Classifying images

D.A. Forsyth

Image classification - features

- 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 (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)

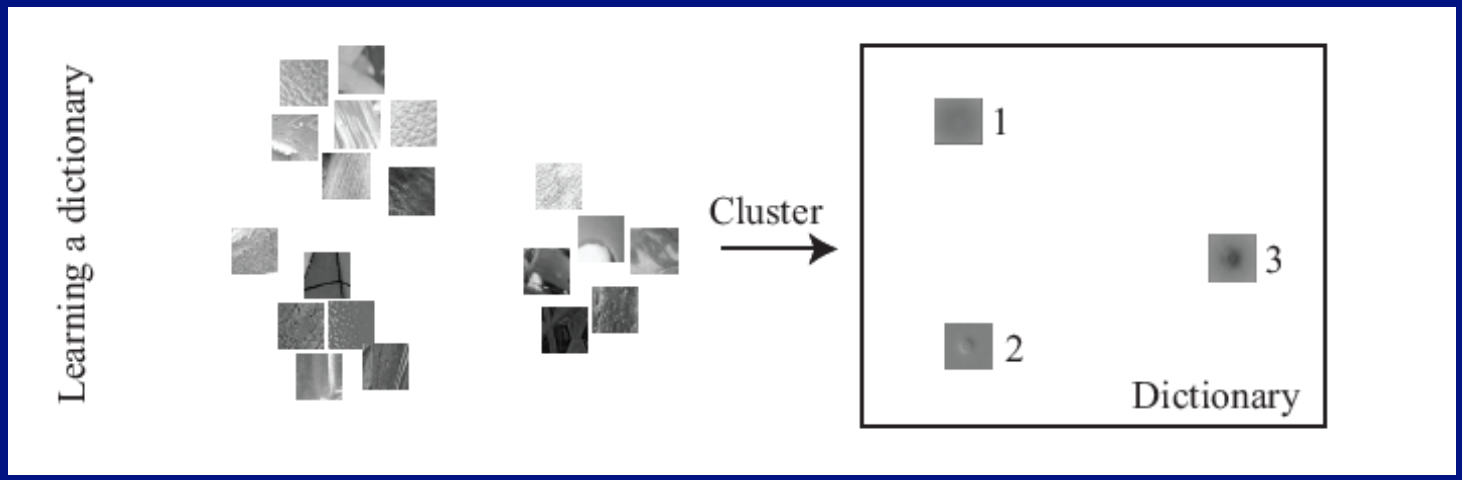
K-means

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.

