Content-Based Retrieval in Digital Libraries

20

20.1 How Should We Retrieve Images?

Consider the image in Fig. 20.1 of a small portion of *The Garden of Delights* by Hieronymus Bosch (1453–1516), now in the Prado museum in Madrid. This is a famous painting, but we may be stumped in understanding the painter's intent. Therefore, if we are aiming at automatic retrieval of images, it should be unsurprising that encapsulating the semantics (meaning) in the image is an even more difficult challenge. A proper annotation of such an image certainly should include the descriptor "people." On the other hand, should this image be blocked by a "Net nanny" screening out "naked people" (as in [1])?

We know very well that web browsers have a web search button for multimedia content (usually images, or video for YouTube and its competitors), as opposed to text. For Bosch's painting, a text-based search will very likely do the best job, should we wish to find this particular image. Yet we may be interested in fairly general searches, say for scenes with deep blue skies and orange sunsets. By pre-calculating some fundamental statistics about images stored in a database, we can usually find simple scenes such as these.

In its inception, retrieval from digital libraries began with ideas borrowed from traditional information retrieval disciplines (see e.g., [2]). This line of inquiry continues [3]. For example, in [4], images are classified into indoor or outdoor classes using basic information-retrieval techniques. For a training set of images and captions, the number of times each word appears in the document is divided by the number of times each word appears over all documents in a class. A similar measure is devised for statistical descriptors of the content of image segments, and the two information-retrieval-based measures are combined for an effective classification mechanism.

However, many multimedia retrieval schemes have moved toward an approach favoring multimedia content itself, either without regard to or reliance upon accompanying textual information, or at least textual-based search bolstered by multimedia evidence. This is commonly known as *CBIR (Content-Based Image Retrieval)*.

Z.-N. Li et al., *Fundamentals of Multimedia*,
Texts in Computer Science, DOI: 10.1007/978-3-319-05290-8_20,

Fig. 20.1 How can we best characterize the information content of an image?

Only recently has attention once more been placed on the deeper problem of addressing semantic content in images, of course also making use of accompanying text (possibly inserted when the media is archived). If data consist of statistical features built from objects in images and also of text associated with the images, each type of modality—text and image—provides semantic content omitted from the other. For example, an image of a red rose will not normally have the manually added keyword "red" since this is generally assumed. Hence, image features and associated words may disambiguate each other (see [5]).

In this chapter, however, we shall focus only on techniques and systems that make use of image features themselves, without text, to retrieve images from databases or from the web. The types of features typically used are such statistical measures as the color histogram for an image. Consider an image that is colorful—say, a Santa Claus plus sled. The combination of bright red and flesh tones and browns might be enough of an image signature to allow us to at least find similar images in our own image database (of office Christmas parties).

Recall that a color histogram is typically a three-dimensional array that counts pixels with specific red, green, and blue values. The nice feature of such a structure is that is does not care about the orientation of the image (since we are simply counting pixel values, not their orientation) and is also fairly impervious to object occlusions. A seminal paper on this subject [6] launched a tidal wave of interest in such so-called "low-level" features for images.

Other simple features used are such descriptors as *color layout*, meaning a simple sketch of where in a checkerboard grid covering the image to look for blue skies and

orange sunsets. Another feature used is *texture*, meaning some type of descriptor typically based on an edge image, formed by taking partial derivatives of the image itself—classifying edges according to closeness of spacing and orientation. An interesting version of this approach uses a histogram of such edge features. *Texture layout* can also be used. Search engines devised on these features are said to be *content-based*: the search is guided by image similarity measures based on the statistical content of each image.

Typically, we might be interested in looking for images similar to our current favorite Santa. A more industry-oriented application would typically be seeking a particular image of a postage stamp, say. Subject fields associated with image database search include art galleries and museums, fashion, interior design, remote sensing, geographic information systems, meteorology, trademark databases, criminology, and an increasing number of other areas.

A more difficult type of search involves looking for a particular *object* within images, which we can term a *search-by-object* model. This involves a much more complete catalog of image contents and is a much more difficult goal. Generally, users will base their searches on *search by association* [7], meaning a first cut search followed by refinement based on similarity to some of the query results. For general images representative of a kind of desired picture, a *category search* returns one element of the requested set, such as one or several trademarks in a database of such logos. Alternatively, the query may be based on a very specific image, such as a particular piece of art—a *target search*. Lately, there are also efforts to retrieve three-dimensional shapes and objects [8,9].

Another axis to bear in mind in understanding the many existing search systems is whether the domain being searched is narrow, such as the database of trademarks, or wide, such as a set of commercial stock photos.

For any system, we are up against the fundamental nature of machine systems that aim to replace human endeavors. The main obstacles are neatly summarized in what the authors of the summary in [7] term the *sensory gap* and the *semantic gap*:

> The sensory gap is the gap between the object in the world and the information in a (computational) description derived from a recording of that scene.
>
> The semantic gap is the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation.

Image features record specifics about images, but the images themselves may elude description in such terms. And while we may certainly be able to describe images linguistically, the message in the image, the semantics, is difficult to capture for machine applications.

20.2 Synopsis of Early CBIR Systems

The following provides examples of some early CBIR systems. It is by no means a complete synopsis. Most of these engines are experimental, but all those included here are interesting in some way. A good summary appears in [7].

- QBIC

  Query by Image Content (*QBIC*), developed by Niblack and colleagues [10] at IBM's Almaden Research Center in San Jose, was arguably the most famous early search engine.

  One interesting feature in QBIC is the metric it uses for color histogram difference. The basic metric used for histogram difference is *histogram intersection*, basically an L_1-norm based measure. Instead of simple histogram intersection, the QBIC metric recognizes that colors that are *similar*, such as red and orange, should not have a zero intersection. Instead, a color-distance matrix $\boldsymbol{A}$ is introduced, with elements

  $$a_{ij} = (1 - d_{ij}/d_{\max}) \tag{20.1}$$

  Here, d_{ij} is defined as a three-dimensional color difference (using Euclidean distance, or any other likely distance—sum of absolute values, say).

  Then a histogram-difference D^2 is defined as follows [11]:

  $$D^2 = z^T \boldsymbol{A}\, z \tag{20.2}$$

  Vector z is a histogram-difference vector (for vectorized histograms). For example, the histogram-difference vectors z would be of length 256 if we compared two-dimensional chromaticity histograms of size 16×16.

- Chabot

  Chabot was an early system from UC-Berkeley that aimed to include 500,000 digitized multiresolution images. Chabot uses the relational database management system POSTGRES to access these images and associated textual data. The system stores both text and color histogram data. Instead of color percentages, a "mostly red" type of simple query is acceptable.

- Blobworld

  Blobworld [12] was also developed at UC-Berkeley. It attempts to capture the idea of objects by segmenting images into regions. To achieve a good segmentation, an *expectation maximization (EM)* algorithm derives the maximum likelihood for a good clustering in the feature space. Blobworld allows for both textual and content-based searching. The system has some degree of feedback, in that it displays the internal representation of the submitted image and the query results, so the user can better guide the algorithm.

- WebSEEk

  A team at Columbia University developed several search engines, of which WebSEEk was better known. It collects images (and text) from the web. The emphasis is on making a searchable catalogue with such topics as animals, architecture, art, astronomy, cats, and so on. Relevance feedback is provided in the form of thumbnail images and motion icons. For video, a good form of feedback is also inclusion of small, short video sequences as animated GIF files.

- Photobook and FourEyes

  Photobook [13] was one of the earlier CBIR systems developed by the MIT Media Laboratory. It searches for three different types of image content (faces, two-dimensional shapes, and texture images) using three mechanisms. For the first two types, it creates an eigenfunction space—a set of "eigenimages". Then new images are described in terms of their coordinates in this basis. For textures, an image is treated as a sum of three orthogonal components in a decomposition denoted as *Wold* features [14].

  With relevance feedback added, Photobook became FourEyes [15]. Not only does this system assign positive and negative weight changes for images, but given a similar query to one it has seen before, it can search faster than previously.

- Informedia

  The Informedia (and later Informedia-II) Digital Video Library project at Carnegie Mellon University centers on "video mining." It was funded by a consortium of government and corporate sponsors. It uniquely combines speech recognition, image understanding, and natural language processing technologies. Features include video and audio indexing, navigation, video summarization and visualization, and search and retrieval of the video media.

- UC Santa Barbara Search Engines

  The Alexandria Digital Library (ADL) was a seasoned image search engine devised at the University of California, Santa Barbara. The ADL is concerned with geographical data: "spatial data on the web." The user can interact with a map and zoom into a map, then retrieve images as a query result type that pertain to the selected map area. This approach mitigates the fact that terabytes, perhaps, of data need to be stored for LANDSAT satellite images, say. Instead, ADL uses a multiresolution approach that allows fast browsing by making use of image thumbnails. Multiresolution images means that it is possible to select a certain region within an image and zoom in on it.

- MARS

  MARS (Multimedia Analysis and Retrieval System) [16] was developed at the University of Illinois at Urbana-Champaign. The idea was to create a dynamic system of feature representations that could adapt to different applications and different users. Relevance feedback, with changes of weightings directed by the user, is the main tool used.

- Virage

  The Visual Information Retrieval (VIR) image search engine [17] operates on objects within images. Image indexing is performed after several preprocessing operations, such as smoothing and contrast enhancement. The details of the feature vector are proprietary; however, it is known that the computation of each feature is made by not one but several methods, with a composite feature vector composed of the concatenation of these individual computations.

20.3 C-BIRD: A Case Study

Let us consider the specifics of how image queries are carried out. To make the discussion concrete, we underpin our discussion by using the image database search engine devised by one of the authors of this text (see [18]). This system is called *Content-Based Image Retrieval from Digital libraries* (*C-BIRD*), an acronym devised from *content-based image retrieval*, or *CBIR*.

The C-BIRD image database contains approximately 5,000 images, many of them keyframes from videos. The database can be searched using a selection of tools: text annotations, color histograms, illumination-invariant color histograms, color density, color layout, texture layout, and model-based search.

Although the system was developed in early years, it still serves as a good example in illustrating the common techniques in CBIR based on image similarity. Moreover, it offers some unique features such as Search by Illumination Invariance, Feature Localization, and Search by Object Model.

Let's step through these options.

20.3.1 Color Histogram

In C-BIRD, features are precomputed for each image in the database. The most prevalent feature that is utilized in image database retrieval is the color histogram [6], a type of *global* image feature, that is, the image is not segmented; instead, every image region is treated equally.

A color histogram counts pixels with a given pixel value in red, green, and blue (RGB). For example, in pseudocode, for images with 8-bit values in each of R, G, B, we can fill a histogram that has 256^3 bins:

```
int hist[256][256][256];  // reset to 0
//image is an appropriate struct
//with byte fields red,green,blue

for i=0..(MAX_Y-1)
  for j=0..(MAX_X-1)
    {
     R = image[i][j].red;
     G = image[i][j].green;
     B = image[i][j].blue;
     hist[R][G][B]++;
    }
```

Usually, we do not use histograms with so many bins, in part because fewer bins tend to smooth out differences in similar but unequal images. We also wish to save storage space.

How image search proceeds is by matching the *feature vector* for the sample image, in this case the color histogram, with the feature vector for every—or at least many of—the images in the database.

C-BIRD calculates a color histogram for each target image as a preprocessing step, then references it in the database for each user query image. The histogram is defined coarsely, with bins quantized to 8 bits, with 3 bits for each of red and green and two for blue.

For example, Fig. 20.2 shows that the user has selected a particular image—one with red flowers. The result obtained, from a database of some 5,000 images, is a set of 60 matching images. Most CBIR systems return as the result set either the top few matches or the match set with a similarity measure above a fixed threshold value. C-BIRD uses the latter approach and thus may return zero search results.

How matching proceeds in practice depends on what measure of similarity we adopt. The standard measure used for color histograms is called the *histogram intersection*. First, a color histogram $\mathbf{H}_i$ is generated for each image i in the database. We like to think of the histogram as a three-index array, but of course the machine thinks of it as a long vector—hence the term "feature vector" for any of these types of measures.

