Classification and Detection in Images

D.A. Forsyth

Classifying Images

- Motivating problems
  - detecting explicit images
  - classifying materials
  - classifying scenes
- Strategy
  - build appropriate image features
  - train classifier
  - test, evaluate

FIGURE 16.1: Material is not the same as object category (the three cars on the **top** are each made of different materials), and is not the same as texture (the three checkered objects on the **bottom** are made of different materials). Knowing the material that makes up an object gives us a useful description, somewhat distinct from its identity and its texture. *This figure was originally published as Figures 2 and 3 of "Exploring Features in a Bayesian Framework for Material Recognition," by C. Liu, L. Sharan, E. Adelson, and R. Rosenholtz Proc. CVPR 2010, 2010* © *IEEE, 2010.*

FIGURE 16.2: Some scenes are easily identified by humans. These are examples from the SUN dataset (Xiao *et al.* 2010) of scene categories that people identify accurately from images; the label above each image gives its scene type. *This figure was originally published as Figure 2 of "SUN database: Large-scale Scene Recognition from Abbey to Zoo," by J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba, Proc. IEEE CVPR 2010, © IEEE, 2010.*

GIST Features

- Layout is important
  - where stuff is
  - are there walls?
  - is there a floor?
  - is it an open space?
- Texture measures should capture this
- Strategy
  - obtain filter responses, obtain average magnitudes
  - compute linear discriminants for reference dataset
    - to find magnitudes that are most helpful
- There is now a standard set of GIST features
  - no need to re-implement

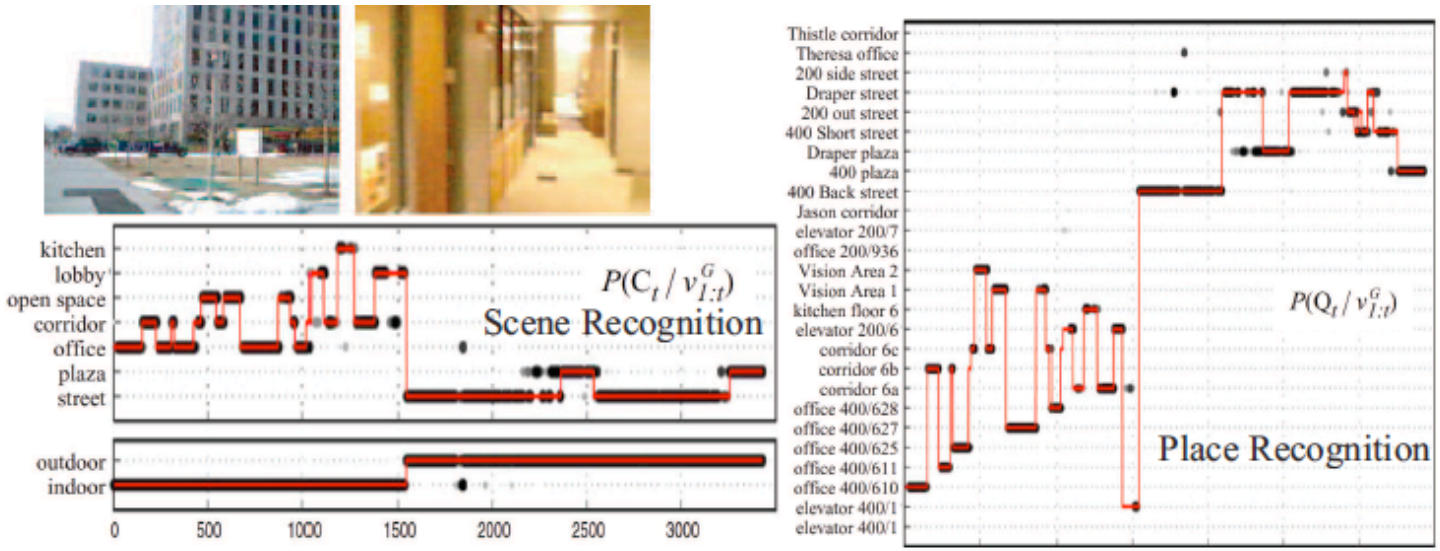


FIGURE 16.3: GIST features can be used to identify scenes, particularly the place where the image was taken. Torralba *et al.* (2003) demonstrated a vision system that moves through a known environment, and can tell where it is from what it sees using scene recognition ideas. Images (examples on the **top left**) are represented with GIST features. These are used to compute a posterior probability of place conditioned on observations and the place of the last image, which is shown on the **right**. The shaded blobs correspond to posterior probability, with darker blobs having higher probability. The thin line superimposed on the figure gives the correct answer; notice that almost all probability lies on the right answer. For places that are not known, the type of place can be estimated (**bottom left**); again, the shaded blobs give posterior probability, darker blobs having higher probability, and the thin line gives the right answer. *This figure was originally published as Figures 2 and 3 of "Context-based vision system for place and object recognition," by A. Torralba, K. Murphy, W.T. Freeman, and M.A. Rubin, Proc. IEEE ICCV 2003, © IEEE 2003.*

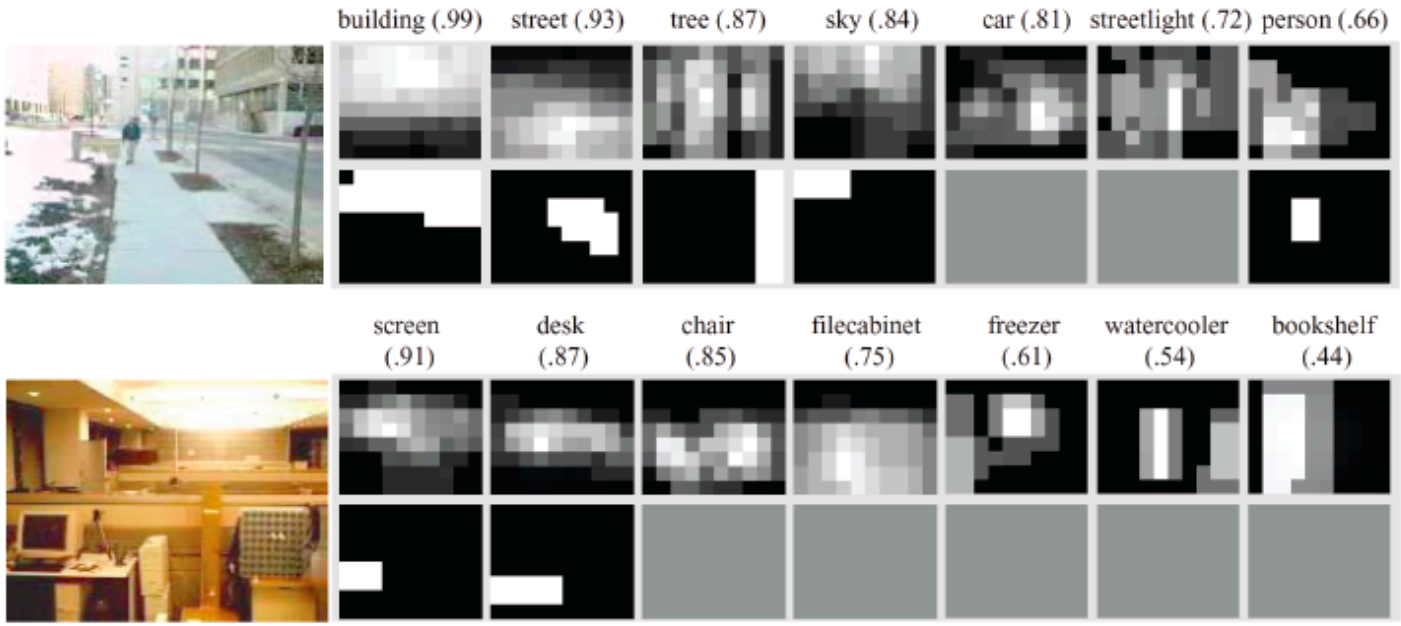


FIGURE 16.4: Scenes are important, because knowing the type of scene shown in an image gives us some information about the objects that are present. For example, street's are typically at the bottom center of street scenes. These maps show probabilities of object locations (**top row**, for each image) extracted from scene information for the image to the left; brighter values are higher probabilities. Compare these with the true support of the object (**bottom row**, for each image); notice that, while knowing the scene doesn't guarantee that an object is present, it does suggest where it is likely to be. This could be used to cue object detection processes. *This figure was originally published as Figure 10 of "Context-based vision system for place and object recognition," by A. Torralba, K. Murphy, W.T. Freeman, and M.A. Rubin, Proc. IEEE ICCV 2003,* © *IEEE 2003.*

Multiclass classification

- Many strategies
  - Easy with k-nearest neighbors
  - 1-vs-all
    - for each class, construct a two class classifier comparing it to all other classes
    - take the class with best output
      - if output is greater than some value
  - Multiclass logistic regression
    - log(P(i|features))-log(P(k|features))=(linear expression)
    - many more parameters
    - harder to train with maximum likelihood
    - still convex

Useful tricks

- Jittering data
  - you can make many useful positives, negatives out of some

FIGURE 15.7: A single positive example can be used to generate numerous positive examples by slight rescaling and cropping, small rotations and crops, or flipping. These transformations can be combined, too. For most applications, these positive examples are informative, because objects usually are not framed and scaled precisely in images. In effect, these examples inform the classifier that, for example, the stove could be slightly more or slightly less to the right of the image or even to the left. *Jake Fitzjones © Dorling Kindersley, used with permission.*

