is small ($M \approx 200$) compared to the number of visual words ($V \approx 20,000$), the distance between pairs of (sparse) *tf-idf* vectors can be computed quite quickly.

After retrieving the top $N_s = 500$ documents based on word frequencies, Sivic and Zisserman (2009) re-rank these results using spatial consistency. This step involves taking every matching feature and counting the number of $k = 15$ nearest adjacent features that also match between the two documents. (This latter process is accelerated using inverted files, which we discuss in more detail below.) As shown in Figure 14.29, this step helps remove spurious false positive matches and produces a better estimate of which frames and regions in the video are actually true matches. Algorithm 14.2 summarizes the processing steps involved in image retrieval using visual words.

While this approach works well for tens of thousand of visual words and thousands of keyframes, as the size of the database continues to increase, both the time to quantize each feature and to find potential matching frames or images can become prohibitive. Nistér and Stewénius (2006) address this problem by constructing a hierarchical *vocabulary tree*, where feature vectors are hierarchically clustered into a k-way tree of prototypes. (This technique is also known as *tree-structured vector quantization* (Gersho and Gray 1991).) At both database construction time and query time, each descriptor vector is compared to several prototypes at a given level in the vocabulary tree and the branch with the closest prototype is selected for further refinement (Figure 14.30). In this way, vocabularies with millions (10^6) of words can be supported, which enables individual words to be far more discriminative, while only requiring $10 \cdot 6$ comparisons for quantizing each descriptor.

At query time, each node in the vocabulary tree keeps its own inverted file index, so that features that match a particular node in the tree can be rapidly mapped to potential matching images. (Interior leaf nodes just use the inverted indices of their corresponding leaf-node descendants.) To score a particular query *tf-idf* vector $\boldsymbol{t}_q$ against all document vectors $\{\boldsymbol{t}_j\}$ using an L_p metric,[18] the non-zero t_{iq} entries in $\boldsymbol{t}_q$ are used to fetch corresponding non-zero t_{ij} entries, and the L_p norm is efficiently computed as

$$\|\boldsymbol{t}_q - \boldsymbol{t}_j\|_p^p = 2 + \sum_{i|t_{iq}>0 \wedge t_{ij}>0} (|t_{iq} - t_{ij}|^p - |t_{iq}|^p - |t_{ij}|^p). \tag{14.35}$$

In order to mitigate quantization errors due to noise in the descriptor vectors, Nistér and Stewénius (2006) not only score leaf nodes in the vocabulary tree (corresponding to visual words), but also score interior nodes in the tree, which correspond to clusters of similar visual words.

Because of the high efficiency in both quantizing and scoring features, their vocabulary-tree-based recognition system is able to process incoming images in real time against a

[18] In their actual implementation, Nistér and Stewénius (2006) use an L_1 metric.

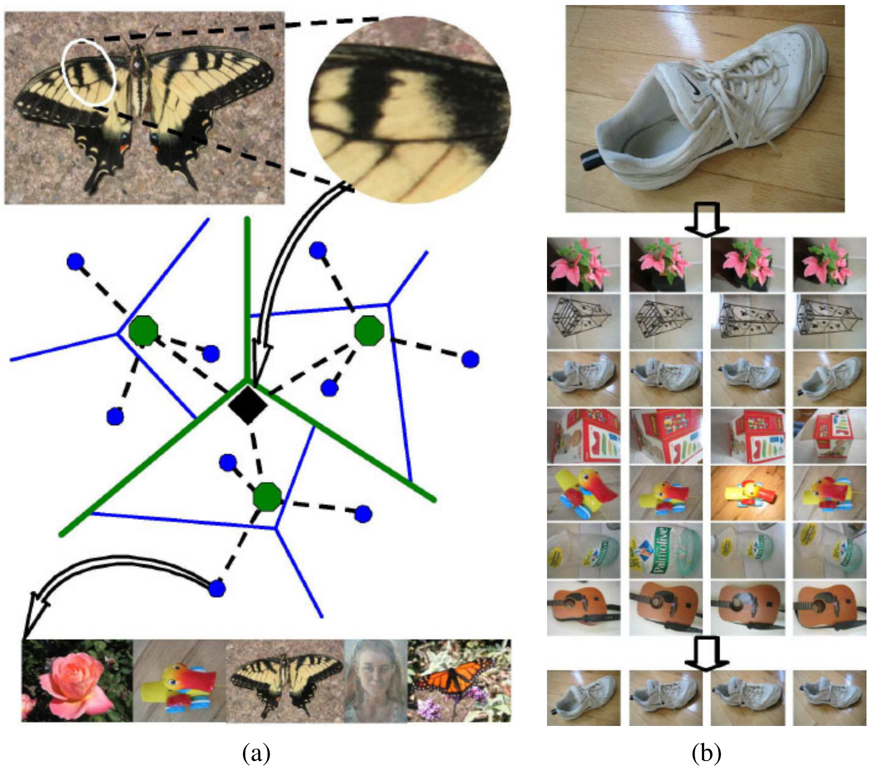


Figure 14.30 Scalable recognition using a vocabulary tree (Nistér and Stewénius 2006) © 2006 IEEE. (a) Each MSER elliptical region is converted into a SIFT descriptor, which is then quantized by comparing it hierarchically to some prototype descriptors in a vocabulary tree. Each leaf node stores its own inverted index (sparse list of non-zero *tf-idf* counts) into images that contain that feature. (b) A recognition result, showing a query image (top row) being indexed into a database of 6000 test images and correctly finding the corresponding four images.

database of 40,000 CD covers and at 1Hz when matching a database of one million frames taken from six feature-length movies. Figure 14.30b shows some typical images from the database of objects taken under varying viewpoints and illumination that was used to train and test the vocabulary tree recognition system.

The state of the art in instance recognition continues to improve rapidly. Philbin, Chum, Isard *et al.* (2007) have shown that randomized forest of k-d trees perform better than vocabulary trees on a large location recognition task (Figure 14.31). They also compare the effects of using different 2D motion models (Section 2.1.2) in the verification stage. In follow-on work, Chum, Philbin, Sivic *et al.* (2007) apply another idea from information retrieval, namely *query expansion*, which involves re-submitting top-ranked images from the initial query as additional queries to generate additional candidate results, to further improve recognition rates for difficult (occluded or oblique) examples. Philbin, Chum, Sivic *et al.* (2008) show how to mitigate quantization problems in visual words selection using *soft assignment*, where each feature descriptor is mapped to a number of visual words based on its distance from the cluster prototypes. The soft weights derived from these distances are used, in turn, to weight the counts used in the *tf-idf* vectors and to retrieve additional images for later verification.

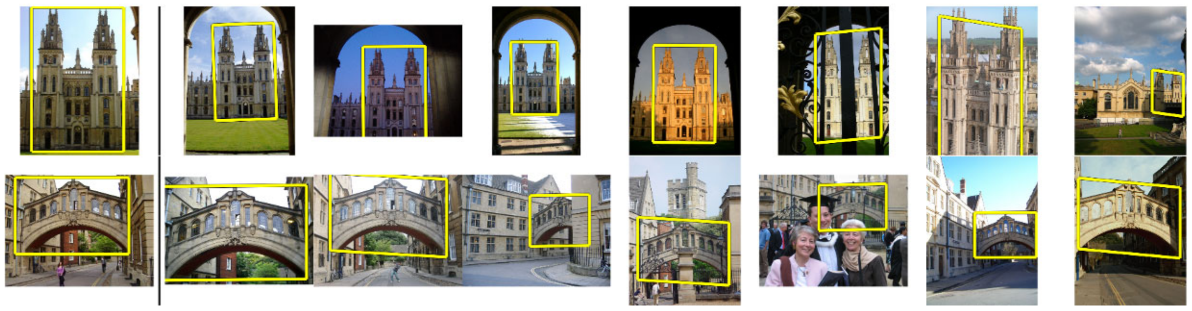


Figure 14.31 Location or building recognition using randomized trees (Philbin, Chum, Isard *et al.* 2007) © 2007 IEEE. The left image is the query, the other images are the highest-ranked results.

Taken together, these recent advances hold the promise of extending current instance recognition algorithms to performing Web-scale retrieval and matching tasks (Agarwal, Snavely, Simon *et al.* 2009; Agarwal, Furukawa, Snavely *et al.* 2010; Snavely, Simon, Goesele *et al.* 2010).

14.3.3 *Application*: Location recognition

One of the most exciting applications of instance recognition today is in the area of location recognition, which can be used both in desktop applications (where did I take this holiday snap?) and in mobile (cell-phone) applications. The latter case includes not only finding out your current location based on a cell-phone image but also providing you with navigation directions or annotating your images with useful information, such as building names and restaurant reviews (i.e., a portable form of *augmented reality*).

Some approaches to location recognition assume that the photos consist of architectural scenes for which vanishing directions can be used to pre-rectify the images for easier matching (Robertson and Cipolla 2004). Other approaches use general affine covariant interest points to perform *wide baseline matching* (Schaffalitzky and Zisserman 2002). The Photo Tourism system of Snavely, Seitz, and Szeliski (2006) (Section 13.1.2) was the first to apply these kinds of ideas to large-scale image matching and (implicit) location recognition from Internet photo collections taken under a wide variety of viewing conditions.

The main difficulty in location recognition is in dealing with the extremely large community (user-generated) photo collections on Web sites such as Flickr (Philbin, Chum, Isard *et al.* 2007; Chum, Philbin, Sivic *et al.* 2007; Philbin, Chum, Sivic *et al.* 2008; Turcot and Lowe 2009) or commercially captured databases (Schindler, Brown, and Szeliski 2007). The prevalence of commonly appearing elements such as foliage, signs, and common architectural elements further complicates the task. Figure 14.31 shows some results on location recognition from community photo collections, while Figure 14.32 shows sample results from denser commercially acquired datasets. In the latter case, the overlap between adjacent database images can be used to verify and prune potential matches using "temporal" filtering, i.e., requiring the query image to match nearby overlapping database images before accepting the match.

Another variant on location recognition is the automatic discovery of *landmarks*, i.e.,

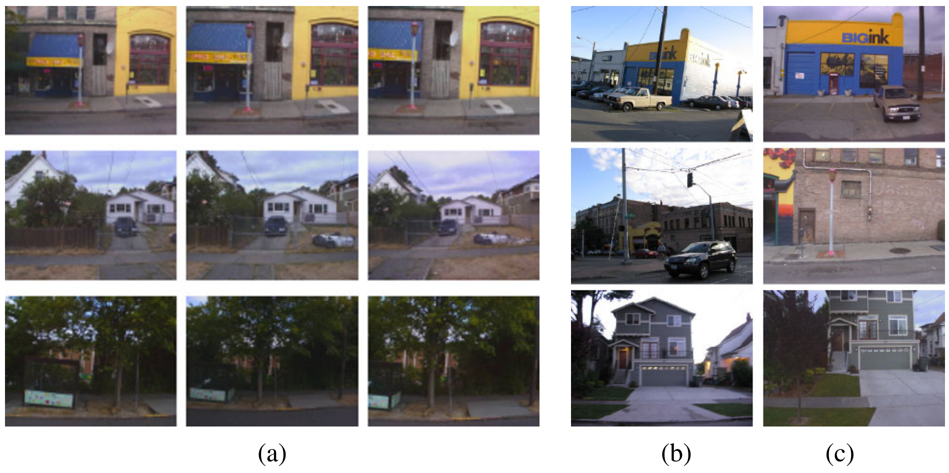


Figure 14.32 Feature-based location recognition (Schindler, Brown, and Szeliski 2007) © 2007 IEEE: (a) three typical series of overlapping street photos; (b) handheld camera shots and (c) their corresponding database photos.

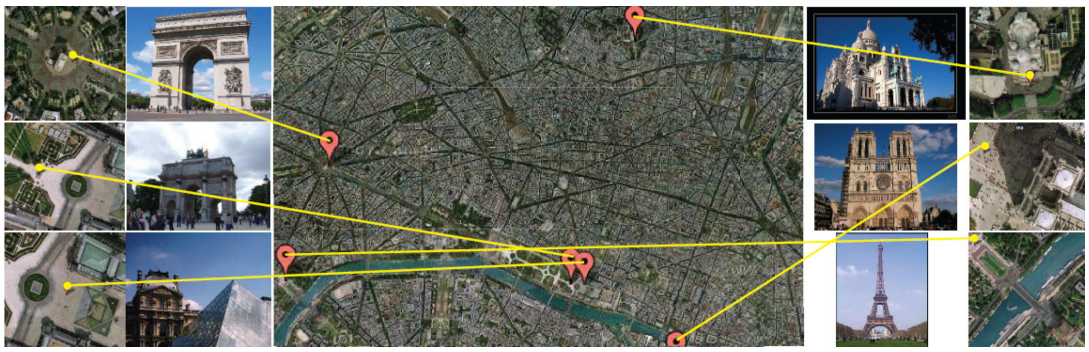


Figure 14.33 Automatic mining, annotation, and localization of community photo collections (Quack, Leibe, and Van Gool 2008) © 2008 ACM. This figure does not show the textual annotations or corresponding Wikipedia entries, which are also discovered.

frequently photographed objects and locations. Simon, Snavely, and Seitz (2007) show how these kinds of objects can be discovered simply by analyzing the matching graph constructed as part of the 3D modeling process in Photo Tourism. More recent work has extended this approach to larger data sets using efficient clustering techniques (Philbin and Zisserman 2008; Li, Wu, Zach *et al.* 2008; Chum, Philbin, and Zisserman 2008; Chum and Matas 2010) as well as combining meta-data such as GPS and textual tags with visual search (Quack, Leibe, and Van Gool 2008; Crandall, Backstrom, Huttenlocher *et al.* 2009), as shown in Figure 14.33. It is now even possible to automatically associate object tags with images based on their co-occurrence in multiple loosely tagged images (Simon and Seitz 2008; Gammeter, Bossard, Quack *et al.* 2009).

The concept of organizing the world's photo collections by location has even been recently extended to organizing all of the universe's (astronomical) photos in an application called *astrometry*, http://astrometry.net/. The technique used to match any two star fields is

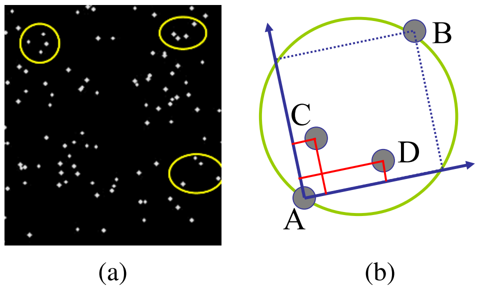


Figure 14.34 Locating star fields using astrometry, http://astrometry.net/. (a) Input star field and some selected star quads. (b) The 2D coordinates of stars C and D are encoded relative to the unit square defined by A and B.

to take quadruplets of nearby stars (a pair of stars and another pair inside their diameter) to form a 30-bit *geometric hash* by encoding the relative positions of the second pair of points using the inscribed square as the reference frame, as shown in Figure 14.34. Traditional information retrieval techniques (k-d trees built for different parts of a sky atlas) are then used to find matching quads as potential star field location hypotheses, which can then be verified using a similarity transform.

