Histogram Refinement for Content-Based Image Retrieval

Greg Pass Ramin Zabih*

Computer Science Department
Cornell University
Ithaca, NY 14853
gregpass,rdz@cs.cornell.edu
http://www.cs.cornell.edu/home/rdz/refinement.html

Abstract

Color histograms are widely used for content-based image retrieval. Their advantages are efficiency, and insensitivity to small changes in camera viewpoint. However, a histogram is a coarse characterization of an image, and so images with very different appearances can have similar histograms. We describe a technique for comparing images called histogram refinement*, which imposes additional constraints on histogram based matching. Histogram refinement splits the pixels in a given bucket into several classes, based upon some local property. Within a given bucket, only pixels in the same class are compared. We describe a split histogram called a* color coherence vector *(CCV), which partitions each histogram bucket based on spatial coherence. CCV's can be computed at over 5 images per second on a standard workstation. A database with 15,000 images can be queried using CCV's in under 2 seconds. We demonstrate that histogram refinement can be used to distinguish images whose color histograms are indistinguishable.*

1 Introduction

Many applications require methods for comparing images based on their overall appearance. Color histograms are a popular solution to this problem, and are used in systems like QBIC [2] and Chabot [6]. Color histograms are computationally efficient, and generally insensitive to small changes in camera position. However, a color histogram provides only a very coarse characterization of an image; images with similar histograms can have dramatically different appearances. For example, the images shown in figure 1 have similar color histograms.

In this paper we describe a method which imposes additional constraints on histogram based matching. In *histogram refinement*, the pixels within a given bucket are split into classes based upon some local property. Split histograms are compared on a bucket by bucket basis, similar to standard histogram matching. Within a given bucket, only pixels with the same property are compared. Two images with identical color histograms can have different split histograms; thus, split histograms create a finer distinction than color histograms. This is particularly important for large image databases, in which many images can have similar color histograms.

Figure 1: Two images with similar color histograms

We have experimented with a split histogram called a *color coherence vector* (CCV), which partitions pixels based upon their spatial coherence. A coherent pixel is part of some sizable contiguous region, while an incoherent pixel is not. While the two images shown in figure 1 have similar color histograms, their CCV's are very different.[1] For example, red pixels appear in both images in similar quantities. In the left image the red pixels (from the flowers) are widely scattered, while in the right image the red pixels (from the golfer's shirt) form a single coherent region.

We begin with a review of color histograms. In section 3 we describe histogram refinement, and present two examples that capture spatial information. Section 4 provides examples of refinement-based image queries and shows that they can give superior results to color histograms. We compare our work with some recent algorithms [5, 8, 9, 10] that also combine spatial information with color histograms.

2 Color Histograms

Color histograms are frequently used to compare images. Examples of their use in multimedia applications include scene break detection and querying a database of images [7, 6, 2]. Color histograms are popular because they are trivial to compute, and tend to

*To whom correspondence should be addressed

[1] The color images used in this paper can be found at http://www.cs.cornell.edu/home/rdz/refinement.html.

be robust against small changes in camera viewpoint. For example, Swain and Ballard [12] describe the use of color histograms for identifying objects. Stricker and Swain [11] analyze the information capacity of color histograms.

We will assume that all images are scaled to contain the same number of pixels M. We discretize the colorspace of the image such that there are n distinct (discretized) colors. A color histogram H is a vector $\langle h_1, h_2, \ldots, h_n \rangle$, in which each bucket h_j contains the number of pixels of color j in the image. Typically images are represented in the RGB colorspace, with a few of the most significant bits per color channel.

For a given image I, the color histogram H_I is a compact summary of the image. A database of images can be queried to find the most similar image to I, and can return the image I' with the most similar color histogram $H_{I'}$. Color histograms are typically compared using the L_1-distance or the L_2-distance, although more complex measures have also been considered [4].

3 Histogram Refinement

In *histogram refinement* the pixels of a given bucket are subdivided into classes based on local features. There are many possible features, including texture, orientation, distance from the nearest edge, relative brightness, etc. Histogram refinement prevents pixels in the same bucket from matching each other if they do not fall into the same class. Pixels in the same class can be compared using any standard method for comparing histogram buckets (such as the L_1 distance). This allows fine distinctions that cannot be made with color histograms.

As a simple example of histogram refinement, consider a positional refinement where each pixel in a given color bucket is classified as either "in the center" of the image, or not. Specifically, the centermost 75% of the pixels are defined as the "center". This produces a split histogram in which the pixels of color buckets are loosely constrained by their location in the image. The resulting split histograms can be compared using the L_1 distance. We will call this simple form of histogram refinement *centering refinement*.