- Hard negative mining
  - negatives are often common - find ones that you get wrong by a search

Visual words

- Issue:
  - category will not produce a single, simple pattern
  - but it might have components that are distinctive, but move around
- Idea:
  - look for distinctive local patches
    - found using methods from domain slides
    - described with HOG/SIFT style features
  - vector quantize, to form Visual Words
  - build a histogram

Important trick: K-Means

- Choose a fixed number of clusters
- Choose cluster centers and point-cluster allocations to minimize error

$$\sum_{i \in \text{clusters}} \left\{ \sum_{j \in \text{elements of i'th cluster}} \left\| x_j - \mu_i \right\|^2 \right\}$$

- can't do this by search
  - there are too many possible allocations.
- Algorithm
  - fix cluster centers; allocate points to closest cluster
  - fix allocation; compute best cluster centers
  - x could be any set of features for which we can compute a distance (careful about scaling)

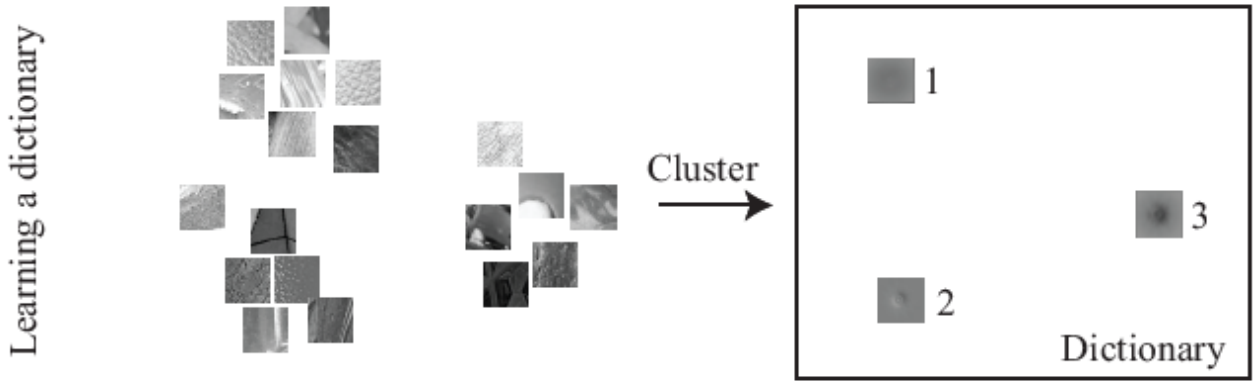
K-means

Choose k data points to act as cluster centers
Until the cluster centers change very little
    Allocate each data point to cluster whose center is nearest.
    Now ensure that every cluster has at least
        one data point; one way to do this is by
        supplying empty clusters with a point chosen at random from
        points far from their cluster center.
    Replace the cluster centers with the mean of the elements
        in their clusters.
end

Algorithm 6.3: Clustering by K-Means.

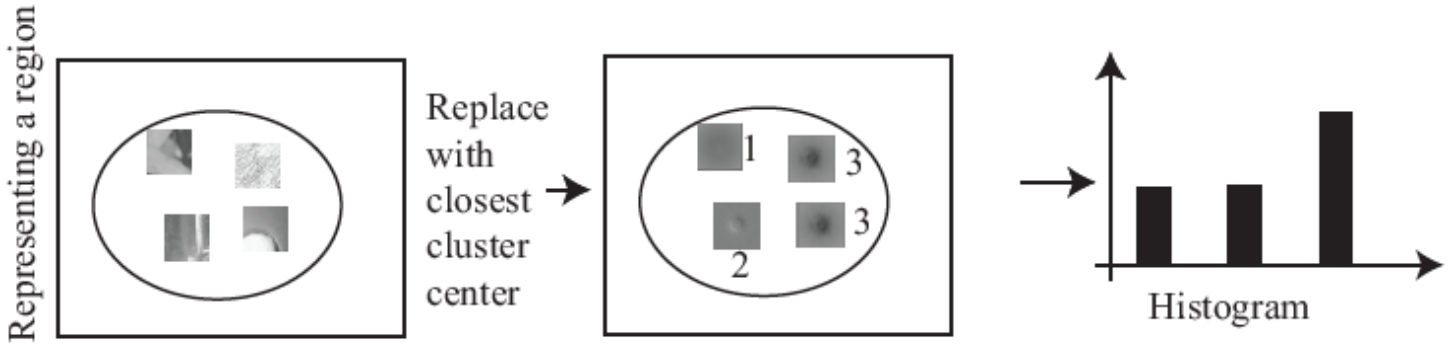
Building visual words - I

- Learn a dictionary
  - cluster patch representations with k-means
  - k will be big (1000's-100,000's)



Building visual words - II

- Encode an image
  - find all interest points
  - for each patch around each interest point
    - map patch to closest cluster center
  - build histogram of interest points



Visual words

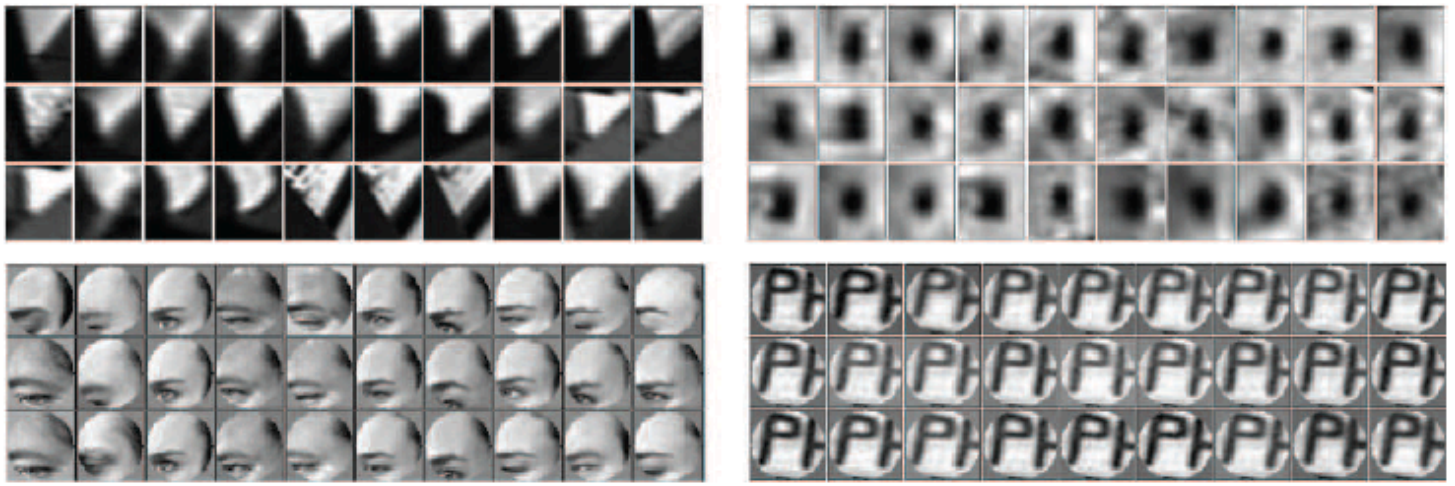


FIGURE 16.6: Visual words are obtained by vector quantizing neighborhoods like those shown in Figure 16.5. This figure shows 30 examples each of instances of four different visual words. Notice that the words represent a moderate-scale local structure in the image (an eye, one and a half letters, and so on). Typical vocabularies are now very large, which means that the instances of each separate word tend to look a lot like one another. *This figure was originally published as Figure 3 of "Efficient Visual Search for Objects in Videos," by J. Sivic and A. Zisserman, Proc. IEEE, Vol. 96, No. 4, April 2008* © *IEEE 2008.*

Visual words

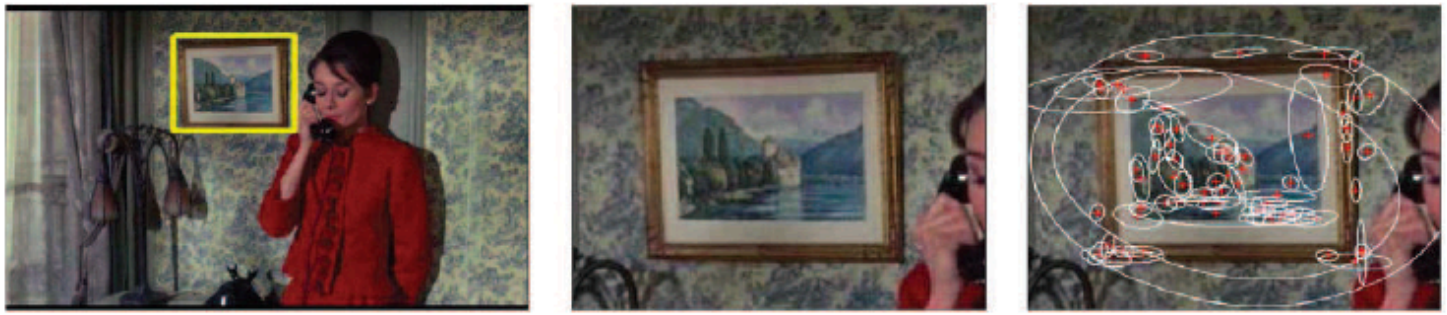


FIGURE 16.5: The original application of visual word representations was to search video sequences for particular patterns. On the **left**, a user has drawn a box around a pattern of interest in a frame of video; the **center** shows a close-up of the box. On the **right**, we see neighborhoods computed from this box. These neighborhoods are ellipses, rather than circles; this means that they are covariant under affine transforms. Equivalently, the neighborhood constructed for an affine transformed patch image will be the affine transform of the neighborhood constructed for the original patch (definition in Section 5.3.2). *This figure was originally published as Figure 11 of J. Sivic and A. Zisserman "Efficient Visual Search for Objects in Videos," Proc. IEEE, Vol. 96, No. 4, April 2008*