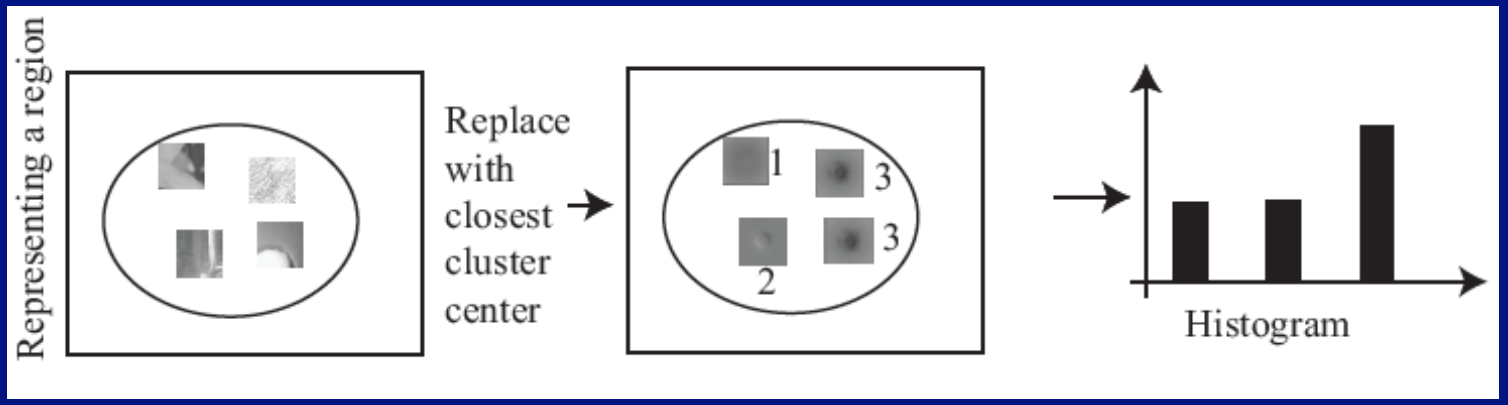
Building visual words - I

- 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

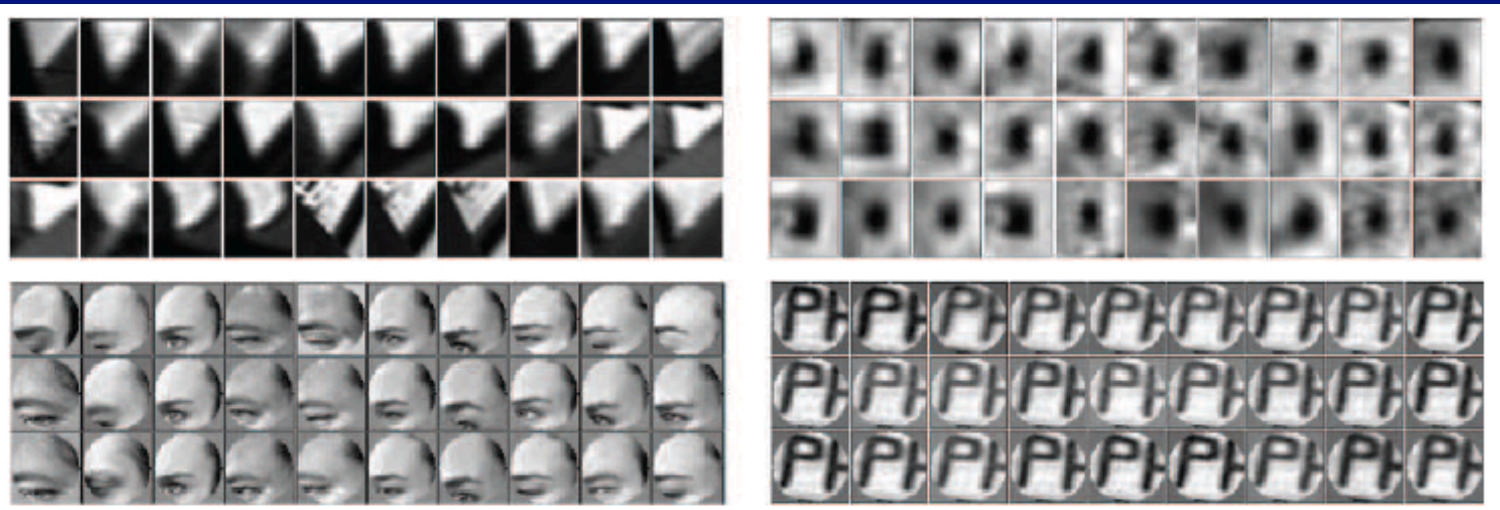


FIGURE 16.6: Visual words are obtained by vector quantizing neighborhoods like those shown in Figure 16.5. This figure shows 30 examples each of instances of four different visual words. Notice that the words represent a moderate-scale local structure in the image (an eye, one and a half letters, and 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