Color coherence vectors

CCV's are a more sophisticated form of histogram refinement, in which histogram buckets are partitioned based on spatial coherence. Our coherence measure classifies pixels as either coherent or incoherent. A coherent pixel is a part of a sizable contiguous region, while an incoherent pixel is not. A *color coherence vector* represents this classification for each color in the image.

The initial stage in computing a CCV is similar to the computation of a color histogram. We first blur the image slightly by replacing pixel values with the average value in a small local neighborhood (currently including the 8 adjacent pixels). We then discretize the colorspace, such that there are only n distinct colors in the image.

The next step is to classify the pixels within a given color bucket as either coherent or incoherent. A coherent pixel is part of a large group of pixels of the same color, while an incoherent pixel is not. We determine the pixel groups by computing connected components. A connected component C is a maximal set of pixels such that for any two pixels $p, p' \in C$, there is a path in C between p and p'. We compute connected components using 4-connected neighbors within a given discretized color bucket. We classify pixels as either coherent or incoherent depending on the size in pixels of its connected component. A pixel is coherent if the size of its connected component exceeds a fixed value τ; otherwise, the pixel is incoherent.

For a given discretized color, some of the pixels with that color will be coherent and some will be incoherent. Let us call the number of coherent pixels of the j'th discretized color α_j and the number of incoherent pixels β_j. Clearly, the total number of pixels with that color is $\alpha_j + \beta_j$, and so a color histogram would summarize an image as $\langle \alpha_1 + \beta_1, \ldots, \alpha_n + \beta_n \rangle$. Instead, for each color we compute the pair (α_j, β_j) which we will call the *coherence pair* for the j'th color. The *color coherence vector* for the image consists of $\langle (\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n) \rangle$. This is a vector of coherence pairs, one for each discretized color.

In our experiments, all images were scaled to contain $M = 38,976$ pixels, and we have used $\tau = 300$ pixels (so a region is classified as coherent if its area is about 1% of the image). With this value of τ, an average image in our database consists of approximately 75% coherent pixels, with a standard deviation of 11%.

Two images I, I' can be compared using their CCV's, for example by using the L_1 distance. Let the coherence pairs for the j'th color bucket be (α_j, β_j) in I and (α'_j, β'_j) in I'. Using the L_1 distance to compare CCV's, the j'th bucket's contribution to the distance between I and I' is

$$\Delta_{CCV} = \left|(\alpha_j - \alpha'_j)\right| + \left|(\beta_j - \beta'_j)\right|. \quad (1)$$

Note that when using color histograms to compare I and I', the j'th bucket's contribution is

$$\Delta_{CH} = \left|(\alpha_j + \beta_j) - (\alpha'_j + \beta'_j)\right|. \quad (2)$$

It follows that CCV's create a finer distinction than color histograms. A given color bucket j can contain the same number of pixels in I as in I', but these pixels may be entirely incoherent in I and entirely coherent in I' (i.e., $\alpha = \beta' = 0$). Formally, $\Delta_{CH} \leq \Delta_{CCV}$ follows from equations 1 and 2, and the fact that the L_1 distance obeys the triangle inequality.

4 Experimental Results

We have implemented histogram refinement, and have used it for image retrieval from a large database. Our database consists of 14,554 images, which are drawn from a variety of sources. Our largest sources include the 11,667 images used in Chabot [6], the 1,440 images used in QBIC [2], and a 1,005 image database available from Corel. In addition, we included a few groups of images in PhotoCD format. Finally, we have taken a number of MPEG videos from the Web and segmented them into scenes. We have added one or

two images from each scene to the database, totaling 349 images. The image database thus contains a wide variety of imagery.

We have compared our results with a number of color histogram variants. These include the L_1 and L_2 distances, with both 64 and 512 color buckets. We include a small amount of smoothing as it empirically improved performance. On our database, the L_1 distance with the 64-bucket RGB colorspace gave the best results, and is used as a benchmark.

Hand examination of our database revealed 75 pairs of images which contain different views of the same scene. Examples are shown in figures 2 and 3. One image is selected as a query image, and the other represents a "correct" answer. In each case, we have shown where the second image ranks, when similarity is computed using color histograms or using histogram refinement. Specifically, results are shown using CCV's, centering refinement, and a successive refinement technique described in section 6.1. The color images shown are available at http://www.cs.cornell.edu/home/rdz/refinement.html.