Visual words

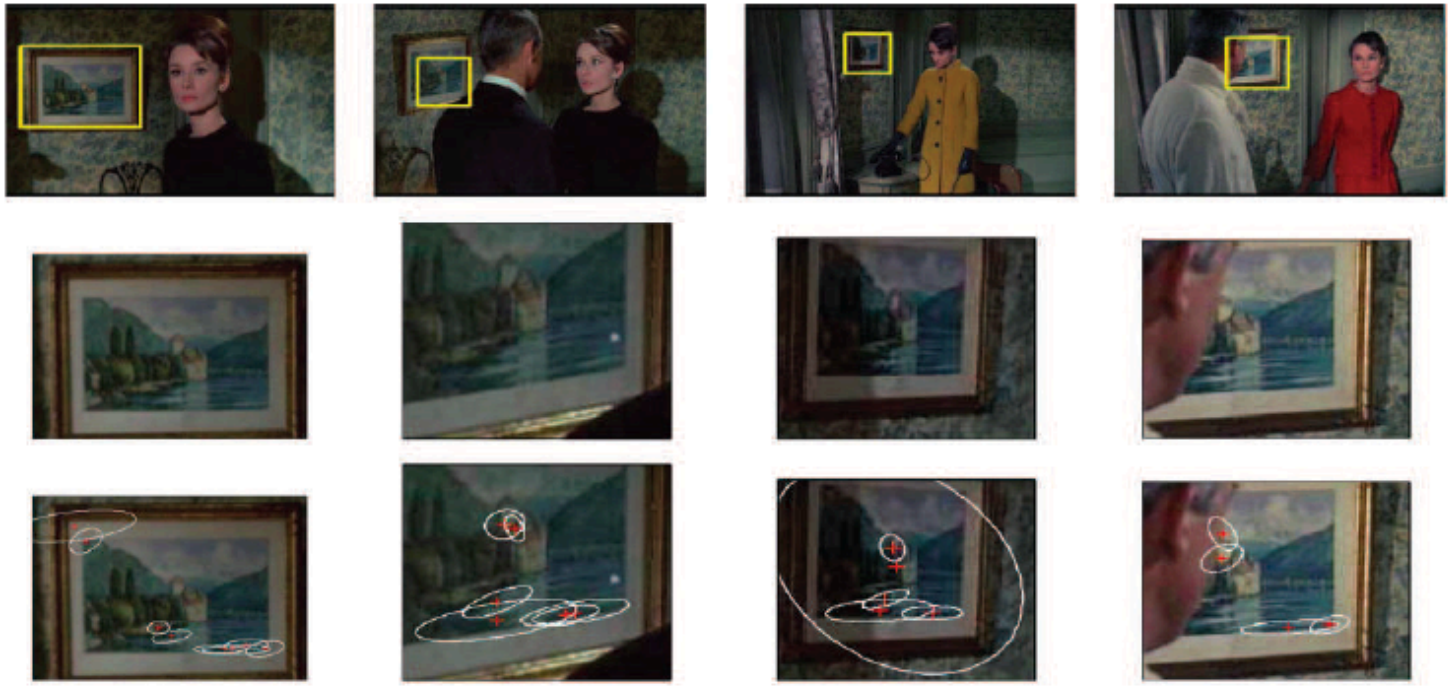


FIGURE 16.7: This figure shows results from the query of Figure 16.5, obtained by looking for image regions that have a set of visual words strongly similar to those found in the query region. The first row shows the whole frame from the video sequence; the second row shows a close-up of the box that is the result (indicated in the first row); and the third row shows the neighborhoods in that box that generated visual words that match those in the query. Notice that some, but not all, of the neighborhoods in the query were matched. *This figure was originally published as Figure 11 of J. Sivic and A. Zisserman "Efficient Visual Search for Objects in Videos," Proc. IEEE, Vol. 96, No. 4, April 2008*

Features from visual words

- Histogram
  - good summary of what is in image; quick and efficient
  - insensitive to spatial reorganization
- Spatial pyramid
  - build histograms of local blocks at various scales
  - less insensitive to spatial reorganization

Spatial pyramids

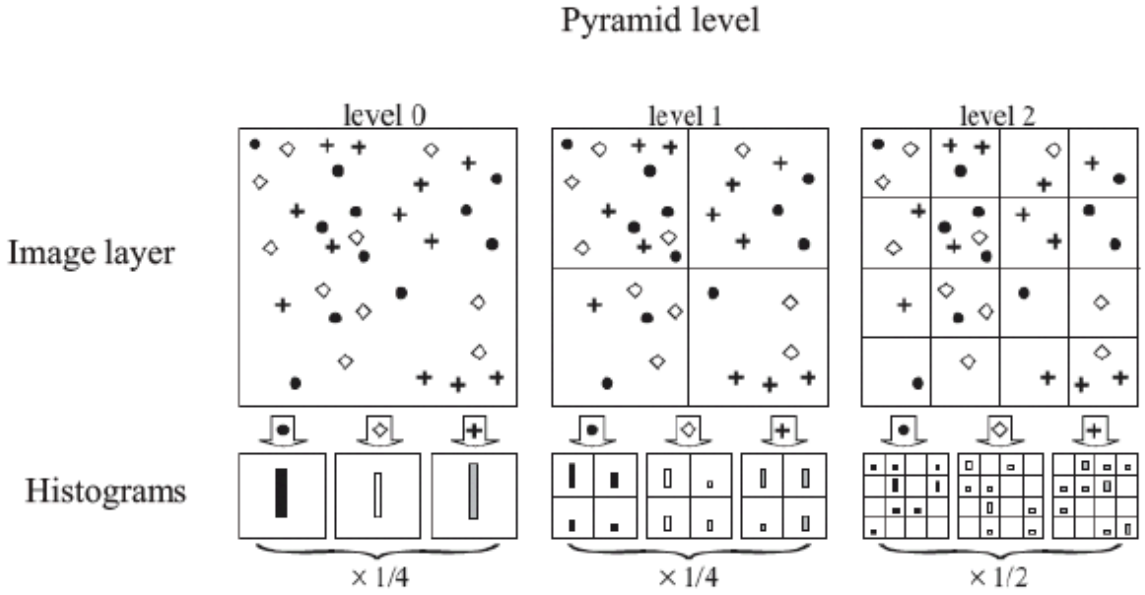


FIGURE 16.8: A simplified example of constructing a spatial pyramid kernel, with three levels. There are three feature types, too (circles, diamonds, and crosses). The image is subdivided into one-, four-, and sixteen-grid boxes. For each level, we compute a histogram of how many features occur in each box for each feature type. We then compare two images by constructing an approximate score of the matches from these histograms. *This figure was originally published as Figure 1 of "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," by S. Lazebnik, C. Schmid, and J. Ponce, Proc. IEEE CVPR 2006, © IEEE 2006.*

Evaluation

- Precision
  - percentage of items in retrieved set that are relevant
- Recall
  - percentage of relevant items that are retrieved
- Precision vs recall
  - use classifier to label a collection of images
  - now plot precision against recall for different classifier thresholds

Evaluation

- AP
  - average precision
  - average of precision as a function of recall

Write rel(r) for the binary function that is one when the rth document is relevant, and otherwise zero; $P(r)$ for the precision of the first r documents in the ranked list; N for the number of documents in the collection; and N_r for the total number of relevant documents. Then, average precision is given by

$$A = \frac{1}{N_r} \sum_{r=1}^{N} (P(r)\text{rel}(r))$$

Notice that average precision is highest (100%) when the top N_r documents are the relevant documents. Averaging over all the relevant documents means the