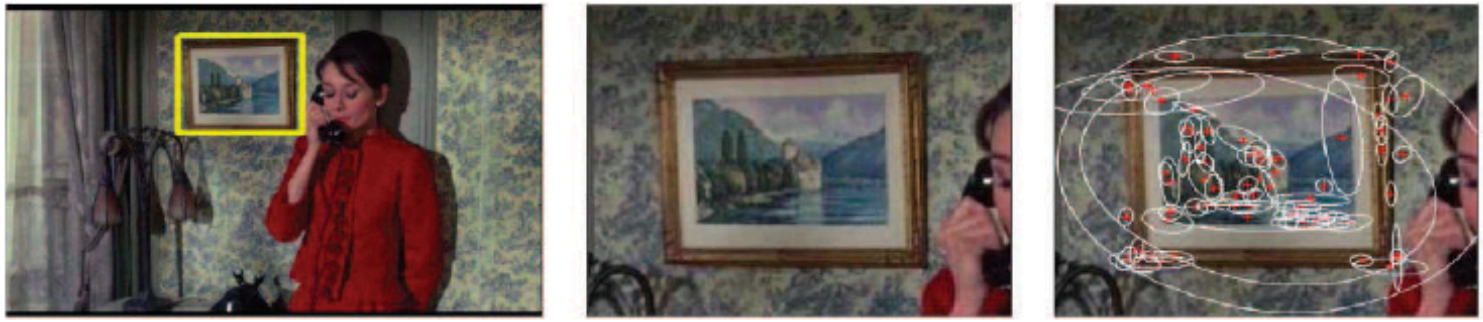


FIGURE 16.5: The original application of visual word representations was to search video sequences for particular patterns. On the **left**, a user has drawn a box around a pattern of interest in a frame of video; the **center** shows a close-up of the box. On the **right**, we see neighborhoods computed 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

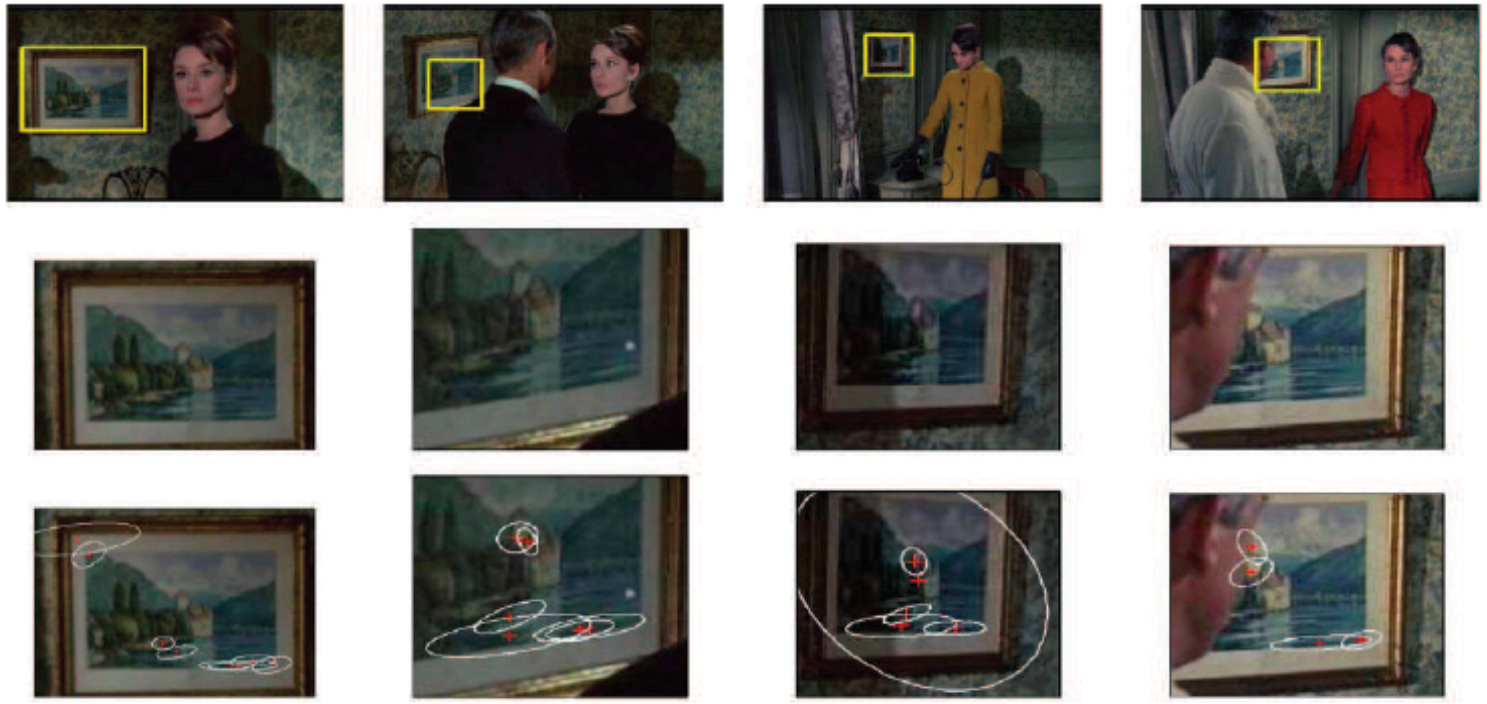


FIGURE 16.7: This figure shows results from the query of Figure 16.5, obtained by looking for image regions that have a set of visual words strongly similar to those found in the query region. The first row shows the whole frame from the video sequence; the second row shows a close-up of the box that 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