4.1 Centering refinement results

In 69 of the 75 cases, centering refinement produced better results, while in 4 cases it produced worse results (there were 2 cases where the ranks did not change). The average change in rank due to centering refinement was an improvement of 55 positions (this included all 75 cases). The average percentage change in rank was an improvement of 41%. In the 69 cases where centering refinement performed better than color histograms, the average improvement in rank was 60 positions, and the average percentage improvement was 49%. In the 4 cases where color histograms performed better than centering refinement, the average rank improvement was 10 positions. We have not yet analyzed these 4 cases to determine why centering refinement fails.

To analyze the statistical significance of this data, we formulate the null hypothesis H_0 which states that centering refinement is equally likely to cause a positive change in ranks (i.e., an improvement) or a negative change. We will discard the 2 ties to simplify the analysis. Under H_0, the expected number of positive changes is 36.5, with a standard deviation of $\sqrt{73}/2 \approx 4.27$. The actual number of positive changes is 69, which is more than 7.6 standard deviations greater than the number expected under H_0. We can therefore reject H_0 at any standard significance level (such as 99.9%).

4.2 CCV results

In 68 of the 75 cases, CCV's produced better results, while in 7 cases they produced worse results. The average change in rank due to CCV's was an improvement of 68 positions (note that this included the 7 cases where CCV's do worse). The average percentage change in rank was an improvement of 35%. In the 68 cases where CCV's performed better than color histograms, the average improvement in rank was 77 positions, and the average percentage improvement was 56%. In the 7 cases where color histograms performed better, the average improvement was 17 positions.

The null hypothesis H_0 states that CCV's are equally likely to cause a positive change in ranks (i.e., an improvement) or a negative change. Under H_0, the expected number of positive changes is 37.5, with a standard deviation of $\sqrt{75}/2 \approx 4.33$. The actual number of positive changes is 68, which is more than 7 standard deviations greater than the number expected under H_0. We can therefore reject H_0 at any standard significance level (such as 99.9%).

When CCV's produced worse results, it was always due to a change in overall image brightness (i.e., the two images were almost identical, except that one was brighter than the other). Because CCV's use discretized color buckets for segmentation, they are more sensitive to changes in overall image brightness than color histograms. We believe that this difficulty can be overcome by using a better colorspace than RGB, as we discuss in section 6.2.

4.3 Efficiency

We have experimented with a number of different techniques for histogram refinement. CCV's are the most computationally expensive method of these, and will be our focus in discussing efficiency.

There are two phases to the computation involved in querying an image database. First, when an image is inserted into the database, a CCV must be computed. Second, when the database is queried, some number of the most similar images must be retrieved. Most methods for content-based indexing include these distinct phases. For both color histograms and CCV's, these phases can be implemented in linear time with a single pass over the image.

We ran our experiments on a 50 MHz SPARCstation 20, and provide the results from color histogramming for comparison. Color histograms can be computed at 67 images per second, while CCV's can be computed at 5 images per second. Using color histograms, 21,940 comparisons can be performed per second, while with CCV's 7,746 can be performed per second. The images used for benchmarking are 232×168. Both implementations are preliminary, and the performance can definitely be improved.

5 Related Work

Our work has focused on the use of spatial information to refine color histograms. Recently, several authors have proposed algorithms for comparing images that combine spatial information with color histograms. Hsu *et al.* [5] attempts to capture the spatial arrangement of the different colors in the image, in order to perform more accurate content-based image retrieval. Rickman and Stonham [8] randomly sample the endpoints of small triangles and compare the distributions of these triplets. Smith and Chang [9] concentrate on queries that combine spatial information with color. Stricker and Dimai [10] divide the image into five partially overlapping regions and compute the first three moments of the color distributions in each image. We will discuss each approach in turn.

Hsu [5] begins by selecting a set of representative colors from the image. Next, the image is partitioned


Histogram: 198. Centering refinement: 42. CCV: 33. Successive refinement: 6.

Histogram: 78. Centering refinement: 54. CCV: 12. Successive refinement: 7.


Histogram: 119. Centering refinement: 60. CCV: 36. Successive refinement: 25.


Histogram: 38. Centering refinement: 17. CCV: 4. Successive refinement: 1.

Figure 2: Example queries with their partner images, plus ranks under various methods. Lower ranks indicate better performance.


Histogram: 88. Centering refinement: 35. CCV: 20. Successive refinement: 13.


