

Content Based Image Retrieval Feature Vector Selection

Isaac Case

November 7, 2008

Abstract

As the number of digital images increase, the ability to search through all of those images decreases. One potential solution is content based image retrieval or CBIR. CBIR attempts to search databases of images based on decomposed features of the image data. Although there are many different ways to decompose an image into a feature vector, there does not exist, at this time, a best method for all cases. CBIR is able to process larger quantities of images faster than is humanly possible with the intent to find a subset of similar images out of a much larger set of images.

1 Introduction

With ever increasing numbers of images available and grouped together, the ability to search through them to find similar images, or to automatically categorize them, becomes a difficult task. Content based image retrieval, or CBIR, is the name given to the category or potential solutions to this task. One way of describing the goal of CBIR is to explain an example use case scenario. One very simple to explain scenario is, given a large database of images, being able to submit a query into the database, where the query is an image, and the query result is a set of images that are similar to the query image, see Figure 1.

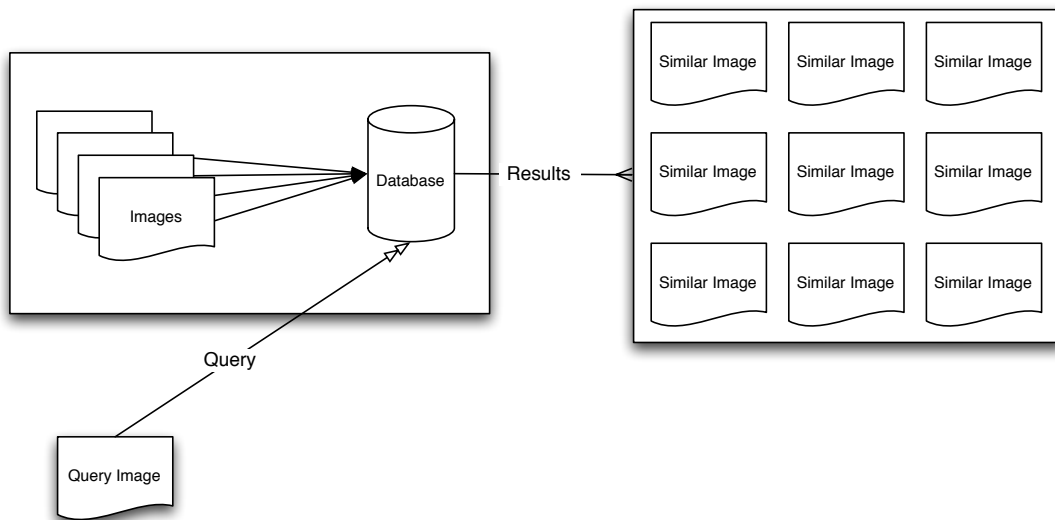


Figure 1: High level view of CBIR

2 Content Based Image Retrieval

One main difficulty in determining similar images is having contextual information about every image in a database. One possible solution is to use the data that is already contained in the image as a means to obtain contextual information. The data contained within an image can be thought of as a two dimensional grid of values, or pixels. This grid of pixels can be referred to as a raster. These pixels may be a single value, or a vector of values in some N -dimensional space. An example of a single value is a grey scale image, where each pixel value is just a percentage of grey ranging from white to black. The most common example of a pixel being a vector is that of RGB images where each pixel has a red, green and blue component of which this combination of red, green and blue values describe the color for a given pixel in RGB space. Trying to match images using only these pixel values, without any further decomposition, appears to be an almost impossible task. An image whose resolution is 1024 by 768 pixels has 786,432 total pixels. If each of these pixels was taken separately, then this image has 786,432 distinct values, or 786,432 dimensions if described as a vector. Comparing this data naively would require an analysis of all 786,432 dimensions. The approach taken currently is similar to that of pattern recognition. It is assumed that there is some information that can be gleaned from the image data, that can be described using less data than the entire contents of the original image. This is a similar problem to lossy compression, where an attempt is made to extract the “important” features from a set of data in less space than is required for the original data set. A quick example is if there is an image of the number zero, then the character “0” is a much smaller representation of the information within the image than the original raster image. This single character representation does not contain the entire information contained in the raster, such as the shape or color of the character, but it may be much faster to compare images for similarity using only the single character encoding.

3 Feature Vectors

This idea that data can be decomposed into a set of values that describe specific features is the basis for a tool called a feature vector. A feature vector is an N -Dimensional vector for which the components of the vector are simplified representations of a more complex data set. Applied to image data, a feature vector for image data is a simpler representation of the data contained within an image. This feature vector can then become the data for which a CBIR database is queried on. The images contained within a CBIR database must be indexed by this feature vector. This then allows us to define CBIR as a function of matching an input feature vector to a database of feature vectors, see Figure 2, and returning the images for which the feature vectors are similar.