Precision vs recall

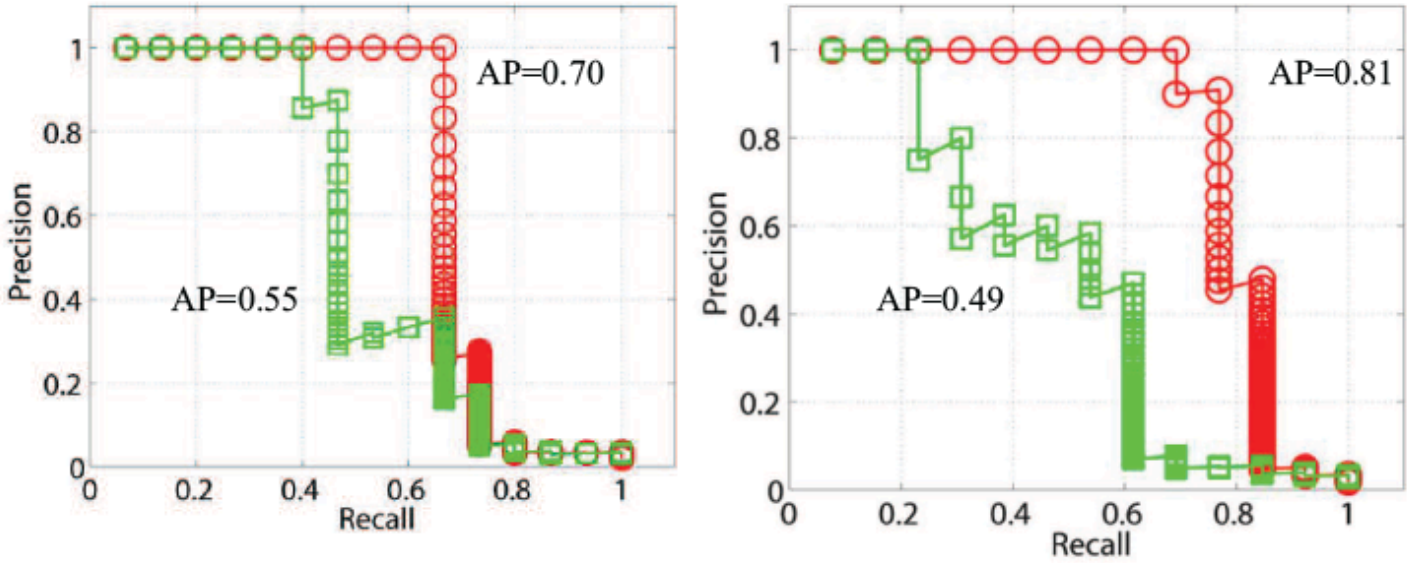


FIGURE 16.19: Plots of precision as a function of recall for six object queries. Notice how precision generally declines as recall goes up (the occasional jumps have to do with finding a small group of relevant images; such jumps would become arbitrarily narrow and disappear in the limit of an arbitrarily large dataset). Each query is made using the system sketched in Figure 16.5. Each graph shows a different query, for two different configurations of that system. On top of each graph, we have indicated the average precision for each of the configurations. Notice how the average precision is larger for systems where the precision is higher for each recall value. *This figure was originally published as Figure 9 of J. Sivic and A. Zisserman "Efficient Visual Search for Objects in Videos," Proc. IEEE, Vol. 96, No. 4, April 2008* © *IEEE 2008.*

Category	2007	2008	2009	2010
aeroplane	0.775	0.811	0.881	0.933
bicycle	0.636	0.543	0.686	0.790
bird	0.561	0.616	0.681	0.716
boat	0.719	0.678	0.729	0.778
bottle	0.331	0.300	0.442	0.543
bus	0.606	0.521	0.795	0.859
car	0.780	0.595	0.725	0.804
cat	0.588	0.599	0.708	0.794
chair	0.535	0.489	0.595	0.645
cow	0.426	0.336	0.536	0.662
diningtable	0.549	0.408	0.575	0.629
dog	0.458	0.479	0.593	0.711
horse	0.775	0.673	0.731	0.820
motorbike	0.640	0.652	0.723	0.844
person	0.859	0.871	0.853	0.916
pottedplant	0.363	0.318	0.408	0.533
sheep	0.447	0.423	0.569	0.663
sofa	0.509	0.454	0.579	0.596
train	0.792	0.778	0.860	0.894
tvmonitor	0.532	0.647	0.686	0.772
# methods	2	5	4	6
# comp	17	18	48	32

TABLE 16.1: Average precision of the best classification method for each category for the Pascal image classification challenge by year (per category; the method that was best at "person" might not be best at "pottedplant"), summarized from `http://pascallin.ecs.soton.ac.uk/challenges/VOC/`. The **bottom** rows show the number of methods in each column and the total number of methods competing (so, for example, in 2007, only 2 of 17 total methods were best in category; each of the other 15 methods was beaten by something for each category). Notice that the average precision grows, but not necessarily monotonically (this is because the test set changes). Most categories now work rather well.

Can be hard

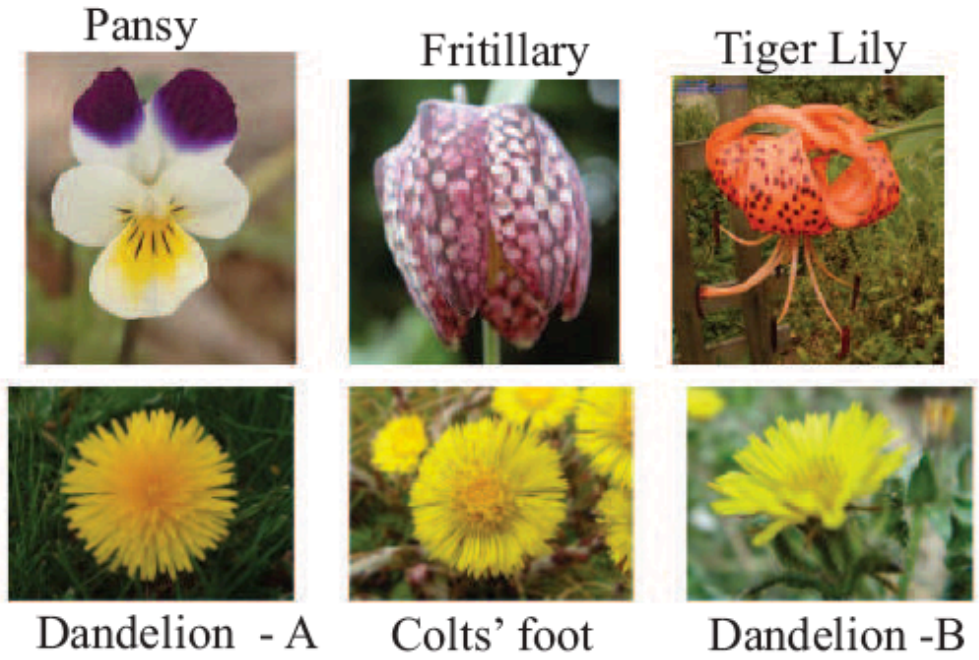


FIGURE 16.21: Identifying a flower from an image is one useful specialized application for image classification techniques. This is a challenging problem. Although some flowers have quite distinctive features (for example, the colors and textures of the pansy, the fritillary, and the tiger lily), others are easy to confuse. Notice that dandelion-A (**bottom**) looks much more like the colts' foot than like dandelion-B. Here the within-class variation is high because of changes of aspect, and the between-class variation is small. *This figure was originally published as Figures 1 and 8 of "A Visual Vocabulary for Flower Classification," by M.E. Nilsback and A. Zisserman, Proc. IEEE CVPR 2006,* © *IEEE 2006.*

Detection with a classifier

- Search
  - all windows
  - at relevant scales
- Prepare features
- Classify

- Issues
  - how to get only one response
  - speed
  - accuracy

Detection with a classifier

Train a classifier on $n \times m$ image windows. Positive examples contain the object and negative examples do not.
Choose a threshold t and steps Δx and Δy in the x and y directions

Construct an image pyramid.

For each level of the pyramid
  Apply the classifier to each $n \times m$ window, stepping by Δx and Δy, in this level to get a response strength c.
  If $c > t$
    Insert a pointer to the window into a ranked list $\mathcal{L}$, ranked by c.

For each window $\mathcal{W}$ in $\mathcal{L}$, starting with the strongest response
  Remove all windows $\mathcal{U} \neq \mathcal{W}$ that overlap $\mathcal{W}$ significantly,
    where the overlap is computed in the original image by expanding windows in coarser scales.

$\mathcal{L}$ is now the list of detected objects.

Algorithm 17.1: Sliding Window Detection.

Non-maximum suppression

- Compute “strength of response”
  - SVM value
  - LR value
- Threshold
  - small values are not faces
- Find largest value (over location, scale)
  - suppress nearby values
  - repeat

Core applications

- Face detection
  - now very successful for frontal faces
  - less so for 3/4, profile views
- Pedestrian detection
  - eg make cars safer
- Generic object detection
  - explain pictures
    - image search
    - robotics applications
    - surveillance applications

In-plane rotation

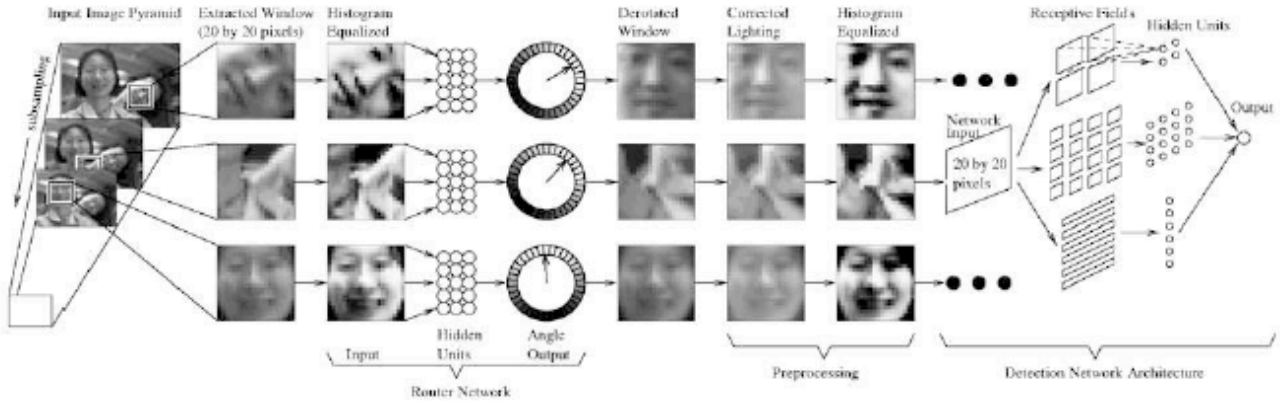


FIGURE 17.2: The architecture of Rowley, Baluja, and Kanade's system for finding faces. Image windows of a fixed size are corrected to a standard illumination using histogram equalization; they are then passed to a neural net that estimates the orientation of the window. The windows are reoriented and passed to a second net that determines whether a face is present. *This figure was originally published as Figure 2 from "Rotation invariant neural-network based face detection," H.A. Rowley, S. Baluja, and T. Kanade, Proc. IEEE CVPR, 1998, © IEEE, 1998.*