14.4 Category recognition

While instance recognition techniques are relatively mature and are used in commercial applications, such as Photosynth (Section 13.1.2), generic category (class) recognition is still a largely unsolved problem. Consider for example the set of photographs in Figure 14.35, which shows objects taken from 10 different visual categories. (I'll leave it up to you to name each of the categories.) How would you go about writing a program to categorize each of these images into the appropriate class, especially if you were also given the choice "none of the above"?

As you can tell from this example, visual category recognition is an *extremely* challenging problem; no one has yet constructed a system that approaches the performance level of a two-year-old child. However, the progress in the field has been quite dramatic, if judged by how much better today's algorithms are compared to those of a decade ago.

Figure 14.54 shows a sample image from each of the 20 categories used in the 2008 PASCAL Visual Object Classes Challenge. The yellow boxes represent the extent of each of the objects found in a given image. On such *closed world* collections where the task is to decide among 20 categories, today's classification algorithms can do remarkably well.

In this section, we look at a number of approaches to solving category recognition. While historically, *part-based* representations and recognition algorithms (Section 14.4.2) were the preferred approach (Fischler and Elschlager 1973; Felzenszwalb and Huttenlocher 2005; Fergus, Perona, and Zisserman 2007), we begin by describing simpler *bag-of-features* approaches (Section 14.4.1) that represent objects and images as unordered collections of feature descriptors. We then look at the problem of simultaneously segmenting images while recognizing objects (Section 14.4.3) and also present some applications of such techniques to photo manipulation (Section 14.4.4). In Section 14.5, we look at how context and scene un-

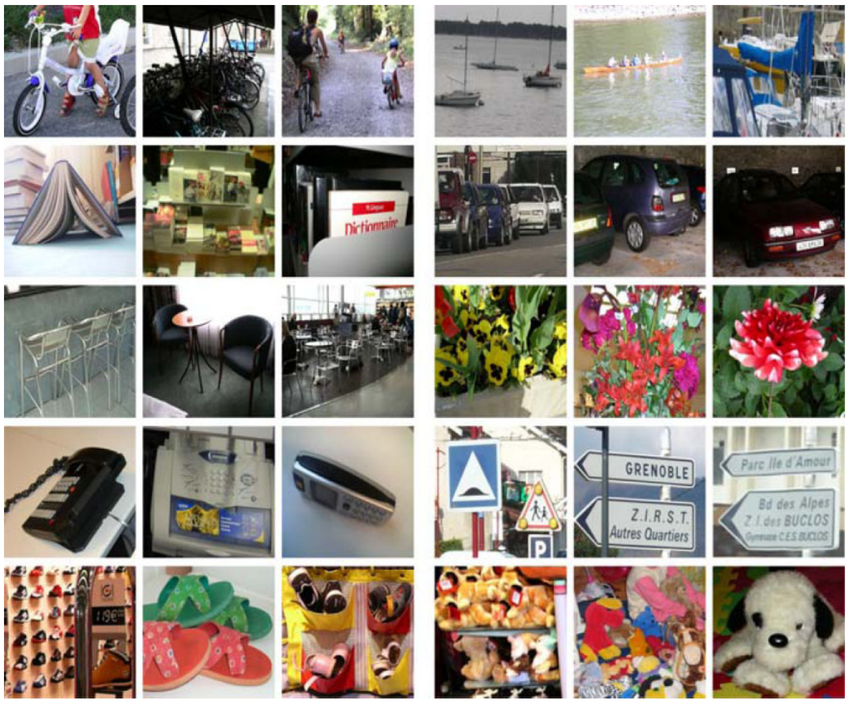


Figure 14.35 Sample images from the Xerox 10 class dataset (Csurka, Dance, Perronnin *et al.* 2006) © 2007 Springer. Imagine trying to write a program to distinguish such images from other photographs.

derstanding, as well as machine learning, can improve overall recognition results. Additional details on the techniques presented in this section can be found in (Pinz 2005; Ponce, Hebert, Schmid *et al.* 2006; Dickinson, Leonardis, Schiele *et al.* 2007; Fei-Fei, Fergus, and Torralba 2009).

14.4.1 Bag of words

One of the simplest algorithms for category recognition is the *bag of words* (also known as *bag of features* or *bag of keypoints*) approach (Csurka, Dance, Fan *et al.* 2004; Lazebnik, Schmid, and Ponce 2006; Csurka, Dance, Perronnin *et al.* 2006; Zhang, Marszalek, Lazebnik *et al.* 2007). As shown in Figure 14.36, this algorithm simply computes the distribution (histogram) of visual words found in the query image and compares this distribution to those found in the training images. We have already seen elements of this approach in Section 14.3.2, Equations (14.33–14.35) and Algorithm 14.2. The biggest difference from instance recognition is the absence of a geometric verification stage (Section 14.3.1), since individual instances of generic visual categories, such as those shown in Figure 14.35, have relatively little spatial coherence to their features (but see the work by Lazebnik, Schmid, and Ponce (2006)).

Csurka, Dance, Fan *et al.* (2004) were the first to use the term *bag of keypoints* to describe such approaches and among the first to demonstrate the utility of frequency-based techniques for category recognition. Their original system used affine covariant regions and SIFT de-

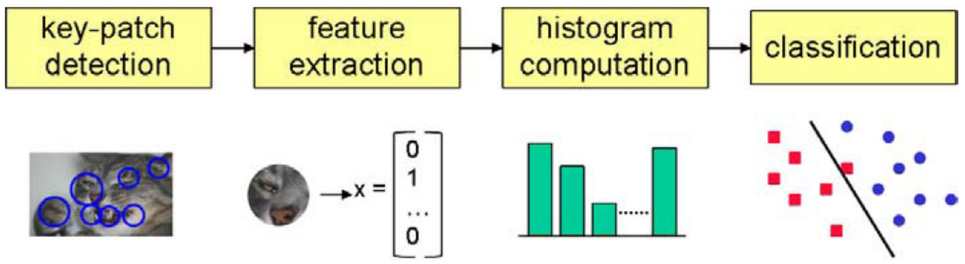


Figure 14.36 A typical processing pipeline for a bag-of-words category recognition system (Csurka, Dance, Perronnin *et al.* 2006) © 2007 Springer. Features are first extracted at keypoints and then quantized to get a distribution (histogram) over the learned *visual words* (feature cluster centers). The feature distribution histogram is used to learn a decision surface using a classification algorithm, such as a support vector machine.

scriptors, k-means visual vocabulary construction, and both a naïve Bayesian classifier and support vector machines for classification. (The latter was found to perform better.) Their newer system (Csurka, Dance, Perronnin *et al.* 2006) uses regular (non-affine) SIFT patches, boosting instead of SVMs, and incorporates a small amount of geometric consistency information.

Zhang, Marszalek, Lazebnik *et al.* (2007) perform a more detailed study of such bag of features systems. They compare a number of feature detectors (Harris–Laplace (Mikolajczyk and Schmid 2004) and Laplacian (Lindeberg 1998b)), descriptors (SIFT, RIFT, and SPIN (Lazebnik, Schmid, and Ponce 2005)), and SVM kernel functions. To estimate distances for the kernel function, they form an *image signature*

$$S = ((t_1, \boldsymbol{m}_1), \ldots, (t_m, \boldsymbol{m}_m)), \tag{14.36}$$

analogous to the *tf-idf* vector $\boldsymbol{t}$ in (14.34), where the cluster centers $\boldsymbol{m}_i$ are made explicit. They then investigate two different kernels for comparing such image signatures. The first is the *earth mover's distance* (EMD) (Rubner, Tomasi, and Guibas 2000),

$$EMD(S, S') = \frac{\sum_i \sum_j f_{ij} d(\boldsymbol{m}_i, \boldsymbol{m}'_j)}{\sum_i \sum_j f_{ij}}, \tag{14.37}$$

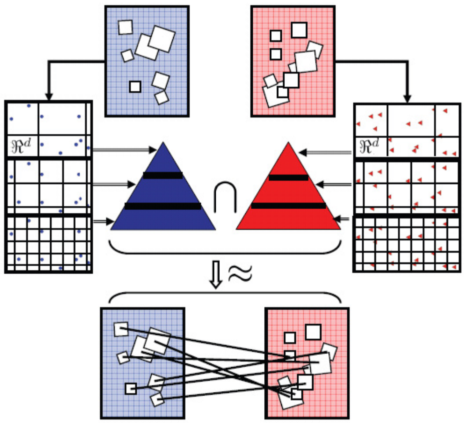
where f_{ij} is a *flow* value that can be computed using a linear program and $d(\boldsymbol{m}_i, \boldsymbol{m}'_j)$ is the *ground distance* (Euclidean distance) between $\boldsymbol{m}_i$ and $\boldsymbol{m}'_j$. Note that the EMD can be used to compare two signatures of different lengths, where the entries do not need to correspond. The second is a χ^2 distance

$$\chi^2(S, S') = \frac{1}{2} \sum_i \frac{(t_i - t'_i)^2}{t_i + t'_i}, \tag{14.38}$$

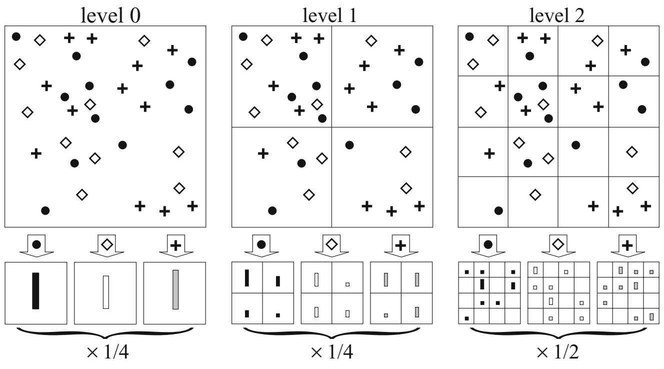
which measures the likelihood that the two signatures were generated from consistent random processes. These distance metrics are then converted into SVM kernels using a generalized Gaussian kernel

$$K(S, S') = \exp\left(-\frac{1}{A} D(S, S')\right), \tag{14.39}$$

where A is a scaling parameter set to the mean distance between training images. In their experiments, they find that the EMD works best for visual category recognition and the χ^2 measure is best for texture recognition.



(a)



(b)

Figure 14.37 Comparing collections of feature vectors using pyramid matching. (a) The feature-space pyramid match kernel (Grauman and Darrell 2007b) constructs a pyramid in high-dimensional feature space and uses it to compute distances (and implicit correspondences) between sets of feature vectors. (b) Spatial pyramid matching (Lazebnik, Schmid, and Ponce 2006) © 2006 IEEE divides the image into a pyramid of pooling regions and computes separate visual word histograms (distributions) inside each spatial bin.

Instead of quantizing feature vectors to visual words, Grauman and Darrell (2007b) develop a technique for directly computing an approximate distance between two variably sized collections of feature vectors. Their approach is to bin the feature vectors into a multi-resolution pyramid defined in feature space (Figure 14.37a) and count the number of features that land in corresponding bins B_{il} and B'_{il} (Figure 14.38a–c). The distance between the two sets of feature vectors (which can be thought of as points in a high-dimensional space) is computed using histogram intersection between corresponding bins

$$C_l = \sum_i \min(B_{il}, B'_{il}) \tag{14.40}$$

(Figure 14.38d). These per-level counts are then summed up in a weighted fashion

$$D_\triangle = \sum_l w_l N_l \quad \text{with} \quad N_l = C_l - C_{l-1} \quad \text{and} \quad w_l = \frac{1}{d2^l} \tag{14.41}$$

(Figure 14.38e), which discounts matches already found at finer levels while weighting finer matches more heavily. (d is the dimension of the embedding space, i.e., the length of the feature vectors.) In follow-on work, Grauman and Darrell (2007a) show how an explicit construction of the pyramid can be avoided using hashing techniques.

Inspired by this work, Lazebnik, Schmid, and Ponce (2006) show how a similar idea can be employed to augment bags of keypoints with loose notions of 2D spatial location analogous to the pooling performed by SIFT (Lowe 2004) and "gist" (Torralba, Murphy, Freeman *et al.* 2003). In their work, they extract affine region descriptors (Lazebnik, Schmid, and Ponce 2005) and quantize them into visual words. (Based on previous results by Fei-Fei and Perona (2005), the feature descriptors are extracted densely (on a regular grid) over the image, which can be helpful in describing textureless regions such as the sky.) They then form

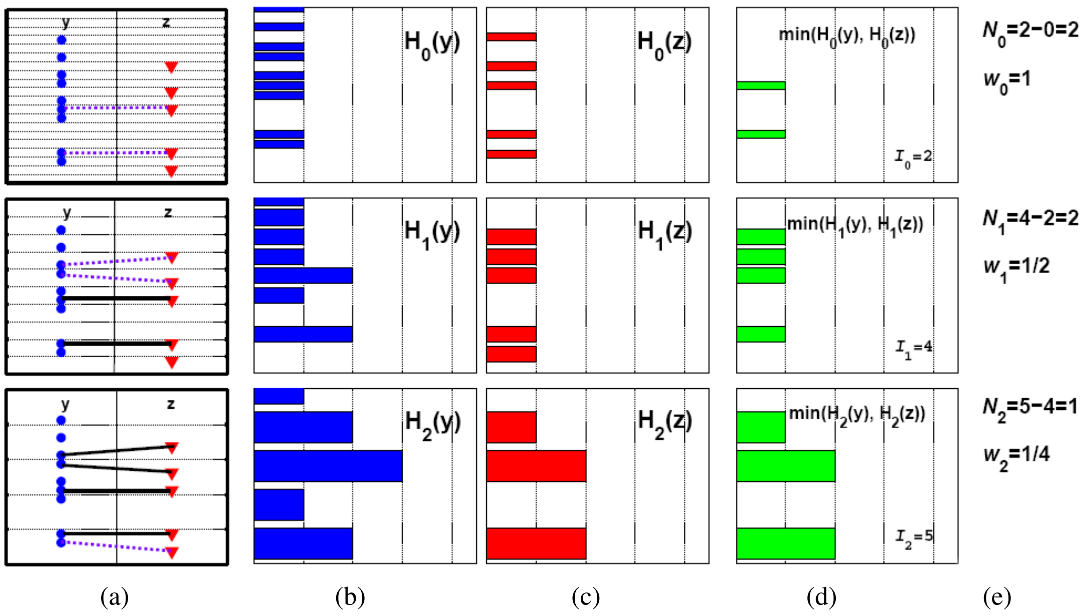


Figure 14.38 A one-dimensional illustration of comparing collections of feature vectors using the pyramid match kernel (Grauman and Darrell 2007b): (a) distribution of feature vectors (point sets) into the pyramidal bins; (b–c) histogram of point counts in bins B_{il} and B'_{il} for the two images; (d) histogram intersections (minimum values); (e) per-level similarity scores, which are weighted and summed to form the final distance/similarity metric.

a spatial pyramid of bins containing word counts (histograms), as shown in Figure 14.37b, and use a similar pyramid match kernel to combine histogram intersection counts in a hierarchical fashion.