The histogram is *normalized*, so that its sum (now a `double`) equals unity. This normalization step is interesting: it effectively removes the *size* of the image. The reason is that if the image has, say, resolution 640×480, then the histogram entries sum to 307, 200. But if the image is only one-quarter that size, or 320×240, the sum is only 76,800. Division by the total pixel count removes this difference. In fact, the normalized histograms can be viewed as *probability density functions* (*pdf*s) . The histogram is then stored in the database.

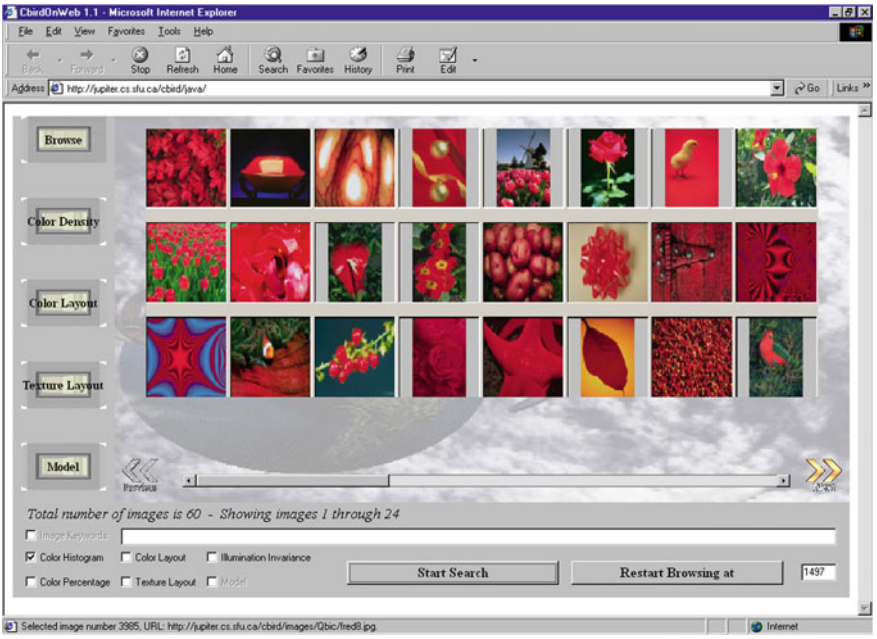

Fig. 20.2 Search by color histogram results. (Some thumbnail images are from the Corel gallery and are copyright Corel. All rights reserved)

Now suppose we select a "model" image—the new image to match against all possible targets in the database. Its histogram $\mathbf{H}_m$ is intersected with all database image histograms $\mathbf{H}_i$, according to the equation [6]

$$\text{intersection} = \sum_{j=1}^{n} \min(\mathbf{H}_i^j, \mathbf{H}_m^j) \tag{20.3}$$

where superscript j denotes histogram bin j, with each histogram having n bins. The closer the intersection value is to 1, the better the images match. This intersection value is fast to compute, but we should note that the intersection value is sensitive to color quantization.

20.3.2 Color Density and Color Layout

To specify the desired colors by their density, the user selects the percentage of the image having any particular color or set of colors, using a color picker and sliders. We can choose from either conjunction (ANDing) or disjunction (ORing) a simple color percentage specification. This is a coarse search method.

The user can also set up a scheme of how colors should appear in the image, in terms of coarse blocks of color. The user has a choice of four grid sizes: 1×1, 2×2, 4×4 and 8×8. Search is specified on one of the grid sizes, and the grid can be filled with any RGB color value—or no color value at all, to indicate that the cell

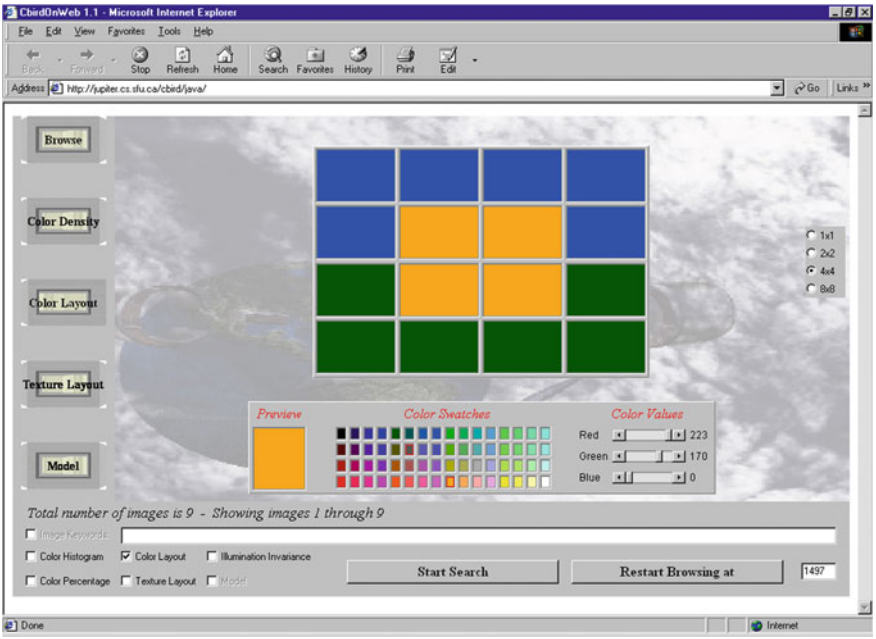

Fig. 20.3 Color layout grid

should not be considered. Every database image is partitioned into windows four times, once for each window size. A clustered color histogram is used inside each window, and the five most frequent colors are stored in the database. Each query cell position and size corresponds to the position and size of a window in the image. Figure 20.3 shows how this layout scheme is used.

20.3.3 Texture Layout

Similar to color layout search, this query allows the user to draw the desired texture distribution. Available textures are zero density texture, medium-density edges in four directions (0°, 45°, 90°, 135°) and combinations of them, and high-density texture in four directions and combinations of them. Texture matching is done by classifying textures according to directionality and density (or separation) and evaluating their correspondence to the texture distribution selected by the user in the texture block layout. Figure 20.4 shows how this layout scheme is used.

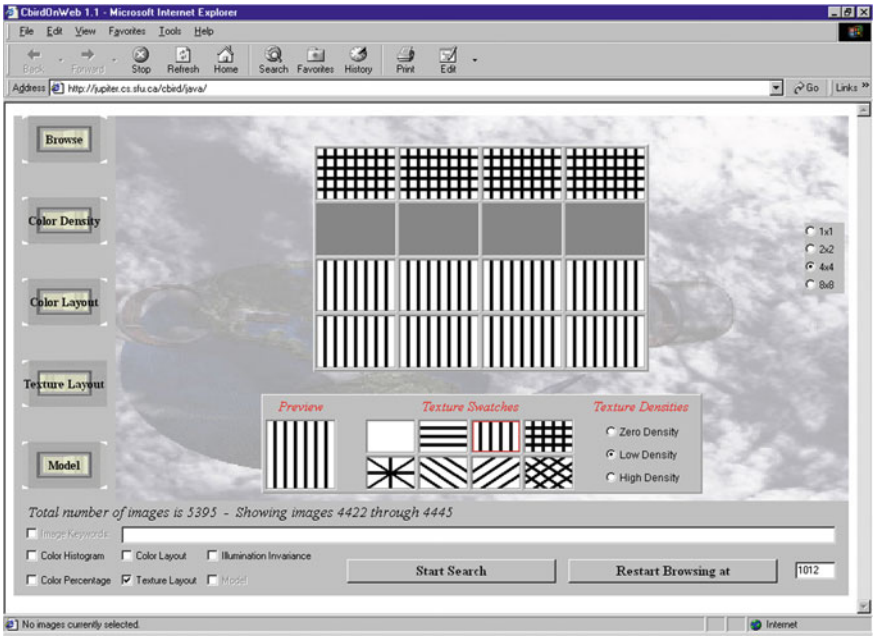

Fig. 20.4 Texture layout grid

20.3.4 Texture Analysis Details

It is worthwhile considering some of the details for a texture-based content analysis aimed at image search. These details give a taste of typical techniques systems employ to work in practical situations.

First, we create a texture histogram. A typical set of indices for comprehending texture is Tamura's [19]. Human perception studies show that "repetitiveness," "directionality," and "granularity" are the most relevant discriminatory factors in human textural perception [20]. Here, we use a two-dimensional texture histogram based on *directionality* ϕ and *edge separation* ξ, which is closely related to "repetitiveness." ϕ measures the edge orientations, and ξ measures the distances between parallel edges.

To extract an edge map, the image is first converted to luminance Y via $Y = 0.299R + 0.587G + 0.114B$. A *Sobel edge operator* [21] is applied to the Y-image by sliding the following 3×3 weighting matrices (*convolution masks*) over the image:

$$d_x : \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad d_y : \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array} \tag{20.4}$$

If we average around each pixel with these weights, we produce approximations to derivatives.

The edge magnitude D and the edge gradient ϕ are given by

$$D = \sqrt{d_x^2 + d_y^2}, \qquad \phi = \arctan \frac{d_y}{d_x} \tag{20.5}$$

Next, the edges are thinned by suppressing all but maximum values. If a pixel i with edge gradient ϕ_i and edge magnitude D_i has a neighbor pixel j along the direction of ϕ_i with gradient $\phi_j \approx \phi_i$ and edge magnitude $D_j > D_i$, then pixel i is suppressed to 0.

To make a binary edge image, we set all pixels with D greater than a threshold value to 1 and all others to 0.

For edge separation ξ, for each edge pixel i we measure the distance along its gradient ϕ_i to the nearest pixel j having $\phi_j \approx \phi_i$ within 15 degrees. If such a pixel j doesn't exist, the separation is considered infinite.

Having created edge directionality and edge separation maps, C-BIRD constructs a two-dimensional texture histogram of ξ versus ϕ. The initial histogram size is 193×180, where separation value $\xi = 193$ is reserved for a separation of infinity (as well as any $\xi > 192$). The histogram size is then reduced by three for each dimension to size 65×60, where joined entries are summed together.

The histogram is "smoothed" by replacing each pixel with a weighted sum of its neighbors and is then reduced again to size 7×8, with separation value 7 reserved for infinity. At this stage, the texture histogram is also normalized by dividing by the number of pixels in the image segment.

20.3.5 Search by Illumination Invariance

Illumination change can dramatically alter the color measured by camera RGB sensors, from *pink* under daylight to *purple* under fluorescent lighting, for example.

To deal with illumination change from the query image to different database images, each color-channel band of each image is first normalized, then compressed to a 36-vector [22]. Normalizing each of the R, G, and B bands of an image serves as a simple yet effective guard against color changes when the lighting color changes. A two-dimensional color histogram is then created using the *chromaticity*, which is the set of band ratios $\{R, G\}/(R + G + B)$. Chromaticity is similar to the chrominance in video, in that it captures color information only, not luminance (or brightness).

A 128×128–bin two-dimensional color histogram can then be treated as an image and compressed using a wavelet-based compression scheme [23]. To further reduce the number of vector components in a feature vector, the DCT coefficients for the smaller histogram are calculated and placed in zigzag order, then all but 36 components are dropped.

Matching is performed in the compressed domain by taking the Euclidean distance between two DCT-compressed 36-component feature vectors. (This illumination-invariant scheme and the object-model-based search described next are unique to C-BIRD.) Figure 20.5 shows the results of such a search.