FIGURE 17.3: Typical responses for the Rowley, Baluja, and Kanade system for face finding; a mask icon is superimposed on each window that is determined to contain a face. The orientation of the face is indicated by the configuration of the eye holes in the mask. *This figure was originally published as Figure 7 from "Rotation invariant neural-network based face detection," H.A. Rowley, S. Baluja, and T. Kanade, Proc. IEEE CVPR, 1998,* © *IEEE, 1998.*

Fast Features for Faces

FIGURE 17.4: Face detection can be made extremely fast using features that are easy to evaluate, and that can reject most windows early. On the **left**, features can be built up out of sums of the image within boxes, weighted by 1 or -1. The drawings show two two-box features (some readers might spot a relationship to Haar wavelets). On the **right**, the features used for the first two tests (equivalently, the first two classifiers in the cascade) by Viola and Jones (2001). Notice how they check for the distinctive dark bar at the eyes with the lighter bar at the cheekbones, then the equally distinctive vertical specularity along the nose and forehead. *This figure was originally published as Figures 1 and 3 from "Rapid Object Detection using a Boosted Cascade of Simple Features," by P. Viola and M. Jones, Proc. IEEE CVPR 2001* © *IEEE 2001.*

Pedestrians: Scissors and Lollipops

FIGURE 17.5: Examples of pedestrian windows from the INRIA pedestrian dataset, collected and published by Dalal and Triggs (2005). Notice the relatively strong and distinctive curve around the head and shoulders; the general "lollipop" shape, caused by the upper body being wider than the legs; the characteristic "scissors" appearance of separated legs; and the strong vertical boundaries around the sides. These seem to be the cues used by classifiers. *This figure was originally published as Figure 2 of "Histograms of Oriented Gradients for Human Detection," N. Dalal and W. Triggs, Proc. IEEE CVPR 2005, © IEEE, 2005.*

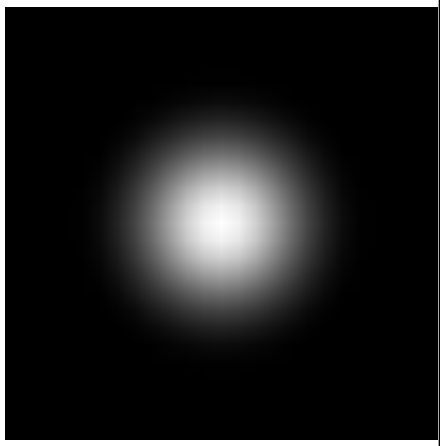
Convolution

- Each pixel in output image is
  - weighted average of window of pixels in input image
  - weights stay the same
  - window centered on pixel
- Important operation
  - Example: smoothing by averaging
  - Example: smoothing by weighted average
  - Example: taking a derivative

Convolution - Example II

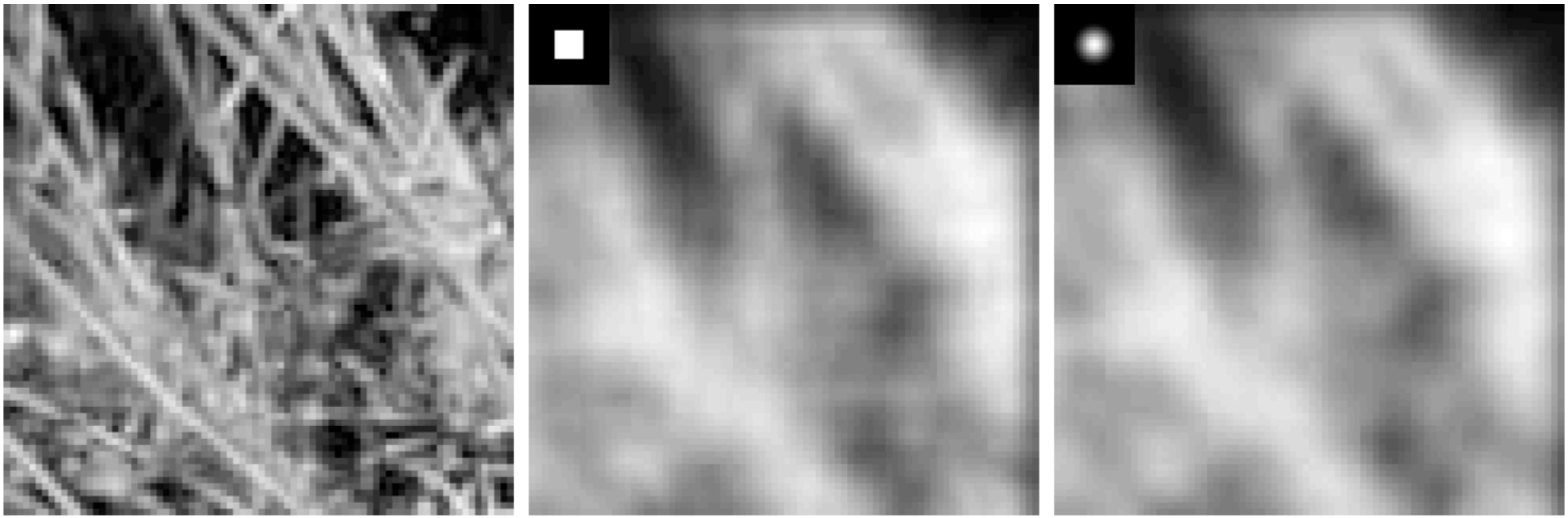
- Averaging neighbors yields poor smoothing
  - look at picture - ringing effects
  - distant neighbors have the same effect as nearby neighbors
- Idea:
  - distant neighbors have small weights, nearby have large
  - weights from Gaussian

$$H_{uv} = \left(\frac{1}{2\pi\sigma^2}\right) \exp\left(\frac{-\left[u^2+v^2\right]}{2\sigma^2}\right)$$



Smoothing with a Gaussian

Inset: smoothing weights



Input image | Output image, average | Output image, gaussian weights

Figure 4.1

Recall: Edges

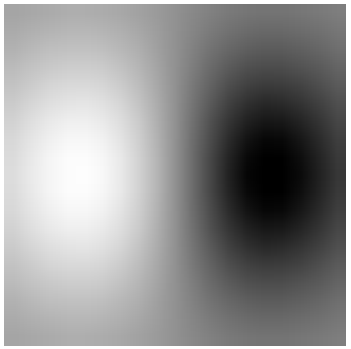
- Idea:
  - points where image value change very sharply are important
    - changes in surface reflectance
    - shadow boundaries
    - outlines
- Finding Edges:
  - Estimate gradient magnitude using appropriate smoothing
  - Mark points where gradient magnitude is
    - Locally biggest and
    - big

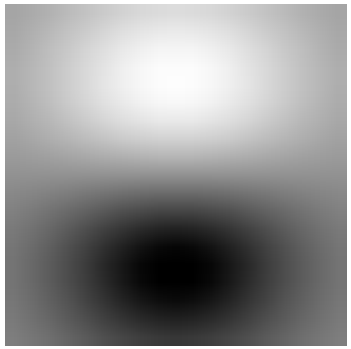
Recall: Smoothed gradients

- Fact: These two are the same
  - Smooth, then differentiate
  - Filter with derivative of Gaussian

$$\frac{\partial \left(G_\sigma * * I\right)}{\partial x} = \left(\frac{\partial G_\sigma}{\partial x}\right) * * I$$

- Exploit:
  - Filter image with derivative of Gaussian filters to get smoothed gradient





Edge Maps Depend on Shading

- If the image is brighter (resp. darker)
  - because the camera gain is higher (resp. lower)
  - because there is more (resp. less) light
  - because the pixel values got multiplied by a constant
- Then the gradient magnitude is bigger (resp. smaller)

- So scaling image brightness changes the edge map
  - because some magnitudes will go above (resp. below) the test threshold

- Edge maps differ for brighter/darker copies of a picture

Orientations - I

- Gradient magnitude is affected by illumination changes
  - but gradient direction isn't