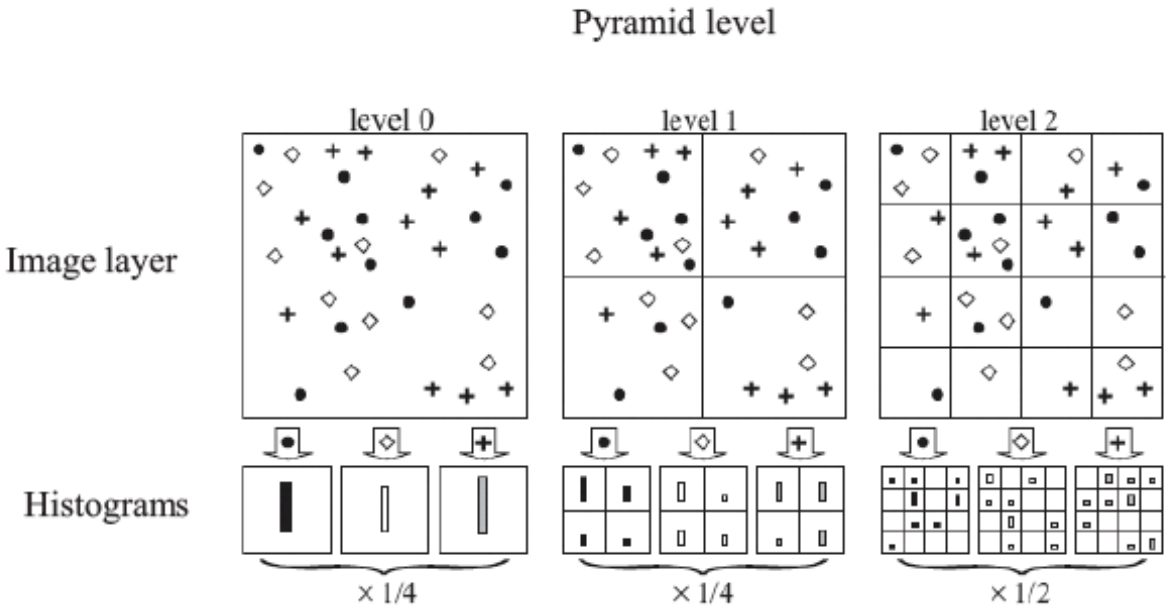


FIGURE 16.8: A simplified example of constructing a spatial pyramid kernel, with three levels. There are three feature types, too (circles, diamonds, and crosses). The image is subdivided into one-, four-, and sixteen-grid boxes. For each level, we compute a histogram of how many features occur in each 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.*

Some standard tasks

- Caltech 101, 256
  - images of isolated objects in 101, 256 categories
- Imagenet
  - same, 1000's of categories
- Pascal image classification
  - not isolated, fewer categories
- Scenes
  - eg indoor, etc.
- Materials

Evaluation methods

- Total error rate
  - percentage of classification attempts that get the wrong answer
- Accuracy
  - percentage of classification attempts that get the right answer
- Class confusion matrix
  - table showing how classes are mixed up
- Look at errors