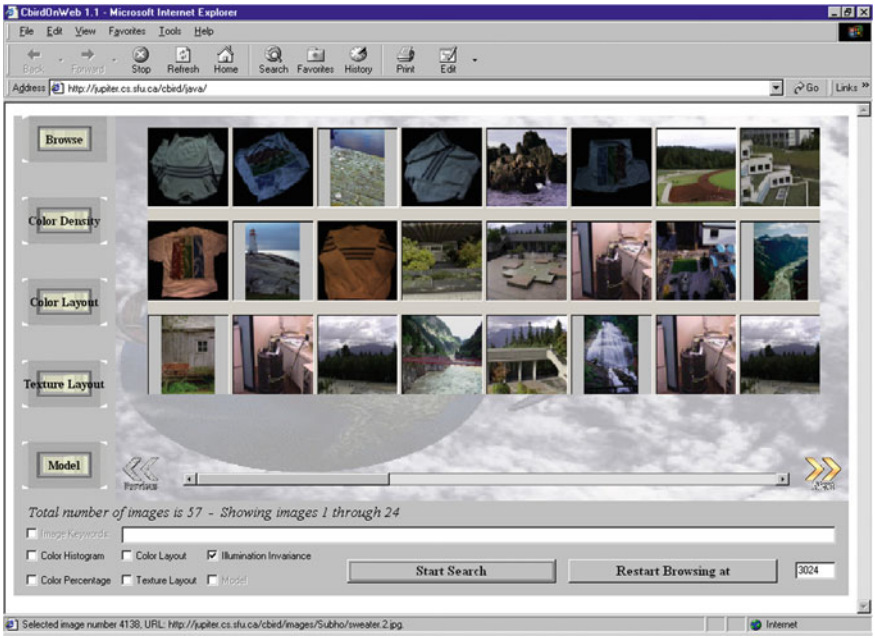

Fig. 20.5 Search with illumination invariance. (Some thumbnail images are from the Corel gallery and are copyright Corel. All rights reserved)

Several of the above types of searches can be done at once by checking multiple checkboxes. This returns a reduced list of images, since the list is the conjunction of all resulting separate return lists for each method.

20.3.6 Search by Object Model

The most important search type C-BIRD supports is the model-based object search. The user picks a sample image and interactively selects a region for object searching. Objects photographed under different scene conditions are still effectively matched. This search type proceeds by the user selecting a thumbnail and clicking the Model tab to enter Object Selection mode. An object is then interactively selected as a portion of the image; this constitutes an object query by example.

Figure 20.6 shows a sample object selection. An image region can be selected using primitive shapes such as a rectangle or ellipse, a magic wand tool that is basically a seed-based flooding algorithm, an active contour (a "snake"), or a brush tool, where the painted region is selected. All the selections can be combined with each other using Boolean operations such as union, intersection, or exclusion.

Once the object region is defined to a user's satisfaction, it can be dragged to the right pane, showing all current selections. Multiple regions can be dragged to the

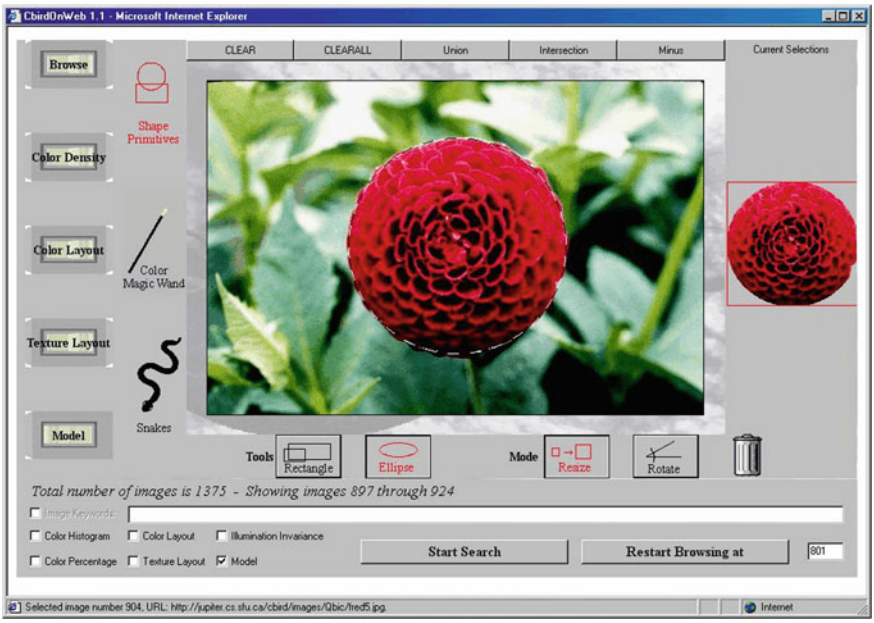

Fig. 20.6 C-BIRD interface, showing object selection using an ellipse primitive. (Image is from the Corel gallery and is copyright Corel. All rights reserved)

selection pane, but only the active object in the selection pane will be searched on. The user can also control parameters such as flooding thresholds, brush size, and active contour curvature.

Details of the underlying mechanisms of this Search by Object Model are set out in [23] and introduced below as an example of a working system. Figure 20.7 shows a block diagram for how the algorithm proceeds. First, the user-selected model image is processed and its features are localized (details are discussed in [18]). Color histogram intersection, based on the reduced chromaticity histogram described in the previous section is then applied as a first "screen." Further steps estimate the pose (scale, translation, and rotation) of the object inside a target image from the database. This is followed by verification by intersection of texture histograms and then a final check using an efficient version of a Generalized Hough Transform for shape verification.

A possible model image and one of the target images in the database might be as in Fig. 20.8, where the scene in (b) was illuminated with a dim fluorescent light. Figure 20.9 shows some search results for the pink book in C-BIRD.

While C-BIRD is an experimental system, it does provide a proof in principle that the difficult task of search by object model is possible.

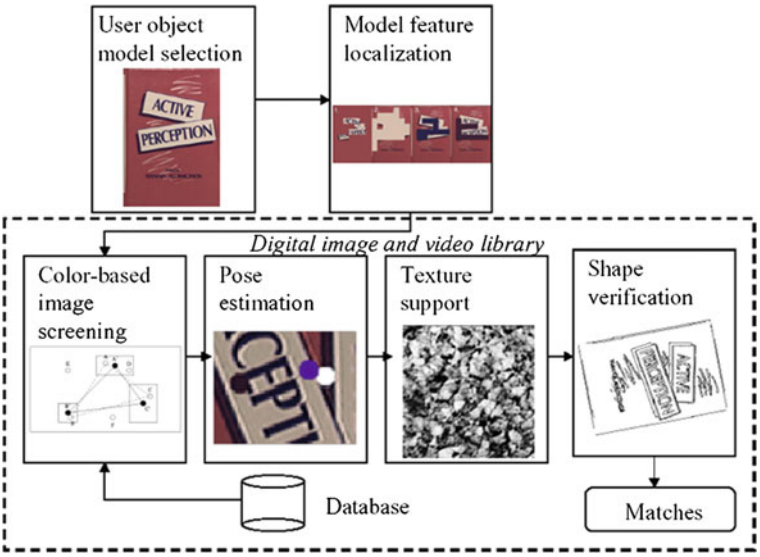

Fig. 20.7 Block diagram of object matching steps

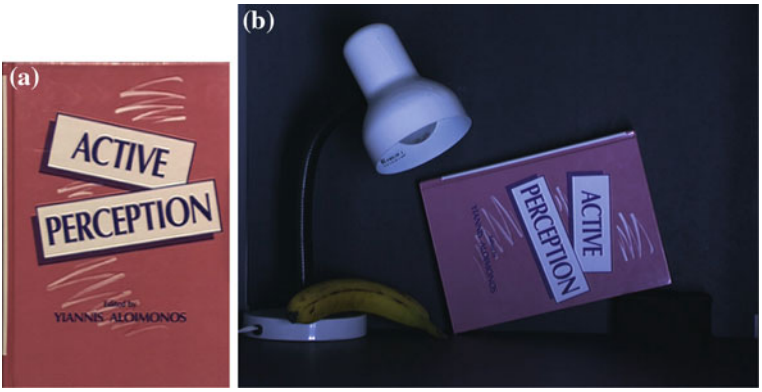

Fig. 20.8 Model and target images: **a** Sample model image; **b** Sample database image containing the model book. Active Perception (textbook cover courtesy Lawrence Erlbaum Associates, Inc.)

20.4 Quantifying Search Results

Generally speaking, some simple expression of the performance of image search engines is desirable. In information retrieval, *Precision* is the percentage of relevant documents retrieved compared to the number of all the documents retrieved, and *Recall* is the percentage of relevant documents retrieved out of all relevant documents. Recall and Precision are widely used for reporting retrieval performance for image retrieval systems as well. However, these measures are affected by the database size

Fig. 20.9 Search result for the pink book model with illumination change support: **a** search results using pose estimation only; **b** search results using pose estimation and texture support; **c** search results using GHT shape verification. (Some thumbnail images are from the Corel gallery and are copyright Corel. All rights reserved)

and the amount of similar information in the database. Also, they do not consider fuzzy matching or search result ordering.

In equation form, these quantities are defined as:

$$\begin{aligned} \text{Precision} &= \frac{\text{Relevant images returned}}{\text{All retrieved images}} \\ \text{Recall} &= \frac{\text{Relevant images returned}}{\text{All relevant images}} \end{aligned} \tag{20.6}$$

Alternatively, they may also be written as:

$$\begin{aligned} \text{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}} \\ \text{Recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}} \end{aligned} \tag{20.7}$$

where TP (True Positives) is the number of relevant images returned, FP (False Positives) is the number of irrelevant images returned, and FN (False Negatives) is the number of relevant images not returned.

In general, the more we relax thresholds and allow more images to be returned, the smaller the Precision, but the larger the Recall; and vice versa. Apparently, it is not quite meaningful to talk about either the Precision or Recall number by itself. Instead, they can be combined to provide a good measure, e.g., Precision when Recall is at 50 %, Recall when Precision is at 90 %, etc.

When multiple queries are involved, the numbers of the Precision and Recall values will again increase. To measure the overall performance of a CBIR system, the most common way is to summarize these values into a single value, i.e., the *Mean Average Precision (MAP)*. The Average Precision (AP) of a single query q is defined as:

$$AP(q) = \frac{1}{N_R} \sum_{n=1}^{N_R} \text{Precision}(n), \tag{20.8}$$

where $Precision(n)$ is the Precision value after the nth relevant image was retrieved, and N_R is the total number of relevant images. The MAP is the mean of Average Precisions over all query images:

$$MAP = \frac{1}{N_Q} \sum_{q \in Q} AP(q), \tag{20.9}$$

where Q is the query image set, and N_Q is its size. The MAP has the advantage of reflecting both Precision and Recall oriented aspects and is sensitive to the entire ranking [24].

In the above definitions of Precision and Recall, one thing is missing: the value of TN (True Negatives). In the context of CBIR, if we are searching for dogs and the image database contains 100 relevant dog images, the fact that whether the database contains another 100 non-dog images or a million non-dog images, i.e., $TN = 100$ or $TN = 1,000,000$, often matters, because a larger TN tends to yield more distractions (noise).

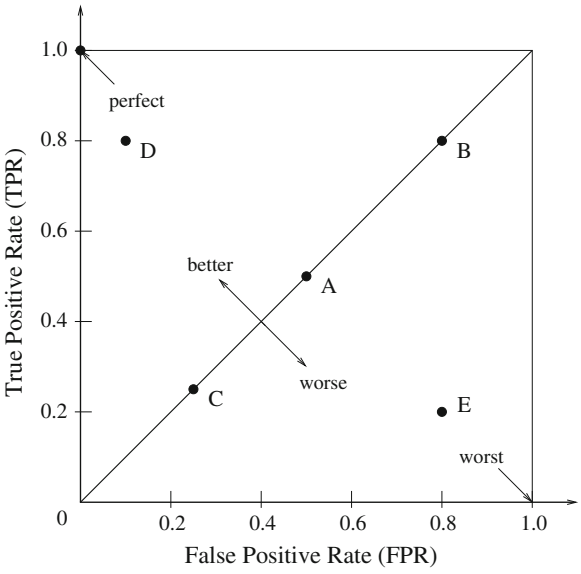

Fig. 20.10 The ROC space

Many other measures for the performance of CBIR systems have been devised, and among them a popular one is *Receiver Operating Characteristic (ROC)*. Given a database containing 100 relevant images (e.g., horses) and 1,000 other images (i.e., non-horses), if a CBIR system correctly retrieves 70 horse images from the 100 horse images, and incorrectly identified 300 "horse" images from the 1,000 other images, then the *True Positive Rate* $TPR = 70\,\%$ and the *False Positive Rate* $FPR = 30\,\%$. Obviously, $TPR = Recall$ as defined in Eq. 20.7, and $FPR = 1 - TNR$, where TNR (True Negative Rate) is the percentage (70 %) of the non-horses correctly identified. Figure 20.10 depicts the so-called ROC space, which plots TPR against FPR.

