Comparing Images Using Color Coherence Vectors

Greg Pass
Ramin Zabih*
Justin Miller

Computer Science Department
Cornell University
Ithaca, NY 14853

Abstract

Color histograms are used to compare images in many applications. Their advantages are efficiency, and insensitivity to small changes in camera viewpoint. However, color histograms lack spatial information, and this can cause images with very different appearances to have similar histograms. For example, a picture of fall foliage might contain a large number of scattered red pixels; this could have a similar color histogram to a picture of a single large red object. We describe a histogram-based method for comparing images that incorporates spatial information. While a color histogram counts the number of pixels with a given color, a color coherence vector (CCV) measures the spatial coherence of the pixels with a given color. If the red pixels in an image are members of large red regions, this color will have high coherence, while if the red pixels are widely scattered it will have low 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 for the images with the most similar CCV's in under 2 seconds. We demonstrate that CCV's can be used to distinguish objects whose color histograms are indistinguishable. We show that CCV's can give superior results to color histograms for image retrieval.

1 Introduction

Many applications require simple methods for comparing pairs of images based on their overall appearance. For example, a user may wish to retrieve all images similar to a given image from a large database of images. Color histograms are a popular solution to this problem, and are used in systems like QBIC [3] and Chabot [9]. Color histograms are computationally efficient, and generally insensitive to small changes in camera position.

*To whom correspondence should be addressed

Color histograms also have some limitations. A color histogram provides no spatial information; it merely describes which colors are present in the image, and in what quantities. In addition, color histograms are sensitive to both compression artifacts and camera auto-gain.

In this paper we describe a color-based method for comparing images which is similar to color histograms, but which also takes spatial information into account. We begin with a review of color histograms. In section 3 we describe *color coherence vectors* (CCV's) and how to compare them. We compare our method with some recent algorithms [6, 13] that also combine spatial information with color histograms. Section 4 provides examples of CCV-based image queries and shows that they can give superior results to color histograms.

2 Color Histograms

Color histograms are frequently used to compare images. Examples of their use in multimedia applications include scene break detection [18, 5, 1, 8, 10] and querying a database of images [11, 9, 3, 2]. Their popularity stems from several factors.

- Color histograms are computationally trivial to compute.
- Small changes in camera viewpoint tend not to effect color histograms.
- Different objects often have distinctive color histograms.

A number of papers have been published concerning color histograms. For example, Swain and Ballard [15] describe the use of color histograms for identifying objects. Hafner *et al.* [4] provide an efficient method for weighted-distance indexing of color histograms. Stricker and Swain [14] analyze the information capacity of color histograms, as well as their sensitivity.

2.1 Definitions

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, and a few of the most significant bits are used from each color channel. For example, Zhang [18] uses the 2 most significant bits of each color channel, for a total of $n = 64$ buckets in the histogram.

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 $H_{I'}$. Typically color histograms are compared using the sum of squared differences (L_2-distance) or the sum of absolute value of differences (L_1-distance). So the most similar image to I would be the image I' minimizing

$$\|H_I - H_{I'}\| = \sum_j^n (H_I[j] - H_{I'}[j])^2,$$

for the L_2-distance, or

$$|H_I - H_{I'}| = \sum_j^n |H_I[j] - H_{I'}[j]|,$$

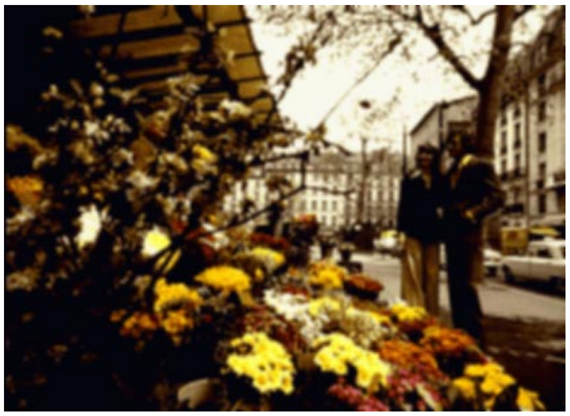
for the L_1-distance. The L_2-distance is at most $\sqrt{2}M$, and the L_1-distance at most $2M$.

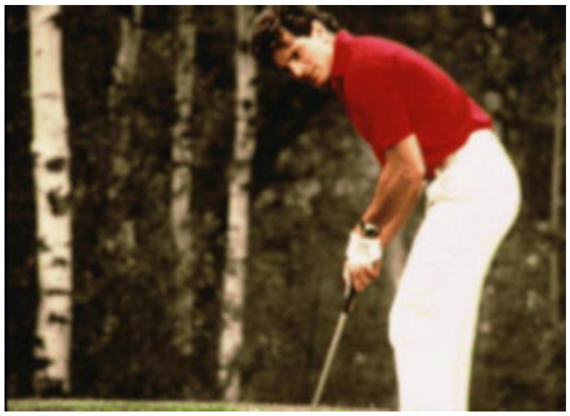
Several authors have considered alternative colorspaces to RGB. For example, Swain and Ballard [15] make use of the opponent-axis colorspace, while QBIC [3] uses the Munsell colorspace. We have experimented with a few different colorspaces, but in our experience the differences are not significant.

3 Color Coherence Vectors

Intuitively, we define a color's *coherence* as the degree to which pixels of that color are members of large similarly-colored regions. We refer to these significant regions as *coherent regions*, and observe that they are of significant importance in characterizing images.