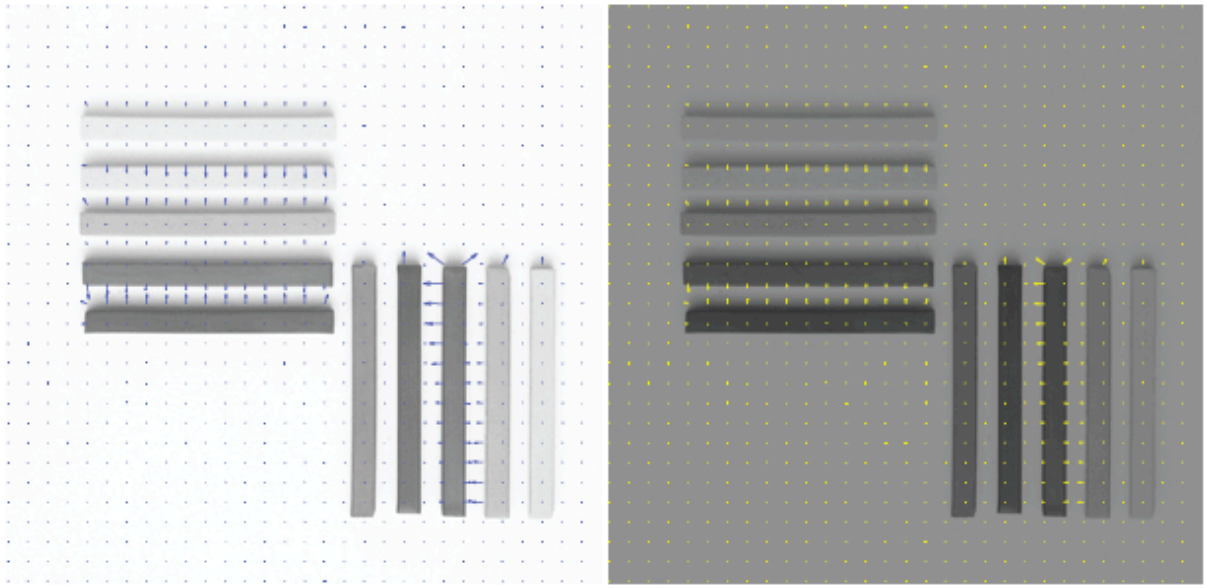


FIGURE 5.7: The magnitude of the image gradient changes when one increases or decreases the intensity. The orientation of the image gradient does not change; we have plotted every 10th orientation arrow, to make the figure easier to read. Note how the directions of the gradient arrows are fixed, whereas the size changes. *Philip Gatward © Dorling Kindersley, used with permission.*

Orientations - II

- Notice larger gradients are “better”
  - we know the orientation better; associated image points “more interesting”

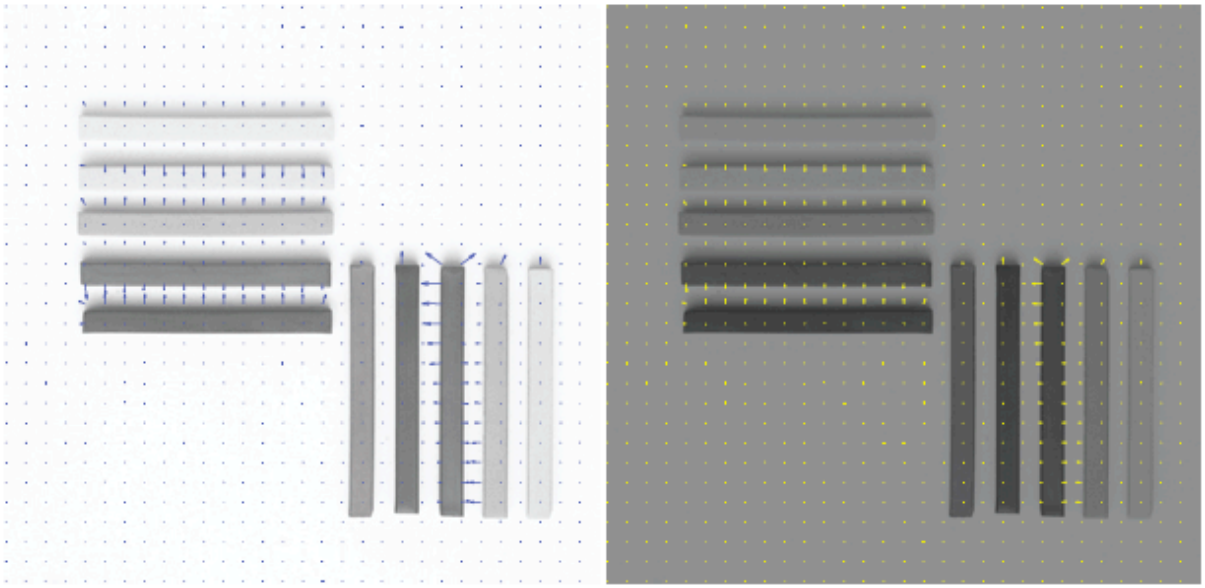


FIGURE 5.7: The magnitude of the image gradient changes when one increases or decreases the intensity. The orientation of the image gradient does not change; we have plotted every 10th orientation arrow, to make the figure easier to read. Note how the directions of the gradient arrows are fixed, whereas the size changes. *Philip Gatward © Dorling Kindersley, used with permission.*

Orientations at different scales

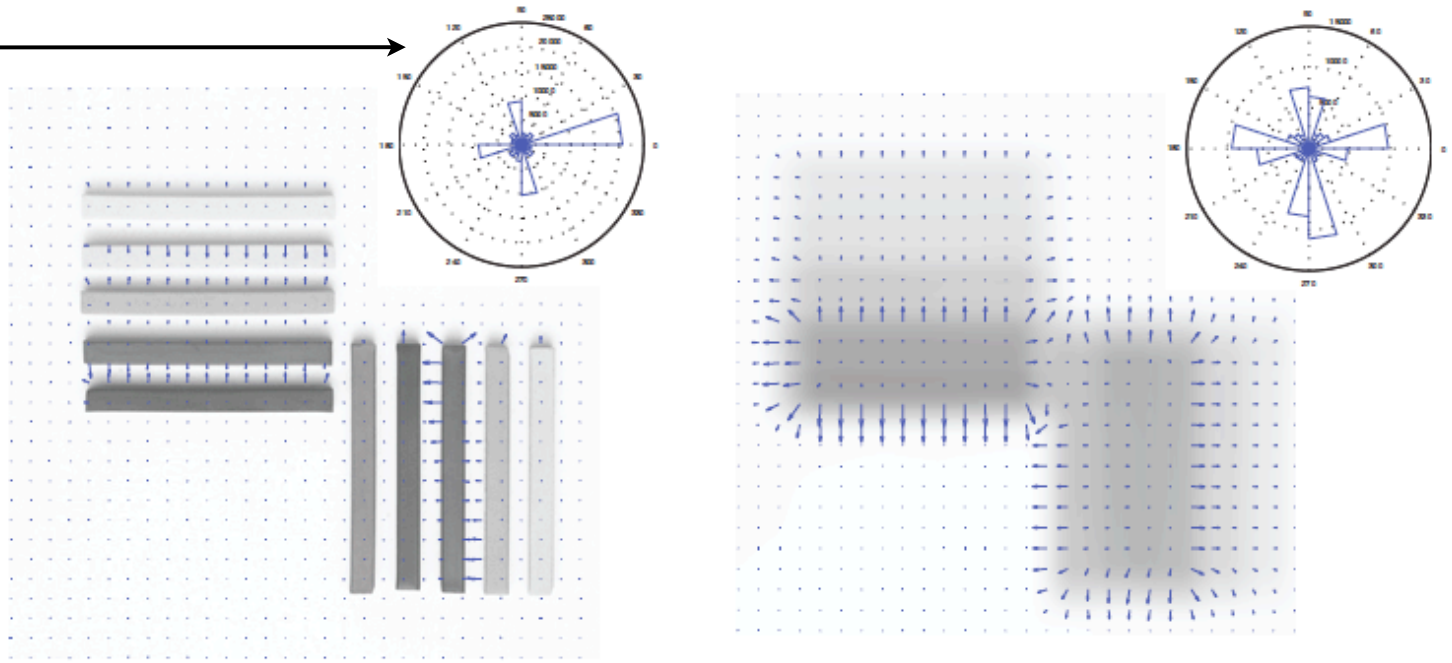


FIGURE 5.8: The scale at which one takes the gradient affects the orientation field. We show the overall trend of the orientation field by plotting a rose plot, where the size of a wedge represents the relative frequency of that range of orientations. **Left** shows an image of artists pastels at a fairly fine scale; here the edges are sharp, and so only a small set of orientations occurs. In the heavily smoothed version on the **right**, all edges are blurred and corners become smooth and blobby; as a result, more orientations appear in the rose plot. *Philip Gatward © Dorling Kindersley, used with permission.*