Histogram: 310. Centering refinement: 214. CCV: 177. Successive refinement: 160.


Histogram: 411. Centering refinement: 282. CCV: 84. Successive refinement: 56.


Histogram: 50. Centering refinement: 37. CCV: 27. Successive refinement: 22.

Figure 3: Additional example queries with ranks. Lower ranks indicate better performance.

into rectangular regions, where each region is predominantly a single color. The partitioning algorithm makes use of maximum entropy. The similarity between two images is the degree of overlap between regions of the same color. Hsu presents results from a database with 260 images, which show that their approach can give better results than color histograms.

While the authors do not report running times, it appears that Hsu's method requires substantially more computation than the approach we describe. A CCV can be computed in a single pass over the image, with a small number of operations per pixel. Hsu's partitioning algorithm in particular appears much more computationally intensive than our method. Hsu's approach can be extended to be independent of orientation and position, but the computation involved is quite substantial. In contrast, our method is naturally invariant to orientation and position.

Rickman and Stonham [8] randomly sample pixel triples arranged in an equilateral triangle with a fixed side length. They use 16 levels of color hue, with non-uniform quantization. Approximately a quarter of the pixels are selected for sampling, and their method stores 372 bits per image. They report results from a database of 100 images.

Smith and Chang's algorithm also partitions the image into regions, but their approach is more elaborate than Hsu's. They allow a region to contain multiple different colors, and permit a given pixel to belong to several different regions. Their computation makes use of histogram back-projection [12] to back-project sets of colors onto the image. They then identify color sets with large connected components.

Smith and Chang's image database contains 3,100 images. Again, running times are not reported, although their algorithm does speed up back-projection queries by pre-computing the back-projections of certain color sets. Their algorithm can also handle certain kinds of queries that our work does not address; for example, they can find all the images where the sun is setting in the upper left part of the image.

Stricker and Dimai [10] compute moments for each channel in the HSV colorspace, where pixels close to the border have less weight. They store 45 floating point numbers per image. Their distance measure for two regions is a weighted sum of the differences in each of the three moments. The distance measure for a pair of images is the sum of the distance between the center regions, plus (for each of the 4 side regions) the minimum distance of that region to the corresponding region in the other image, when rotated by 0, 90, 180 or 270 degrees. Because the regions overlap, their method is insensitive to small rotations or translations. Because they explicitly handle rotations of 0, 90, 180 or 270 degrees, their method is not affected by these particular rotations. Their database contains over 11,000 images, but the performance of their method is only illustrated on 3 example queries. Like Smith and Chang, their method is designed to handle certain kinds of more complex queries that we do not consider.

6 Extensions

There are a number of ways in which our histogram refinement could be extended and improved. One generalization is to further subdivide split histograms based on additional features; we refer to this process as *successive refinement*. Another extension centers on improving the choice of colorspace.

6.1 Successive refinement

In *successive refinement* the buckets in a split histogram are further subdivided based on additional features. Much as we distinguish between pixels of similar color by coherence, we can distinguish between pixels of similar coherence by some additional feature. We can apply this method repeatedly; each refinement imposes an additional constraint on what it means for two pixels to be similar.

We have implemented a simple successively refined histogram. A color histogram was first split with coherence constraints (to create a CCV). Successive refinement was enforced on both the coherent and incoherent pixels of the CCV. We used the centering refinement introduced in section 3. With successive refinement, pixels are divided into four classes based on coherence versus incoherence, and on whether or not they were in the centermost 75% of the image. The L_1 distance was used as a comparison measure. Examples of the successively refined histogram's performance are shown in figures 2 and 3. These preliminary results seem promising.

We have also investigated successive refinement based on intensity gradients. Again, the initial refinement was based on coherence, and the successive refinement was enforced identically on coherent and incoherent pixels. We have further classified pixels based on the gradient magnitude or on the gradient direction. The results we obtained are quite preliminary, but they seem to indicate a statistically significant improvement over CCV's.

The best system of constraints to impose on the image is an open issue. Any combination of features might give effective results, and there are many possible features to choose from. However, it is possible to take advantage of the temporal structure of a successively refined histogram. One feature might serve as a filter for another feature, by ensuring that the second feature is only computed on pixels which already possess the first feature.

For example, the perimeter-to-area ratio can be used to classify the relative shapes of color regions. If we used this ratio as an initial refinement on color histograms, incoherent pixels would result in statistical outliers, and thus give questionable results. This feature is better employed after the coherent pixels have been segregated. Refining a histogram not only makes finer distinctions between pixels, but functions as a statistical filter for successive refinements.