The 45-degree diagonal line in Fig. 20.10 indicates the performance of a random guess. In the above example, if we were to use a (fair) coin-flip to determine the outcome, then it will return a result of 50 true horse images and 500 false "horse" images from the two groups of images, i.e., $TPR = 50\,\%$ and $FPR = 50\,\%$, which is indicated by point A (0.5, 0.5) on the diagonal line in the figure. A biased coin that is weighted to produce more positive outcomes (e.g., heads) may yield performance that is indicated by point B (0.8, 0.8). Conversely, it may produce C (0.25, 0.25).

Clearly, we would like to have CBIR systems that will produce results better than a coin-flip. Namely, their performance should be above the diagonal line. Point D (0.1, 0.8) is an example of what would be very good performance. In general, we aim to be as close as possible to the ultimately perfect point (0, 1.0). Point E (0.8, 0.2) indicates poor performance, and obviously (1.0, 0) would be the worst.

If we measure the CBIR system at multiple TPR (or FPR) values, we will derive an *ROC curve*, which often provides a more comprehensive analysis of the system behavior. The perfect ROC curve would consist of two straight line segments, one

vertical from (0, 0) to (0, 1.0) and one horizontal from (0, 1.0) to (1.0, 1.0), which indicates that there are no false positives or false negatives whatsoever. A normal ROC curve will be in the area between this ideal performance and the 45-degree diagonal line.

ROC is a general statistical measure for various classifiers. It has its applications in many disciplines, e.g., psychology, physics, medicine, and increasingly in machine learning and data mining.

20.5 Key Technologies in Current CBIR Systems

The field of CBIR has witnessed rapid growth and progress in the new millennium. Unlike in the early years, users are no longer satisfied with query results returning scenes containing "blue skies," or red blobs that may or may not be flowers. They are more and more looking for objects, people, and often search for human activities as well. Lew et al. [25] and Datta et al. [26] provide comprehensive surveys of the CBIR field.

In this section, we will briefly describe some technologies and issues that are key to the success of current and future CBIR systems.

20.5.1 Robust Image Features and Their Representation

Many feature descriptors have been developed beyond the ones specified in MPEG-7. Here, we will only discuss *SIFT* and *Visual Words*.

SIFT (Scale Invariant Feature Transform)

In addition to global features such as color and texture histograms, local features characterized by SIFT (Scale Invariant Feature Transform) [27] have been developed because they are more suitable for searches emphasizing visual objects. SIFT has been shown to be robust against image noise; it also has a fair degree of invariance to translation, rotation, and scale change.

The process of extracting SIFT features from an image starts with building a multiresolution scale space, as shown in Fig. 20.11. At each scale, stacks of images are generated by applying Gaussian smoothing (filtering) operations. Each stack (the so-called octave) has $s+3$ images—so $s = 2$ for the example shown in the Fig. 20.11. The standard deviation of the Gaussian filters in the stack increases by a factor of $2^{1/s}$. In this example, they are σ, $\sqrt{2}\sigma$, 2σ, $2\sqrt{2}\sigma$, and 4σ in the first octave. The third image from the top of the stack is used as the bottom image of the next stack by reducing its image resolution by half. The same Gaussian filtering process will continue, and hence at the next octave the Gaussian filters have standard deviations

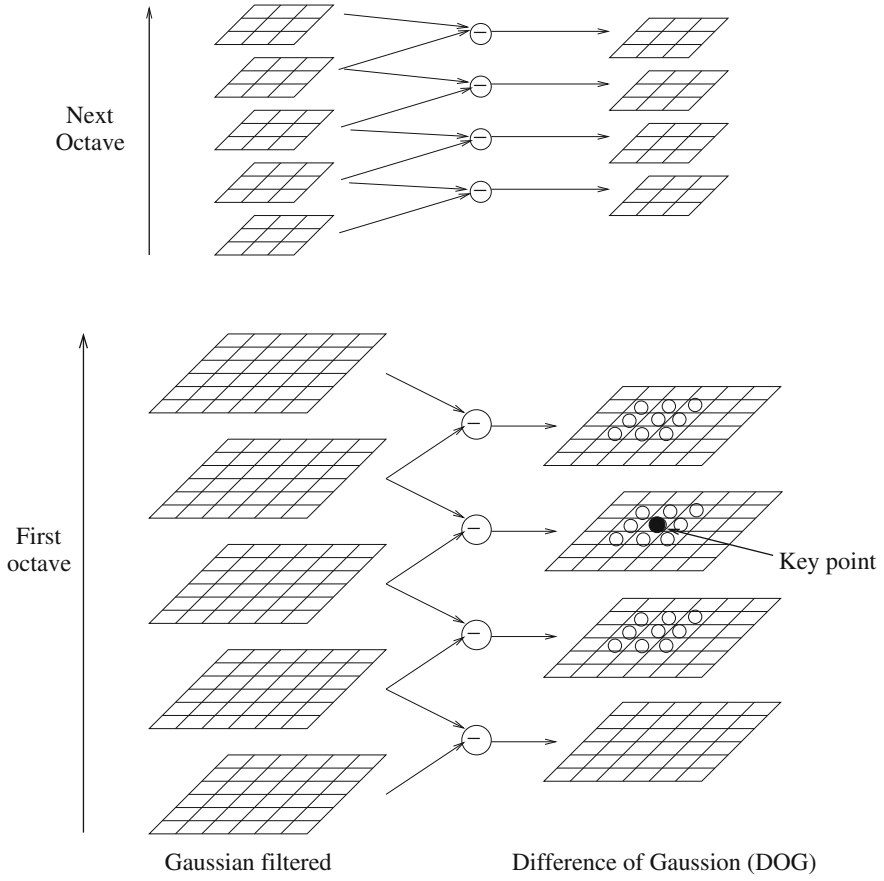

Fig. 20.11 The scale space and SIFT key point

$2\sigma, 2\sqrt{2}\sigma, 4\sigma, 4\sqrt{2}\sigma$, and 8σ. A simple operation of image subtraction, as indicated in the figure, generates Difference of Gaussian (DOG) images.

Now we are ready to talk about *Key points* for SIFT. In the DOG images, if a pixel's DOG value is the maximum or minimum when compared to its 26 neighbors in scale space (see Fig. 20.11), it is considered a possible Key point. Further screening steps are introduced to make sure that the Key points are more distinct, e.g., are at the corners rather than distributed over a long edge. Then, the histogram of the edge (gradient) directions near the Key point are analyzed to yield a dominant direction θ (the so-called *canonical orientation*) for the Key point.

The local patch (16×16 pixels) at the Key point is now examined to produce a 128-dimensional SIFT descriptor. The patch is divided into 4×4 subwindows. In each subwindow, a histogram of edge (gradient) directions is derived. The quantization is for every 45 degrees, so the histogram in each subwindow produces a vector of 8 dimensions. In total, we obtain a $4 \times 4 \times 8 = 128$ dimensional SIFT descriptor.

Here, all the edge directions are calculated relative to the canonical orientation θ. Hence, this facilitates rotation invariance.

Visual Words

The *Bag of Words (BoW)* concept was originally a technique for document classification and text retrieval. As the name suggests, a mixed bag of words can be extracted from a query sentence and be used for a query. The stems of the words are used, e.g., "talk" for "talk," "talking," "talked," etc. In this way, the details of the word (e.g., tense, singular or plural), the word order, and the grammar of the sentence are all ignored. The advantage is that this tends to be more robust against any variations of the text.

Analogously, bags of *Visual Words* can be extracted to represent image features. Fei-Fei and Perona [28] presented an earlier work in Computer Vision in which a bag of codewords is used to represent various texture features in images.

In CBIR, a common way of generating Visual Words is to use SIFT because of its good properties discussed above. This can be (a) object based or (b) video frame based. If the search is based on a given object model, then clusters of its SIFT features can be used as the Visual Words describing the object. Commonly, the vector quantization method as outlined in Chap. 8 can be used to turn these Visual Words into codewords in a codebook. If the search is aimed at finding similar frames from a video or movie, then all SIFT features in the query video frame can be used to generate the Visual Words. Alternatively, Sivic and Zisserman [29] divided the video frame into dozens (or hundreds) of regions, and each region will produce a mean SIFT descriptor $\bar{x_i}$ and be used as a Visual Word. The clustering and matching of a large number of SIFT descriptors is shown to be computationally challenging.

Visual Words are rich in encapsulating essential visual features. However, compared to words from text, Visual Words are even more ambiguous. In general, a small-sized codebook will have limited discriminative power, not good enough to handle CBIR for large image and video databases. On the other hand, a large codebook also has its own problems, because the same feature contaminated by noise can easily be quantized to different codewords.

20.5.2 Relevance Feedback

Relevance feedback, a well-known technique from Information Retrieval has been brought to bear in CBIR systems [16]. Briefly, the idea is to involve the user in a loop, whereby images retrieved are used in further rounds of convergence onto correct returns. The usual situation is that the user identifies images as good, bad, or don't care, and weighting systems are updated according to this user guidance.

The basic advantage of putting the user into the loop by using relevance feedback is that this way, the user need not provide a completely accurate initial query. Relevance feedback establishes a more accurate link between low-level features and high-level

concepts, somewhat closing the semantic gap. As a result, the retrieval performance of the CBIR system is improved.

20.5.3 Other Post-processing Techniques

Beside Relevance Feedback, other post-processing methods have been developed after the initial query results are in.

Spatial Verification

The quality of the query results can be verified and improved by *Spatial Verification*. Modern cameras often provide information about the location where pictures were taken, so it is trivial to check for consistency or relevance in certain query results if the user is only interested in images from certain locations, e.g., Paris or Rome.

In their paper, Philbin et al. [30] go well beyond simple checks on geometric locations. They aim at verifying that image regions from the query image and the retrieved images are from the same object or scene region. It is argued that unlike words in Information Retrieval (e.g., "animal," "flower"), *Visual Words* inherently contain much more spatial information. For example, it is known from the theory of image geometry that two views of a rigid object are related by epipolar geometry, two views of a planar patch are related by a homography, etc. They show that verifying mappings-based geometric transformations can indeed improve the quality of results by spatial re-ranking.

Zhou et al. [31] describe a spatial coding method that records relative spatial information (e.g., left or right, above or below) of each pair of image features. A rotating spatial map is proposed that is more efficient than a simple x- and y-map.

Query Expansion

Another approach is to move the query toward positively marked content. *Query Expansion* proposed by Chum et al. [32] is such a method. Query Expansion is again a well-known method in Information Retrieval in which some combination of high-ranked relevant documents can be presented as a new query to boost the performance of the retrieval. The combination could simply be averages (means) of the feature descriptors from the returned documents. The problem is that if one of the high-ranked document is a false-positive, then it will immediately harm the performance of the new query. Some robust statistical methods or even simply taking the median (instead of the mean) can help.

As mentioned above, the Visual Words used in CBIR often contain useful spatial information. Hence, high-ranked retrieved images can be verified before being used in forming a new query image. Chum et al. [32] use recursive average query expansion, which recursively generates new query images from all spatially verified returned images.

QA Paradigm

Question-answering (QA) tries to replace the large number of images or multimedia content in general, that is returned by a search query, by making use of media content, knowledge about the search domain, and also linguistic analysis to return hits based on users' natural-language questions. Since QA has traditionally been focused on text, bridging this technique over to multimedia content is referred to as MMQA [33]. Generally, MMQA attempts to combine traditional QA, based on textual metadata, with multimedia-oriented approaches if such are indeed more intuitive answers to users' queries.