Similar to other pattern recognition techniques, some desired properties of these feature vectors are related to their invariance to specific differences. Some of these, are translation invariance, or the ability for the pattern to be matched even if it is shifted in the x or y direction, rotation, and dilation, or scaling. Brousil and Smith described the necessity of invariance in this manner, given a desired shape for which we want to recognize as a specific pattern, if the size of the shape is less than the size of the overall raster, then there can exist many multiple distinct images for which the same pattern exists and should be recognizable, but are distinct due to the placement, rotation, or size of the pattern[4].

Because the feature vector is a reduced representation of the original data set, it is often not possible to describe all possible characteristics of the original data in one reduced representation. It is in this area that there is currently a wide variety of research into what constitutes an adequate feature vector.

3.1 Color

One distinguishing feature of an image is its color. Swain and Ballard’s paper on using color histogram[15] to identify an object is one of the fundamental works in this area. Although it may be thought that form only follows function and therefore color be deemed irrelevant, this is not always the case. According to Swain and Ballard, there are many cases in nature where specific colors are associated with class identity[15]. Examples of this include the fact that chlorophyll, a pigment found in plants used to help in the absorption of light, is

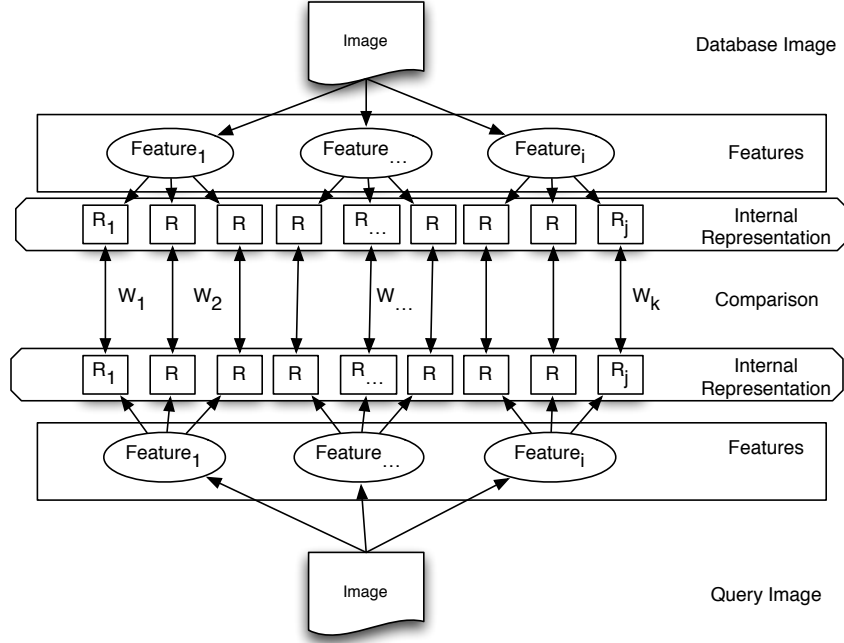


Figure 2: Image Comparison using features

green. This color is then associated with a specific function in plants. Also the idea was proposed that when one is searching for a well known object, the first thing looked for is something that is of the same color as the desired object. If one were to look for a strawberry, red objects with possibly a little bit of green would be prioritized in search above orange and blue objects of the same shape. This then suggests the idea that images can be indexed by color or average color and then searches on those images could use a feature vector containing the color histogram. Color histograms also possess the highly sought after qualities of translation and rotation invariance. They also change only slightly under changes in viewing angle, scale and occlusion.

The comparison of images using this color histogram feature vector can be done by histogram intersection[15]. The histogram intersection is defined, given a pair of histograms H and I , each with n bins, as

$$\sum_{j=1}^n \min(I_j, M_j)$$

The number corresponding to the intersection is the number of pixels for which there is a corresponding pixel in the other image that has the same color. To reduce this to a value between 0 and 1 where 0 is a complete failure and 1 is an absolute match, where match is defined as having the exact same color histogram, this value needs to be normalized relative to the number of pixels in the image[15], defined as

$$H(I, M) = \frac{\sum_{j=1}^n \min(I_j, M_j)}{\sum_{j=1}^n M_j}$$

where $H(I, M)$ is the normalized histogram intersection.