Orientation Histograms Vary

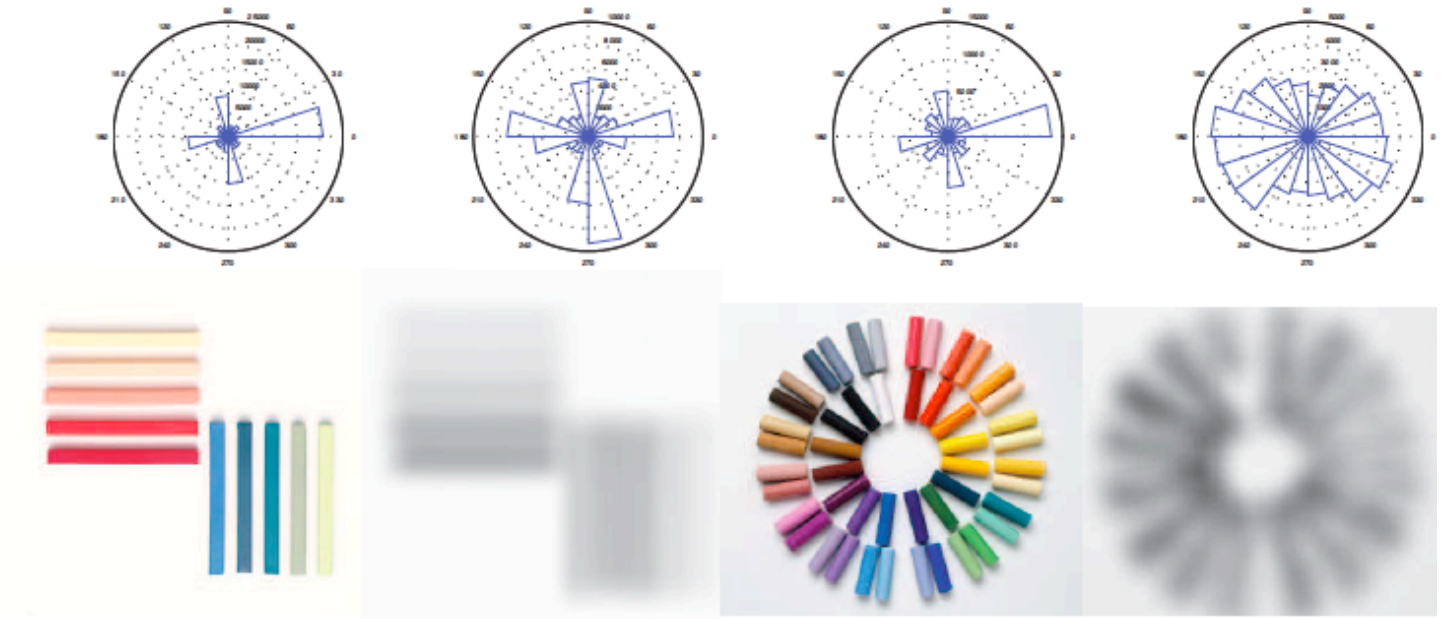


FIGURE 5.9: Different patterns have quite different orientation histograms. The **left** shows rose plots and images for a picture of artists pastels at two different scales; the **right** shows rose plots and images for a set of pastels arranged into a circular pattern. Notice how the pattern of orientations at a particular scale, and also the changes across scales, are quite different for these two very different patterns. *Philip Gatward © Dorling Kindersley, used with permission.*

Building Orientation Representations

- We would like to represent a pattern in an image patch
  - to detect things in images
  - to match points in one image to corresponding points in another image
- Necessary properties
  - we have to know which patch to describe
  - think of this as knowing the center and size of an image window
- Desirable features
  - representation doesn't change much if the center is slightly wrong
  - representation doesn't change much if the size is slightly wrong
  - representation is distinctive
  - representation doesn't change much if the patch gets brighter/darker
  - large gradients are more important than small gradients

Histograms of Oriented Gradients

- Necessary properties
  - we have to know which patch to describe
  - think of this as knowing the center and size of an image window

For the moment, assume window is known

- Desirable features
  - representation doesn't change much if the center is slightly wrong
  - representation doesn't change much if the size is slightly wrong
  - representation is distinctive
  - representation doesn't change much if the patch gets brighter/darker
  - large gradients are more important than small gradients

Use histograms

Use orientations

Break window into boxes, describe each separately

Weight orientation histogram entries

Histograms of Oriented Gradients

- Strategy:
  - break patch up into blocks
  - construct histogram representing gradient orientations in that block
    - which won't change much if the patch moves slightly
    - entries weighted by magnitude
- Variants
  - histogram of angles
  - histogram of gradient vectors, length normalized by block averages

HOG features

Given a grid cell $\mathcal{G}$ for patch with center $\boldsymbol{c} = (x_c, y_c)$ and radius r

Create an orientation histogram
For each point $\boldsymbol{p}$ in an $m \times m$ subgrid spanning $\mathcal{G}$
  Compute a gradient estimate $\nabla \mathcal{I} \mid_{\boldsymbol{p}}$ estimate at $\boldsymbol{p}$
    as a weighted average of $\nabla \mathcal{I}$, using bilinear weights centered at $\boldsymbol{p}$.
  Add a vote with weight $\| \nabla \mathcal{I} \| \frac{1}{r\sqrt{2\pi}} \exp\left(-\frac{\|\boldsymbol{p}-\boldsymbol{c}\|^2}{r^2} \right)$
    to the orientation histogram cell for the orientation of $\nabla \mathcal{I}$.

Algorithm 5.5: Computing a Weighted q Element Histogram for a SIFT Feature.

HOG features

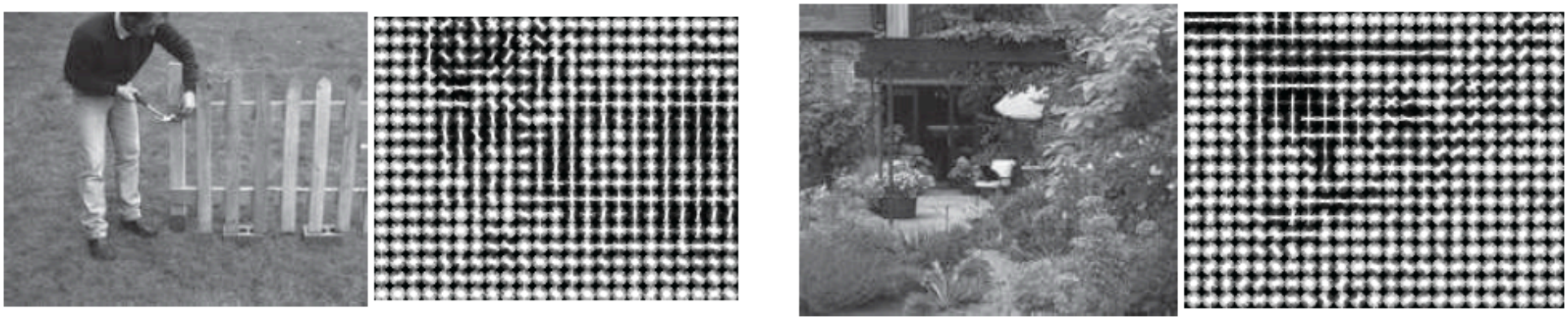


FIGURE 5.15: The HOG features for each the two images shown here have been visualized by a version of the rose diagram of Figures 5.7–5.9. Here each of the cells in which the histogram is taken is plotted with a little rose in it; the direction plotted is at right angles to the gradient, so you should visualize the overlaid line segments as edge directions. Notice that in the textured regions the edge directions are fairly uniformly distributed, but strong contours (the gardener, the fence on the **left**; the vertical edges of the french windows on the **right**) are very clear. This figure was plotted using the toolbox of Dollár and Rabaud. *Left:* © *Dorling Kindersley, used with permission. Right: Geoff Brightling* © *Dorling Kindersley, used with permission.*

HOG - Crucial Points

- Gradient orientations are not affected by intensity
- Orientations with larger magnitude are more important
- Describe an image window of known location, size
  - Histograms reduce the effect of poor estimate of location, size
  - Break window into subwindows
    - for each, compute an orientation histogram, weighting orientations by magnitude
  - Numerous variants available

Visualizing Distinctive Features

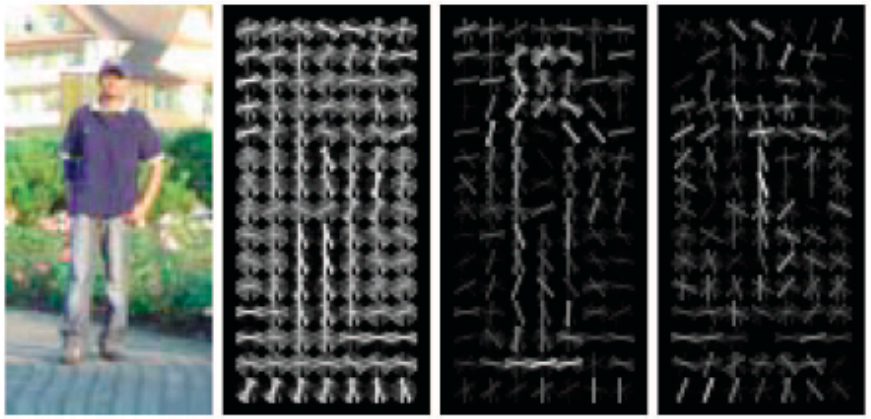


FIGURE 17.7: As Figure 17.6 indicates, a linear SVM works about as well as the best detector for a pedestrian detector. Linear SVMs can be used to visualize what aspects of the feature representation are distinctive. On the **left**, a typical pedestrian window, with the HOG features visualized on the **center left**, using the scheme of Figure 5.15. Each of the orientation buckets in each window is a feature, and so has a corresponding weight in the linear SVM. On the **center right**, the HOG features weighted by positive weights, then visualized (so that an important feature is light). Notice how the head and shoulders curve and the lollipop shape gets strong positive weights. On the **right**, the HOG features weighted by the absolute value of negative weights, which means a feature that strongly suggests a person is not present is light. Notice how a strong vertical line in the center of the window is deprecated (because it suggests the window is not centered on a person). *This figure was originally published as Figure 6 of "Histograms of Oriented Gradients for Human Detection," N. Dalal and W. Triggs, Proc. IEEE CVPR 2005, © IEEE, 2005.*