6.2 Choice of colorspace

Many researchers spend considerable effort on selecting a good set of colors. Hsu [5], for example, assumes that the colors in the center of the image are more important than those at the periphery, while Smith and Chang [9] use several different thresholds to

extract colors and regions. A wide variety of different colorspaces have also been investigated for content-based image retrieval, such as the opponent-axis colorspace [12] and the Munsell colorspace [2].

The choice of colorspace is a particularly significant issue for CCV's, since they use the discretized color buckets to segment the image. A perceptually uniform colorspace, such as CIE Lab, should result in better segmentations and improve the performance of CCV's. A related issue is the color constancy problem, which causes objects of the same color to appear rather differently depending upon the lighting conditions. The simplest effect of color constancy is a change in overall image brightness; this is responsible for the negative examples obtained in our experiments with CCV's. Standard histogramming methods are sensitive to image gain. More sophisticated methods, such as color ratio histograms [3] or the use of color moments [10], might alleviate this problem. These methods, like most proposed improvements to color histograms, can also be used in histogram refinement. For example, color moments could be computed separately for coherent and incoherent pixels.

7 Conclusions

We have described a method for imposing additional constraints on histogram based matching called histogram refinement. This idea can be extended by placing further constraints on the split histogram itself. Both histogram refinement and successive refinement are general methods for improving the performance of histogram based matching. If the initial histogram is a color histogram, and it is refined based on coherence, then the resulting split histogram is a CCV. But there is no requirement that this refinement be based on coherence, or even that the initial histogram be based on color.

Most research in content-based image retrieval has focused on query by example (where the system automatically finds images similar to an input image). However, other types of queries are also important. For example, it is often useful to search for images in which a subset of another image (e.g. a particular object) appears. This would be particularly useful for queries on a database of videos. One approach to this problem might be to generalize histogram backprojection [12] to separate pixels based on spatial coherence, or some other local property.

It is clear that larger and larger image databases will demand more complex similarity measures. This added time complexity can be offset by using efficient, coarse measures that prune the search space by removing images which are clearly not the desired answer. Measures which are less efficient but more effective can then be applied to the remaining images. Baker and Nayar [1] have begun to investigate similar ideas for pattern recognition problems. To effectively handle large image databases will require a balance between increasingly fine measures (such as histogram refinement) and efficient coarse measures.

Acknowledgments

We wish to thank Virginia Ogle for giving us access to the Chabot imagery, and Thorsten von Eicken for supplying additional images. Greg Pass has been supported by Cornell's Alumni-Sponsored Undergraduate Research Program. We also thank Vera Kettnaker and Justin Miller for helping produce the data.

References

[1] Simon Baker and Shree Nayar. Pattern rejection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 544–549, 1996.

[2] M. Flickner *et al.* Query by image and video content: The QBIC system. *IEEE Computer*, 28(9):23–32, September 1995.

[3] Brian V. Funt and Graham D. Finlayson. Color constant color indexing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):522–529, May 1995.

[4] J. Hafner, H. Sawhney, W. Equitz, M. Flickner, and W. Niblack. Efficient color histogram indexing for quadratic form distance functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(7):729–736, July 1995.

[5] Wynne Hsu, T. S. Chua, and H. K. Pung. An integrated color-spatial approach to content-based image retrieval. In *ACM Multimedia Conference*, pages 305–313, 1995.

[6] Virginia Ogle and Michael Stonebraker. Chabot: Retrieval from a relational database of images. *IEEE Computer*, 28(9):40–48, September 1995.

[7] Alex Pentland, Rosalind Picard, and Stan Sclaroff. Photobook: Content-based manipulation of image databases. *International Journal of Computer Vision*, 18(3):233–254, June 1996.

[8] Rick Rickman and John Stonham. Content-based image retrieval using color tuple histograms. *SPIE proceedings*, 2670:2–7, February 1996.

[9] John Smith and Shih-Fu Chang. Tools and techniques for color image retrieval. *SPIE proceedings*, 2670:1630–1639, February 1996.

[10] Markus Stricker and Alexander Dimai. Color indexing with weak spatial constraints. *SPIE proceedings*, 2670:29–40, February 1996.

[11] Markus Stricker and Michael Swain. The capacity of color histogram indexing. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 704–708, 1994.

[12] Michael Swain and Dana Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.