Accuracy on Caltech

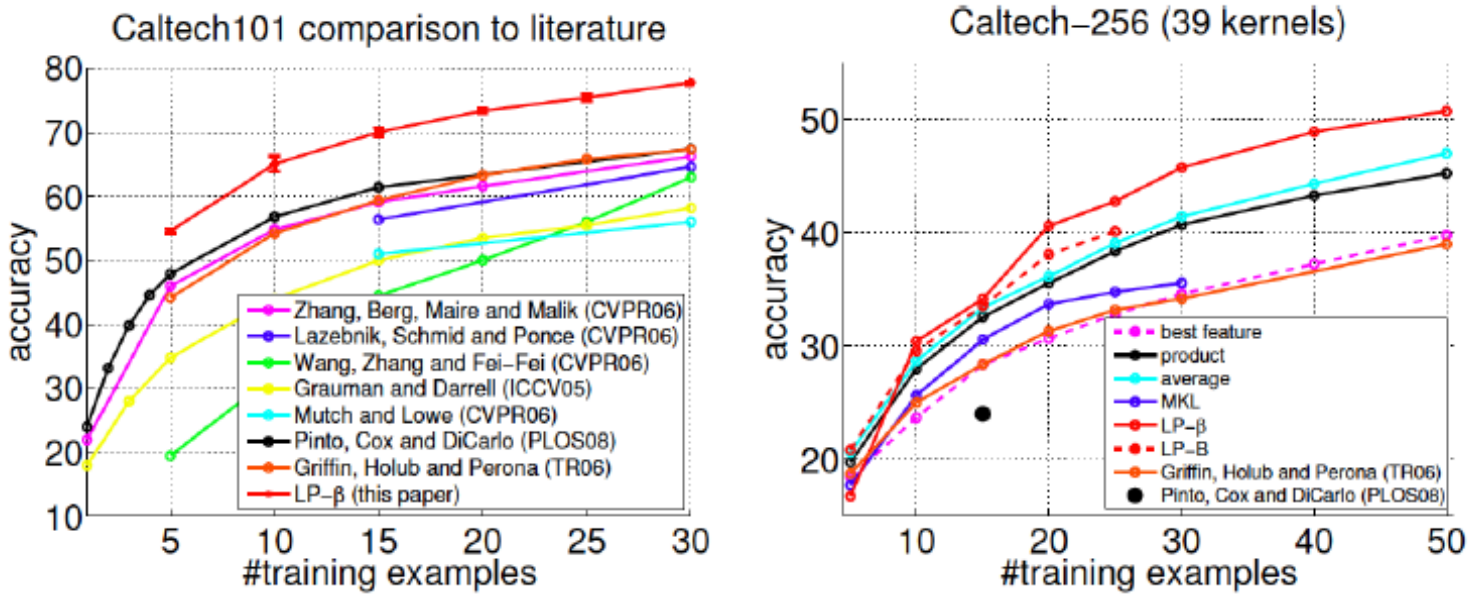


FIGURE 16.20: Graphs illustrating typical performance on Caltech 101 for single descriptor types (**left**) and on Caltech 256 for various types of descriptor (**right**; notice the vertical scale is different), plotted against the number of training examples. Although these figures are taken from a paper advocating nearest neighbor methods, they illustrate performance for a variety of methods. Notice that Caltech 101 results, while not perfect, are now quite strong; the cost of going to 256 categories is quite high. Methods compared are due to: Zhang *et al.* (2006*b*), Lazebnik *et al.* (2006), Wang *et al.* (2006), Grauman and Darrell (2005), Mutch and Lowe (2006), Griffin *et al.* (2007), and Pinto *et al.* (2008); the graph is from Gehler and Nowozin (2009), which describes multiple methods (anything without a named citation on the graph). *This figure was originally published as Figure 2 of "On Feature Combination for Multiclass Object Classification," by P. Gehler and S. Nowozin Proc. ICCV 2009, 2009 © IEEE 2009.*