The debate about whether to use quantized feature descriptors or continuous descriptors and also whether to use sparse or dense features continues to this day. Boiman, Shechtman, and Irani (2008) show that if query images are compared to *all* the features representing a given class, rather than just each class image individually, nearest-neighbor matching followed by a naïve Bayes classifier outperforms quantized visual words (Figure 14.39). Instead of using generic feature detectors and descriptors, some authors have been investigating *learning* class-specific features (Ferencz, Learned-Miller, and Malik 2008), often using randomized forests (Philbin, Chum, Isard *et al.* 2007; Moosmann, Nowak, and Jurie 2008; Shotton, Johnson, and Cipolla 2008) or combining the feature generation and image classification stages (Yang, Jin, Sukthankar *et al.* 2008). Others, such as Serre, Wolf, and Poggio (2005) and Mutch and Lowe (2008) use hierarchies of dense feature transforms inspired by biological (visual cortical) processing combined with SVMs for final classification.

14.4.2 Part-based models

Recognizing an object by finding its constituent parts and measuring their geometric relationships is one of the oldest approaches to object recognition (Fischler and Elschlager 1973; Kanade 1977; Yuille 1991). We have already seen examples of part-based approaches being used for face recognition (Figure 14.18) (Moghaddam and Pentland 1997; Heisele, Ho, Wu

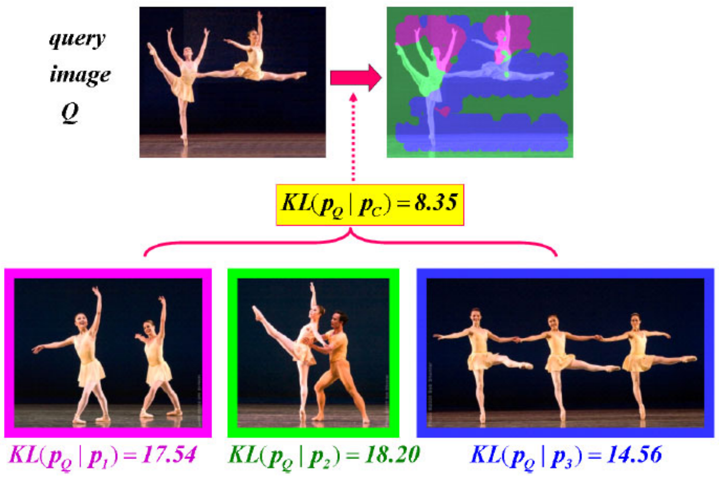

query
image
Q
$KL(p_Q \mid p_C) = 8.35$
$KL(p_Q \mid p_1) = 17.54$
$KL(p_Q \mid p_2) = 18.20$
$KL(p_Q \mid p_3) = 14.56$

Figure 14.39 "Image-to-Image" vs. "Image-to-Class" distance comparison (Boiman, Shechtman, and Irani 2008) © 2008 IEEE. The query image on the upper left may not match the feature distribution of any of the database images in the bottom row. However, if each feature in the query is matched to its closest analog in *all* the class images, a good match can be found.

et al. 2003; Heisele, Serre, and Poggio 2007) and pedestrian detection (Figure 14.9) (Felzenszwalb, McAllester, and Ramanan 2008).

In this section, we look more closely at some of the central issues in part-based recognition, namely, the representation of geometric relationships, the representation of individual parts, and algorithms for learning such descriptions and recognizing them at run time. More details on part-based models for recognition can be found in the course notes of Fergus (2007b, 2009).

The earliest approaches to representing geometric relationships were dubbed *pictorial structures* by Fischler and Elschlager (1973) and consisted of spring-like connections between different feature locations (Figure 14.1a). To fit a pictorial structure to an image, an energy function of the form

$$E = \sum_i V_i(\boldsymbol{l}_i) + \sum_{ij \in E} V_{ij}(\boldsymbol{l}_i, \boldsymbol{l}_j) \tag{14.42}$$

is minimized over all potential part locations or poses $\{\boldsymbol{l}_i\}$ and pairs of parts (i, j) for which an edge (geometric relationship) exists in E. Note how this energy is closely related to that used with Markov random fields (3.108–3.109), which can be used to embed pictorial structures in a probabilistic framework that makes parameter learning easier (Felzenszwalb and Huttenlocher 2005).

Part-based models can have different topologies for the geometric connections between the parts (Figure 14.41). For example, Felzenszwalb and Huttenlocher (2005) restrict the connections to a tree (Figure 14.41d), which makes learning and inference more tractable. A tree topology enables the use of a recursive Viterbi (dynamic programming) algorithm (Pearl 1988; Bishop 2006), in which leaf nodes are first optimized as a function of their parents, and the resulting values are then plugged in and eliminated from the energy function—see Appendix B.5.2. The Viterbi algorithm computes an optimal match in $O(N^2|E| + NP)$ time,

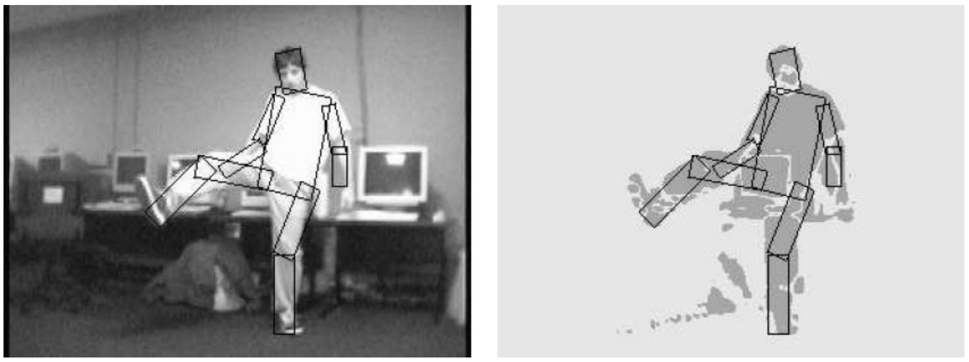


Figure 14.40 Using pictorial structures to locate and track a person (Felzenszwalb and Huttenlocher 2005) © 2005 Springer. The structure consists of articulated rectangular body parts (torso, head, and limbs) connected in a tree topology that encodes relative part positions and orientations. To fit a pictorial structure model, a binary silhouette image is first computed using background subtraction.

where N is the number of potential locations or poses for each part, $|E|$ is the number of edges (pairwise constraints), and $P = |V|$ is the number of parts (vertices in the graphical model, which is equal to $|E| + 1$ in a tree). To further increase the efficiency of the inference algorithm, Felzenszwalb and Huttenlocher (2005) restrict the pairwise energy functions $V_{ij}(\boldsymbol{l}_i, \boldsymbol{l}_j)$ to be Mahalanobis distances on functions of location variables and then use fast distance transform algorithms to minimize each pairwise interaction in time that is closer to linear in N.

Figure 14.40 shows the results of using their pictorial structures algorithm to fit an articulated body model to a binary image obtained by background segmentation. In this application of pictorial structures, parts are parameterized by the locations, sizes, and orientations of their approximating rectangles. Unary matching potentials $V_i(\boldsymbol{l}_i)$ are determined by counting the percentage of foreground and background pixels inside and just outside the tilted rectangle representing each part.

Over the last decade, a large number of different graphical models have been proposed for part-based recognition, as shown in Figure 14.41. Carneiro and Lowe (2006) discuss a number of these models and propose one of their own, which they call a *sparse flexible model*; it involves ordering the parts and having each part's location depend on at most k of its ancestor locations.

The simplest models, which we saw in Section 14.4.1, are bags of words, where there are no geometric relationships between different parts or features. While such models can be very efficient, they have a very limited capacity to express the spatial arrangement of parts. Trees and stars (a special case of trees where all leaf nodes are directly connected to a common root) are the most efficient in terms of inference and hence also learning (Felzenszwalb and Huttenlocher 2005; Fergus, Perona, and Zisserman 2005; Felzenszwalb, McAllester, and Ramanan 2008). Directed acyclic graphs (Figure 14.41f–g) come next in terms of complexity and can still support efficient inference, although at the cost of imposing a causal structure on the part model (Bouchard and Triggs 2005; Carneiro and Lowe 2006). k-fans, in which a clique of size k forms the root of a star-shaped model (Figure 14.41c) have inference complexity $O(N^{k+1})$, although with distance transforms and Gaussian priors, this can be lowered to

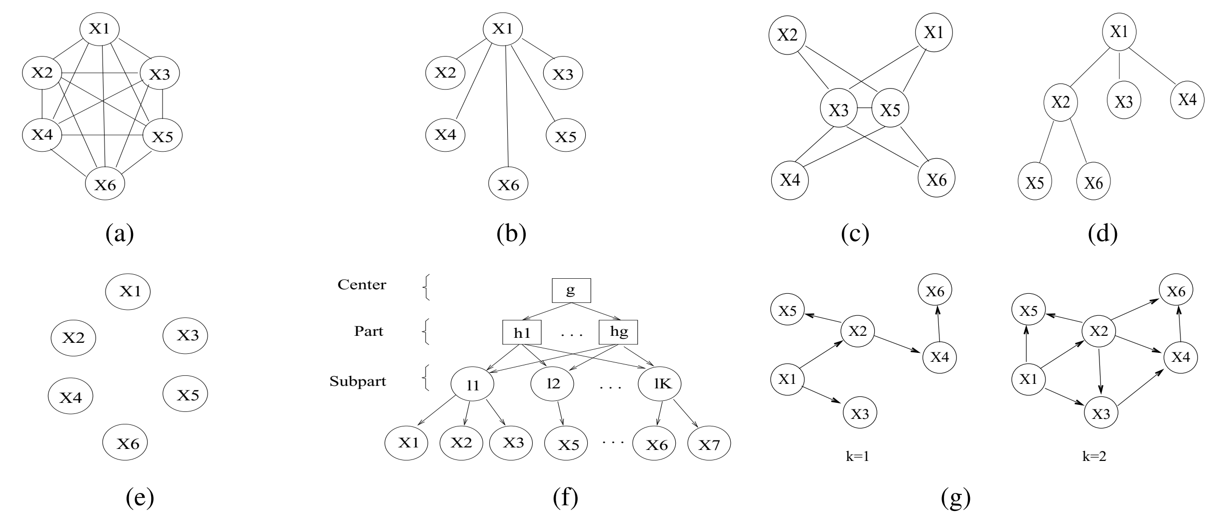


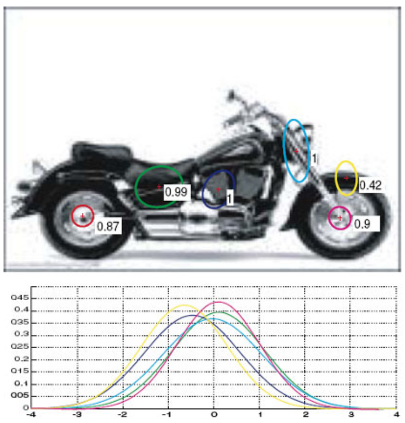
Figure 14.41 Graphical models for geometric spatial priors (Carneiro and Lowe 2006) © 2006 Springer: (a) constellation (Fergus, Perona, and Zisserman 2007); (b) star (Crandall, Felzenszwalb, and Huttenlocher 2005; Fergus, Perona, and Zisserman 2005); (c) k-fan ($k = 2$) (Crandall, Felzenszwalb, and Huttenlocher 2005); (d) tree (Felzenszwalb and Huttenlocher 2005); (e) bag of features (Csurka, Dance, Fan *et al.* 2004); (f) hierarchy (Bouchard and Triggs 2005); (g) sparse flexible model (Carneiro and Lowe 2006).

$O(N^k)$ (Crandall, Felzenszwalb, and Huttenlocher 2005; Crandall and Huttenlocher 2006). Finally, fully connected *constellation* models (Figure 14.41a) are the most general, but the assignment of features to parts becomes intractable for moderate numbers of parts P, since the complexity of such an assignment is $O(N^P)$ (Fergus, Perona, and Zisserman 2007).

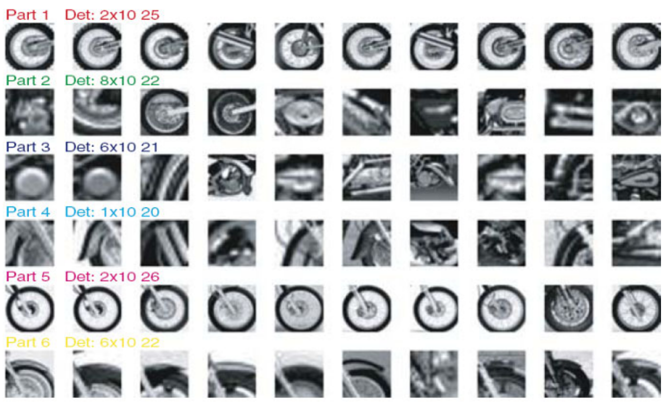
The original constellation model was developed by Burl, Weber, and Perona (1998) and consists of a number of parts whose relative positions are encoded by their mean locations and a full covariance matrix, which is used to denote not only positional uncertainty but also potential correlations (covariance) between different parts (Figure 14.42a). Weber, Welling, and Perona (2000) extended this technique to a weakly supervised setting, where both the appearance of each part and its locations are automatically learned given only whole image labels. Fergus, Perona, and Zisserman (2007) further extend this approach to simultaneous learning of appearance and shape models from scale-invariant keypoint detections.

Figure 14.42a shows the shape model learned for the motorcycle class. The top figure shows the mean relative locations for each part along with their position covariances (inter-part covariances are not shown) and likelihood of occurrence. The bottom curve shows the Gaussian PDFs for the relative log-scale of each part with respect to the "landmark" feature. Figure 14.42b shows the appearance model learned for each part, visualized as the patches around detected features in the training database that best match the appearance model. Figure 14.42c shows the features detected in the test database (pink dots) along with the corresponding parts that they were assigned to (colored circles). As you can see, the system has successfully learned and then used a fairly complex model of motorcycle appearance.