20.5.4 Visual Concept Search

Search by concept is another major step in closing the semantic gap. Typically, after local features are turned into words, the words are converted to semantic concepts with the aid of machine learning algorithms.

In an interesting work presented by Wang et al. [34], image similarity is learned from 103 Flickr image groups on the Internet by adopting a Support Vector Machine (SVM) classifier, with a histogram intersection kernel. The image groups range from *objects* such as Aquarium, Boat, Car, and Penguin, *scenes* such as Sunset and Urban, to *concepts* such as Christmas and Smile. They show that such a system performs better than others that directly measure image similarity with simple low-level visual features such as color, texture, etc.

To assist in the development and testing of these systems, researchers and practitioners have developed many multimedia databases and benchmarks. The best known is perhaps the TREC Video Retrieval Evaluation (TRECVID) benchmark. TRECVID started in 2003, originally from the conference TREC (Text REtrieval Conference), co-sponsored by the National Institute of Standards (NIST) and the U.S. Department of Defense. Initially, TRECVID provided video data from professional sources such as broadcast news, TV programs, and surveillance systems, thus having limited styles and content: for example, the very typical head-shot of a news person, an overhead view of a variety of indoor scenes, etc. In recent years, the benchmarks have expanded in terms of test data and objectives. For example, TRECVID 2013 evaluated the following tasks:

- Semantic indexing
- Interactive surveillance event detection
- Instance search
- Multimedia event detection
- Multimedia event recounting

Myers et al. [35] reported good performance for Multimedia Event Detection in their project, entitled SESAME. To start, multiple event classifiers were developed based on the bags of words from single data types, e.g., low-level visual features, motion features, and audio features; and high-level visual (semantic) concepts such as Birthday-party, Making-a-sandwich, etc. Various fusion methods (Arithmetic Mean,

Geometric Mean, MAP (Mean Average Precision) Weighted, Weighted Mean Root, Conditional Mixture Model, Sparse Mixture Model, etc.) were then tested. It was shown that some simple fusion methods, e.g., Arithmetic Mean, deliver as good a performance (or better) than more complex ones.

20.5.5 The Role of Users in Interactive CBIR Systems

Beside mechanisms such as Relevance Feedback, it has been argued that users should play a more active role in CBIR systems. In the 2012 ICMR (International Conference on Multimedia Retrieval), the panelists (also the authors of [3]) again raised the question: Where is the User in Multimedia Retrieval? They pointed out the dominance of MAP, common in evaluations such as TRECVID, may have hindered the development of better and more useful multimedia retrieval systems. Although MAP has the merit of being objective and reproducible, it is unlikely that a single number will meet the needs of most users, who tend to have very different and dynamic tasks in mind.

One way to comprehend how people view images as similar is to study using user groups just what forms our basis of perception of image similarity [36]. The function used in this approach can be a type of "perceptual similarity measure" and is *learned* by finding the best set of features (color, texture, etc.) to capture "similarity" as defined via the groups of similar images identified.

Another way to understand users is to talk to them and carefully analyze their search patterns. In other words, in addition to *content-based*, we need to be *context-based*, because the user's interpretation of the content is often influenced (or even determined) by the context.

20.6 Querying on Videos

Video indexing can make use of *motion* as the salient feature of temporally changing images for various types of queries. We shall not examine video indexing in any detail here but refer the reader to the excellent surveys in [25] and [26].

In brief, since temporality is the main difference between a video and just a collection of images, dealing with the time component is first and foremost in comprehending the indexing, browsing, search, and retrieval of video content. A direction taken by the QBIC group [37] is a new focus on storyboard generation for automatic understanding of video—the so-called "inverse Hollywood" problem. In production of a video, the writer and director start with a visual depiction of how the story proceeds. In a video understanding situation, we would ideally wish to regenerate this storyboard as the starting place for comprehending the video.

The first place to start, then, would be dividing the video into *shots*, where each shot consists roughly of the video frames between the on and off clicks of the Record button. However, transitions are often placed between shots—fade-in, fade-out, dissolve,

wipe, and so on—so detection of shot boundaries may not be so simple as for abrupt changes.

Generally, since we are dealing with digital video, if at all possible we would like to avoid uncompressing MPEG files, say, to speed throughput. Therefore, researchers try to work on the compressed video. A simple approach to this idea is to uncompress just enough to recover the DC term, generating a thumbnail 64 times smaller than the original. Since we must consider P- and B-frames as well as I-frames, even generating a good approximation of the best DC image is itself a complicated problem.

Once DC frames are obtained from the whole video—or, even better, are obtained on the fly—many approaches have been used for finding shot boundaries. Features used have typically been color, texture, and motion vectors, although such concepts as trajectories traversed by objects have also been used [38].

Shots are grouped into *scenes*. A scene is a collection of shots that belong together and that are contiguous in time. Even higher-level semantics exist in so-called "film grammar" [39]. Semantic information such as the basic elements of the story may be obtainable. These are (at the coarsest level) the story's exposition, crisis, climax, and denouement.

Audio information is important for scene grouping. In a typical scene, the audio has no break within a scene, even though many shots may take place over the course of the scene. General timing information from movie creation may also be brought to bear.

Text may indeed be the most useful means of delineating shots and scenes, making use of closed-captioning information already available. However, relying on text is unreliable, since it may not exist, especially for legacy video.

Different schemes have been proposed for organizing and displaying storyboards reasonably succinctly. The most straightforward method is to display a two-dimensional array of *keyframes*. Just what constitutes a good keyframe has of course been subject to much debate. One approach might be to simply output one frame every few seconds. However, action has a tendency to occur between longer periods of inactive story. Therefore, some kind of clustering method is usually used, to represent a longer period of time that is more or less the same within the temporal period belonging to a single keyframe.

Some researchers have suggested using a graph-based method. Suppose we have a video of two talking heads, the interviewer and the interviewee. A sensible representation might be a digraph with directed arcs taking us from one person to the other, then back again. In this way, we can encapsulate much information about the video's structure and also have available the arsenal of tools developed for graph pruning and management.

Other "proxies" have also been developed for representing shots and scenes. A grouping of *sets* of keyframes may be more representative than just a sequence of keyframes, as may keyframes of variable sizes. Annotation by text or voice, of each set of keyframes in a "skimmed" video, may be required for sensible understanding of the underlying video.

Fig. 20.12 Digital video and associated keyframes, `beach` video: **a** frames from a digital video; **b** keyframes selected

A *mosaic* of several frames may be useful, wherein frames are combined into larger ones by matching features over a set of frames. This results in set of larger keyframes that are perhaps more representational of the video.

An even more radical approach to video representation involves selecting (or creating) a *single* frame that best represents the entire movie! This could be based on making sure that people are in the frame, that there is action, and so on. In [40], Dufaux proposes an algorithm that selects shots and keyframes based on measures of motion-activity (via frame difference), spatial activity (via entropy of the pixel value distribution), skin-color pixels, and face detection.

By taking into account skin color and faces, the algorithm increases the likelihood of the selected keyframe including people and portraits, such as close-ups of movie actors, thereby producing interesting keyframes. Skin color is learned using labeled image samples. Face detection is performed using a neural net.

Figure 20.12a shows a selection of frames from a video of beach activity (see [41]). Here, the keyframes in Fig. 20.12b are selected based mainly on color information (but being careful with respect to the changes incurred by changing illumination conditions when videos are shot).

A more difficult problem arises when changes between shots are gradual and when colors are rather similar overall, as in Fig. 20.13a. The keyframes in Fig. 20.13b are sufficient to show the development of the whole video sequence.

Other approaches attempt to deal with more profoundly human aspects of video, as opposed to lower-level visual or audio features. Much effort has gone into applying

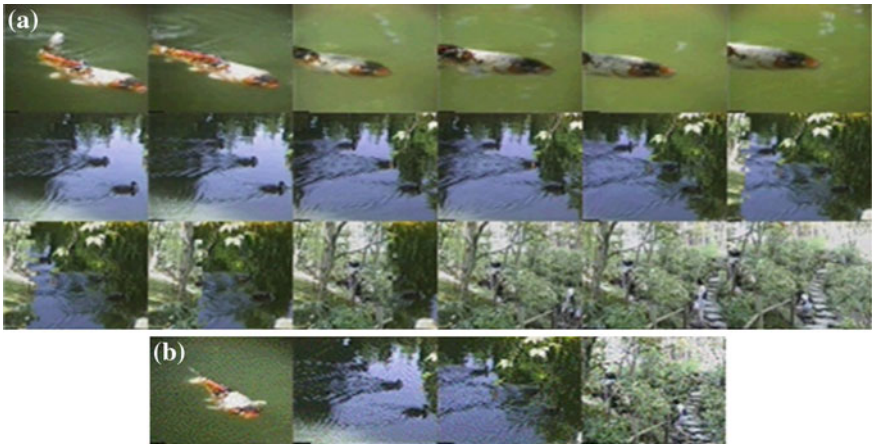

Fig. 20.13 Garden video: **a** frames from a digital video; **b** keyframes selected

data mining or knowledge-base techniques to *classifying* videos into such categories as sports, news, and so on, and then subcategories such as football and basketball.

20.7 Querying on Videos Based on Human Activity

Thousands of hours of video are being captured every day by CCTV cameras, web cameras, broadcast cameras, etc. However, most of the activities of interest (e.g., a soccer player scores a goal) only occur in a relatively small region along the spatial and temporal extent of the video. In this scenario, effective retrieval of a small spatial/temporal segment of the video containing a particular activity from a large collection of videos is very important.

Lan et al.'s work [42] on activity retrieval is directly inspired by the application of searching for activities of interest from broadcast sports videos. For example, consider the scene shown in Fig. 20.14. In terms of human activities, there is a variety of questions one can ask. Who is the attacker? What are the players in the bottom right corner doing? How many people are running? Which players are defending (marking) members of the opposing team? What is the overall game situation? Note that potential queries often involve social roles such as "defender," "attacker," or "man-marking." Lan et al. [42] present a model toward answering queries such as these.

The representation of human activity is a challenging, open problem. Much of the work focuses on recognition of low-level single-person actions (e.g., [43]). Lan et al. rely on low-level features and representations used by these methods to predict the actions of individuals, but build higher-level models upon them. It is arguable that multiple levels of detail and types of labels are required depending on the problem

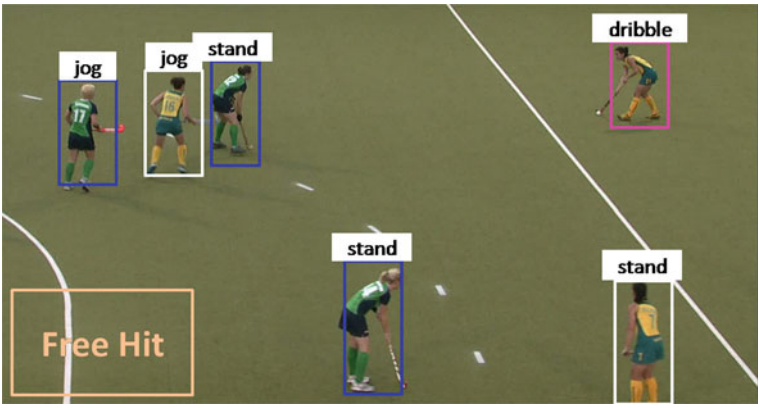

Fig. 20.14 An example of human activity in realistic scenes. Beyond the general scene-level activity description (e.g. Free Hit), we can explain the scene at multiple levels of detail such as low-level actions (e.g., standing and jogging) and mid-level social roles (e.g., attacker and first defenders). The social roles are denoted by different colors. In this example, we use *magenta*, *blue* and *white* to represent attacker, man-marking and players in the same team as the attacker respectively

focus. A model is presented that can be used to capture a variety of levels of detail in a unified framework. In addition to modeling of low-level actions (e.g., running or standing) and high-level events ("attack play", "penalty corner"), we model social roles. Social roles take into account inter-related people and are a complementary representation to the low-level actions typically used in the activity recognition literature. For example, a player engaged in the social role of "man-marking" is likely to have an opponent nearby. Further, the notions of low-level actions, social roles, and high-level events naturally require a contextual representation—the actions and social roles of all the people in a scene are interdependent, and related to the high-level event taking place. The model captures these relationships, and allows flexible inference of the social roles and their dependencies in a given scene.