Material classification

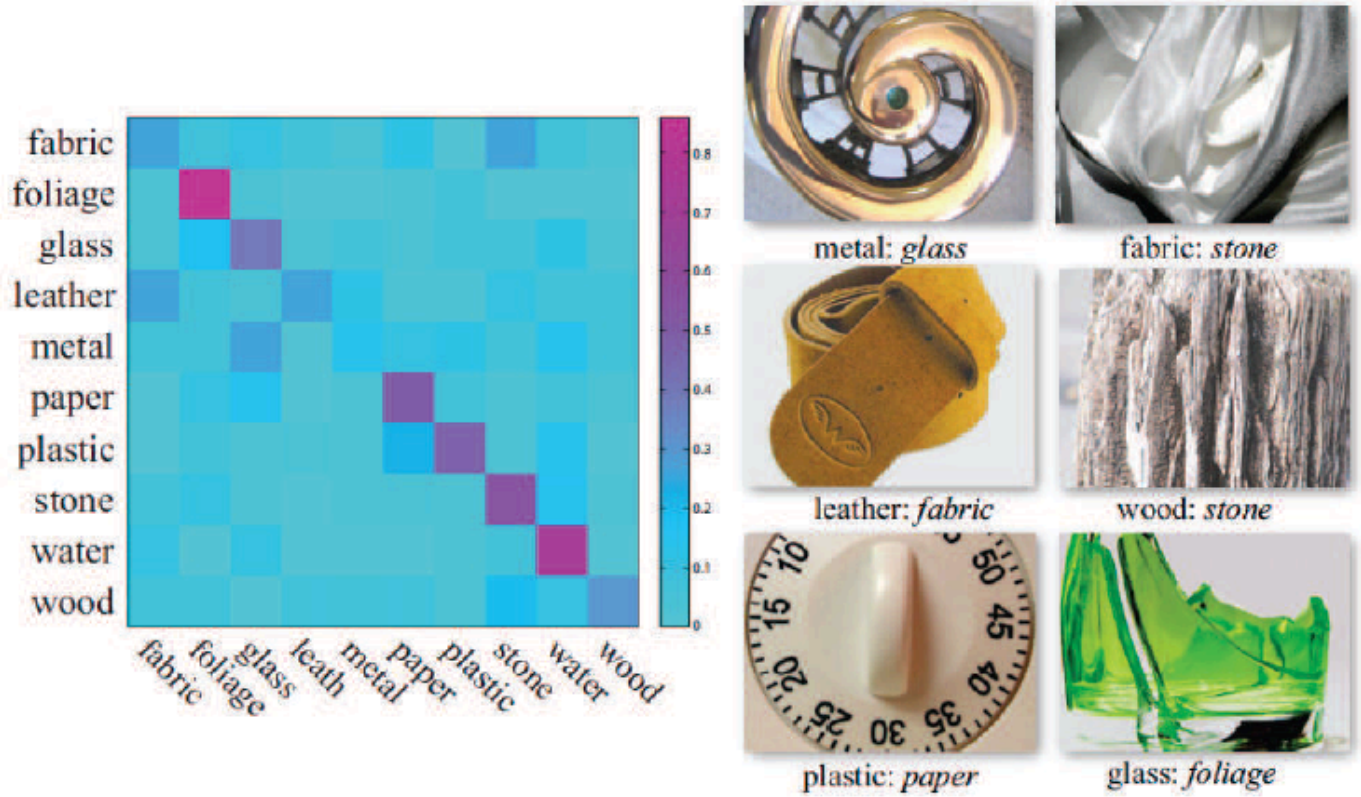


FIGURE 16.17: Liu *et al.* (2010) prepared a material classification dataset from flickr images, and used a combination of SIFT features and novel features to classify the materials. This is a difficult task, as the class confusion matrix on the **left** shows; for example, it is quite easy to mix up metal with most other materials, particularly glass. On the **right**, examples of misclassified images (the italic label is the incorrect prediction). *This figure was originally published as Figure 12 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.*