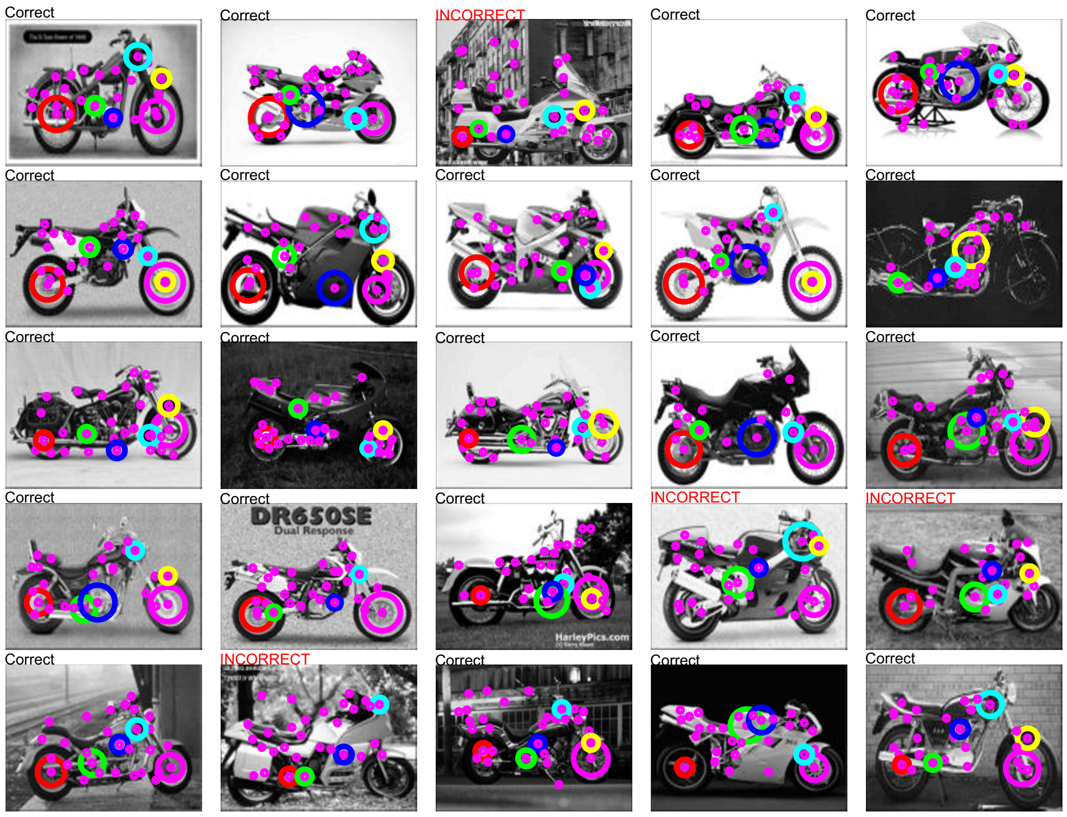
The part-based approach to recognition has also been extended to learning new categories from small numbers of examples, building on recognition components developed for other classes (Fei-Fei, Fergus, and Perona 2006). More complex hierarchical part-based models can



(a)



(b)



(c)

Figure 14.42 Part-based recognition (Fergus, Perona, and Zisserman 2007) © 2007 Springer: (a) locations and covariance ellipses for each part, along with their occurrence probabilities (top) and relative log-scale densities (bottom); (b) part examples drawn from the training images that best match the average appearance; (c) recognition results for the motorcycle class, showing detected features (pink dots) and parts (colored circles).

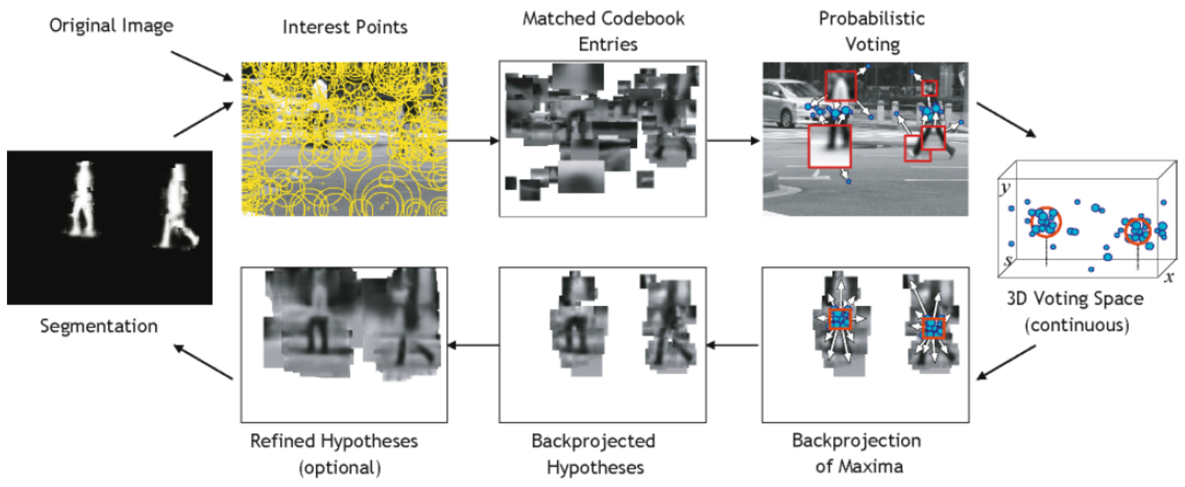


Figure 14.43 Interleaved recognition and segmentation (Leibe, Leonardis, and Schiele 2008) © 2008 Springer. The process starts by re-recognizing visual words (codebook entries) in a new image (scene) and having each part vote for likely locations and size in a 3D (x, y, s) voting space (top row). Once a maximum has been found, the parts (features) corresponding to this instance are determined by *backprojecting* the contributing votes. The foreground–background segmentation for each object can be found by backprojecting probabilistic masks associated with each codebook entry. The whole recognition and segmentation process can then be repeated.

be developed using the concept of grammars (Bouchard and Triggs 2005; Zhu and Mumford 2006). A simpler way to use parts is to have keypoints that are recognized as being part of a class vote for the estimated part locations, as shown in the top row of Figure 14.43 (Leibe, Leonardis, and Schiele 2008). (Implicitly, this corresponds to having a star-shaped geometric model.)

14.4.3 Recognition with segmentation

The most challenging version of generic object recognition is to simultaneously perform recognition with accurate boundary segmentation (Fergus 2007a). For instance recognition (Section 14.3.1), this can sometimes be achieved by backprojecting the object model into the scene (Lowe 2004), as shown in Figure 14.1d, or matching portions of the new scene to pre-learned (segmented) object models (Ferrari, Tuytelaars, and Van Gool 2006b; Kannala, Rahtu, Brandt *et al.* 2008).

For more complex (flexible) object models, such as those for humans Figure 14.1f, a different approach is to pre-segment the image into larger or smaller pieces (Chapter 5) and then match such pieces to portions of the model (Mori, Ren, Efros *et al.* 2004; Mori 2005; He, Zemel, and Ray 2006; Gu, Lim, Arbelaez *et al.* 2009).

An alternative approach by Leibe, Leonardis, and Schiele (2008), which we introduced in the previous section, votes for potential object locations and scales based on the detection of features corresponding to pre-clustered visual codebook entries (Figure 14.43). To support segmentation, each codebook entry has an associated foreground–background mask, which is learned as part of the codebook clustering process from pre-labeled object segmentation masks. During recognition, once a maximum in the voting space is found, the masks associated with the entries that voted for this instance are combined to obtain an object segmentation, as shown on the left side of Figure 14.43.

A more holistic approach to recognition and segmentation is to formulate the problem as one of labeling every pixel in an image with its class membership, and to solve this problem using energy minimization or Bayesian inference techniques, i.e., conditional random fields (Section 3.7.2, (3.118)) (Kumar and Hebert 2006; He, Zemel, and Carreira-Perpiñán 2004). The TextonBoost system of Shotton, Winn, Rother *et al.* (2009) uses unary (pixel-wise) potentials based on image-specific color distributions (Section 5.5) (Boykov and Jolly 2001; Rother, Kolmogorov, and Blake 2004), location information (e.g., foreground objects are more likely to be in the middle of the image, sky is likely to be higher, and road is likely to be lower), and novel texture-layout classifiers trained using shared boosting. It also uses traditional pairwise potentials that look at image color gradients (Veksler 2001; Boykov and Jolly 2001; Rother, Kolmogorov, and Blake 2004). The texton-layout features first filter the image with a series of 17 oriented filter banks and then cluster the responses to classify each pixel into 30 different texton classes (Malik, Belongie, Leung *et al.* 2001). The responses are then filtered using offset rectangular regions trained with joint boosting (Viola and Jones 2004) to produce the texton-layout features used as unary potentials.

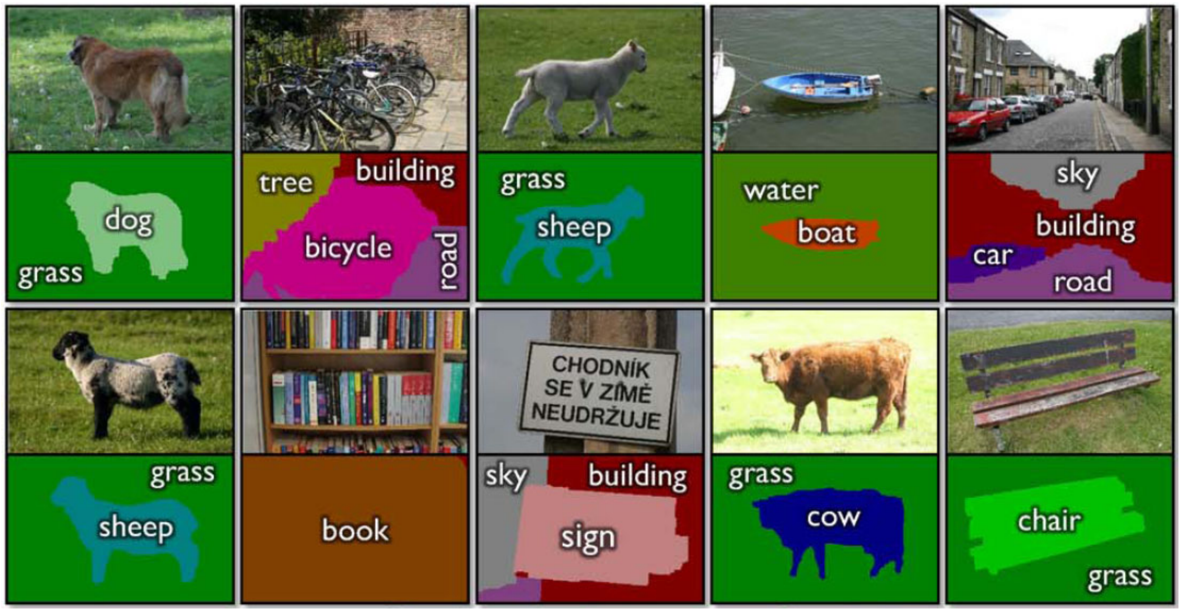
Figure 14.44a shows some examples of images successfully labeled and segmented using TextonBoost, while Figure 14.44b shows examples where it does not do as well. As you can see, this kind of semantic labeling can be extremely challenging.

The TextonBoost conditional random field framework has been extended to LayoutCRFs by Winn and Shotton (2006), who incorporate additional constraints to recognize multiple object instances and deal with occlusions (Figure 14.45), and even more recently by Hoiem, Rother, and Winn (2007) to incorporate full 3D models.

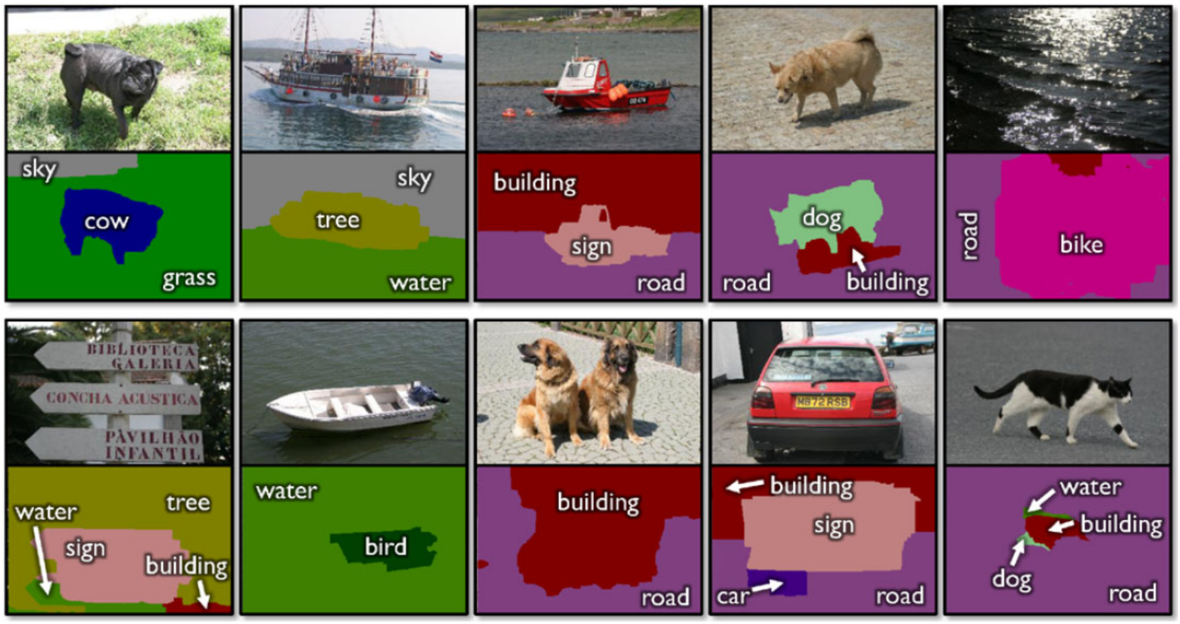
Conditional random fields continue to be widely used and extended for simultaneous recognition and segmentation applications (Kumar and Hebert 2006; He, Zemel, and Ray 2006; Levin and Weiss 2006; Verbeek and Triggs 2007; Yang, Meer, and Foran 2007; Rabinovich, Vedaldi, Galleguillos *et al.* 2007; Batra, Sukthankar, and Chen 2008; Larlus and Jurie 2008; He and Zemel 2008; Kumar, Torr, and Zisserman 2010), producing some of the best results on the difficult PASCAL VOC segmentation challenge (Shotton, Johnson, and Cipolla 2008; Kohli, Ladický, and Torr 2009). Approaches that first segment the image into unique or multiple segmentations (Borenstein and Ullman 2008; He, Zemel, and Ray 2006; Russell, Efros, Sivic *et al.* 2006) (potentially combined with CRF models) also do quite well: Csurka and Perronnin (2008) have one of the top algorithms in the VOC segmentation challenge. Hierarchical (multi-scale) and grammar (parsing) models are also sometimes used (Tu, Chen, Yuille *et al.* 2005; Zhu, Chen, Lin *et al.* 2008).