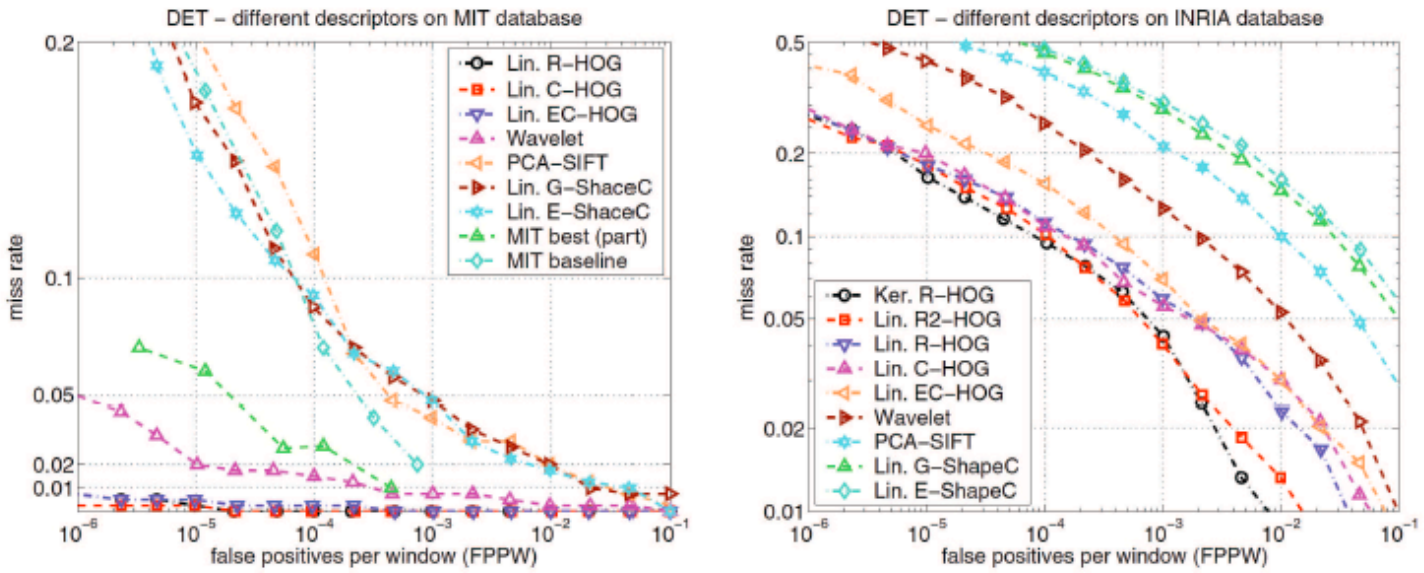


FIGURE 17.6: The performance of the pedestrian detector of Dalal and Triggs (2005), for various choices of features and two different datasets. On the **left**, results using the MIT pedestrian dataset, and on the **right**, results using the INRIA dataset. The results are reported as the miss rate (so smaller is better) against the false positive per window (FPPW) rate, and so evaluate the classifier rather than the system. Overall system performance will depend on how many windows are presented to the detector in an average image (details in the text; see Figure 17.8). Notice that different datasets result in quite different performance levels. The best performance on the INRIA dataset (which is quite obviously the harder dataset) is obtained with a kernel SVM (circles, Ker. R-HOG), but there is very little difference between this and a linear SVM (squares, Lin. R2-HOG). *This figure was originally published as Figure 3 of "Histograms of Oriented Gradients for Human Detection," N. Dalal and W. Triggs, Proc. IEEE CVPR 2005, © IEEE, 2005.*

16.3.1 Codes for Image Features

Oliva and Torralba provide GIST feature code at `http://people.csail.mit.edu/torralba/code/spatialenvelope/`, together with a substantial dataset of outdoor scenes.

Color descriptor code, which computes visual words based on various color SIFT features, is published by van de Sande *et al* at `http://koen.me/research/colordescriptors/`.

The pyramid match kernel is an earlier variant of the spatial pyramid kernel described in Section 16.1.4; John Lee provides a library, `libpmk`, that supports this kernel at `http://people.csail.mit.edu/jjl/libpmk/`. There are a variety of extension libraries written for `libpmk`, including implementations of the pyramid kernel, at this URL.

Li Fei-Fei, Rob Fergus, and Antonio Torralba publish example codes for core object recognition methods at `http://people.csail.mit.edu/torralba/shortCourseRLOC/`. This URL is the online repository associated with their very successful short course on recognizing and learning object categories.

VLFeat is an open-source library that implements a variety of popular computer vision algorithms, initiated by Andrea Vedaldi and Brian Fulkerson; it can be found at `http://www.vlfeat.org`. VLFeat comes with a set of tutorials that show how to use the library, and there is example code showing how to use VLFeat to classify Caltech-101.

There is a repository of code links at `http://featurespace.org`.

At the time of writing, multiple-kernel learning methods produce the strongest results on standard problems, at the cost of quite substantial learning times. Section 15.3.3 gives pointers to codes for different multiple-kernel learning methods.

16.3.2 Image Classification Datasets

There is now a rich range of image classification datasets, covering several application topics. Object category datasets have images organized by category (e.g., one is distinguishing between "bird"s and "motorcycle"s, rather than between particular species of bird). Five classes (motorbikes, airplanes, faces, cars, spotted cats, together with background, which isn't really a class) were introduced by Fergus *et al.* (2003) in 2003; they are sometimes called Caltech-5. Caltech-101 has 101 classes, was introduced in Perona *et al.* (2004) and by Fei-Fei *et al.* (2006), and can be found at `http://www.vision.caltech.edu/Image_Datasets/Caltech101/`. This dataset is now quite well understood, but as Figure 16.20 suggests, it is not yet exhausted. Caltech-256 has 256 classes, was introduced by (Griffin *et al.* 2007), and can be found at `http://www.vision.caltech.edu/Image_Datasets/Caltech256/`. This dataset is still regarded as challenging.

LabelMe is an image annotation environment that has been used by many users to mark out and label objects in images; the result is a dataset that is changing and increasing in size as time goes on. LabelMe was introduced by Russell *et al.* (2008), and can be found at `http://labelme.csail.mit.edu/`.

The Graz-02 dataset contains difficult images of cars, bicycles, and people in natural scenes; it is originally due to Opelt *et al.* (2006), but has been recently reannotated Marszalek and Schmid (2007). The reannotated edition can be found at `http://lear.inrialpes.fr/people/marszalek/data/ig02/`.

Imagenet contains tens of millions of examples, organized according to the Wordnet hierarchy of nouns; currently, there are examples for approximately 17,000 nouns. Imagenet was originally described in Deng *et al.* (2009), and can be found at `http://www.image-net.org/`.

The Lotus Hill Research Institute publishes a dataset of images annoted in detail at `http://www.imageparsing.com`; the institute is also available to prepare datasets on a paid basis.

Each year since 2005 has seen a new Pascal image classification dataset; these are available at `http://pascallin.ecs.soton.ac.uk/challenges/VOC/`.

There are numerous specialist datasets. The Oxford visual geometry group publishes two flower datasets, one with 17 categories and one with 102 categories; each can be found at `http://www.robots.ox.ac.uk/~vgg/data/flowers/`. Other datasets include a "things" dataset, a "bottle" dataset, and a "camel" dataset, all from Oxford (`http://www.robots.ox.ac.uk/~vgg/data3.html`).

There is a bird dataset published by Caltech and UCSD jointly at `http://www.vision.caltech.edu/visipedia/CUB-200.html`.

Classifying materials has become a standard task, with a standard dataset. The Columbia-Utrecht (or CURET) material dataset can be found at `http://www.cs.columbia.edu/CAVE/software/curet/`; it contains image textures from over 60 different material samples observed with over 200 combinations of view and light direction. Details on the procedures used to obtain this dataset can be found in Dana *et al.* (1999). More recently, Liu *et al.* (2010) offer an alternative and very difficult material dataset of materials on real objects, which can be found at `http://people.csail.mit.edu/celiu/CVPR2010/FMD/`.

We are not aware of collections of explicit images published for use as research datasets, though such a dataset would be easy to collect.

There are several scene datasets now. The largest is the SUN dataset (from MIT; `http://groups.csail.mit.edu/vision/SUN/`; Xiao *et al.* (2010)) contains 130,519 images of 899 types of scene; 397 categories have at least 100 examples per category. There is a 15-category scene dataset used in the original spatial pyramid kernel work at `http://www-cvr.ai.uiuc.edu/ponce_grp/data/`.

It isn't possible (at least for us!) to list all currently available datasets. Repositories that contain datasets, and so are worth searching for a specialist dataset, include: the pilot European Image Processing Archive, currently at `http://peipa.essex.ac.uk/index.html`; Keith Price's comprehensive computer vision bibliography, whose root is `http://visionbib.com/index.php`, and with dataset pages at `http://datasets.visionbib.com/index.html`; the Featurespace dataset pages at `http://www.featurespace.org/`; and the Oxford repository at `http:`