Spatial pyramids + scenes

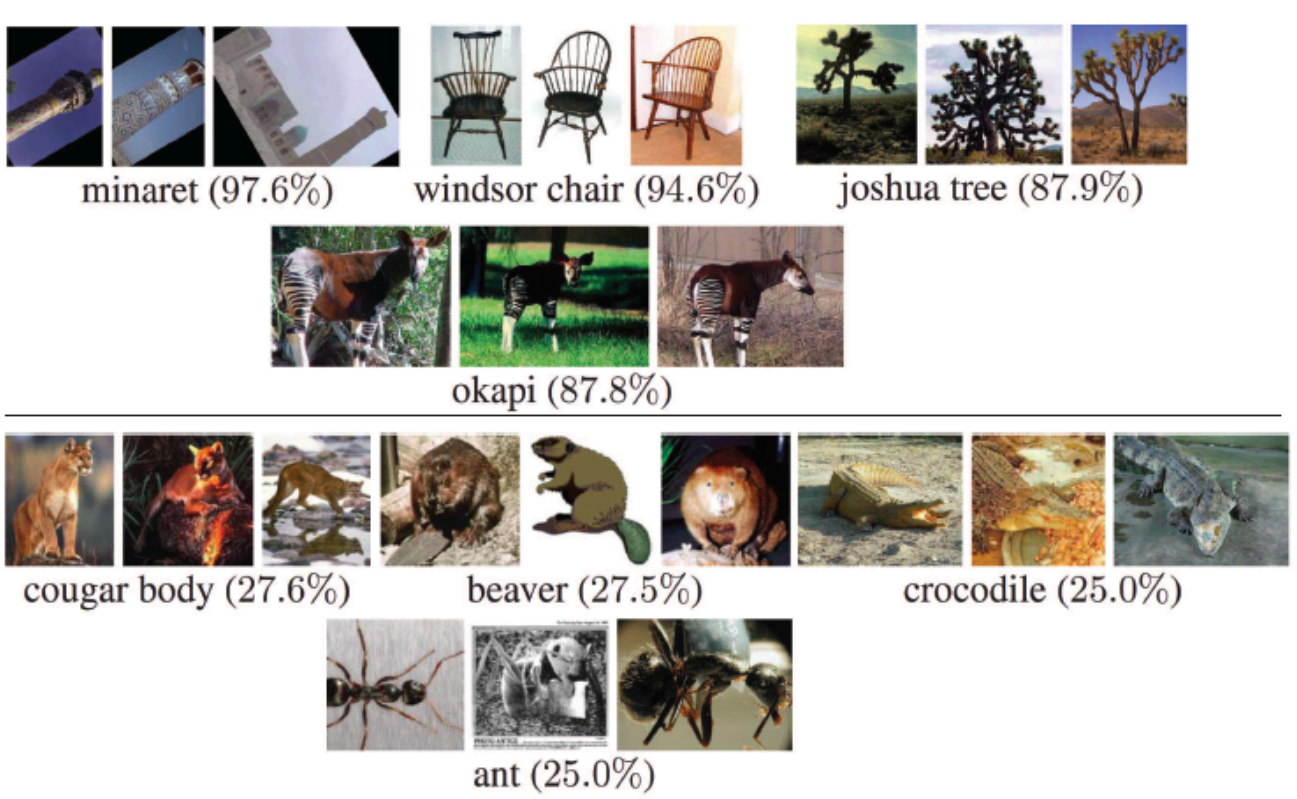


FIGURE 16.10: The spatial pyramid kernel is capable of complex image classification tasks. Here we show some examples of categories from the Caltech 101 collection on which the method does well (**top row**) and poorly (**bottom row**). The number is the percentage of images of that class classified correctly. Caltech 101 is a set of images of 101 categories of objects; one must classify test images into this set of categories (Section 16.3.2). *This figure was originally published as Figure 5 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
- AP
  - average precision
  - average of precision as a function of recall