14.4.4 *Application*: Intelligent photo editing

Recent advances in object recognition and scene understanding have greatly increased the power of intelligent (semi-automated) photo editing applications. One example is the Photo Clip Art system of Lalonde, Hoiem, Efros *et al.* (2007), which recognizes and segments objects of interest, such as pedestrians, in Internet photo collections and then allows users to paste them into their own photos. Another is the scene completion system of Hays and Efros (2007), which tackles the same *inpainting* problem we studied in Section 10.5. Given an image in which we wish to erase and fill in a large section (Figure 14.46a–b), where do you



(a)



(b)

Figure 14.44 Simultaneous recognition and segmentation using TextonBoost (Shotton, Winn, Rother *et al.* 2009) © 2009 Springer: (a) successful recognition results; (b) less successful results.

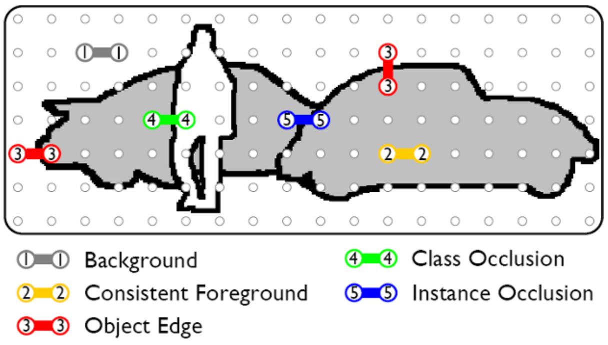


Figure 14.45 Layout consistent random field (Winn and Shotton 2006) © 2006 IEEE. The numbers indicate the kind of neighborhood relations that can exist between pixels assigned to the same or different classes. Each pairwise relationship carries its own likelihood (energy penalty).

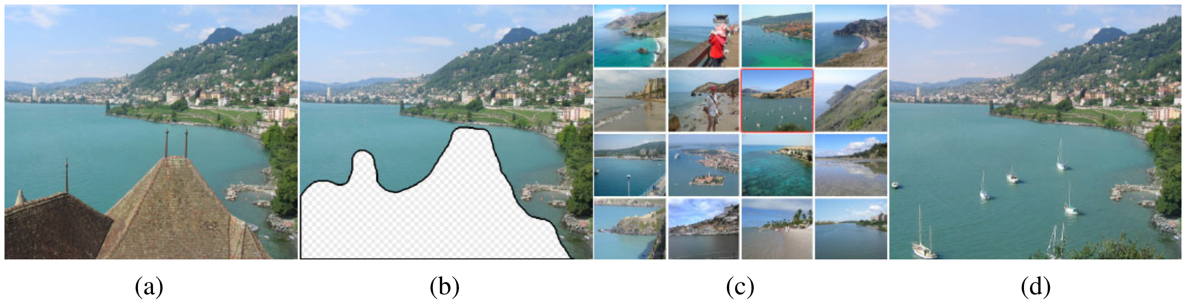


Figure 14.46 Scene completion using millions of photographs (Hays and Efros 2007) © 2007 ACM: (a) original image; (b) after unwanted foreground removal; (c) plausible scene matches, with the one the user selected highlighted in red; (d) output image after replacement and blending.

get the pixels to fill in the gaps in the edited image? Traditional approaches either use smooth continuation (Bertalmio, Sapiro, Caselles *et al.* 2000) or borrowing pixels from other parts of the image (Efros and Leung 1999; Criminisi, Pérez, and Toyama 2004; Efros and Freeman 2001). With the advent of huge repositories of images on the Web (a topic we return to in Section 14.5.1), it often makes more sense to find a *different* image to serve as the source of the missing pixels.

In their system, Hays and Efros (2007) compute the *gist* of each image (Oliva and Torralba 2001; Torralba, Murphy, Freeman *et al.* 2003) to find images with similar colors and composition. They then run a graph cut algorithm that minimizes image gradient differences and composite the new replacement piece into the original image using Poisson image blending (Section 9.3.4) (Pérez, Gangnet, and Blake 2003). Figure 14.46d shows the resulting image with the erased foreground rooftops region replaced with sailboats.

A different application of image recognition and segmentation is to infer 3D structure from a single photo by recognizing certain scene structures. For example, Criminisi, Reid, and Zisserman (2000) detect vanishing points and have the user draw basic structures, such as walls, in order infer the 3D geometry (Section 6.3.3). Hoiem, Efros, and Hebert (2005a) on the other hand, work with more "organic" scenes such as the one shown in Figure 14.47.

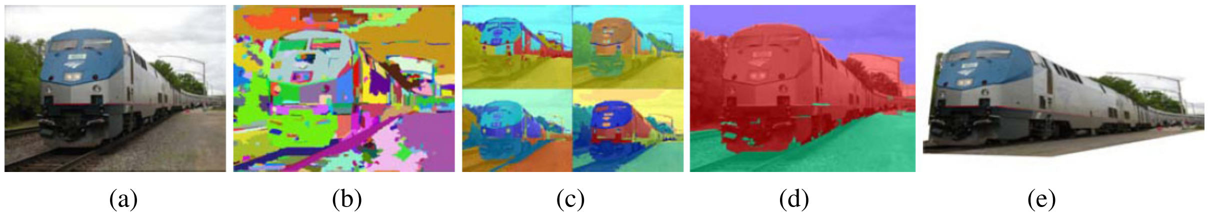


Figure 14.47 Automatic photo pop-up (Hoiem, Efros, and Hebert 2005a) © 2005 ACM: (a) input image; (b) superpixels are grouped into (c) multiple regions; (d) labelings indicating ground (green), vertical (red), and sky (blue); (e) novel view of resulting piecewise-planar 3D model.

Their system uses a variety of classifiers and statistics learned from labeled images to classify each pixel as either ground, vertical, or sky (Figure 14.47d). To do this, they begin by computing superpixels (Figure 14.47b) and then group them into plausible regions that are likely to share similar geometric labels (Figure 14.47c). After all the pixels have been labeled, the boundaries between the vertical and ground pixels can be used to infer 3D lines along which the image can be folded into a "pop-up" (after removing the sky pixels), as shown in Figure 14.47e. In related work, Saxena, Sun, and Ng (2009) develop a system that directly infers the depth and orientation of each pixel instead of using just three geometric class labels.

Face detection and localization can also be used in a variety of photo editing applications (in addition to being used in-camera to provide better focus, exposure, and flash settings). Zanella and Fuentes (2004) use active shape models (Section 14.2.2) to register facial features for creating automated morphs. Rother, Bordeaux, Hamadi *et al.* (2006) use face and sky detection to determine regions of interest in order to decide which pieces from a collection of images to stitch into a collage. Bitouk, Kumar, Dhillon *et al.* (2008) describe a system that matches a given face image to a large collection of Internet face images, which can then be used (with careful relighting algorithms) to replace the face in the original image. Applications they describe include de-identification and getting the best possible smile from everyone in a "burst mode" group shot. Leyvand, Cohen-Or, Dror *et al.* (2008) show how accurately locating facial features using an active shape model (Cootes, Edwards, and Taylor 2001; Zhou, Gu, and Zhang 2003) can be used to warp such features (and hence the image) towards configurations resembling those found in images whose facial attractiveness was highly rated, thereby "beautifying" the image without completely losing a person's identity.

Most of these techniques rely either on a set of labeled training images, which is an essential component of all learning techniques, or the even more recent explosion in images available on the Internet. The assumption in some of this work (and in recognition systems based on such very large databases (Section 14.5.1)) is that as the collection of accessible (and potentially partially labeled) images gets larger, finding a close match gets easier. As Hays and Efros (2007) state in their abstract "Our chief insight is that while the space of images is effectively infinite, the space of semantically differentiable scenes is actually not that large." In an interesting commentary on their paper, Levoy (2008) disputes this assertion, claiming that "features in natural scenes form a heavy-tailed distribution, meaning that while some features in photographs are more common than others, the relative occurrence of less common features drops slowly. In other words, there are many unusual photographs in the world." He does, however agree that in computational photography, as in many other applications such

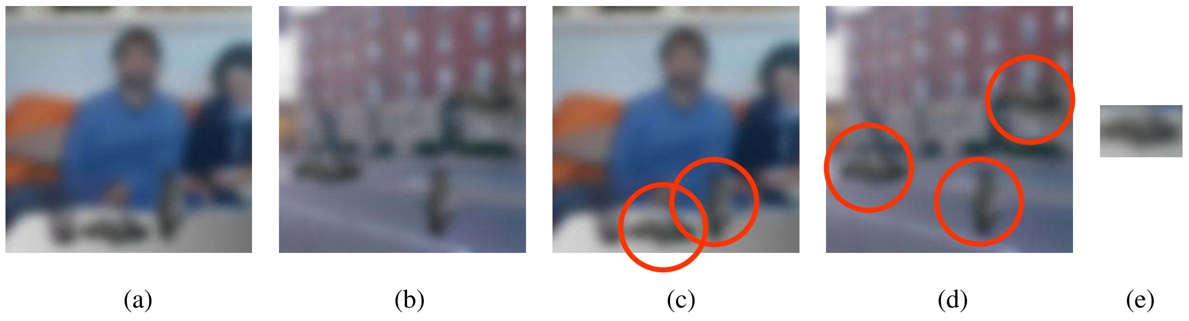


Figure 14.48 The importance of context (images courtesy of Antonio Torralba). Can you name all of the objects in images (a–b), especially those that are circled in (c–d). Look carefully at the circled objects. Did you notice that they all have the same shape (after being rotated), as shown in column (e)?

as speech recognition, synthesis, and translation, "simple machine learning algorithms often outperform more sophisticated ones if trained on large enough databases." He also goes on to point out both the potential advantages of such systems, such as better automatic color balancing, and potential issues and pitfalls with the kind of image fakery that these new approaches enable.

For additional examples of photo editing and computational photography applications enabled by Internet computer vision, please see recent workshops on this topic,[19] as well as the special journal issue (Avidan, Baker, and Shan 2010), and the course on Internet Vision by Tamara Berg (2008).

14.5 Context and scene understanding

Thus far, we have mostly considered the task of recognizing and localizing objects in isolation from that of understanding the scene (context) in which the object occur. This is a severe limitation, as context plays a very important role in human object recognition (Oliva and Torralba 2007). As we will see in this section, it can greatly improve the performance of object recognition algorithms (Divvala, Hoiem, Hays *et al.* 2009), as well as providing useful semantic clues for general scene understanding (Torralba 2008).

Consider the two photographs in Figure 14.48a–b. Can you name all of the objects, especially those circled in images (c–d)? Now have a closer look at the circled objects. Do see any similarity in their shapes? In fact, if you rotate them by 90°, they are all the same as the "blob" shown in Figure 14.48e. So much for our ability to recognize object by their shape! Another (perhaps more artificial) example of recognition in context is shown in Figure 14.49. Try to name all of the letters and numbers, and then see if you guessed right.

Even though we have not addressed context explicitly earlier in this chapter, we have already seen several instances of this general idea being used. A simple way to incorporate spatial information into a recognition algorithm is to compute feature statistics over different regions, as in the spatial pyramid system of Lazebnik, Schmid, and Ponce (2006). Part-based models (Section 14.4.2, Figures 14.40–14.43), use a kind of local context, where various parts need to be arranged in a proper geometric relationship to constitute an object.

[19] http://www.internetvisioner.org/.

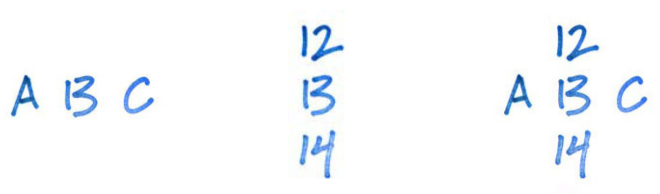


Figure 14.49 More examples of context: read the letters in the first group, the numbers in the second, and the letters and numbers in the third. (Images courtesy of Antonio Torralba.)

The biggest difference between part-based and context models is that the latter combine objects into scenes and the number of constituent objects from each class is not known in advance. In fact, it is possible to combine part-based and context models into the same recognition architecture (Murphy, Torralba, and Freeman 2003; Sudderth, Torralba, Freeman *et al.* 2008; Crandall and Huttenlocher 2007).

Consider the street and office scenes shown in Figure 14.50a–b. If we have enough training images with labeled regions, such as buildings, cars, and roads or monitors, keyboards, and mice, we can develop a geometric model for describing their relative positions. Sudderth, Torralba, Freeman *et al.* (2008) develop such a model, which can be thought of as a two-level constellation model. At the top level, the distributions of objects relative to each other (say, buildings with respect to cars) is modeled as a Gaussian (Figure 14.50c, upper right corners). At the bottom level, the distribution of parts (affine covariant features) with respect to the object center is modeled using a mixture of Gaussians (Figure 14.50c, lower two rows). However, since the number of objects in the scene and parts in each object is unknown, a *latent Dirichlet process* (LDP) is used to model object and part creation in a generative framework. The distributions for all of the objects and parts are learned from a large labeled database and then later used during inference (recognition) to label the elements of a scene.

Another example of context is in simultaneous segmentation and recognition (Section 14.4.3) (Figures 14.44–14.45), where the arrangements of various objects in a scene are used as part of the labeling process. Torralba, Murphy, and Freeman (2004) describe a conditional random field where the estimated locations of building and roads influence the detection of cars, and where boosting is used to learn the structure of the CRF. Rabinovich, Vedaldi, Galleguillos *et al.* (2007) use context to improve the results of CRF segmentation by noting that certain adjacencies (relationships) are more likely than others, e.g., a person is more likely to be on a horse than on a dog.

Context also plays an important role in 3D inference from single images (Figure 14.47), using computer vision techniques for labeling pixels as belonging to the ground, vertical surfaces, or sky (Hoiem, Efros, and Hebert 2005a,b). This line of work has been extended to a more holistic approach that simultaneously reasons about object identity, location, surface orientations, occlusions, and camera viewing parameters (Hoiem, Efros, and Hebert 2008a,b).