20.7.1 Modeling Human Activity Structures

Here we introduce the model in [42]. To illustrate, we describe an instantiation applicable to modeling field hockey videos.

We first describe the labeling. Assume an image has been pre-processed, so the location of the goal and persons in the image have been found. We separate the players into two teams according to their color histograms. Each person is associated with two labels: action and social role. Let $h_i \in \mathcal{H}$ and $r_i \in \mathcal{R}$ be the action and social roles of the person i respectively, where $\mathcal{H}$ and $\mathcal{R}$ are the sets of all possible action and social role labels, respectively. Each video frame is associated with an event label $y \in \mathcal{Y}$, where $\mathcal{Y}$ is the set of all possible event labels.

The model is hierarchical, and includes various levels of detail: low-level actions, mid-level social roles, and high-level events. The relationships and interactions

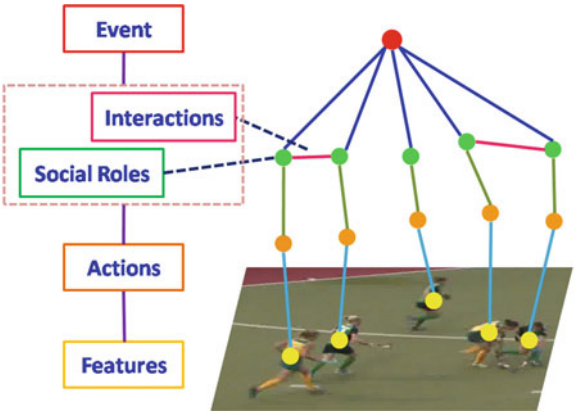

Fig. 20.15 Graphical illustration of the model. Different types of potentials are denoted by *lines* with different colors. Details of the potential functions are contained in Eq. 20.10

between these are included in the model. We define the score of interpreting a video frame I with the hierarchical event representation as:

$$F_w(\mathbf{x}, y, \mathbf{r}, \mathbf{h}, I) = w^\top \Phi(\mathbf{x}, y, \mathbf{r}, \mathbf{h}, I) = \sum_j w_1^\top \phi_1(x_j, h_j)$$
$$+ \sum_j w_2^\top \phi_2(h_j, r_j) + \sum_{j,k} w_3^\top \phi_3(y, r_j, r_k) \quad (20.10)$$

Now we go over the model formulation using the graphical illustration shown in Fig. 20.15. The first term denotes standard linear models trained to predict the action labels of the persons in the scene (indicated by blue lines in Fig. 20.15). The second term captures the dependence between action labels and social roles (green lines in Fig. 20.15). The third term explores contextual information by modeling interactions between people in terms of their social roles under an event y (magenta and dark blue lines in Fig. 20.15). Social roles naturally capture important interactions, e.g., *first defenders* tend to appear in the neighborhood of an *attacker*, *man-marking* happens when there is a player from the opposing team.

The model parameters w are learned in a structured SVM framework [44]; for details, the reader is referred to [42].

Inference

Given a test video there are a variety of queries one might wish to answer. Using our hierarchical model, one can formulate queries about any individual variable at any level of detail. For instance, one can query on the overall event label for the scene, or the social role label of a particular person. Figure 20.16 shows an overview of the testing phase.

For a given video and query variable q, the inference problem is to find the best hierarchical event representation that maximizes the scoring function $F_w(\mathbf{x}, y, \mathbf{h}, \mathbf{r}, I)$ while fixing the value of q to its possible values. For example, if q is the action of one

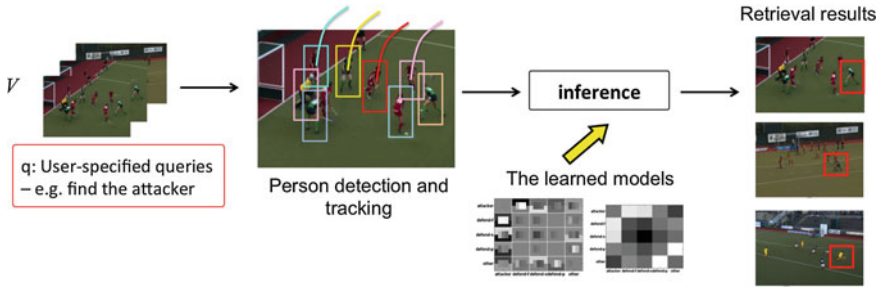

Fig. 20.16 An overview of the testing phase. Given a new video and a query, we first run the pretrained person detector and tracker to extract the tracklet of each player. Then the tracklet features are fed into the inference framework. Finally, the retrieval results are obtained based on the inference scores computed using Eq. 20.10

person (one of the h_i), we would compute the maximum value of the scoring function F_w when fixing q to each possible action. We define the optimization problem as follows:

$$\max_{y,\mathbf{h},\mathbf{r}\backslash q} F_w(\mathbf{x}, y, \mathbf{h}, \mathbf{r}, I) = \max_{y,\mathbf{h},\mathbf{r}\backslash q} w^\top \Phi(\mathbf{x}, y, \mathbf{h}, \mathbf{r}, I) \tag{20.11}$$

The score is used to represent the relevance of the instance (e.g., video frame) to the query. The goal of an activity retrieval system is to rank the data according to the relevance scores and return the top-ranked instances.

20.7.2 Experimental Results

The challenging *Broadcast Field Hockey Dataset* (developed in [42]) that consists of sporting activities captured from broadcast cameras is used to demonstrate the efficacy of the model. The model can carry out different inferences based on a user's queries. The method is directly applicable to multilevel human activity recognition. The goal is to predict events, social roles or actions of each person. In this case, the query doesn't specify a particular event or social roles, but consists of more general questions, such as what is the overall game situation, or what are the social roles of each player. Figure 20.17 shows the visualizations of the predicted events and social roles.

20.8 Quality-Aware Mobile Visual Search

With the increasing popularity of mobile phones and tablets, *Mobile Visual Search* has attracted growing interest in the field of content-based image retrieval (CBIR). In this section, we present a novel framework for quality-aware mobile CBIR by Peng et al. [45]. On the mobile-client side, a query image is compressed to a certain quality level to accommodate the network conditions and then uploaded onto a server with its

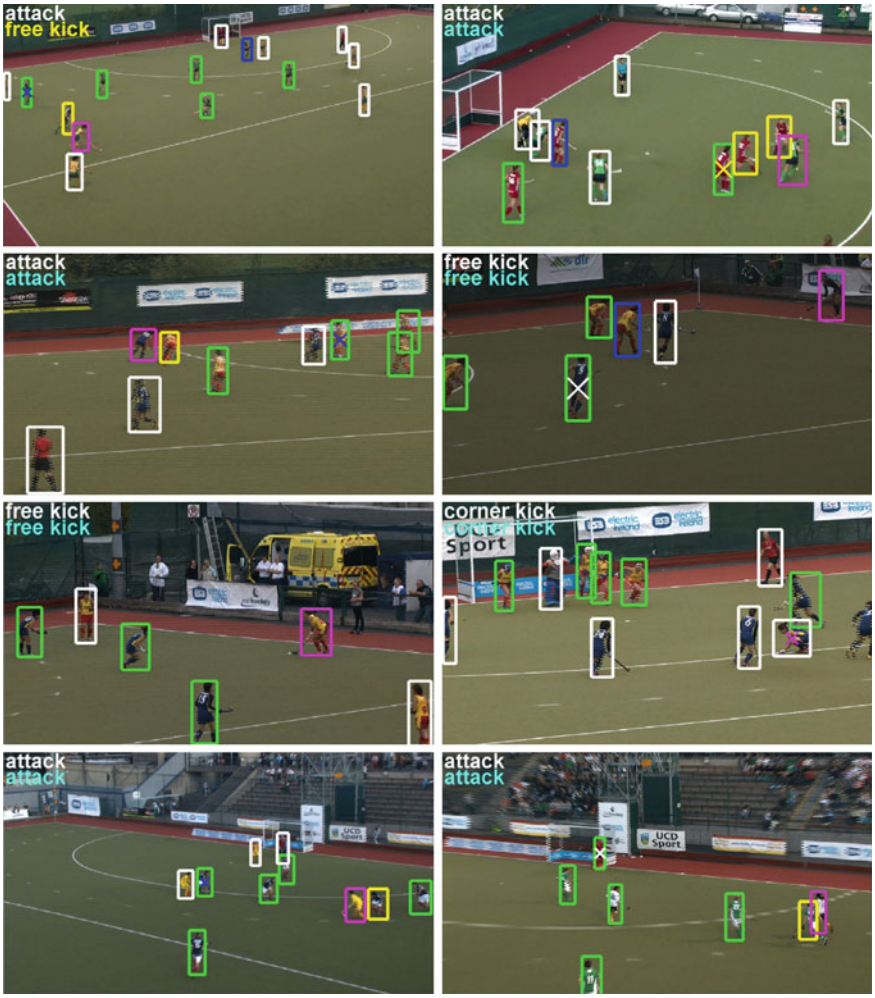

Fig. 20.17 Visualization of results on broadcast field hockey dataset. The ground truth event (*white*) and the predicted event are shown in the *left* corner of each image. Correct predictions are visualized in *blue*, otherwise *yellow*. Each bounding box is represented by a color, which denotes the predicted social roles. We use *magenta*, *yellow*, *green*, *blue* and *white* to represent the social roles attacker, first defenders, defenders defend against space, defenders defend against person and other, respectively. The cross sign in the *middle* of a bounding box indicates incorrect predictions, and the ground truth social roles are indicated by the color of the cross sign

quality level transferred as side information. On the server side, a set of features are extracted from the query image and then compared against the features of the images in the database. As the efficacy of different features changes over query quality, we leverage the side information about the query quality to select a quality-specific

similarity function that is learned offline using a *Support Vector Machine (SVM)* method.

Mobile Visual Search enables people to look for visually similar products or find information about movies or CDs online by initiating a search request from a camera phone. Depending on which part of the visual search is performed on the mobile devices, there are several possible client-server architectures [46]:

1. A query image is transmitted to the server, and then feature extraction and retrieval are done on the server.
2. The mobile client extracts some features from the query image and uploads only the features to the server. The retrieval is performed on the server.
3. The mobile client maintains a cache of the image database. The retrieval is done locally on the client. Only if no match is found does the client sends a query to the server.

In each case, the system performance is constrained on the bandwidth, computation, memory, and power of mobile devices. Recently, there has been work focusing on designing compact descriptors for visual search. One representative work is the Compressed Histogram of Gradients (CHoG) descriptor proposed by Chandrasekhar et al. [47], which is shown to be highly discriminative at a low bitrate. Besides, there has been exploratory work by the MPEG committee toward defining a standard for visual search applications since 2011. This standardization initiative is referred to as "Compact Descriptors for Visual Search (CDVS)" [48]. It is evident that a low-bitrate descriptor can leads to shorter transmission latency, smaller memory overload, and potentially faster matching. Therefore, all the three aforementioned client-server architectures of mobile visual search can benefit from the advancement of compact-descriptor technology. Besides the great amount of research effort devoted to the design of visual descriptors, fusion methods for visual search has also attracted lots of attention in the CBIR community. Given that a descriptor is a set of characteristics of an image, such as color, shape, and texture, fusion techniques are shown to be effective in reducing the semantic gap of image retrieval based on feature similarity.

Here, we outline a framework for mobile visual search using a client-server architecture [45]. Specifically, a query image is compressed to a certain quality level on the mobile client and then uploaded to the server with its quality level transmitted as side information at the same time. A query quality-dependent retrieval algorithm based on fusion of multiple features is then performed on the server. The motivations behind proposing such a framework are as follows:

- Although the computational capacity of mobile devices has become more and more powerful, there are several advantages to performing descriptor extraction on the server. Certainly, it eliminates the waiting time caused by computing descriptors on a mobile device of limited computing resource. More importantly, given the abundant computing resources on the server, it greatly relaxes the stringent constraint on the complexity and memory usage of the descriptor(s), which make a fusion method computationally feasible in this framework.
- As bandwidth is also an important concern for visual search on wireless networks, the framework allows the client to compress a query image at a certain bitrate to accommodate the network condition.

- Since a specific descriptor is not equally important for images of different quality levels, the side information about the query quality could be leveraged to enhance the retrieval performance of the fusion method.

20.8.1 Related Work

Chatzichristofis et al. investigated the behavior of some compact composite descriptors in early fusion (a new descriptor is constructed based on multiple existing descriptors), late fusion (the retrieved results with different descriptors are fused to form a final result list), and distributed image retrieval [49]. The experimental results show that the fusion methods are able to present better results than individual descriptors. Singh and Pooja present a fusion image retrieval method using the global angular radial transform and local polar Hough transform based features [50]. Chen et al. propose an image retrieval method based on similarity score fusion of color and texture features using a genetic algorithm [51]. All of these methods perform query independent fusion. Since a special feature has different importance in reflecting the content of different images, Huang et al. propose a query-dependent feature fusion method for medical image retrieval based on a one-class SVM method [52]. Zhang et al. also propose a graph-based query specific fusion approach where multiple retrieval sets are merged and reranked by conducting a link analysis on a fused graph [53]. It is worth mentioning that none of these fusion methods takes into consideration the quality of the query images.

There are several methods that explicitly deal with distorted query images. For example, Liao and Chen propose a complementary retrieval method based on fusion of multiple features to resist various types of processing, such as geometric transformation, compression, changing of illumination, and noise corruption. In their method, the distortion types of the query image are assumed to be unknown. A complementary analysis is proposed to determine the distortion category for each query image and the feature resistant to the predicted category is used to retrieve the desired original image [54]. Unlike the work outlined here, that focuses on the quality levels of the query images, their method focuses on the distortion type and is designed for copy detection rather than general visual search. Besides, Singh et al. [55] propose a method that combines color and shape features to retrieval images with incomplete or distorted queries. In their studies, query images are categorized into two classes: "complete" and "incomplete." The same fusion method is used for all incomplete queries.

20.8.2 Quality-Aware Method

We outline a query quality-dependent fusion approach for mobile visual search. Specifically, it is based on the fusion of five common image features [56]:

- Tiny Images: This is the most trivial descriptor that compares images directly in color space after reducing the image dimensions drastically.

- Color Histograms: The histograms have 16 bins for each component of RGB color space, yielding a total of 48 dimensions.
- GIST: This descriptor computes the output of a bank of 24 Gabor-like filters tuned to eight orientations at four scales. The squared output of each filter is then averaged on a 4×4 grid.
- Texton Histograms: For each image build a 512-dimensional histogram using a 512-entry universal texton dictionary.
- SSIM [57]: The self-similarity descriptors are quantized into 300 visual words by k-means. Unlike the descriptors mentioned before, SSIM provide a complementary measure of scene layout that is somewhat appearance invariant.

A query quality-dependent method for fusion image retrieval based on the five descriptors is then employed, as follows.

1. Categorize the query images into different quality levels based on the side information. [1]
2. For each "query image-retrieved image" pair, compute a 5-dimensional similarity-score vector $\vec{x}$ based on the five descriptors. The C-SVM formulation is adopted to learn the weights $\vec{w}_k$ to map $\vec{x}$ into a final score $s_f = \vec{w}_k \cdot \vec{x}$. Specifically, a set of positive (relevant) image pairs P is selected, and also a set of negative (irrelevant) image pairs N. For each image pair $p_i \in P \cup N$, compute a vector $\vec{x}_i$, and assign to $\vec{x}_i$ a label y_i ("1" for $p_i \in P$, and "-1" for $p_i \in N$). The weights $\vec{w}$ can be learned by solving the following optimization problem:

$$\begin{aligned} \min_{\vec{w},b,\xi} \quad & \frac{1}{2}\vec{w}^T\vec{w} + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\vec{w}^T\phi(\vec{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0. \end{aligned} \tag{20.12}$$

   This optimization is run for each quality level k to learn $\vec{w}_k$ using a query dataset of the corresponding quality.
3. At the test stage, the learned weight vector $\vec{w}_k$ is used to compute the final similarity score for a "query image-retrieved image" pair, with k being the quality level of the query image. The retrieved images are returned in decreasing order of the final similarity scores.

20.8.3 Experimental Results

Datasets

Consider the Wang image database [58] that contains 1,000 images of 10 classes (100 for each class). Images that belong to the same class are considered to be relevant. Ten copies of the Wang database are constructed at different quality levels. Specifically, images are compressed using JPEG compression with quality factor

[1] For example, the JPEG standard uses a scalar to adjust a set of well-defined quantization tables.

Table 20.1 File size ranges of the images (386 × 256) from the Wang database compressed at different quality levels

Quality factor	100	75	50	30	20	15	10	8	5	3
Size range (Kb)	7–56	6–38	4–24	4–18	3–14	3–12	3–9	3–8	3–6	3–5

$k \in \{100, 75, 50, 30, 20, 15, 10, 8, 5, 3\}$. Let D_k be the database containing images at quality level k. D_{100} corresponds to the original Wang database, and D_3 contains images of the lowest quality. The file size ranges of the images after compression are listed in Table 20.1.

In the experiments, the query images can be from any quality level, whereas the images to be retrieved are always from D_{100}. To learn the weights $\vec{w}_k$ for quality level k, we selected 500 query images (50 for each class) from D_k, and pair them with the images in D_{100} for training. Let $(q_{k,i}, r_j)$ be such a pair, where $q_{k,i} \in D_k$ and $r_j \in D_{100}$. Let $c(\cdot)$ denote the class of an image. A pair $(q_{k,i}, r_j)$ is labeled positive or "1", if $c(q_{k,i}) = c(r_j)$. Otherwise, it is labeled negative or "−1." As there are nine negative classes and only one positive class for each query image, we randomly select one-ninth of the negative pairs in order to balance the positive and negative samples during training. The remaining 500 images from each D_k that have not been selected for training are used for test. In the implementation, the dictionaries used to compute the descriptors are built using images from the SUN database [56]. We measure the similarity of two descriptors based on the histogram intersection distance and use a linear kernel for the SVM-based quality-aware fusion method.

Retrieval Metric

Let $s_f(q, r_n)$ be the final similarity score between a query image q and an image r_n in the database. The database images r_n are then sorted according to the similarity scores such that $s_f(q, r_n) \geq s_f(q, r_{n+1})$.

As the Precision and Recall values vary with the query images and the numbers of returned images, instead use is made of the Mean Average Precision (MAP), as defined in Eq. 20.9.

Results

The MAP results on the Wang database of 10 different quality levels are shown in Table 20.2 and also in Fig. 20.18. We can see that the performances of the Color Histograms, Tiny Images, and GIST do not change drastically as the image quality drops. On the contrary, the Texton Histograms and SSIM descriptors achieve the best performance near the higher end of the quality range (level 100, 75, and 50)

Table 20.2 MAP results of different quality levels

Query quality	100	75	50	30	20	15	10	8	5	3
Color histograms	0.3874	0.3874	0.3831	0.3835	0.3800	0.3775	0.3746	0.3752	0.3672	0.3626
Tiny images	0.3054	0.3052	0.3052	0.3047	0.3046	0.3043	0.3044	0.3038	0.3021	0.3021
GIST	0.3292	0.3276	0.3279	0.3264	0.3247	0.3228	0.3174	0.3125	0.2990	0.2824
Texton histograms	0.4260	0.4171	0.4256	0.4206	0.4135	0.4056	0.3843	0.3597	0.3119	0.2694
SSIM	0.4804	0.4805	0.4154	0.3656	0.3463	0.3202	0.2949	0.2777	0.2456	0.2178
Quality-aware fusion	**0.5730**	**0.5726**	**0.5462**	**0.5330**	**0.5259**	**0.5168**	**0.5081**	**0.4975**	**0.4808**	**0.4701**

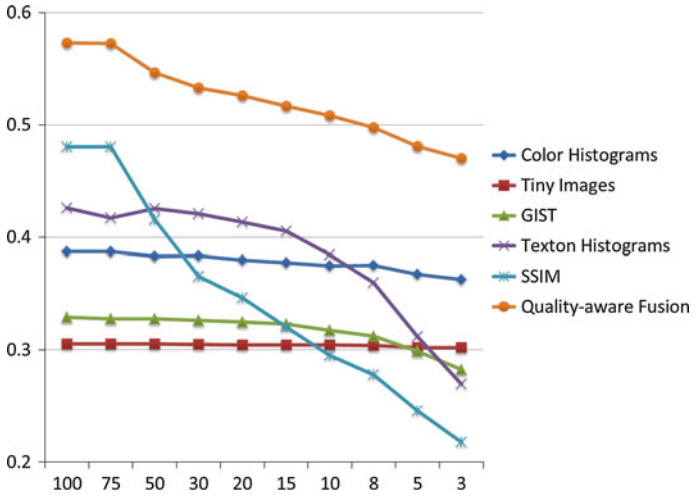

Fig. 20.18 Map results of different quality levels

and perform poorly near the lower-end of the quality range (level 5 and 3). On these quality levels, the quality-aware fusion method is able to achieve better retrieval performance than a mean-based fusion method, which indicates that the quality-dependent weights learned by the SVM method are better than uniform weights in fusing the different descriptors. For the medium quality levels, mean-based fusion is slightly better than the quality-aware fusion, but there is no significant performance gap between them. This indicates that the uniform weights are closer to the optimal weights than the learned quality-dependent weights for the medium quality levels. This is not surprising considering the relatively smaller performance differences among the five descriptors. Nevertheless, there is clearly much room left for improving such a quality-aware fusion method by exploring better ways to estimate the optimal weights for combination.

It can also be observed that both fusion methods achieve significantly better MAPs than any individual descriptors on each quality level. Noticeably, the quality-aware

fusion algorithm for the lowest-quality query images achieves comparable performance with the best-performing individual descriptor SSIM with the highest-quality query images. This strongly supports the advantages of fusing multiple descriptors for visual search.

The above discussion presented a quality-aware framework for Mobile Visual Search based on a query quality-dependent fusion method. The experimental results demonstrate the potential of taking into consideration the quality of query images to improve the performance of fusion image retrieval.

Current progressive coding techniques allow a mobile client to upload a bitstream that successively refines the reconstructed query image. In this case, the server can perform image retrieval using a query image of reduced quality, and then update the retrieved results as the query quality gets better and better.

20.9 Exercises

1. Devise a text-annotation taxonomy (categorization) for image descriptions, starting your classification using the set of Yahoo! categories, say.
2. Examine several web site image captions. How useful would you say the textual data is as a cue for identifying image contents? (Typically, search systems use *word stemming*, for eliminating tense, case, and number from words—the word *stemming* becomes the word *stem*.)
3. Suppose a color histogram is defined coarsely, with bins quantized to 8 bits, with 3 bits for each red and green and two for blue. Set up an appropriate structure for such a histogram, and fill it from some image you read. Template Visual C++ code for reading an image is on the text web site, as `sampleCcode.zip` under "Sample Code."
4. Try creating a texture histogram as described in Sect. 20.3.4. You could try a small image and follow the steps given there, using MATLAB, say, for ease of visualization.
5. Describe how you may find an image containing some two-dimensional "brick pattern" in an image database, assuming the color of the "brick" is yellow and the color of the "gaps" is blue. (Make sure you discuss the limitations of your method and possible improvements.)
   (a) Use color only.
   (b) Use edge-based texture measures only.
   (c) Use color, texture, and shape.
6. The main difference between a static image and video is the availability of motion in the latter. One important part of CBR from video is motion estimation (e.g., the direction and speed of any movement). Describe how you could estimate the movement of an object in a video clip, say a car, if MPEG (instead of uncompressed) video is used.