list. 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

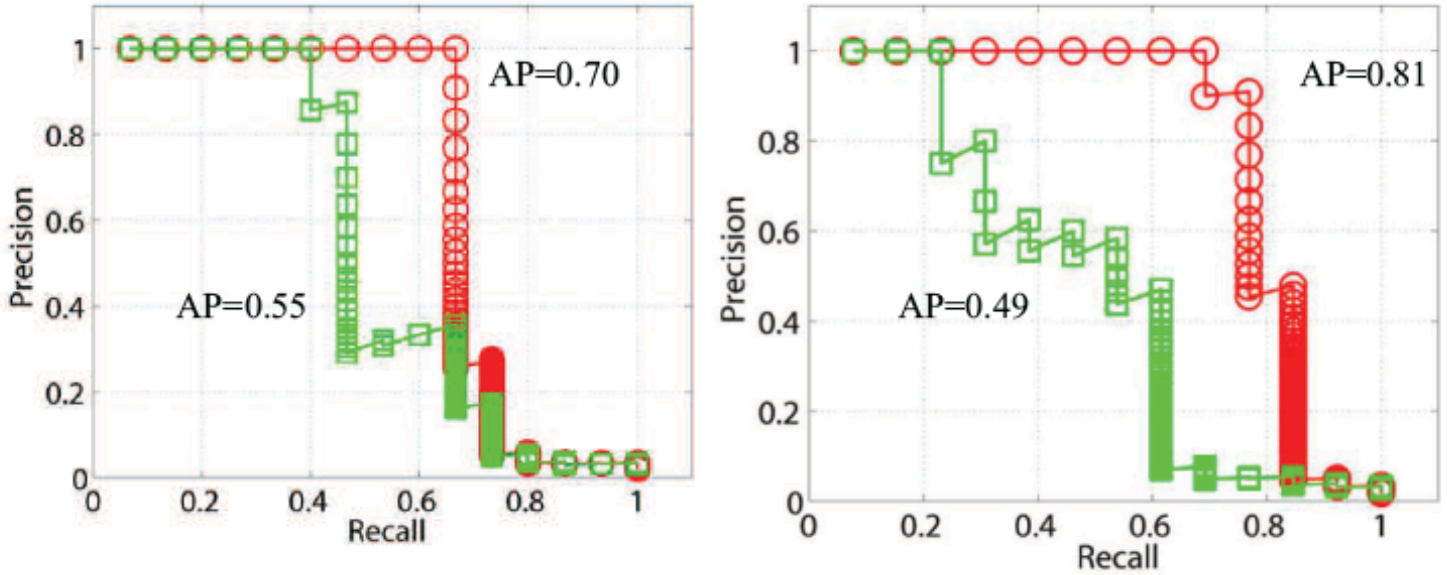


FIGURE 16.19: Plots of precision as a function of recall for six object queries. Notice how precision generally declines as recall goes up (the occasional jumps have to do with finding a small group of relevant images; such jumps would become arbitrarily narrow and disappear in the limit of an arbitrarily 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*

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

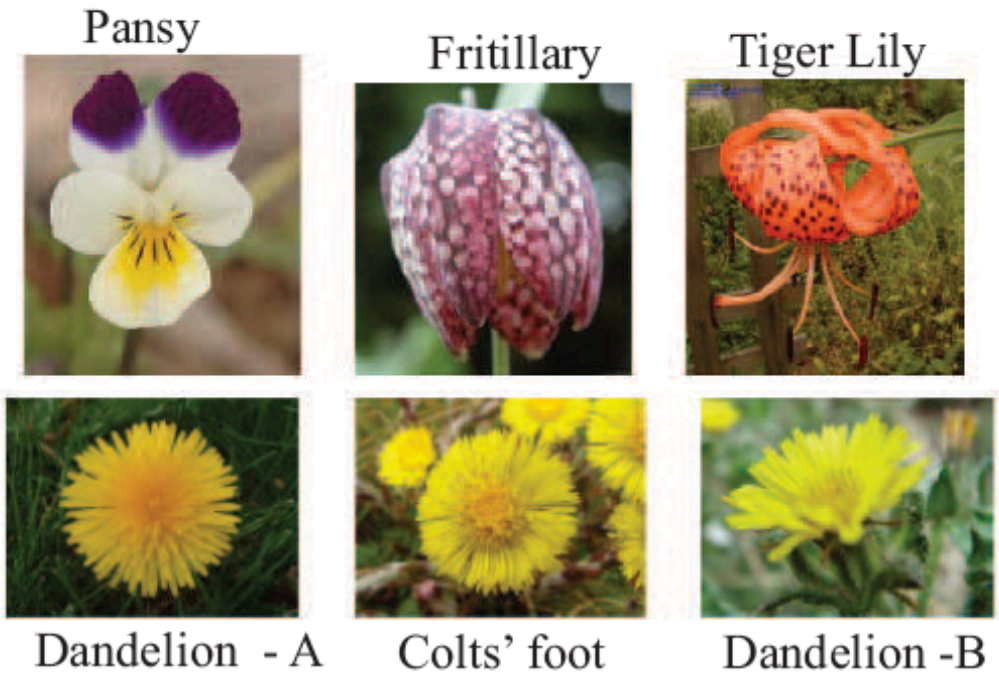


FIGURE 16.21: Identifying a flower from an image is one useful specialized application for image classification techniques. This is a challenging problem. Although some flowers have quite distinctive features (for example, the colors and textures of the pansy, the fritillary, and the tiger lily), others 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.*

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: