Image Database Query Techniques: A Survey

Josh Hyman
University of California
Los Angeles, CA 90095
josh@cs.ucla.edu

April 18, 2007

Abstract

Image databases attempt to allow users to effectively search through large sets of images. These images are typically captured intentionally by a human (eg. vacation pictures) or automatically by cameras attached to sensors (eg. a security camera in an airport). As with all other searchable end-to-end database systems, image databases need the following facilities: storage, indexing, querying, and result display. In this survey, I will focus on the method by which image databases can be queried as well as all supporting technologies such as feature extraction, query matching, and ranking.

1 Introduction

The corpus of images available is growing quickly, and users need effective means to query this dataset. We would like to ask semantic questions about the images in the corpus, but computers are currently unable to discern meaning from images the way humans can. Image database systems attempt to bridge the gap between semantic meaning of a query and a set of image features which can be measured and compared quantitatively.

Though the field of image database and semantic image querying is relative new, it can be easily related to the much more developed field of semantic textual retrieval. Specifically, image features are similar to n-gram features of text [14]. When searching a text document, we are not concerned about letter frequency; instead we are interested in the meaning of the words and how they form thoughts. Just as semantic textual queries can be transformed into measurements of textual features (eg. locality of query terms), semantic image queries can be transformed into image features. Even though the general method to compute results for semantic textural queries is similar to that of semantic image queries, the systems which support these facilities are largely different.

Image database systems are comprised of a number of pieces which span the entire field of computer science. When images are collected, they need to be put into the database which raises storage concerns because images are relatively large in comparison to text. Next, features must be extracted from these images such that they can be compared quantitatively. The user must then be provided with a meaningful interface to easily ask semantically meaningful about the images. The results of the query must be ordered by some feature dependent metric and displayed so that the user can effectively digest the results. To make the user experience more interactive, features should be precomputed off-line and indexed to enhance the wall-time speed of on-line computation.

There are a number of frameworks like MARS [13] and Virage [2] which provide a base on which an image database system can be built. They implement the common functions described above to allow researchers to focus on connecting semantic meanings to image features. These frameworks focus on building real software systems which can answer semantic queries based on current computer vision techniques. Though they may serve as a useful testbed for vision research, they do not attempt to solve the classic vision problems like general object recognition.

In this paper, I will survey techniques to perform semantic image queries, extract image features, and compare extracted features. Other previous surveys of this field like [23] attempt to summarize the largest possible set of current image database systems in the literature. In comparison, this survey will concentrate on techniques and ideas common to many of the systems developed in this field.

2 Extracted Features

Though users interact with image databases via queries, it is impossible to meaningfully talk about such queries without a fundamental understanding of the extracted image features which form their foundation. Though some of these features may seem capable of answering semantic questions about images, it is through their combination that semantic meaning can be derived. Again, consider these features to be similar to word/letter frequencies or locality in the context of text search. Feature are just crude, easily comparable representation of the image in question.

The Virage framework [2] describes features (or primitives as they call them) to be: meaningful, compact in representation, efficient in computation, efficient in comparison, accurate, and indexable. Further, it is important that these features can be automatically be extracted from images. Typically these features are extracted from the image as a whole. However, in some cases features are extracted from sub-images to localize their properties, and preserve their spacial relationship.

2.1 Color

Color is the most obvious feature of an image. Typically, color is the first thing that we humans notice about a picture. Further, it can easily be counted and there has been significant research into color bases and the human perception of color [11].

2.1.1 Color Spaces

The most common representation of color is in the RGB color space. However, this color space is not the most natural way to represent color it as a feature. A more natural color space is HSV (Hue, Saturation, Value) also known as HSL (Hue, Saturation, and Luminescence). This color space maps all colors in the spectrum into the H direction which is intended to be independent of its relative brightness. Because of this property, systems which use this color basis tend to sample the H direction more densely than the S and V directions [7].

Another commonly used color basis is the L*a*b color basis because it closely corresponds to the human perception of color [11]. In order to use this color basis, one must first convert the RGB color into the XYZ color space using a matrix transformation. Then, the L*a*b color can be computed from the XYZ coordinates.

2.1.2 Color Histogram

This simplest means of quantifying the colors in an image is to compute the color histogram over all the pixels in the image. Various systems choose to compute histograms over different color spaces; for example, the Focus [7] and MARS [18] systems choose to compute histograms in the HSV color space where as the Blobworld [5] and ImageRover [21] systems use the L*a*b color space. These histograms, by their nature, are three dimensional and are cumbersome in both size and computational complexity of comparison. To reduce this burden, adjacent histogram buckets can be merged; however, this technique limits the total number of colors the histogram can discern.

A more complex representation of a color histogram is its wavelet decomposition [24]. This technique considers a histogram to be a signal and attempts approximate its form using wavelets. By design, wavelets can provide varying resolution levels with the addition or subtraction of wavelet coefficients. In some cases, the variable amount of information contained in a wavelet decomposition may help strike an optimal balance between discarding information for computational efficiency and the ability to discern as many colors as possible.

2.1.3 Color Locality

Similar to the wavelet decomposition used in WBIIS, the CANDID system [14] attempts to model the 3-D color histogram (and other image features) using a Gaussian Mixture Model. They would like to compute the probability density function which describes the distribution, but since such estimation is quite difficult, they instead fit a Gaussian Mixture Model to the data. To fit the mixture, they cluster the data in high-dimensional feature space using the k-means clustering algorithm. Then for each cluster, they compute the mean μ and standard deviation σ. The Gaussians are then combined into a mixture by weighting Gaussian using the percentage of elements in its corresponding cluster. Such a probability density function will preserve the location of features in the image relative to each other.

Another means of preserving color locality is using

Color Adjacency Graphs as described in [16]. However, this approach is too computationally expensive because of the complexity of images typically found in am image databases. Instead, the Focus system [7] developed a similar approach called a Spacial Proximity Graph (SPG). Such a graph attempts to represent the general layout of colors in an image while being scale independent. The graph is constructed by partitioning the image into a fixed number of sub-images and computing the number of distinct peaks in the local sub-image histograms. The color m and location i represented by a peak in a histogram become a node c_m^i in the graph.

$$E(c_m^i, c_n^j) = \begin{cases} 1 & \text{if } i == j \\ 1 & \text{if } m == n \text{ and } (i,j) \text{ neighbors} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Edges are drawn between peaks in the same sub-image or similar colors in adjacent sub-images as described in equation 1.

2.1.4 Image Color Preprocessing

There are numerous image formats available today. Typically, their goal is preserve humanly observable image quality while maximizing compression. However, image compression increases the computational complexity of accessing individual pixel's color values because of the decompression and change of basis overhead incurred. To overcome this obstacle, images are generally converted to a common representation and color space when added to the database [21]. In addition to image format, some systems normalize image size, correct for object position, and even attempt to align the orientation of all images [19]. Though more ambitious, these techniques attempt to introduce more regularity of image representation in the database, easing the comparison of the associated image features.

Another technique to introduce regularity of image representation is color normalization. Poor lighting, over exposure, and various camera induced effects may cause images appear too dark or too light. To fix these issues basic color properties can be corrected [17] and local non-linear color effects (typically caused by compression) can be removed [19].

As mentioned above, color histograms in any color space are typically too large to be used without further processing. One technique to reduce the size of color histograms is to ignore all colors which don't exist in a given image. In addition, common colors can be clustered and represented by a single color point. This type of color clustering is more effective than histogram bin merging because the boundaries between cluster are placed where logical color changes occur rather than arbitrarily where bin boundaries occur. This technique is called Color Sets [22], a Color Codebook [15], or indexed color [6] and can greatly reduce the representation cost of an image's color histogram.

Though less general than previous techniques, color thresholding (especially gray-level thresholding) can be used to discard parts of an image which are known to be uninteresting. In some cases, it has been found that for a given image set, the image's background satisfies certain color and intensity properties. In these cases, the background can be filtered out in an image preprocessing step to prevent it from interfering with other extracted features. PhotoBook [19] leverages this technique when attempting to distinguish fish species on a common background.

2.2 Texture

Texture of the various objects represented in an image is another important feature. Texture is typically represented by three components: coarseness, contrast, and directionality (CCD) [1]. The QBIC system [8] describes these features as follows: coarseness measures the scale of a texture (pebbles versus boulders), contrast describes its vividness, and directionality describes whether it has a favored direction (like grass) or not (like a smooth object). The CCD values can then be computed for each pixel in the image and bucketed into a histogram like colors. Some systems [18] have found such histograms based directly on CCD values too noisy and have resorted to various smoothing methods.

Steerable pyramids [10] are another way of representing the texture of an object. A description of how steerable pyramids can be used to measure texture can be found in the ImageRover [21] system. Briefly, this technique attempts to model the texture of an object by using computed image properties to "steer" first-order approximating functions. The outputs of these functions are then used to "steer" other approximating functions which are in an adjacent level of the pyramid. This way, the model becomes more accurate as more levels are added (with added computational complexity). ImageRover strikes a balance between accuracy and computation with a 4-level pyramid.

Wold-like decomposition [20] is yet another way to represent texture of an image and its constituent parts. As described by the PhotoBook system [19] the 2-D Wold-like decomposition is made up of three field components: a harmonic field, a generalized-evanescent field, and a purely-indeterministic field. These components represent periodicity, directionality, and randomness, respectively. Such a feature can be used to compare the periodicity of two textures or to compare the other two less salient dimensions if no periodicity exists.

2.3 Shape Descriptors

The simplest description of a shape is the Minimum Bounding Rectangle (MBR) which contains that shape. This technique is used in VisualSEEK [22] to identify objects, assigning an MBR, a center, and edge lengths. Given this shape descriptor, the color features described above can be computed in a localized region to allow for shape based image comparison.

A more accurate way of representing shape would be to use snake splines to outline a shape, as used in QBIC [8], or to detect contour boundaries, as done in NeTra [15]. Detecting contour edges can be computed automatically, although imperfectly. In contrast, the application of snake splines must be done by a human and is generally used to query for a shape rather than define such a region during feature extraction. Both of these techniques attempt to more closely bound the shape to help distinguish it from the containing background.

A more complex approach like Modified Fourier Descriptors (MFD) [18] attempts to model the edges and contours of shape's boundary. A method based on the finite element method (FEM) borrowed from Mechanical Engineering [19] computes the eigenvectors of an object's stiffness matrix, encoding an its deformations relative to an average assumed shape. Both of these techniques are more computationally expensive but can provide more accurate shape descriptions which can increase the accuracy of comparison.

2.4 Segmentation

The simplest image segmentation scheme is to predetermine the size and location of all segments in the image, essentially a grid. This technique is used with great success by MARS [18] and ImageRover [21] to localize other color or texture related features in an image. Shoebox [17] has combined this grid segmentation with Voronoi segmentation based on regions of similar color. However, they found that simple grid based image segmentation provided them with sufficient locality for color and texture features that there was no need for the more expensive Voronoi segmentation.

Another approach is to segment greedily based on non-overlapping regions of color and texture as done by NeTra [15]. This technique is relatively fast to compute, leveraging a dynamic boundary detection scheme which searches for abrupt changes in the directionality of texture. A similar technique employed by MARS [13] performs a simple k-means clustering of color and texture features in 6-dimensions (HSV and CCD). Though slightly more computationally expensive, reported results of these feature based segmentation techniques are very positive.

A yet more expensive segmentation approach is to attempt to fix a Gaussian mixture model to the color and texture features as used in Blobworld [5]. The number of Gaussians to fit is determined by the Minimum Description Length (MDL) principle and then the Expectation-Maximization (EM) algorithm is run on the data to actually fit the model. The idea is to have each model in the mixture represent a single segment in the image. Thus, adjacent pixels which belong to the same Gaussian are clustered together. Though more expensive, this computation can be performed off-line and can still be compared to a user query quickly.

2.5 User-Aided Feature Extraction

The Shoebox system [17] devised a feature not based on the image's pixels, but rather a human's interpretation of the image. They attempted to lower the barrier of image annotation by allowing the user to annotate the images using a microphone and speech recognition. They were able to show that with current speech recognition and textural search technology they were able to achieve a good conceptual accuracy. However, this approach is clearly not scalable and avoids the core issue of discerning semantic meaning from the combination of image features.

3 Query Methods

Queries are the fundamental way by which users interact with an image corpus, as other options (like random image browsing) are too cumbersome or computation intractable. In order for a query method to be useful it should be semantically meaningful and efficiently comparable to images in the database. Here, I describe a number different methods by which an image database can be queried which satisfy these properties and make use the features discussed previously. Further, I will motivate why each type of query effectively converts the semantic meaning of the user's question into a set of quantitatively comparable image features.

3.1 Query by Content or Example

The simplest form of query is an example image. In this form of query, the user submits the sample image itself or a pointer to such and image and asks the system to find similar images. In the WBIIS system [24], the sample image is processed in the exact same way that database images are preprocessed. The same features are extracted, in this case a color wavelet decomposition, and are compared to the database image's features directly. Essentially, this sort of query searches for images whose color distribution and layout resemble that of the query image.

The CANDID system [14] takes the user submitted image and computes a probability density function (PDF) over localized color histogram and CCD texture features in the image. This approach avoids the need to compare high-dimensional feature vectors to compute similarity. Though more computationally expensive than computing bare feature histograms, this PDF more accurately describes the content of the image increasing the accuracy of the search.

In the Blobworld system [5], a user submits an image of interest and selects a particular region. Blobworld uses color histograms and CCD texture features compare regions of user interest. In Focus [7], the user is required to submit just a sub-image (with the background and other unimportant objects subtracted) which is compared to images in the database. This query attempts to find objects who HSV color histogram match that of the queried object. Additionally, in the case of Focus, the location of the object in the image is considered using a Spacial Proximity Graph.

Some systems like [18] only use this sort of example based query during testing to measure the efficiency of their other query methods. In this particular case, they provided ground truth data about which images were actually similar to a particular example image. Then, they provided a particular image's feature vector, containing both CCD texture features and MFD shape features, as a search expression to find other images in the database. From this, they were able to compute the precision and recall of the search. Section 4.4 further discusses how to evaluate the performance of image queries.

3.2 Query by Shape

In addition to querying using an entire example image, some systems like VisualSEEK [22] allow a user to draw a bounding box on a blank canvas and specify properties about that region. For example, one can draw a box in the upper-middle portion of the canvas and specify the color of yellow in an attempt to match images containing a sunset. More than one region may be specified with different properties to further narrow the search. The search is then performed over localized HSV color histograms which have been pruned using Color Sets for computational efficiency.

In the QBIC system [8], the user can outline an object in an image using a snake-split which automatically snaps to inferred edges. Then the user can query for other images which have this sort of shape in the image independent of location. Additionally, the user can sketch a shape free-hand and request a similar query. Similarly, the WBIIS system [24] can perform partial image queries based on user sketches. These systems provide a means for the user to have no example image to work from but instead attempt to find semantically meaningful objects defined by the user's free-hand or assisted sketch.

3.3 Query by Feature

By far, the simplest and least semantically meaningful type of query is direct value thresholding of the features vectors themselves. Though not typically meaningful, this sort of query is generally very efficient. In fact, all other queries reduce to a feature query where the features in question and their thresholds are automatically computed. In VisualSEEK [22] a user can directly query for specific images which contain certain colors and objects of certain sizes. If searching for an ocean scene, for example, a simplistic query for a blue region (the ocean and sky) above a darker region (land) may be sufficient.

Similar facilities are available in most image database systems, though they may be implemented in slightly different ways. In one system [4], object and color location is translated into identifying string (”upper-right” or ”middle”) and then sub-string matching is used to compare location. In NeTra [15], a color codebook can be queried for images with specific colors which can be supplemented by specifying the location as a bounding rectangle.

The QBIC system [8] allows a user to select a texture from a texture palette compiled from all images in the database. These textures are identified by CCD coordinates and are directly compared to the CCD texture feature extracted from the database images. This may be the most semantically meaningful version of direct feature queries as it provides a user with an idea, in this case a representative texture swatch, of what corresponds to a particular feature value.

3.4 Query by Refinement

Though a seemingly inefficient use of a user’s time, query by refinement is a good method to extract features from an entirely semantic query. In the ImageRover system [21], a user is presented with a random subset of images from the database. She then may either request a new set of images if there aren’t any ”interesting” images in the current set or select a particular image to retrieve more similar images. Similar images are judged based on a k-nearest-neighbor clustering of L*a*b color histogram features and steerable pyramid texture features. This process iterates until the user has found images which suitably answer their query. ImageRover reports that typical queries can be answered in this manner within two or three iterations.

A refinement of this strategy is presented in PhotoBook [19] where the initial set of images shown to the user is a result of a text query over the image annotations and then sorted by their relevance. Then the processes iterates by the user choosing a particular image and recursively refining the result set. Images are compared using the eigenvectors of a stiffness matrix for objects, wold-like decompositions for texture, and color.

3.5 Query by Context or Metadata

Though not directly a query of the image content, querying the context or metadata present for a given image can be a good way to refine an image query. In the case of the UC Berkeley Digital Library project [4], many of the images contained text annotations. They were then able to use these annotations to help filter out images which did not match the user’s query which helps them to avoid wasting precious on-line computational resources needed to compare image features. Similarly, PhotoBook [19] starts its query by refinement with a directed textual search as mentioned previously.

The Shoebox system [17] goes a step further and attempts to make it convenient for users to submit text annotations for images as they are being added to the database. Shoebox allows users to speak annotations for their images, transcribing them using speech recognition software. These annotations then become searchable in the Shoebox interface, right along side other image features.

More unique in their metadata discovery, WebSeer [9] collects images from the Internet and extracts metadata about the image from its surrounding web page. Specifically, the web page title and text surrounding the image are captured in addition to the image’s name. All of this metadata is searchable in the WebSeer user interface. This text capture is an attempt to gain semantic meaning from the web page and transfer it to the image for more meaningful search. This technique has been more recently applied by Google Image Search [12].

3.6 Combination of Query Methods

No one particular query method is able to capture the true semantic meaning of a user’s query. Consequently, most image database systems employ multiple query methods to allow a user to effectively express their query. Some systems allow the user to specify a query using multiple methods at once, other systems provide the other query methods as refinement tools. In all cases, the intersection of all query results will be displayed to the user for inspection.

4 Query Matching

As mentioned before, image database queries attempt to map the semantic meaning of the query to a set of feature values. Section 3 describes as number of query methods and the features which they leverage. Though the queries and the features may be different, all image database queries follow the same logical algorithm. First, the semantic query is converted into a set of features; perhaps by extracting features from a sample image (query by example) or by receiving the values directly from the user (query by feature). Second, the features are compared to the features extracted from the images in the database. Lastly, the resulting set of matching images are ranked by some similarity metric which is related to the feature comparison method.

4.1 Feature Comparison

4.1.1 General Feature Distance

Many systems treat features as vectors in a high-dimensional space. By doing so, it is easy to use vector comparison methods to define a distance metric between features. One system [18] defines the distance between two features vectors to be the cosine of the angle between them. The WBIIS system [24] use the euclidean distance between the two feature points in N-dimensional space. This metric is also used by Blobworld [5] and ImageRover [21].

In PhotoBook [19] the distance between two feature vectors is computed as the Root Mean Squared (RMS) Difference. It was found that RMS difference was effective for shape features, but relatively poor for texture features. These metrics are both inexpensive to compute and experimentally accurate. However, not all image features can be represented as a feature vector.

4.1.2 Color Comparison

Color histograms are an example of a image feature which is not easily compared directly as a feature vector. There are many different ways of comparing color histograms which have various different trade-offs. VisualSEEK [22] uses histogram distance, given by equation 2, as a measure of color difference. However, this metric neglects to compare similar colors; for example, a dark red image would be equally dissimilar to both a red and blue image.

$$d(g,h) = \sum_A \sum_B \sum_C (h(a,b,c) - g(a,b,c))^2 \tag{2}$$

One way to include the notion of similar colors into the metric is to use quadratic histogram distance given by equation 3. Here A is the similarity matrix representing the cross-correlation of all colors where $a_{i,j}$ is given by equation 4.

$$d(g,h) = (h-g)^t A (h-g) \tag{3}$$

$$a_{i,j} = \frac{1 - d_{i,j}}{max(d_{i,j})} \tag{4}$$

Here $d_{i,j}$ is the L_1 distance (also known as city block distance) between color i and color j in the RGB color space. A similar value for $a_{i,j}$ can be computed for every color space. Though this metric is much more accurate, it is also far more expensive to compute and is prohibitive in many situations. Even so, system like QBIC [8] and Blobworld [5] rely heavily on histogram quadratic distance.

A similar metric used by VisualSEEk [22] is Color Set Distance. Instead of using a typical 3-D color histogram, they only use a histogram over their limited Color Set. Then they compute the quadratic histogram distance between those to specialized color histograms. This serves to reduce the computation expense of computing this metric and also reduces the amount of data needed to be stored as the color sets are far smaller than typical color histograms.

A much simpler and computationally efficient metric is the histogram intersection given by equation 5, or in the case of [18], its inverse. This measure computes only the overlap between two histograms and is a good first pass to disregard many images which are obviously dissimilar to the query.

$$d(h,g) = \frac{\sum_A \sum_B \sum_C min(h(a,b,c), g(a,b,c)}{min(|h|, |g|)} \tag{5}$$

Another computationally efficient metric is the histogram inner product given by equation 6. Two color histograms which are completely dissimilar (or disjoint) will result in a inner product of zero. However, two identically similar histograms will result in a large value dependent on the number of data points. To resolve this, we can normalize the inner product which then becomes identical to the cosine of the angle between these two histogram functions. This method is used extensively by CANDID [14].

$$d(h,g) = \sum_A \sum_B \sum_C h(a,b,c)g(a,b,c) \qquad (6)$$

The Focus [7] system attempts to find peaks in the 3-D color histogram and then compares the L_2 distance (or euclidean distance) between those peaks. This is both inexpensive computationally and also allows Focus to avoid storing the entire color histogram. Though this may loose accuracy, they argue that this method would be similar to a histogram with all non-zero buckets giving them more discriminating power.

If a Spatial Proximity Graph (SPG) has been computed over the database images, as done by the Focus system [7], then it can be used to compare both colors and the general spacial relationship of those colors in the query image to those images in the database. First, an SPG is computed over the query image as described in Section 2.1.2. Then peaks which don't exist in both graphs are discarded. The remaining subgraphs of the query and database image are compared to see if one contains the other. Unfortunately, this problem reduces to the NP-complete subgraph isomorphism problem. However, since the two graphs have identical labeling, we can compute this in $O(n^m)$ time where n is the number of edges and m is the number of color labels.

4.1.3 Value Thresholding

In general, feature value thresholding is a poor metric for computing the similarity between images. However, it is an excellent way to discard large sets of images which clearly do not match a given query. This lowers the on-line computational burden of a query significantly. This basic strategy can be seen in the FIDS system [3].

The WBIIS system [24] uses a similar method based on the standard deviation of wavelet coefficients in the lowest frequency band. They found that the standard deviation of wavelet coefficients representing color intensity is a good metric of similarity. In the first phase of query matching, they compare the wavelet coefficient standard deviation stored in the database to that of the query image and disregard images whose differed beyond some threshold.

4.1.4 Classifier

The MARS system [18] system applies a probabilistic model to matching user queries. The probability of an image I matching a query variable v_i is defined to be $P(v_i|I)$ where each v_i is derived from the query image. Then we can define a query to be $Q(v_1, v_2, ..., v_n)$ and the probability that an image matches a query to be $P(Q(v_1, v_2, ..., v_n)|I)$. This formulation assumes that all of the features are independent so as not to skew the results. To make this formulation meaningful, all of the distance measures between image features are converted into a probability. Two images are defined to be similar if the probability of similarity is above a predefined threshold. We can treat this probabilistic matching as a rudimentary classifier where each feature becomes a weak-classifier and equal weights are used to combine them.

The WebSeer system [9] uses a battery of tests of classify an image as either a photo, a drawing, or an artificial computer generated image. They have trained this classifier on hundreds of images fetched from the Internet which were classified by hand. Though this comparison alone won't be able to answer a semantic query, it can be used to reduce the number of images which must be considered for a particular query.

4.2 Increasing Performance

The largest barrier that image databases face is the large amount of on-line computation they must perform to return accurate results to semantic queries. Anything that can be done to reduce the amount or complexity of on-line computation pays huge dividends in usability. One technique used by nearly all image database systems is to precompute as much feature data about the stored images as possible off-line. Though some [19] say that this approach limits the queries which can be posed to such a system, it is a necessary step to scaling such systems to deal with real image datasets.

Another approach described in Section 4.1.3 and used extensively in WBIIS [24] is to discard images which don't match a given query. This way, the more computationally expensive operations can be performed on a much smaller set of images. Though this only delivers a constant factor improvement, it is still extremely important with large image sets.

Indexing the extracted features is yet another approach common to nearly all image database systems. However, indexing alone cannot effectively answer queries because the image features are so complex that only the simplest amoung them can be meaningfully indexed. Instead, indexes are used as a more intelligent value threshold; they are used as a first cut to narrow the search and bound the amount of required on-line computation. This method is well illustrated by Focus [7].

A similar approach to reducing on-line computation is to reduce the dimensionality of the feature vectors using techniques like Principle Component Analysis (PCA). ImageRover [21] and other systems use such algorithms to discard part of the feature vectors which don't contain much "useful" information. This reduces both the feature storage cost as well as the feature comparison cost.

4.3 Ranking in Multiple Feature Dimensions

Since nearly all semantic image database queries use more than one query methods and image feature, there needs to be a way to rank the results of query taking into account all of the feature similarity comparisons. This ranking value is often called the compound query score. The VisualSEEK system [22] independently searches for matches in each feature dimension and then computes the intersection of all matched images. Images are then ranked by applying weights to each of the features, where the weights represent the query's confidence in that feature for discerning similarity with respect to the perceived semantic meaning of the query. In the Blobworld system [5], the compound query score is calculated both by using fuzzy-logic operations to aid in clustering and user provided feature weights.

A similar technique is employed by the FIDS system [3] where they try to develop a metric space of the feature similarity results. To do this they combine feature distance measures using the following operations: sum, $weight$, min, and max. This method is more powerful than the simple weighting employed by VisualSEEK or the fuzzy weights used by Blobworld. Further, this scheme can be expanded to represent much more complex relationships between features.

In order to combine the similarity of multiple image features into a total order, the MARS system [18] uses a process called feature sequence normalization so that all features have the same effective weight. First, the system normalizes the values of all features to $[0, 1]$ individually. Second, the variance of the different features are used to remap all features onto a single $[0, 1]$ range such that direct comparison of features values has meaning. If particular feature are found to be more representative of the semantic meaning of a user's query, then those particular features can be weighted during the second step of the normalization. This process allows a total order of image similarity to be trivially computed which is the natural ranking for results to a given user query.

4.4 Evaluation of Matching Accuracy

Without and objective measure of the accuracy features and queries, it is difficult to trade-off the computational complexity of a given features with its expressive power with respect to semantic queries. Here, I will describe the techniques many of these systems leverage to measure this performance

4.4.1 Feature Matching Accuracy

This VisualSEEK system [22] attempts to identify shapes using a minimum bounding rectangle and color histogram. To measure the effectiveness of these features, they built 500 synthetic images composed of twelve different types of shapes randomly placed in the image and filled with a random solid color. They then randomly generated queries, a set of bounding rectangles, and computed which images should have been matched by those queries. Lastly, they extracted features from the synthetic images and compared the result of a query to the system with the ground truth data they computed. This method showed that localized color histograms using minimum bounding rectangles had consistently better precision and recall when compared to global color histograms.

Similarly, in Blobworld [5], they compared the accuracy of their system which leveraged local features computed from image segments against a system which only used a global histogram. Again, they found their system to have consistently better precision and recall.

4.4.2 Query Matching Accuracy

The canonical method to measure the accuracy of a semantic query is to measure it's precision and recall from a test dataset whose ground truth is known. The precision of a query is the ratio of the number of relevant images in the database retrieved to the total number of images retrieved. The recall of a query is the ratio of the number of relevant images retrieved to the total number of relevant images [18]. High precision or high recall alone is not indicative of a good methodology. Instead, the goal is to maximize the sum of precision and recall. Nearly all systems which measured their query accuracy use this method.

Another method for measuring accuracy of a query is to measure the improvement over random retrieval. Though illustrative, it is expected that any system using image feature for query matching should do far better than a simple random retrieval of images from the database. This approach is briefly considered by Shoebox [17].

5 Conclusion

Image database systems are becoming increasingly important as the typical user has access to more images. Further, since computer vision has not been solved in general, we are left to engineer systems which can leverage what little semantic knowledge we can get from both the user's query and the images themselves. In this paper, I have described a number of different methods for querying image database, extracting image features, and comparing extracted features to build an ordered result set.

Though all of the systems surveyed paid attention to the computational complexity of searching image databases, they typically tested their systems on small image sets (approximately 5000 images). In today's world, users have access to millions of images on the Internet and thousands in their personal archive. These number are going to continue to grow in the coming years as cameras are integrated into more personal devices. Future systems will have to leverage the computational power of a cluster of computers if there is any hope to keep up with the abundance of images.

Recent works in the field of image databases largely ignore the performance implications of having extremely large image sets. Though some systems attempt to extract features from images in the compressed domain to avoid some computation, this savings will not be enough to make up for the enormity of the image set. Other systems focus on applying the general techniques described here to specific fields, like the field of medical imaging. Doctors need automated techniques to sift through the huge mounds of data produced by MRI, X-Ray, and CT imagers. Quering image databases is not a solved problem by any meas, and the needs of commercial and personal applications will continue to drive innovation in the image database field for the foreseeable future.

References

[1] M. Amadasun and R. King. Textural features corresponding to textural properties. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):1264–1274, 1989.

[2] J. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R. Jain, and C. Shu. The Virage image search engine: An open framework for image management. *Proceedings of SPIE, Storage and Retrieval for Still Image and Video Databases IV*, pages 76–87, 1996.

[3] A. Berman and L. Shapiro. Flexible image database system for content-based retrieval. *Computer Vision and Image Understanding*, 75(1):175–195, 1999.

[4] C. Carson and V. Ogle. Storage and Retrieval of Feature Data for a Very Large Online Image Collection. *Bulletin of the Technical Committee on Data Engineering*, 19, 1996.

[5] C. Carson, M. Thomas, S. Belongie, J. Hellerstein, and J. Malik. Blobworld: A system for region-based image indexing and retrieval. *Third International Conference on Visual Information Systems*, pages 509–516, 1999.

[6] CompuServe. Graphics Interchange Format 89a. *http://www.w3.org/Graphics/GIF/spec-gif89a.txt*, 1990.

[7] M. Das, E. Riseman, and B. Draper. FOCUS: Searching for multi-colored objects in a diverse imagedatabase. *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition*, pages 756–761, 1997.

[8] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, et al. Query by Image and Video Content: The QBIC System. *Computer*, 28(9):23–32, 1995.

[9] C. Frankel, M. Swain, and V. Athitsos. WebSeer: An Image Search Engine for the World Wide Web. 1996.

[10] W. Freeman and E. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:891–906, 1991.

[11] U. Gargi and R. Kasturi. An evaluation of color histograms based methods of video indexing. *Proceedings of International Workshop on Image Database and Multimedia Search*, 1996.

[12] Google. Google Image Search. *http://images.google.com*, 2007.

[13] T. Huang, S. Mehrotra, and K. Ramchandran. Multimedia Analysis and Retrieval System (MARS) Project. *Prococeedings of Library Application of Data Processing-Digital Image Access and Retrieval*, 1996.

[14] P. Kelly, M. Cannon, and D. Hush. Query by image example: The CANDID approach. *SPIE, Vol 2420, Storage and Rerieval of image and Video Databases III*, pages 238–248, 1995.

[15] W. Ma and B. Manjunath. NeTra: A toolbox for navigating large image databases. *Multimedia Systems*, 7(3):184–198, 1999.

[16] J. Matas, R. Marik, and J. Kittler. On Representation and Matching of Multi-Coloured Objects. *International Conference on Computer Vision*, pages 726–732, 1995.

[17] T. Mills, D. Pye, D. Sinclair, and K. Wood. Shoebox: A digital photo management system. *AT&T Laboratories Cambridge*, 2000.

[18] M. Ortega, Y. Rui, K. Chakrabarti, S. Mehrotra, and T. Huang. Supporting similarity queries in MARS. *Proceedings of the fifth ACM international conference on Multimedia*, pages 403–413, 1997.

[19] A. Pentland, R. Picard, and S. Sclaroff. Photobook: Content-based manipulation of image databases. *International Journal of Computer Vision*, 18(3):233–254, 1996.

[20] R. Picard and F. Liu. A new Wold ordering for image similarity. *IEEE Conference on ASSP*, 1994.

[21] S. Sclaroff, L. Taycher, and M. La Cascia. ImageRover: a content-based image browser for the World Wide Web. *Proceedings IEEE Workshop on Content-Based Access of Image and Video Libraries*, pages 2–9, 1997.

[22] J. Smith and S. Chang. VisualSEEk: a fully automated content-based image query system. *Proceedings of the fourth ACM international conference on Multimedia*, pages 87–98, 1997.

[23] R. Veltkamp and M. Tanase. Content-Based Image Retrieval Systems: A Survey. *Rapport Technique*, 2002.

[24] J. Wang, G. Wiederhold, O. Firschein, and S. Wei. Wavelet-based image indexing techniques with partial sketch retrieval capability. *The 1997 IEEE International Forum on Research and Technology Advances in Digital Libraries, ADL'97*, pages 13–24, 1997.