7. Color is three-dimensional, as Newton pointed out. In general, we have made use of several different color spaces, all of which have some kind of brightness axis, plus two intrinsic-color axes.

   Let's use a *chromaticity* two-dimensional space, as defined in Eq. (4.7). We'll use just the first two dimensions, $\{r, g\}$. Devise a two-dimensional color histogram for a few images, and find their histogram intersections. Compare image similarity measures with those derived using a three-dimensional color histogram, comparing over several different color resolutions. Is it worth keeping all three dimensions, generally?
8. Suggest at least three ways in which audio analysis can assist in video retrieval-system-related tasks.
9. Implement an image search engine using low-level image features such as color histogram, color moments, and texture. Construct an image database that contains at least 500 images from at least 10 different categories. Perform retrieval tasks using a single low-level feature as well as a combination of features. Which feature combination gives the best retrieval results, in terms of both Precision and Recall, for each category of images?
10. Another way of combining Precision and Recall is the F-score measure. The F-score is the harmonic mean of Precision P and Recall R, defined as

$$F = 2(P * R)/(P + R)$$

    Experiment and determine how F behaves as P and R change.

References

1. M.M. Fleck, D.A. Forsyth, C. Bregler, in *Finding Naked People*, European Congress on Computer Vision, vol 2 (1996), pp. 593–602
2. C.C. Chang, S.Y. Lee, Retrieval of similar pictures on pictorial databases. Pattern Recognit. **24**, 675–680 (1991)
3. M. Worring, P. Sajda, S. Santini, D. Shamma, A.F. Smeaton, Q. Yang, Where is the user in multimedia retrieval? IEEE Multimedia **19**(4), 6–10 (2012)
4. S. Paek, C.L. Sable, V. Hatzivassiloglou, A. Jaimes, B.H. Schiffman, S.-F. Chang, K.R. McKeown, in *Integration of visual and text based approaches for the content labeling and classification of photographs*, ACM SIGIR'99 Workshop on Multimedia Indexing and Retrieval, (1991), pp. 423–444
5. K. Barnard, D.A. Forsyth, in *Learning the semantics of words and pictures*, Proceedings of International Conference on Computer Vision, vol 2 (2001), p. 408–415
6. M.J. Swain, D.H. Ballard, Color indexing. Int. J. Comput. Vision **7**, 11–32 (1991)
7. A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, R. Jain, in *Content-based image retrieval at the end of the early years*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 22 (2000), pp. 1349–1380
8. J.W.H. Tangelder, R.C. Veltkamp, A survey of content based 3d shape retrieval methods. Multimedia Tools Appl. **39**, 441–471 (2008)

9. P. Huang, A. Hilton, J. Starck, Shape similarity for 3d video sequences of people. Int. J. Comput. Vision **89**(2–3), 362–381 (2010)
10. M. Flickner et al., Query by image and video content: the qbic system. IEEE Comput. **28**(9), 23–32 (1995)
11. J. Hafner, H.S. Sawhney, W. Equitz, M. Flickner, W. Niblack, in *Efficient color histogram indexing for quadratic form distance functions*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 17 (1995), pp. 729–736
12. C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: image segmentation using expectation-maximization and its application to image querying. IEEE Trans. Pattern Anal. Mach. Intell. **24**(8), 1026–1038 (2002)
13. A. Pentland, R. Picard, S. Sclaroff, in *System One, Photobook: tools for content-based manipulation of image databases*, Proceedings of SPIE, Storage and Retrieval for Image and Video Databases, vol 2185 (1994), pp. 34–47
14. F. Liu, R.W. Picard, Periodicity, directionality, and randomness: wold features for image modeling and retrieval. IEEE Trans. Pattern Anal. Mach. Intell. **18**, 722–733 (1996)
15. R.W. Picard, T.P. Minka, M. Szummer, Modeling user subjectivity in image libraries. IEEE Int. Conf. Im. Proc. **2**, 777–780 (1996)
16. Y. Rui, T.S. Huang, M. Ortega, S. Mehrotra, Relevance feedback: a power tool for interactive content-based image retrieval. IEEE Trans. Circ. Sys. Video Tech. **8**(5), 644–655 (1998)
17. A. Hampapur, A. Gupta, B. Horowitz, C.F. Shu, in *The Virage Image Search Engine: an open framework for image management*, Proceedings of SPIE, Storage and Retrieval for Image and Video Databases, vol 3022 (1997), pp. 188–198
18. Z.N. Li, O.R. Zaïane, Z. Tauber, Illumination invariance and object model in content-based image and video retrieval. J. Vis. Commun. Image Rep. **10**, 219–244 (1999)
19. H. Tamura, S. Mori, T. Yamawaki, Texture features corresponding to visual perception. IEEE Trans. Syst. Man Cybern. **8**(6), 460–473 (1978)
20. A.R. Rao, G.L. Lohse, in *Towards a Texture Naming System: identifying relevant dimensions of texture*, IEEE Conference Visualization, (1993), pp. 220–227
21. R. Jain, R. Kasturi, B.G. Schunck, *Machine Vision* (McGraw-Hill Inc, New York, 1995), p. 549
22. M.S. Drew, J. Wei, Z.N. Li, Illumination-invariant image retrieval and video segmentation. Pattern Recognit. **32**, 1369–1388 (1999)
23. M.S. Drew, Z.N. Li, Z. Tauber, Illumination color covariant locale-based visual object retrieval. Pattern Recognit. **35**(8), 1687–1704 (2002)
24. T. Deselaers, D. Keysers, H. Ney, Features for image retrieval: an experimental comparison. Inf. Retrieval **11**(2), 77–107 (2008)
25. M.S. Lew, N. Sebe, C. Djeraba, R. Jain, Content-based multimedia information retrieval: state of the art and challenges. ACM Trans. Multimedia Comput. Commun. Appl. **2**(1), 1–19 (2006)
26. R. Datta, D. Joshi, J. Li, J.Z. Wang, Image retrieval: ideas, influences, and trends of the new age. ACM Comput. Surveys, **40**(2), 5:1–5:60 (2008)
27. D. Lowe, Distinctive image features form scale-invariant keypoints. Int. J. Comput. Vision **20**(2), 91–110 (2004)
28. L. Fei-Fei, P. Perona, in *A Bayesian Hierarchical Model for Learning Natural Scene Categories*, Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, (2005)
29. J. Sivic, A. Zisserman, in *Video Google: a text retrieval approach to object matching in videos*, Proceedings of International Conference on Computer Vision (2003)
30. J. Philbin, O. Chum, M. Isard, J. Sivic, A. Zisserman, in *Object Retrieval with Large Vocabularies and Fast Spatial Matching*, Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (2007)
31. W. Zhou, Y. Lu, H. Li, Y. Song, Q. Tian, in *Spatial Coding for Large Scale Partial-duplicate Web Image Search*, Proceedings of ACM Conference on Multimedia (ACM Multimedia) (2010)

32. O. Chum, J. Philbin, J. Sivic, M. Isard, A. Zisserman, *Total Recall: automatic query expansion with a generative feature model for object retrieval*, Proceedings of International Conference on Computer Vision (2007)
33. T.-S. Chua, R. Hong, G. Li, J. Tang, in *From Text Question-Answering to Multimedia QA on Web-scale Media Resources*, Proceedings of the First ACM Workshop on Large-scale Multimedia Retrieval and Mining (2009), pp. 51–58
34. G. Wang, D. Hoiem, D. Forsyth, Learning image similarity from flickr group using fast kernel machines. IEEE Trans. Pattern Anal. Mach. Intell. **34**(11)2, 177–2188 (2012)
35. G.K. Meyers et al., Evaluating multimedia features and fusion for example-based event detection. Mach. Vis. Appl. 25(1), 17–32 (2014)
36. B. Li, E. Chang, C.-T. Wu, in *DPF: A perceptual distance function for image retrieval*, IEEE International Conference on Image Processing, (2002), pp. 597–600
37. W. Niblack, X. Zhu, J.L. Hafner, T. Breuel, D. Ponceleon, D. Petkovic, M.D. Flickner, E. Upfal, S.I. Nin, S. Sull, B. Dom, B.-. Yeo, A. Srinivasan, D. Zivkovic, M. Penner, in *Updates to the QBIC System*, Proceedings of SPIE Storage and Retrieval for Image and Video Databases VI, vol. 3312 (1998), pp. 150–161
38. S.F. Chang et al., Videoq: an automated content based video search system using visual cues. Proc. ACM Multimedia **97**, 313–324 (1997)
39. D. Bordwell, K. Thompson, *Film Art: An Introduction*, 9th edn. (McGraw-Hill, New york, 2009)
40. F. Dufaux, in *Key Frame Selection to Represent a Video*, International Conference on Image Processing (2000), pp. 275–278
41. M.S. Drew, J. Au, Video keyframe production by efficient clustering of compressed chromaticity signatures. ACM Multimedia **2000**, 365–368 (2000)
42. T. Lan, L. Sigal, G. Mori, in *Social Roles in Hierarchical Models for Human Activity Recognition*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2012)
43. C. Schuldt, I. Laptev, B. Caputo, in *Recognizing Human Actions: a local SVM approach*, Proceedings of the International Conference on Pattern Recognition (2004)
44. T. Joachims, in *Training Linear SVMs in Linear Time*, Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (2006)
45. P. Peng, J. Li, Z.N. Li, in *Quality-Aware Mobile Visual Search*, The 3rd International Conference on Integrated Information (2013)
46. B. Girod, V. Chandrasekhar, D.M. Chen, N.M. Cheung, R. Grzeszczuk, Y. Reznik, G. Takacs, S.S. Tsai, R. Vedantham, Mobile visual search. IEEE Signal Process. Mag. **28**(4), 61–76 (2011)
47. V. Chandrasekhar, G. Takacs, D.M. Chen, S.S. Tsai, Y. Reznik, R. Grzeszczuk, B. Girod, Compressed histogram of gradients: a low-bitrate descriptor. Int. J. Comput. Vision **96**(3), 384–399 (2012)
48. Y.A. Reznik, in *On MPEG Work Towards a Standard for Visual Search*, Proceedings of SPIE Applications of Digital Image Processing XXXIV, vol. 8135 (2011)
49. S.A. Chatzichristofis, A. Arampatzis, Y.S. Boutalis, Investigating the behavior of compact composite descriptors in early fusion, late fusion, and distributed image retrieval. Radioengineering **19**(4), 725–733 (2010)
50. C. Singh et al., An effective image retrieval using the fusion of global and local transforms based features. Opt. Laser Technol. **44**(7), 2249–2259 (2012)
51. M. Chen, P. Fu, Y. Sun, H. Zhang, in *Image Retrieval Based on Multi-Feature Similarity Score Fusion Using Genetic Algorithm*, 2nd International Conference on Computer and Automation Engineering, vol 2 (2010) pp. 45–49
52. Y. Huang, D. Ma, J. Zhang, Y. Zhao, S. Yi, A new query dependent feature fusion approach for medical image retrieval based on one-class svm. J. Comput. Inform. Syst. **7**(3), 654–665 (2011)

53. S. Zhang, M. Yang, T. Cour, K. Yu, D.N. Metaxas, in *Query Specific Fusion for Image Retrieval*, European Conference on Computer Vision, (Springer, 2012) pp. 660–673
54. C.J. Liao, S.Y. Chen, Complementary retrieval for distorted images. Pattern Recogn. **35**(8), 1705–1722 (2002)
55. B.K. Singh, A.S. Thoke, K. Verma, A. Chandrakar, Image information retrieval from incomplete queries using color and shape features. Signal Image Process. **2**(4), 213 (2011)
56. J. Xiao, J. Hays, K.A. Ehinger, A. Oliva, A. Torralba, in *Sun Database: large-scale scene recognition from abbey to zoo*, IEEE Conference on Computer Vision and Pattern Recognition (2010), pp. 3485–3492
57. Z. Wang et al., Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)
58. J.Z. Wang, J. Li, G. Wiederhold, Simplicity: semantics-sensitive integrated matching for picture libraries. IEEE Trans. Pattern Anal. Mach. Intell. **23**(9), 947–963 (2001)