One possible shortcoming of a color histogram is the fact that color is not invariant to the lighting. Given variable lighting conditions, it is necessary to use some form of color-consistency algorithm. One suggested

by Swain and Ballard is a simple algorithm which scales the color axes by the total intensity. This, it was claimed, removed sensitivity to intensity variations[15].

According to Swain and Ballard even if the color histogram intersection isn't used as the only feature for CBIR it is very useful in pruning large databases to smaller sets of potential matches thereby reducing the search space, or at least trimming the amount of images that need to be analyzed for similarity of all features[15].

3.2 Shape

Shape can be another distinguishing feature of the content in an image. Jain and Vailaya [7] investigated the idea that shape could be used as a distinguishing feature of CBIR. Jain and Vailaya stored the shape information as a histogram of edge directions. Using shape as a feature vector had some benefits as well as drawbacks.

One major difference between using shape as an indexing feature of an image and using color is direction. The color histogram of an image will be the same no matter how an image is rotated, if the number of pixels of a certain color are the same, then the histogram is the same. This cannot be said of storing shape. Techniques used to store the shape of an object are subject to pitfalls if care is not taken. If an object is rotated, the object in question is still the same object, but the way in which the shape of the object is calculated or stored may be different. The same can be said of objects that have been translated or scaled. If the image is a close up of a tree compared to a tree in the distance, it is still a picture of a tree, but the way the shape feature is calculated may not be robust to these types of difference.

Jain and Vailaya's use of edge direction histograms to describe shapes within an image has its strengths and drawbacks as well. Edge directions are invariant to translation, but are not inherently invariant to scale or rotation[7]. Also, the color of the object may have effect on the edge direction histogram. A black square on a white background may be described differently than a white square on a black background. Even though both images are of squares, the edge directions are in opposite directions[7].

Another way to detect, describe and store the shape information contained within an image is the use of Fourier features[2]. In Brandt et al's research they were able to first, create an edge map, then use the fast Fourier transform to produce Fourier features. These features are invariant to translation, but not rotation. Another alternative method suggested by Brandt et al is the use of polar Fourier features[2]. These features are rotation invariant, relative to the center of the image, but are not translation or scale invariant. Further work produced even more invariance. According to Brandt et al, Log-polar Fourier features are invariant to translation, scaling and rotation.

According to Brandt et al, one major issue with any Fourier based feature set which also extends to any shape feature is that they are all sensitive to occlusion. If a portion of the object in question is occluded, it is more likely that it will be calculated as a different shape[2], see Figure 3. The shape features calculations are only able to work with data that is actually in the image. There is no extrapolation, or guessing as to what objects in the image may be hidden, or out of frame. If an object is partially missing, it is described as something different, and will likely fail in indexed lookup.

Bruno et al extend this idea of searching by shape by defining a hierarchy of shapes by importance[5]. Bruno et al's claim is that a shape can be decomposed into sets of closed contours. These contours can be divided up by segmenting the image at optimal threshold levels. For this experiment Bruno et al found gray-levels to be the optimal feature to perform thresholding. The method they used to pick optimal levels was based on the work done by Otsu, which automatically picks threshold values for grayscale images from grayscale image histograms[9]. Given these threshold values, it is then possible to map the contours of the image at those threshold values. This will produce contour images for the most prominent features in the image. Bruno et al's use of this was to compare the value of using more than one of these contour features as part of the feature vector that described the image, see Figure 4. According to their research, the use of the first three most important contours improved accuracy of classification of images from $68.6 \pm 16.5\%$ with only the first to $77 \pm 16.9\%$ using the first two to $87.5 \pm 9.7\%$ using the three most important contours in the image[5], where importance is based on the threshold values arrived at by Otsu's thresholding.

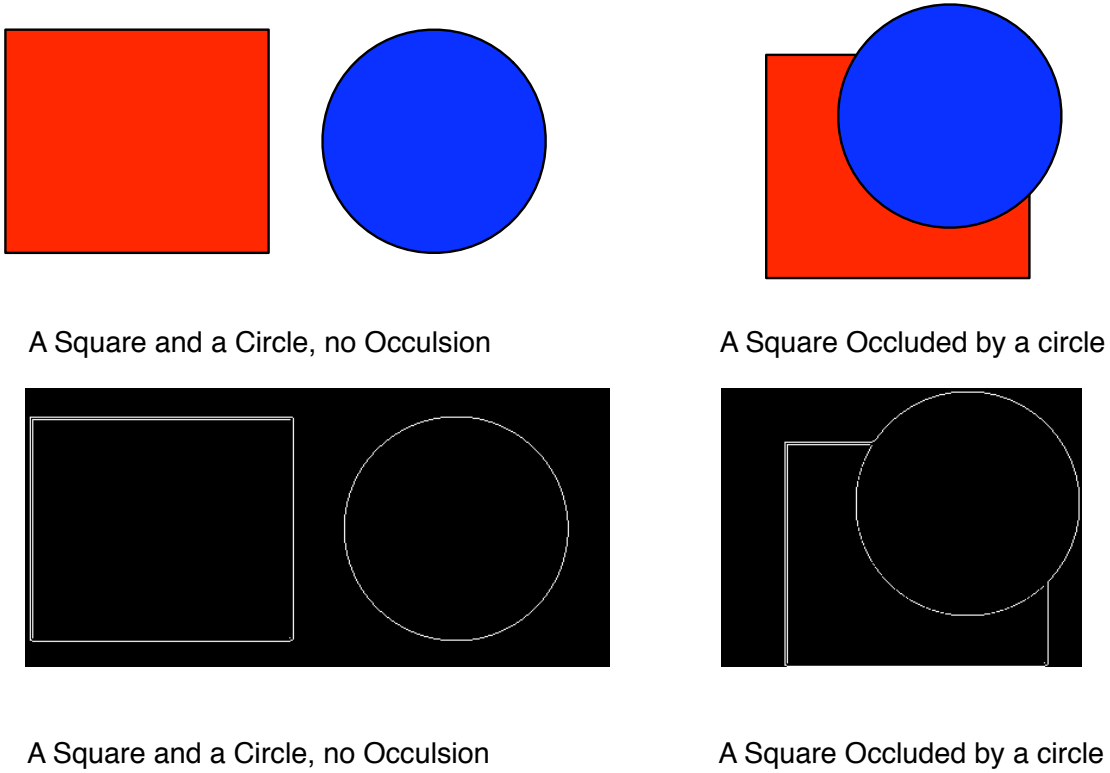


Figure 3: Object Occlusion and their respective edges[2]

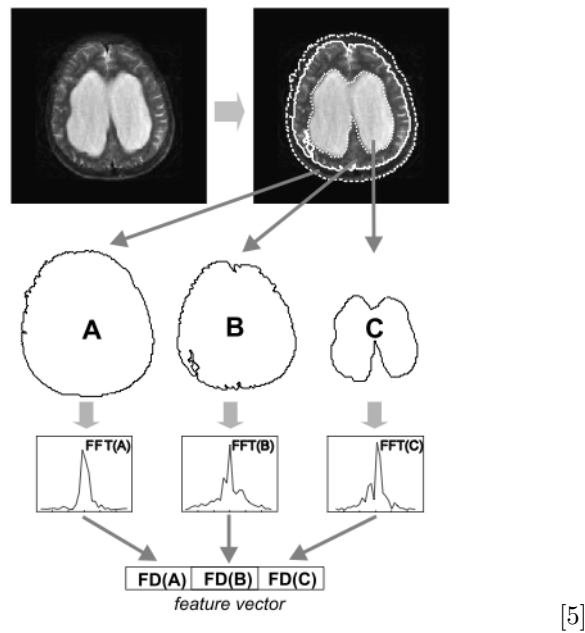


Figure 4: Multiple contours as a Feature Vector[5]

3.3 Texture

Another component in the decomposition of an image into a feature set is that of texture. Smith and Chang describe how this can be accomplished[14]. Smith and Chang performed texture classification using subband energy-based feature sets, namely wavelet subband, uniform subband, discrete cosine transform and spatial positioning. Experimentation was done where of a set of 112 textures, selected from Brodatz's "Textures: A Photographic Album for Artists and Designers"[3], 20 randomly sized and positioned cuts were taken as a sample of these textures. Even though some of the cuts may not accurately represent the original texture that it was derived from, it can be expected that these types of variances will exist in real world images that have only partial information[14]. They then experimented with the classification of these cuts into distinct classes, with correctness being defined as classified cuts should match their original texture classification. Of the energy-based features tested, the wavelet subband and uniform subband performed equally well, with a 92.14% success rate. Smith and Chang claim that this is due to the fact that the textures contain a significant amount of energy in the middle and low frequencies. If the textures were more weighted with less middle or less low energy, then those decompositions might not have fared so well.

Another attempt to isolate the texture features as a component of CBIR is the use of gradient projections as used by Rose and Shah[11]. The gradient was described as the direction of the fastest rate of change from one pixel to any of its eight neighboring pixels. Then, with this gradient data, a one bit edge map can be created for the image based on gradient values and some given threshold where 1 represents gradients larger than the threshold and 0 for gradients smaller than the threshold. Given these gradient based edge maps, the confidence level of a matched image was calculated as the total number of pixels that are the same in both the query and the comparison image, divided by the total number of pixels[11]. This confidence level is then ranged between 0 and 1 and was used to determine the closest matching images. For testing purposes, a database of 82 images including images of beaches, forests, buildings, people, ships and bays was used. Each of the images in the database was submitted as a query into the database. It should be noted that for each image, the first, most similar, image returned was itself. Although no data was provided as part of Rose and Shah research, results were described anecdotally. Images that were take a fraction of a second apart from each other were score with high confidence, as were images with related structures. One weakness of this approach was that it is inherently color blind. Since only texture information was used in the feature vector, no color similarities were examined or used as a measure of success.

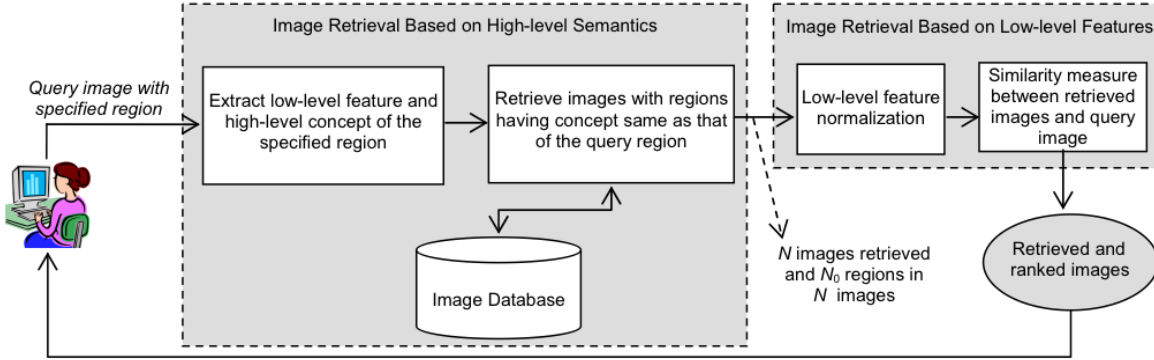
3.4 Semantics

Another potential component of a feature vector is semantic information. This technique was researched by Liu et al[8] as part of a CBIR system. The basic function of the system query based on not just a source image, but also a specified region of the source image from which the system should glean semantic information. The system would then extract both the low level features of the region and the high level concepts. Then once high level concepts have been determined, search for other images having regions containing the same high level concept as the query image. It is only after the search space has been reduced conceptually that there is the standard low level feature comparison performed to determine similarity and results are ranked by similarity and returned to the user, see Figure 5.

According to Liu et al, adding this semantic information, or concept information, increased precision rates by over 10% compared with conventional CBIR systems. They claim that this represents a significant reduction in the semantic gap between low level features and high level concepts, or semantic features.

3.5 Combinations of Features

There does not seem to be one significant feature that dominates of others within a CBIR search. Features such as color, shape and texture, although powerful alone, are more descriptive when combined. For example, a ball described as a circle, and a ball described as a blue object are both accurate descriptions, but a ball described as a blue circle is a much for specific description of what is contained within the image. Jain and Vailaya combined a set of features that described shape with a set of features that described color, or a



[8]

Figure 5: A High Level Semantic Search CBIR System[8]

color histogram[7]. These combined features are what constitutes a feature vector that is a descriptor for the entire image. Since this feature vector contains both shape and color information, queries into the CBIR system can be selective on both overall color and shape for possible matches.

Fu et al also investigated using a combination of textures and shape. Specifically they looked at combining Gabor filters and Zernike moments, where the Gabor filter is used for texture feature extraction and Zernike moments extract shape features[6]. The Gabor filters are a group of wavelets, with each capturing energy for a specific direction and energy frequency. For this system, Fu et al chose comparing Gabor filters to be the simple euclidean distance between feature vectors. The Zernike moments, chosen to extract shape information, and again the euclidean distance was chosen as the similarity metric between two feature vectors. When testing the effectiveness of using these metrics as features in the feature vector, different combinations of features were used in testing. In order to gain some perspective on the utility of using both shape and texture features, Fu et al tested these features on two databases, one being a fingerprint database, and another being a database of MPEG-7 Core Experiment CE-shape-1 part B. For the database containing fingerprints, the Gabor filter was very successful, while the same feature vector used on the MPEG-7 database, scored the worst out of all combinations. Also, in contrast, the feature vector containing only Zernike moments scored very poorly with the fingerprint database, yet scored very well when tested with the MPEG-7 database. These results are not very surprising since the fingerprint database can be thought of as a database of textures, whereas the database of MPEG-7 images is mostly images who's most distinguishing feature is that of shape without any texture. The weakness of both feature vector components are visible, given an appropriate database of images to test against. In contrast to this, using both the Gabor filters and Zernike moments as a feature vector scored very well with both the fingerprint database and the MPEG-7 database. Although Fu et al described certain limitations to these specific feature vector components, namely the fact that this combination lacks the property of size invariance, it was shown that using a combination of features in a given feature vector can help compensate for limitations in either feature component.

3.6 Number of Components

Given all these different types of feature vector components, it might be suggested that a best solution would be to combine all these possible components into one large feature vector. This concept has some flaws, namely that as the number of feature vector components increase, so does the time to compute similarities between different feature vectors since there is more data to compare. In a study done by Rudinac et al, a test was done to compare a large 556 component feature vector system against two reduced component systems. The two reductions were first, a 50 component reduction from which the 50 components were selected from the original 556 components, and second a 25 component system for which the 25 components

described color and texture[12].

To test these two reductions, Rudinac et al described a performance metric P_B as

$$P_B = \frac{R}{B} \times 100$$

where B is the number of returned “best matches” and R as the number of images annotated as relevant. Given this definition of precision, and the reductions it was noted that the reduced set of only 50 components, a subset of the original 556, was able to achieve a 60.25% precision compared to the the full 556 component set which obtained 66.5% precision[12].

The specific way in which the feature vector components were reduced was not in the indices of the images in the database, but rather in the query feature vector. The idea being that it is unknown what features may be similar to a query image, but it is known what features a query image has, therefore it may be possible to reduce the complexity of the query, and the lookup, given some analysis of the query image.

As an alternative to the 50 component set, Rudinac et al proposed and tested a 25 component feature vector which was distinctly different than the original 556 component feature vector. This was an example of a reduced feature vector for indexing where the query feature vector contained all components of the index feature vector. This specific 25 component feature vector consisted of color moments in HSV space and gray-level co-occurrence matrix components[12]. Although the results of testing this reduced component set were positive in a small subset of images, given the full image test that the other feature vectors were exposed to it showed reduced precision. In comparison, the 25 component feature vector achieved a 44.75% precision compared to the 66.5% precision of the full 556 component feature vector. Rudinac et al agreed with the results of this test due to a claim that color moments and components from the gray-level co-occurrence matrix are not enough for the precision needed given the heterogeneous image set they were testing against[12].

3.7 Selection

After a set of features have been selected as “ideal” it is still necessary to determine appropriate weights for each of the features. It also may be possible to reduce the number of components in the feature vector as seen in section 3.6 through further processing. One method known as relevance feedback significantly improves the quality of the CBIR results[13]. According to Rui et al choosing at a high level the features and their importance was limited in usefulness because of the difficulty in representing high level concepts using low level features given human subjectivity.

The relevance feedback method is then an interactive query, for which the user of the system and the system interact to increase the accuracy or relevance of the results of the CBIR. Instead of having the user understand the internal features selected for the feature vectors and the weights associated with those features, the user only needs to be able to respond to a query response. Given a user query, the CBIR system returns a set of images believed to be similar. The user then is given a method to rate the relevance of the returned images. Based on this response, the CBIR system is then able to update associated with the internal representation of the features to produce resultant images which more close match the relevance criteria given by the user. This is done by first initializing the CBIR system with weights for each feature vector, W_0 , for which there is no bias. Upon query, each database image’s similarity is calculated and stored according to a predefined similarity measure for each feature. The images are then sorted according to their similarity measure and reported back to the user in order[13]. In Rui et al’s implementation, the user is then able to rate the images as *highly relevant*, *relevant*, *no opinion*, *nonrelevant*, or *highly nonrelevant*[13], however in other systems, a binary representation of *relevant* or *nonrelevant* may be applicable[1]. The system then updates the weights (see Figure 2) associated with each feature according to the users feedback such that the adjusted results better match the user’s feedback. This process can be repeated over and over again continually updating the query and the weights to better match the desired results.

According to Rui et al this method of feature vector selection has two main advantages over a predefined set of weights. For one, it removes a burden from the user. The user of the system is no longer required to specify an arbitrary set of weights at the same time a query is constructed. This system also removes a

burden from the computer. The computer is no longer required to understand the high level concepts or interactions between different features, instead these high level concepts are determined as part of the user feedback process, and is able to update the weights dynamically as feedback is provided by the user[13].

Using two separate image libraries, one from the Fowler Museum of Cultural History at the University of California, Los Angeles consisting of 286 images, and another from the Corel Corporation containing more than 70,000 images. These image sets differed as the first set contained only images of African and Peruvian artifacts while the set from Corel contained images from over 500 different categories. Given these two distinct sets, this relevance feedback system was then tested on both, exercising the ability to search a relatively small image set, of mostly homogeneous images, and another, a much more heterogeneous set, with a wide coverage of distinct items. Using the relevance feedback approach, the precision of results, as tested on the artifact images, increased on average from 73.3% to 95.3% within the first iteration. Similar results were seen on the Corel set with improvements averaging from 57.1% to 65.9% after the first iteration[13]. For all cases the most significant improvement was seen after the first iteration. After the first iteration minor improvements were visible, but none to the extent of the initial feedback response.

One other response to the issue is the idea presented by Patiño-Escarcina and Costa of using a descriptors hierarchy[10], see Figure 6. This idea is that low level features can be organized into a hierarchy, which is somewhat representational of reducing a problem into multiple subproblems which are separated at a higher level of the hierarchy. If images fails a high level feature, then they are not considered for lower level features. This then reduces the number of comparisons needed, given a fairly large feature vector. The high level features sort out non matches by doing the fewest number of feature vector component comparisons possible with the goal of failing early, then only the successful matches will be compared using all feature vector components.

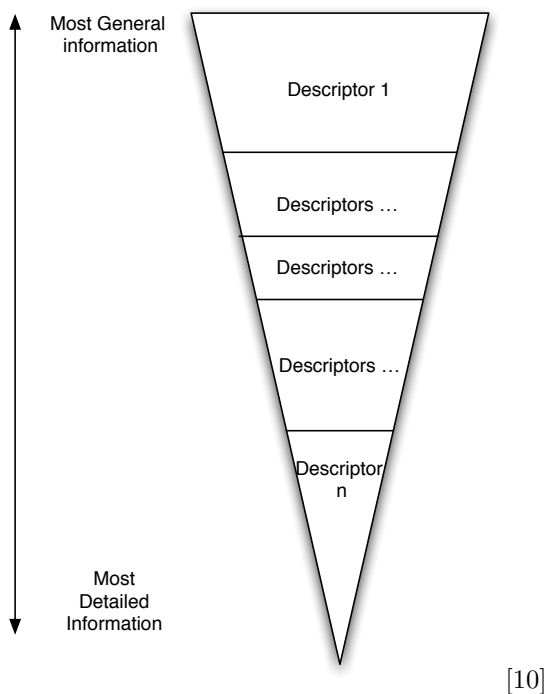


Figure 6: Feature Descriptor Hierarchy

An example of this given by Patiño-Escarcina and Costa is that of images of people. If it is possible at a first pass to determine the gender of the person in the query image, then it is possible to reduce the search space to only those images of people of the same gender. Then a next level descriptor could be hair color, since the space has been reduced, this next feature vector comparison would only be done on the

reduce set of images of people of the same gender. If the next feature vector component was eye color, this comparison would only be done to those images of people with the same gender, and hair color, etc. Not only does this help reduce the computational complexity by reducing the search space, by failing early, it was also show to increase the success rate since the hierarchy intrinsically implies importance to higher level features over lower level features. This method was tested against parallel descriptors, which is where different feature vector components are searched out in parallel and then combined to obtain the matched images, and composed descriptors which is where all feature vector components are tested together as one feature vector undivided. These three methods were tested on an image database of 2200 images, and a query of the first 50 similar images used as the test. Using a hierarchical descriptor resulted in 22 successful matches. Using a parallel descriptor only resulted in 16 successful matches and using a composed descriptor only resulted in 14 successful matches.

4 Conclusion and Future Work

CBIR can be seen as a branch of pattern recognition. It also has the harder job of attempting to match multiple, potentially overlapping patterns within a single raster. In combination with that there is a notion of high level concepts that are tied to these images, and the searches that are requested on them. Like pattern recognition, one common way to approach the problem is to break up the problem into smaller, more mathematically friendly problems, and search on those subproblems. There are many different ways to analyze or decompose an image and many ways to combine these decompositions into feature vectors. The selection of these feature vectors and the organization around how they are used in search is critical to the success of any CBIR algorithm. The mapping of high level concepts to low level features appears to be a key part of this puzzle in accurately matching desired results, and ignoring non relevant images. Some of this difficulty has been mitigated with relevance feedback, however, relevance feedback is an attempt to glean information of desired results from a selection of deemed relevant images. Relevance feedback also requires multiple iterations with the user to feedback the query information. Many of the gaps in CBIR feature vector selection still stem from the difficulty in representing the high level concepts desired by users. This is not only a difficulty in implementing these concepts, but also in dealing with difference between individual preference. Given all of this, it is impressive to see what searches are possible using only the raster information of an image.

For the future of CBIR, work should continue in refining appropriate feature vector components. One difficulty in refining existing systems is having a standardized comparison test for these systems. This test should be both a test of accuracy and of speed since both are vital for the usefulness of CBIR. Given that, a standardized image set should be created for which different CBIR systems can be tested against should be created and agreed upon. Many studies done recently used image sets that were readily available to them at the time. Although this may makes it easier to do testing, it makes it difficult to compare and contrast different systems objectively. Much of the difficulty in creating such an image set is that often different algorithms are targeted to different applications. An image set that is a general representation of consumer photographs may not be a good metric for a CBIR system who's application is geared toward medical imaging. This should not prevent an existence of such an image set, as it may be used as a supplement to the targeted image set, to assist with comparisons. Also, one other difficulty with comparing different CBIR systems is the lack of complexity analysis, or any sort of speed tests. Of the literature, it was noted for some of the systems to give approximate times for indexing or searching, but these numbers were given with respect to whatever test system development was done on. It becomes difficult to compare algorithms when times are given for multiple platforms, e.g. a Sun SPARC workstation vs. an Intel Pentium III machine. Preferably all algorithms should be fast enough for it not to matter, but a reference platform along with a reference image set should help separate algorithms. Given a specific ground level, it then becomes possible to compare different systems and see the progress that has been made without having to implement all other systems of interest to see how they perform on a non standard platform and image suite test. Given stable metrics, it would be of interest to see how far CBIR has come in a relatively short amount of time.

References

- [1] Miguel Arevalillo-Herraez, Mario Zacarez, Xaro Benavent, and Esther de Ves. A relevance feedback cbir algorithm based on fuzzy sets. *Signal Processing: Image Communication*, 23:490–504, 2008.
- [2] S. Brandt, J. Laaksonen, and E. Oja. Statistical shape features in content-based image retrieval. *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, 2:1062–1065, 2000.
- [3] Phil. Brodatz. *Textures; a photographic album for artists and designers*. Dover Publications New York,, 1966.
- [4] James K. Brousil and David R. Smith. A threshold logic network for shape invariance. *IEEE Transactions on Electronic Computers*, EC-16(6):818–828, 1967.
- [5] Odemir Martinez Bruno, Luis Gustavo Nonato, Mario Augusto Pazoti, and Joao Batista Neto. Topological multi-contour decomposition for image analysis and image retrieval. *Pattern Recognition Letters*, 29:1675–1683, 2008.
- [6] X. Fu, Y. Li, R. Harrison, and S. Belkasim. Content-based image retrieval using gabor-zernike features. *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, 2:417–420, 2006.
- [7] Anil K. Jain and Aditya Vailaya. Image retrieval using color and shape. *Pattern Recognition*, 29(8):1233–1244, 1996.
- [8] Ying Liu, Dengsheng Zhang, and Guojun Lu. Region-based image retrieval with high-level semantics using decision tree learning. *Pattern Recognition*, 41:2554–2570, 2008.
- [9] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.
- [10] R.E. Patino-Escarcina and J.A.F. Costa. Content based image retrieval using a descriptors hierarchy. *Hybrid Intelligent Systems, 2007. HIS 2007. 7th International Conference on*, pages 228–233, Sept. 2007.
- [11] J. Rose and M. Shah. Content-based image retrieval using gradient projections. *Southeastcon '98. Proceedings. IEEE*, pages 118–121, Apr 1998.
- [12] S. Rudinac, G. Zajic, M. Uscumlic, M. Rudinac, and B. Reljin. Comparison of cbir systems with different number of feature vector components. *Semantic Media Adaptation and Personalization, Second International Workshop on*, pages 199–204, Dec. 2007.
- [13] Yong Rui, T.S. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: a power tool for interactive content-based image retrieval. *Circuits and Systems for Video Technology, IEEE Transactions on*, 8(5):644–655, Sep 1998.
- [14] John R. Smith and Shih-Fu Chang. Transform features for texture classification and discrimination in large image databases. *Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference*, 3:407–411, 1994.
- [15] M. Swain and D. Ballard. Color indexing. *International Journal of Computer Vision*, 7 (1):11–32, 1991.