A number of approaches use the *gist* of a scene (Torralba 2003; Torralba, Murphy, Freeman *et al.* 2003) to determine where instances of particular objects are likely to occur. For example, Murphy, Torralba, and Freeman (2003) train a regressor to predict the vertical loca-

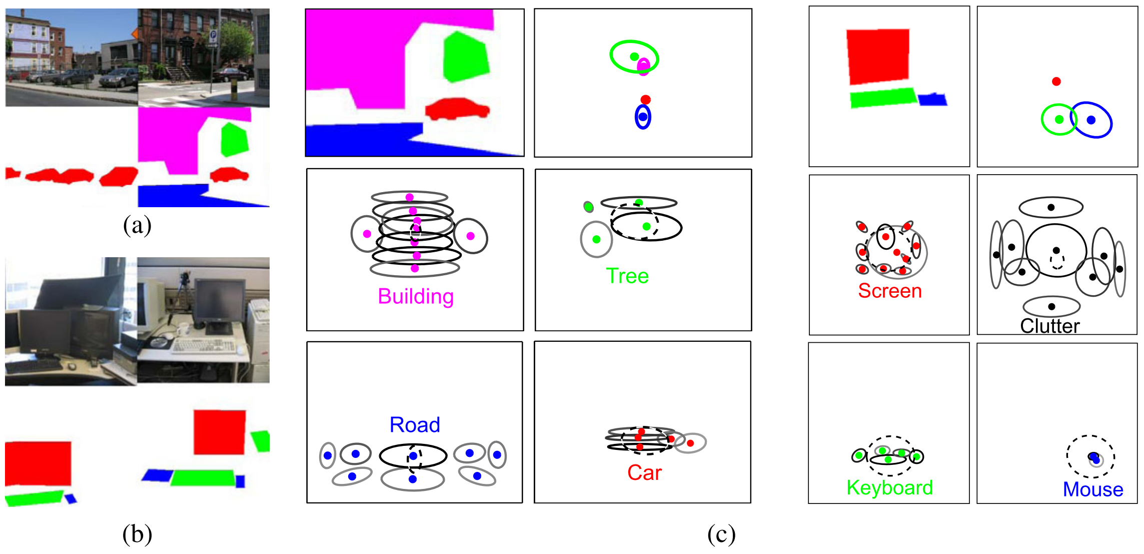


Figure 14.50 Contextual scene models for object recognition (Sudderth, Torralba, Freeman *et al.* 2008) © 2008 Springer: (a) some street scenes and their corresponding labels (magenta = buildings, red = cars, green = trees, blue = road); (b) some office scenes (red = computer screen, green = keyboard, blue = mouse); (c) learned contextual models built from these labeled scenes. The top row shows a sample label image and the distribution of the objects relative to the center red (car or screen) object. The bottom rows show the distributions of parts that make up each object.

tions of objects such as pedestrians, cars, and buildings (or screens and keyboard for indoor office scenes) based on the gist of an image. These location distributions are then used with classic object detectors to improve the performance of the detectors. Gists can also be used to directly match complete images, as we saw in the scene completion work of Hays and Efros (2007).

Finally, some of the most recent work in scene understanding exploits the existence of large numbers of labeled (or even unlabeled) images to perform matching directly against whole images, where the images themselves implicitly encode the expected relationships between objects (Figure 14.51) (Russell, Torralba, Liu *et al.* 2007; Malisiewicz and Efros 2008). We discuss such techniques in the next section, where we look at the influence that large image databases have had on object recognition and scene understanding.

14.5.1 Learning and large image collections

Given how learning techniques are widely used in recognition algorithms, you may wonder whether the topic of learning deserves its own section (or even chapter), or whether it is just part of the basic fabric of all recognition tasks. In fact, trying to build a recognition system without lots of training data for anything other than a basic pattern such as a UPC code has proven to be a dismal failure.

In this chapter, we have already seen lots of techniques borrowed from the machine learning, statistics, and pattern recognition communities. These include principal component, subspace, and discriminant analysis (Section 14.2.1) and more sophisticated discriminative clas-

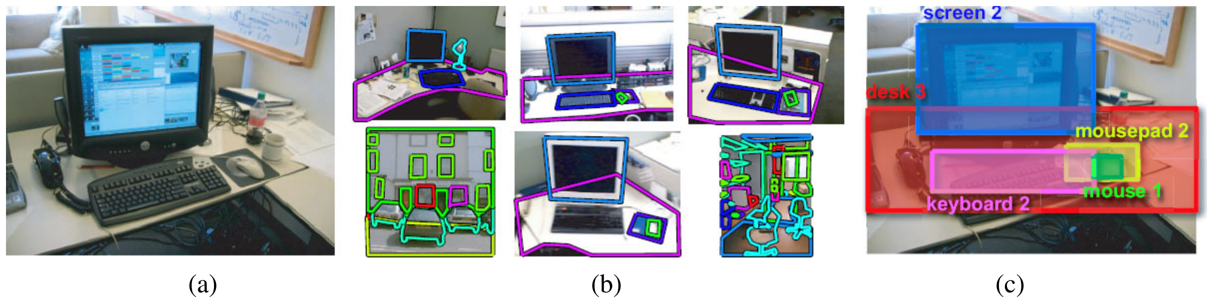


Figure 14.51 Recognition by scene alignment (Russell, Torralba, Liu *et al.* 2007): (a) input image; (b) matched images with similar scene configurations; (c) final labeling of the input image.

sification algorithms such as neural networks, support vector machines, and boosting (Section 14.1.1). Some of the best-performing techniques on challenging recognition benchmarks (Varma and Ray 2007; Felzenszwalb, McAllester, and Ramanan 2008; Fritz and Schiele 2008; Vedaldi, Gulshan, Varma *et al.* 2009) rely heavily on the latest machine learning techniques, whose development is often being driven by challenging vision problems (Freeman, Perona, and Schölkopf 2008).

A distinction sometimes made in the recognition community is between problems where most of the variables of interest (say, parts) are already (partially) labeled and systems that learn more of the problem structure with less supervision (Fergus, Perona, and Zisserman 2007; Fei-Fei, Fergus, and Perona 2006). In fact, recent work by Sivic, Russell, Zisserman *et al.* (2008) has demonstrated the ability to learn visual hierarchies (hierarchies of object parts with related visual appearance) and scene segmentations in a totally unsupervised framework.

Perhaps the most dramatic change in the recognition community has been the appearance of very large databases of training images.[20] Early learning-based algorithms, such as those for face and pedestrian detection (Section 14.1), used relatively few (in the hundreds) labeled examples to train recognition algorithm parameters (say, the thresholds used in boosting). Today, some recognition algorithms use databases such as LabelMe (Russell, Torralba, Murphy *et al.* 2008), which contain tens of thousands of labeled examples.

The existence of such large databases opens up the possibility of matching directly against the training images rather than using them to learn the parameters of recognition algorithms. Russell, Torralba, Liu *et al.* (2007) describe a system where a new image is matched against each of the training images, from which a consensus labeling for the unknown objects in the scene can be inferred, as shown in Figure 14.51. Malisiewicz and Efros (2008) start by over-segmenting each image and then use the LabelMe database to search for similar images and configurations in order to obtain per-pixel category labelings. It is also possible to combine feature-based correspondence algorithms with large labeled databases to perform simultaneous recognition and segmentation (Liu, Yuen, and Torralba 2009).

When the database of images becomes large enough, it is even possible to directly match complete images with the expectation of finding a good match. Torralba, Freeman, and Fergus (2008) start with a database of 80 million tiny (32×32) images and compensate for the poor accuracy in their image labels, which are collected automatically from the Internet, by using

[20] We have already seen some computational photography applications of such databases in Section 14.4.4.

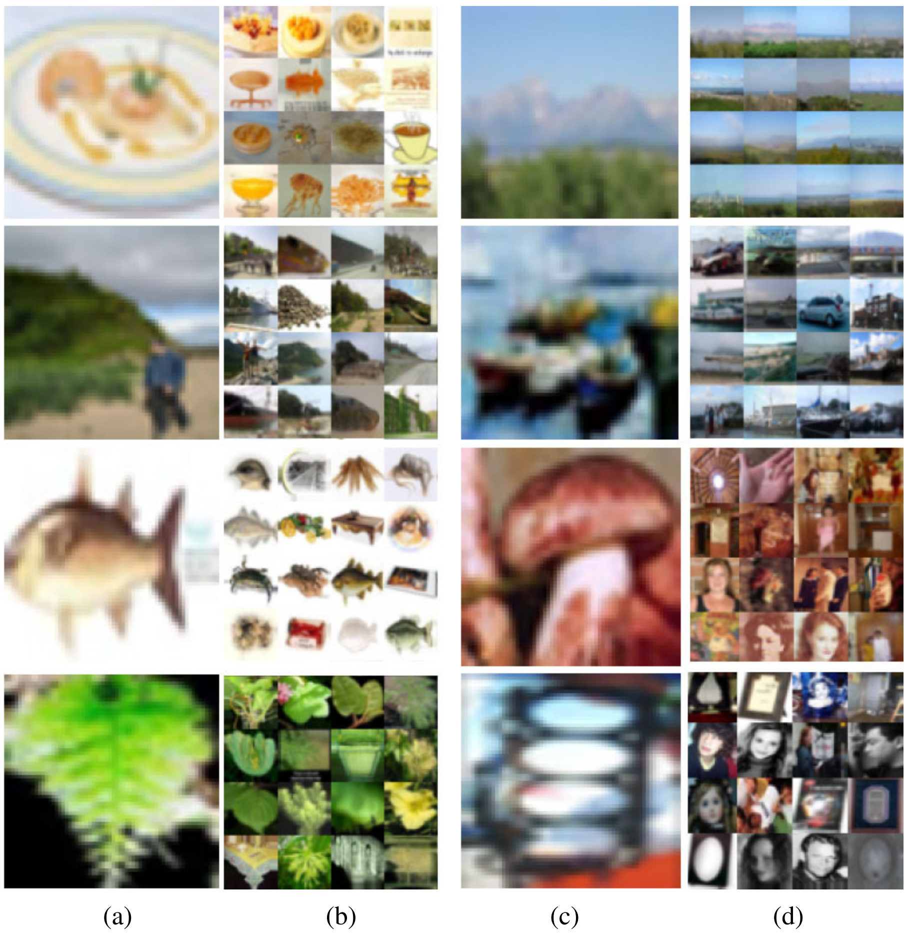


Figure 14.52 Recognition using tiny images (Torralba, Freeman, and Fergus 2008) © 2008 IEEE: columns (a) and (c) show sample input images and columns (b) and (d) show the corresponding 16 nearest neighbors in the database of 80 million tiny images.

a semantic taxonomy (Wordnet) to infer the most likely labels for a new image. Somewhere in the 80 million images, there are enough examples to associate some set of images with each of the 75,000 non-abstract nouns in Wordnet that they use in their system. Some sample recognition results are shown in Figure 14.52.

Another example of a large labeled database of images is ImageNet (Deng, Dong, Socher *et al.* 2009), which is collecting images for the 80,000 nouns (synonym sets) in WordNet (Fellbaum 1998). As of April 2010, about 500–1000 carefully vetted examples for 14841 synsets have been collected (Figure 14.53). The paper by Deng, Dong, Socher *et al.* (2009) also has a nice review of related databases.

As we mentioned in Section 14.4.3, the existence of large databases of partially labeled Internet imagery has given rise to a new sub-field of Internet computer vision, with its own workshops[21] and a special journal issue (Avidan, Baker, and Shan 2010).

[21] http://www.internetvisioner.org/.

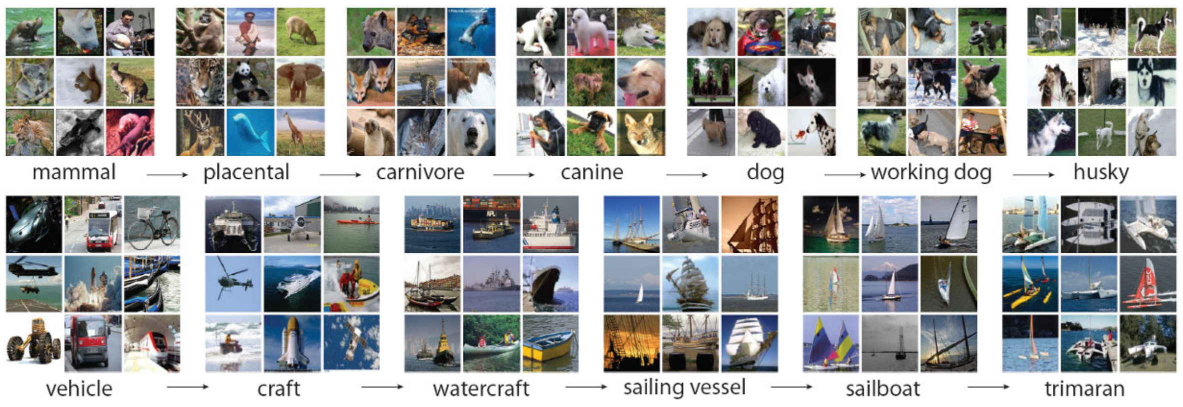


Figure 14.53 ImageNet (Deng, Dong, Socher *et al.* 2009) © 2009 IEEE. This database contains over 500 carefully vetted images for each of 14,841 (as of April, 2010) nouns from the WordNet hierarchy.

14.5.2 *Application*: Image search

Even though visual recognition algorithms are by some measures still in their infancy, they are already starting to have some impact on image search, i.e., the retrieval of images from the Web using combinations of keywords and visual similarity. Today, most image search engines rely mostly on textual keywords found in captions, nearby text, and filenames, augmented by user click-through data (Craswell and Szummer 2007). As recognition algorithms continue to improve, however, visual features and visual similarity will start being used to recognize images with missing or erroneous keywords.

The topic of searching by visual similarity has a long history and goes by a variety of names, including content-based image retrieval (CBIR) (Smeulders, Worring, Santini *et al.* 2000; Lew, Sebe, Djeraba *et al.* 2006; Vasconcelos 2007; Datta, Joshi, Li *et al.* 2008) and query by image content (QBIC) (Flickner, Sawhney, Niblack *et al.* 1995). Original publications in these fields were based primarily on simple whole-image similarity metrics, such as color and texture (Swain and Ballard 1991; Jacobs, Finkelstein, and Salesin 1995; Manjunathi and Ma 1996).

In more recent work, Fergus, Perona, and Zisserman (2004) use a feature-based learning and recognition algorithm to re-rank the outputs from a traditional keyword-based image search engine. In follow-on work, Fergus, Fei-Fei, Perona *et al.* (2005) cluster the results returned by image search using an extension of probabilistic latest semantic analysis (PLSA) (Hofmann 1999) and then select the clusters associated with the highest ranked results as the representative images for that category.

Even more recent work relies on carefully annotated image databases such as LabelMe (Russell, Torralba, Murphy *et al.* 2008). For example, Malisiewicz and Efros (2008) describe a system that, given a query image, can find similar LabelMe images, whereas Liu, Yuen, and Torralba (2009) combine feature-based correspondence algorithms with the labeled database to perform simultaneous recognition and segmentation.

14.6 Recognition databases and test sets

In addition to rapid advances in machine learning and statistical modeling techniques, one of the key ingredients in the continued improvement of recognition algorithms has been the increased availability and quality of image recognition databases.

Tables 14.1 and 14.2, which are based on similar tables in Fei-Fei, Fergus, and Torralba (2009), updated with more recent entries and URLs, show some of the mostly widely used recognition databases. Some of these databases, such as the ones for face recognition and localization, date back over a decade. The most recent ones, such as the PASCAL database, are refreshed annually with ever more challenging problems. Table 14.1 shows examples of databases used primarily for (whole image) recognition while Table 14.2 shows databases where more accurate localization or segmentation information is available and expected.

Ponce, Berg, Everingham *et al.* (2006) discuss some of the problems with earlier datasets and describe how the latest PASCAL Visual Object Classes Challenge aims to overcome these. Some examples of the 20 visual classes in the 2008 challenge are shown in Figure 14.54. The slides from the VOC workshops,[22] are a great source for pointers to the best recognition techniques currently available.

Two of the most recent trends in recognition databases are the emergence of Web-based annotation and data collection tools, and the use of search and recognition algorithms to build up databases (Ponce, Berg, Everingham *et al.* 2006). Some of the most interesting work in human annotation of images comes from a series of interactive multi-person games such as ESP (von Ahn and Dabbish 2004) and Peekaboom (von Ahn, Liu, and Blum 2006). In these games, people help each other guess the identity of a hidden image by giving textual clues as to its contents, which implicitly labels either the whole image or just regions. A more "serious" volunteer effort is the LabelMe database, in which vision researchers contribute manual polygonal region annotations in return for gaining access to the database (Russell, Torralba, Murphy *et al.* 2008).

The use of computer vision algorithms for collecting recognition databases dates back to the work of Fergus, Fei-Fei, Perona *et al.* (2005), who cluster the results returned by Google image search using an extension of PLSA and then select the clusters associated with the highest ranked results. More recent examples of related techniques include the work of Berg and Forsyth (2006) and Li and Fei-Fei (2010).

Whatever methods are used to collect and validate recognition databases, they will continue to grow in size, utility, and difficulty from year to year. They will also continue to be an essential component of research into the recognition and scene understanding problems, which remain, as always, the grand challenges of computer vision.

14.7 Additional reading

Although there are currently no specialized textbooks on image recognition and scene understanding, some surveys (Pinz 2005) and collections of papers (Ponce, Hebert, Schmid *et al.* 2006; Dickinson, Leonardis, Schiele *et al.* 2007) can be found that describe the latest approaches. Other good sources of recent research are courses on this topic, such as the ICCV

[22] http://pascallin.ecs.soton.ac.uk/challenges/VOC/.

Name / URL	Extents	Contents / Reference
Face and person recognition		
Yale face database	Centered face images	Frontal faces
http://www1.cs.columbia.edu/~belhumeur/		Belhumeur, Hespanha, and Kriegman (1997)
Resources for face detection	Various databases	Faces in various poses
http://vision.ai.uiuc.edu/mhyang/face-detection-survey.html		Yang, Kriegman, and Ahuja (2002)
FERET	Centered face images	Frontal faces
http://www.frvt.org/FERET		Phillips, Moon, Rizvi *et al.* (2000)
FRVT	Centered face images	Faces in various poses
http://www.frvt.org/		Phillips, Scruggs, O'Toole *et al.* (2010)
CMU PIE database	Centered face image	Faces in various poses
http://www.ri.cmu.edu/projects/project_418.html		Sim, Baker, and Bsat (2003)
CMU Multi-PIE database	Centered face image	Faces in various poses
http://multipie.org		Gross, Matthews, Cohn *et al.* (2010)
Faces in the Wild	Internet images	Faces in various poses
http://vis-www.cs.umass.edu/lfw/		Huang, Ramesh, Berg *et al.* (2007)
Consumer image person DB	Complete images	People
http://chenlab.ece.cornell.edu/people/Andy/GallagherDataset.html		Gallagher and Chen (2008)
Object recognition		
Caltech 101	Segmentation masks	101 categories
http://www.vision.caltech.edu/Image_Datasets/Caltech101/		Fei-Fei, Fergus, and Perona (2006)
Caltech 256	Centered objects	256 categories and clutter
http://www.vision.caltech.edu/Image_Datasets/Caltech256/		Griffin, Holub, and Perona (2007)
COIL-100	Centered objects	100 instances
http://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php		Nene, Nayar, and Murase (1996)
ETH-80	Centered objects	8 instances, 10 views
http://www.mis.tu-darmstadt.de/datasets		Leibe and Schiele (2003)
Instance recognition benchmark	Objects in various poses	2550 objects
http://vis.uky.edu/~stewe/ukbench/		Nistér and Stewénius (2006)
Oxford buildings dataset	Pictures of buildings	5062 images
http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/		Philbin, Chum, Isard *et al.* (2007)
NORB	Bounding box	50 toys
http://www.cs.nyu.edu/~ylclab/data/norb-v1.0/		LeCun, Huang, and Bottou (2004)
Tiny images	Complete images	75,000 (Wordnet) things
http://people.csail.mit.edu/torralba/tinyimages/		Torralba, Freeman, and Fergus (2008)
ImageNet	Complete images	14,000 (Wordnet) things
http://www.image-net.org/		Deng, Dong, Socher *et al.* (2009)

Table 14.1 Image databases for recognition, adapted and expanded from Fei-Fei, Fergus, and Torralba (2009).

Name / URL	**Extents**	**Contents / Reference**
Object detection / localization		
CMU frontal faces	Patches	Frontal faces
http://vasc.ri.cmu.edu/idb/html/face/frontal_images		Rowley, Baluja, and Kanade (1998a)
MIT frontal faces	Patches	Frontal faces
http://cbcl.mit.edu/software-datasets/FaceData2.html		Sung and Poggio (1998)
CMU face detection databases	Multiple faces	Faces in various poses
http://www.ri.cmu.edu/research_project_detail.html?project_id=419		Schneiderman and Kanade (2004)
UIUC Image DB	Bounding boxes	Cars
http://l2r.cs.uiuc.edu/~cogcomp/Data/Car/		Agarwal and Roth (2002)
Caltech Pedestrian Dataset	Bounding boxes	Pedestrians
http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians/		Dollàr, Wojek, Schiele *et al.* (2009)
Graz-02 Database	Segmentation masks	Bikes, cars, people
http://www.emt.tugraz.at/~pinz/data/GRAZ_02/		Opelt, Pinz, Fussenegger *et al.* (2006)
ETHZ Toys	Cluttered images	Toys, boxes, magazines
http://www.vision.ee.ethz.ch/~calvin/datasets.html		Ferrari, Tuytelaars, and Van Gool (2006b)
TU Darmstadt DB	Segmentation masks	Motorbikes, cars, cows
http://www.vision.ee.ethz.ch/~bleibe/data/datasets.html		Leibe, Leonardis, and Schiele (2008)
MSR Cambridge	Segmentation masks	23 classes
http://research.microsoft.com/en-us/projects/objectclassrecognition/		Shotton, Winn, Rother *et al.* (2009)
LabelMe dataset	Polygonal boundary	>500 categories
http://labelme.csail.mit.edu/		Russell, Torralba, Murphy *et al.* (2008)
Lotus Hill	Segmentation masks	Scenes and hierarchies
http://www.imageparsing.com/		Yao, Yang, Lin *et al.* (2010)
On-line annotation tools		
ESP game	Image descriptions	Web images
http://www.gwap.com/gwap/		von Ahn and Dabbish (2004)
Peekaboom	Labeled regions	Web images
http://www.gwap.com/gwap/		von Ahn, Liu, and Blum (2006)
LabelMe	Polygonal boundary	High-resolution images
http://labelme.csail.mit.edu/		Russell, Torralba, Murphy *et al.* (2008)
Collections of challenges		
PASCAL	Segmentation, boxes	Various
http://pascallin.ecs.soton.ac.uk/challenges/VOC/		Everingham, Van Gool, Williams *et al.* (2010)

Table 14.2 Image databases for detection and localization, adapted and expanded from Fei-Fei, Fergus, and Torralba (2009).

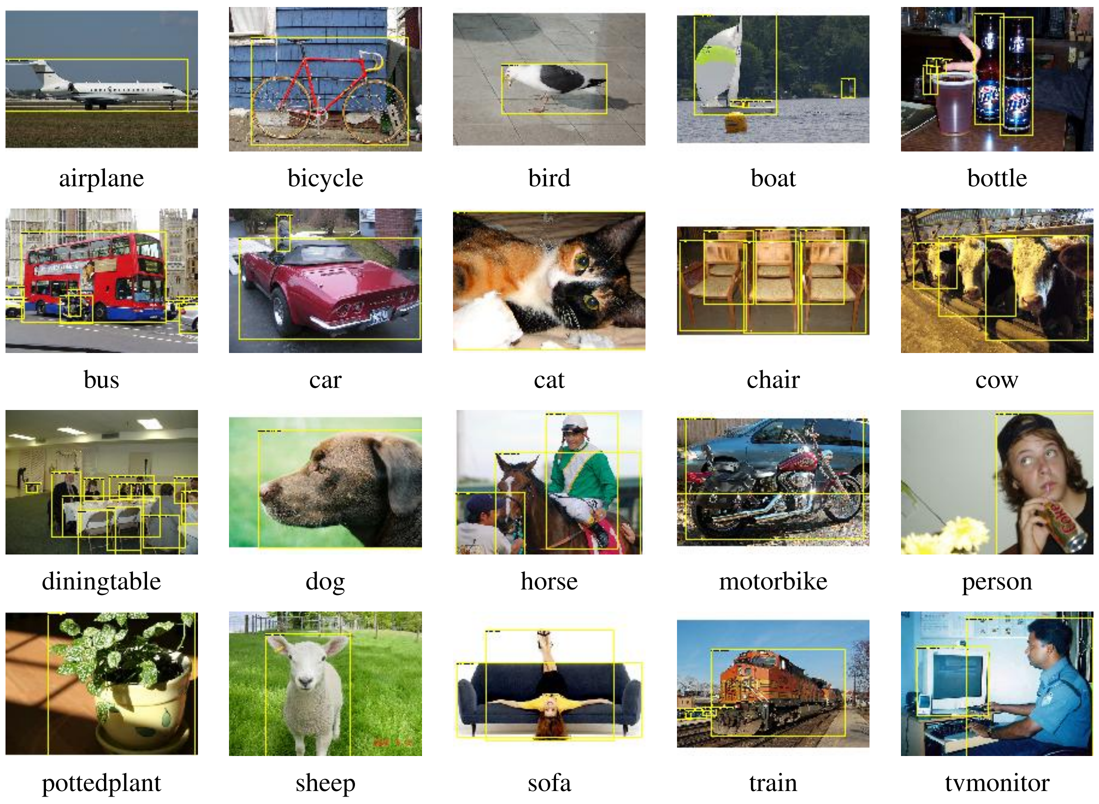


Figure 14.54 Sample images from the PASCAL Visual Object Classes Challenge 2008 (VOC2008) database (Everingham, Van Gool, Williams *et al.* 2008). The original images were obtained from flickr (http://www.flickr.com/) and the database rights are explained on http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2008/.

2009 short course (Fei-Fei, Fergus, and Torralba 2009) and Antonio Torralba's more comprehensive MIT course (Torralba 2008). The PASCAL VOC Challenge Web site contains workshop slides that summarize today's best performing algorithms.

The literature on face, pedestrian, car, and other object detection is quite extensive. Seminal papers in face detection include those by Osuna, Freund, and Girosi (1997), Sung and Poggio (1998), Rowley, Baluja, and Kanade (1998a), and Viola and Jones (2004), with Yang, Kriegman, and Ahuja (2002) providing a comprehensive survey of early work in this field. More recent examples include (Heisele, Ho, Wu *et al.* 2003; Heisele, Serre, and Poggio 2007).

Early work in pedestrian and car detection was carried out by Gavrila and Philomin (1999), Gavrila (1999), Papageorgiou and Poggio (2000), Mohan, Papageorgiou, and Poggio (2001), and Schneiderman and Kanade (2004). More recent examples include the work of Belongie, Malik, and Puzicha (2002), Mikolajczyk, Schmid, and Zisserman (2004), Dalal and Triggs (2005), Leibe, Seemann, and Schiele (2005), Dalal, Triggs, and Schmid (2006), Opelt, Pinz, and Zisserman (2006), Torralba (2007), Andriluka, Roth, and Schiele (2008), Felzenszwalb, McAllester, and Ramanan (2008), Rogez, Rihan, Ramalingam *et al.* (2008), Andriluka, Roth, and Schiele (2009), Kumar, Zisserman, and H.S.Torr (2009), Dollàr, Belongie, and Perona (2010). and Felzenszwalb, Girshick, McAllester *et al.* (2010).

While some of the earliest approaches to face recognition involved finding the distinctive image features and measuring the distances between them (Fischler and Elschlager 1973; Kanade 1977; Yuille 1991), more recent approaches rely on comparing gray-level images, often projected onto lower dimensional subspaces (Turk and Pentland 1991a; Belhumeur, Hespanha, and Kriegman 1997; Moghaddam and Pentland 1997; Moghaddam, Jebara, and Pentland 2000; Heisele, Ho, Wu *et al.* 2003; Heisele, Serre, and Poggio 2007). Additional details on principal component analysis (PCA) and its Bayesian counterparts can be found in Appendix B.1.1 and books and articles on this topic (Hastie, Tibshirani, and Friedman 2001; Bishop 2006; Roweis 1998; Tipping and Bishop 1999; Leonardis and Bischof 2000; Vidal, Ma, and Sastry 2010). The topics of subspace learning, local distance functions, and metric learning are covered by Cai, He, Hu *et al.* (2007), Frome, Singer, Sha *et al.* (2007), Guillaumin, Verbeek, and Schmid (2009), Ramanan and Baker (2009), and Sivic, Everingham, and Zisserman (2009). An alternative to directly matching gray-level images or patches is to use non-linear local transforms such as local binary patterns (Ahonen, Hadid, and Pietikäinen 2006; Zhao and Pietikäinen 2007; Cao, Yin, Tang *et al.* 2010).

In order to boost the performance of what are essentially 2D appearance-based models, a variety of shape and pose deformation models have been developed (Beymer 1996; Vetter and Poggio 1997), including Active Shape Models (Lanitis, Taylor, and Cootes 1997; Cootes, Cooper, Taylor *et al.* 1995; Davies, Twining, and Taylor 2008), Elastic Bunch Graph Matching (Wiskott, Fellous, Krüger *et al.* 1997), 3D Morphable Models (Blanz and Vetter 1999), and Active Appearance Models (Costen, Cootes, Edwards *et al.* 1999; Cootes, Edwards, and Taylor 2001; Gross, Baker, Matthews *et al.* 2005; Gross, Matthews, and Baker 2006; Matthews, Xiao, and Baker 2007; Liang, Xiao, Wen *et al.* 2008; Ramnath, Koterba, Xiao *et al.* 2008). The topic of head pose estimation, in particular, is covered in a recent survey by Murphy-Chutorian and Trivedi (2009).

Additional information about face recognition can be found in a number of surveys and books on this topic (Chellappa, Wilson, and Sirohey 1995; Zhao, Chellappa, Phillips *et al.* 2003; Li and Jain 2005) as well as on the Face Recognition Web site.[23] Databases for face recognition are discussed by Phillips, Moon, Rizvi *et al.* (2000), Sim, Baker, and Bsat (2003), Gross, Shi, and Cohn (2005), Huang, Ramesh, Berg *et al.* (2007), and Phillips, Scruggs, O'Toole *et al.* (2010).

Algorithms for instance recognition, i.e., the detection of static man-made objects that only vary slightly in appearance but may vary in 3D pose, are mostly based on detecting 2D points of interest and describing them using viewpoint-invariant descriptors (Lowe 2004; Rothganger, Lazebnik, Schmid *et al.* 2006; Ferrari, Tuytelaars, and Van Gool 2006b; Gordon and Lowe 2006; Obdržálek and Matas 2006; Kannala, Rahtu, Brandt *et al.* 2008; Sivic and Zisserman 2009).

As the size of the database being matched increases, it becomes more efficient to quantize the visual descriptors into words (Sivic and Zisserman 2003; Schindler, Brown, and Szeliski 2007; Sivic and Zisserman 2009; Turcot and Lowe 2009), and to then use information-retrieval techniques, such as inverted indices (Nistér and Stewénius 2006; Philbin, Chum, Isard *et al.* 2007; Philbin, Chum, Sivic *et al.* 2008), query expansion (Chum, Philbin, Sivic *et al.* 2007; Agarwal, Snavely, Simon *et al.* 2009), and min hashing (Philbin and Zisserman 2008; Li, Wu, Zach *et al.* 2008; Chum, Philbin, and Zisserman 2008; Chum and Matas 2010) to perform efficient retrieval and clustering.

[23] http://www.face-rec.org/.

A number of surveys, collections of papers, and course notes have been written on the topic of category recognition (Pinz 2005; Ponce, Hebert, Schmid *et al.* 2006; Dickinson, Leonardis, Schiele *et al.* 2007; Fei-Fei, Fergus, and Torralba 2009). Some of the seminal papers on the bag of words (bag of keypoints) approach to whole-image category recognition have been written by Csurka, Dance, Fan *et al.* (2004), Lazebnik, Schmid, and Ponce (2006), Csurka, Dance, Perronnin *et al.* (2006), Grauman and Darrell (2007b), and Zhang, Marszalek, Lazebnik *et al.* (2007). Additional and more recent papers in this area include Sivic, Russell, Efros *et al.* (2005), Serre, Wolf, and Poggio (2005), Opelt, Pinz, Fussenegger *et al.* (2006), Grauman and Darrell (2007a), Torralba, Murphy, and Freeman (2007), Boiman, Shechtman, and Irani (2008), Ferencz, Learned-Miller, and Malik (2008), and Mutch and Lowe (2008). It is also possible to recognize objects based on their contours, e.g., using shape contexts (Belongie, Malik, and Puzicha 2002) or other techniques (Jurie and Schmid 2004; Shotton, Blake, and Cipolla 2005; Opelt, Pinz, and Zisserman 2006; Ferrari, Tuytelaars, and Van Gool 2006a).

Many object recognition algorithms use part-based decompositions to provide greater invariance to articulation and pose. Early algorithms focused on the relative positions of the parts (Fischler and Elschlager 1973; Kanade 1977; Yuille 1991) while newer algorithms use more sophisticated models of appearance (Felzenszwalb and Huttenlocher 2005; Fergus, Perona, and Zisserman 2007; Felzenszwalb, McAllester, and Ramanan 2008). Good overviews on part-based models for recognition can be found in the course notes of Fergus 2007b; 2009.

Carneiro and Lowe (2006) discuss a number of graphical models used for part-based recognition, which include trees and stars (Felzenszwalb and Huttenlocher 2005; Fergus, Perona, and Zisserman 2005; Felzenszwalb, McAllester, and Ramanan 2008), k-fans (Crandall, Felzenszwalb, and Huttenlocher 2005; Crandall and Huttenlocher 2006), and constellations (Burl, Weber, and Perona 1998; Weber, Welling, and Perona 2000; Fergus, Perona, and Zisserman 2007). Other techniques that use part-based recognition include those developed by Dorkó and Schmid (2003) and Bar-Hillel, Hertz, and Weinshall (2005).

Combining object recognition with scene segmentation can yield strong benefits. One approach is to pre-segment the image into pieces and then match the pieces to portions of the model (Mori, Ren, Efros *et al.* 2004; Mori 2005; He, Zemel, and Ray 2006; Russell, Efros, Sivic *et al.* 2006; Borenstein and Ullman 2008; Csurka and Perronnin 2008; Gu, Lim, Arbelaez *et al.* 2009). Another is to vote for potential object locations and scales based on object detection (Leibe, Leonardis, and Schiele 2008). One of the currently most popular approaches is to use conditional random fields (Kumar and Hebert 2006; He, Zemel, and Carreira-Perpiñán 2004; He, Zemel, and Ray 2006; Levin and Weiss 2006; Winn and Shotton 2006; Hoiem, Rother, and Winn 2007; Rabinovich, Vedaldi, Galleguillos *et al.* 2007; Verbeek and Triggs 2007; Yang, Meer, and Foran 2007; Batra, Sukthankar, and Chen 2008; Larlus and Jurie 2008; He and Zemel 2008; Shotton, Winn, Rother *et al.* 2009; Kumar, Torr, and Zisserman 2010), which produce some of the best results on the difficult PASCAL VOC segmentation challenge (Shotton, Johnson, and Cipolla 2008; Kohli, Ladický, and Torr 2009).

More and more recognition algorithms are starting to use scene context as part of their recognition strategy. Representative papers in this area include those by Torralba (2003), Torralba, Murphy, Freeman *et al.* (2003), Murphy, Torralba, and Freeman (2003), Torralba, Murphy, and Freeman (2004), Crandall and Huttenlocher (2007), Rabinovich, Vedaldi, Galleguillos *et al.* (2007), Russell, Torralba, Liu *et al.* (2007), Hoiem, Efros, and Hebert (2008a), Hoiem, Efros, and Hebert (2008b), Sudderth, Torralba, Freeman *et al.* (2008), and Divvala, Hoiem, Hays *et al.* (2009).

Sophisticated machine learning techniques are also becoming a key component of successful object detection and recognition algorithms (Varma and Ray 2007; Felzenszwalb, McAllester, and Ramanan 2008; Fritz and Schiele 2008; Sivic, Russell, Zisserman *et al.* 2008; Vedaldi, Gulshan, Varma *et al.* 2009), as is exploiting large human-labeled databases (Russell, Torralba, Liu *et al.* 2007; Malisiewicz and Efros 2008; Torralba, Freeman, and Fergus 2008; Liu, Yuen, and Torralba 2009). Rough three-dimensional models are also making a comeback for recognition, as evidenced in some recent papers (Savarese and Fei-Fei 2007, 2008; Sun, Su, Savarese *et al.* 2009; Su, Sun, Fei-Fei *et al.* 2009). As always, the latest conferences on computer vision are your best reference for the newest algorithms in this rapidly evolving field.

14.8 Exercises

Ex 14.1: Face detection Build and test one of the face detectors presented in Section 14.1.1.

1. Download one or more of the labeled face detection databases in Table 14.2.

2. Generate your own negative examples by finding photographs that do not contain any people.

3. Implement one of the following face detectors (or devise one of your own):

   - boosting (Algorithm 14.1) based on simple area features, with an optional cascade of detectors (Viola and Jones 2004);
   - PCA face subspace (Moghaddam and Pentland 1997);
   - distances to clustered face and non-face prototypes, followed by a neural network (Sung and Poggio 1998) or SVM (Osuna, Freund, and Girosi 1997) classifier;
   - a multi-resolution neural network trained directly on normalized gray-level patches (Rowley, Baluja, and Kanade 1998a).

4. Test the performance of your detector on the database by evaluating the detector at every location in a sub-octave pyramid. Optionally retrain your detector on false positive examples you get on non-face images.

Ex 14.2: Determining the threshold for AdaBoost Given a set of function evaluations on the training examples $\boldsymbol{x}_i$, $f_i = f(\boldsymbol{x}_i) \in \pm 1$, training labels $y_i \in \pm 1$, and weights $w_i \in (0, 1)$, as explained in Algorithm 14.1, devise an efficient algorithm to find values of θ and $s = \pm 1$ that maximize

$$\sum_i w_i y_i h(s f_i, \theta), \tag{14.43}$$

where $h(x, \theta) = \text{sign}(x - \theta)$.

Ex 14.3: Face recognition using eigenfaces Collect a set of facial photographs and then build a recognition system to re-recognize the same people.

1. Take several photos of each of your classmates and store them.
2. Align the images by automatically or manually detecting the corners of the eyes and using a similarity transform to stretch and rotate each image to a canonical position.
3. Compute the average image and a PCA subspace for the face images
4. Take a new set of photographs a week later and use them as your test set.
5. Compare each new image to each database image and select the nearest one as the recognized identity. Verify that the distance in PCA space is close to the distance computed with a full SSD (sum of squared difference) measure.
6. (Optional) Compute different principal components for identity and expression, and use them to improve your recognition results.

Ex 14.4: Bayesian face recognition Moghaddam, Jebara, and Pentland (2000) compute separate covariance matrices $\boldsymbol{\Sigma}_I$ and $\boldsymbol{\Sigma}_E$ by looking at differences between *all* pairs of images. At run time, they select the *nearest* image to determine the facial identity. Does it make sense to estimate statistics for all pairs of images and use them for testing the distance to the nearest exemplar? Discuss whether this is statistically correct.

How is the all-pair intrapersonal covariance matrix $\boldsymbol{\Sigma}_I$ related to the within-class scatter matrix $\boldsymbol{S}_{\mathrm{W}}$? Does a similar relationship hold between $\boldsymbol{\Sigma}_E$ and $\boldsymbol{S}_{\mathrm{B}}$?

Ex 14.5: Modular eigenfaces Extend your face recognition system to separately match the eye, nose, and mouth regions, as shown in Figure 14.18.

1. After normalizing face images to a canonical scale and location, manually segment out some of the eye, nose, and face regions.
2. Build separate detectors for these three (or four) kinds of region, either using a subspace (PCA) approach or one of the techniques presented in Section 14.1.1.
3. For each new image to be recognized, first detect the locations of the facial features.
4. Then, match the individual features against your database and note the locations of these features.
5. Train and test a classifier that uses the individual feature matching IDs as well as (optionally) the feature locations to perform face recognition.

Ex 14.6: Recognition-based color balancing Build a system that recognizes the most important color areas in common photographs (sky, grass, skin) and color balances the image accordingly. Some references and ideas for skin detection are given in Exercise 2.8 and by Forsyth and Fleck (1999), Jones and Rehg (2001), Vezhnevets, Sazonov, and Andreeva (2003), and Kakumanu, Makrogiannis, and Bourbakis (2007). These may give you ideas for how to detect other regions or you can try more sophisticated MRF-based approaches (Shotton, Winn, Rother *et al.* 2009).

Ex 14.7: Pedestrian detection Build and test one of the pedestrian detectors presented in Section 14.1.2.

Ex 14.8: Simple instance recognition Use the feature detection, matching, and alignment algorithms you developed in Exercises 4.1–4.4 and 9.2 to find matching images given a query image or region (Figure 14.26).

Evaluate several feature detectors, descriptors, and robust geometric verification strategies, either on your own or by comparing your results with those of classmates.

Ex 14.9: Large databases and location recognition Extend the previous exercise to larger databases using quantized visual words and information retrieval techniques, as described in Algorithm 14.2.

Test your algorithm on a large database, such as the one used by Nistér and Stewénius (2006) or Philbin, Chum, Sivic *et al.* (2008), which are listed in Table 14.1. Alternatively, use keyword search on the Web or in a photo sharing site (e.g., for a city) to create your own database.

Ex 14.10: Bag of words Adapt the feature extraction and matching pipeline developed in Exercise 14.8 to category (class) recognition, using some of the techniques described in Section 14.4.1.

1. Download the training and test images from one or more of the databases listed in Tables 14.1 and 14.2, e.g., Caltech 101, Caltech 256, or PASCAL VOC.

2. Extract features from each of the training images, quantize them, and compute the *tf-idf* vectors (bag of words histograms).

3. As an option, consider not quantizing the features and using pyramid matching (14.40–14.41) (Grauman and Darrell 2007b) or using a spatial pyramid for greater selectivity (Lazebnik, Schmid, and Ponce 2006).

4. Choose a classification algorithm (e.g., nearest neighbor classification or support vector machine) and "train" your recognizer, i.e., build up the appropriate data structures (e.g., k-d trees) or set the appropriate classifier parameters.

5. Test your algorithm on the test data set using the same pipeline you developed in steps 2–4 and compare your results to the best reported results.

6. Explain why your results differ from the previously reported ones and give some ideas for how you could improve your system.

You can find a good synopsis of the best-performing classification algorithms and their approaches in the report of the PASCAL Visual Object Classes Challenge found on their Web site (http://pascallin.ecs.soton.ac.uk/challenges/VOC/).

Ex 14.11: Object detection and localization Extend the classification algorithm developed in the previous exercise to localize the objects in an image by reporting a bounding box around each detected object. The easiest way to do this is to use a sliding window approach. Some pointers to recent techniques in this area can be found in the workshop associated with the PASCAL VOC 2008 Challenge.

Ex 14.12: Part-based recognition Choose one or more of the techniques described in Section 14.4.2 and implement a part-based recognition system. Since these techniques are fairly involved, you will need to read several of the research papers in this area, select which general approach you want to follow, and then implement your algorithm. A good starting point could be the paper by Felzenszwalb, McAllester, and Ramanan (2008), since it performed well in the PASCAL VOC 2008 detection challenge.

Ex 14.13: Recognition and segmentation Choose one or more of the techniques described in Section 14.4.3 and implement a simultaneous recognition and segmentation system. Since these techniques are fairly involved, you will need to read several of the research papers in this area, select which general approach you want to follow, and then implement your algorithm. Test your algorithm on one or more of the segmentation databases in Table 14.2.

Ex 14.14: Context Implement one or more of the context and scene understanding systems described in Section 14.5 and report on your experience. Does context or whole scene understanding perform better at naming objects than stand-alone systems?

Ex 14.15: Tiny images Download the tiny images database from http://people.csail.mit.edu/torralba/tinyimages/ and build a classifier based on comparing your test images directly against all of the labeled training images. Does this seem like a promising approach?