For example, the images below have similar color histograms, despite their rather different appearances. The color red appears in both images in approximately the same 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.





Our coherence measure classifies pixels as either coherent or incoherent. Coherent pixels are a part of some sizable contiguous region, while incoherent pixels are not. A *color coherence vector* represents this classification for each color in the image. This notion of coherence allows us to make fine distinctions that cannot be made with simple color histograms.

3.1 Computing CCV's

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'. (Formally, a path in C is a sequence of pixels $p = p_1, p_2, \ldots, p_n = p'$ such that each pixel p_i is in C and any two sequential pixels p_i, p_{i+1} are adjacent to each other. We consider two pixels to be adjacent if one pixel is among the eight closest neighbors of the other; in other words, we include diagonal neighbors.) Note that we only compute connected components within a given discretized color bucket.

Connected components can be computed in a single pass over the image (see, for example, [16]). When this is complete, each pixel will belong to exactly one connected component. 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

$$(\alpha_j, \beta_j)$$

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 to date, we have used $\tau = 25$ pixels, although nearby values of τ appear to give similar results. The colorspace we use has $n = 64$ buckets evenly distributed in RGB space, just as Zhang [18] does. Images are scaled to contain $M = 38,976$ pixels. Note that these parameters are not independent of each other (in particular, τ and M are interdependent).

3.1.1 An example CCV

We next demonstrate the computation of a CCV. To keep our example small, we will let $\tau = 4$ and assume that we are dealing with an image in which all 3 color components have the same value at every pixel (in the RGB colorspace this would represent a grayscale image). This allows us to represent a pixel's color with a single number (i.e., the pixel with R/G/B values 12/12/12 will be written as 12).

Suppose that after we slightly blur the input image, the resulting intensities are as follows.

$$\begin{array}{cccccc}
22 & 10 & 21 & 22 & 15 & 16 \\
24 & 21 & 13 & 20 & 14 & 17 \\
23 & 17 & 38 & 23 & 17 & 16 \\
25 & 25 & 22 & 14 & 15 & 14 \\
27 & 22 & 12 & 11 & 17 & 18 \\
24 & 21 & 10 & 12 & 15 & 19
\end{array}$$

Let us discretize the colorspace so that bucket 1 contains intensities 10 through 19, bucket 2 contains 20 through 29, etc. Then after discretization we obtain

$$\begin{array}{cccccc}
2 & 1 & 2 & 2 & 1 & 1 \\
2 & 2 & 1 & 2 & 1 & 1 \\
2 & 1 & 3 & 2 & 1 & 1 \\
2 & 2 & 2 & 1 & 1 & 1 \\
2 & 2 & 1 & 1 & 1 & 1 \\
2 & 2 & 1 & 1 & 1 & 1
\end{array}$$

The next step is to compute the connected components. Individual components will be labeled with letters ($A, B, \ldots$) and we will need to keep a table which maintains the discretized color associated with each label, along with the number of pixels with that label. Of course, the same discretized color can be associated with different labels if multiple contiguous regions of the same color exist. The image may then become

$$\begin{array}{cccccc}
B & C & B & B & A & A \\
B & B & C & B & A & A \\
B & C & D & B & A & A \\
B & B & B & A & A & A \\
B & B & A & A & A & A \\
B & B & A & A & A & A
\end{array}$$

and the connected components table will be

Label	A	B	C	D
Color	1	2	1	3
Size	17	15	3	1

The components A and B have more than τ pixels, and the components C and D less than τ pixels. Therefore the pixels in A and B are classified as coherent, while the pixels in C and D are classified as incoherent.

A given color bucket may thus contain only coherent pixels (as does 2), only incoherent pixels (as does 3), or a mixture of coherent and incoherent pixels (as does 1). The CCV for this image will be

Color	1	2	3
α	17	15	0
β	3	0	3

If we assume there are only 3 possible discretized colors, the CCV can also be written

$$\langle (17,3), (15,0), (0,3) \rangle .$$

3.2 Comparing CCV's

Consider two images I and I', together with their CCV's G_I and $G_{I'}$, and let the number of coherent pixels in color bucket j be α_j (for I) and α'_j (for I'). Similarly, let the number of incoherent pixels be β_j and β'_j. So

$$G_I = \langle (\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n) \rangle$$

and

$$G_I = \langle (\alpha'_1, \beta'_1), \ldots, (\alpha'_n, \beta'_n) \rangle$$

Color histograms will compute the difference between I and I' as

$$\Delta_H = \sum_{j=1}^{n} \left| (\alpha_j + \beta_j) - (\alpha'_j + \beta'_j) \right| . \tag{1}$$

Our method for comparing is based on the quantity

$$\Delta_G = \sum_{j=1}^{n} \left| (\alpha_j - \alpha'_j) \right| + \left| (\beta_j - \beta'_j) \right| . \tag{2}$$

From equations 1 and 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', i.e.

$$\alpha_j + \beta_j = \alpha'_j + \beta'_j,$$

but these pixels may be entirely coherent in I and entirely incoherent in I'. In this case $\beta_j = \alpha'_j = 0$, and while $\Delta_H = 0$, Δ_G will be large.

In general, $\Delta_H \leq \Delta_G$. This is true even if we use squared differences instead of absolute differences in the definitions of Δ_H and Δ_G. This is because both $d(x) = |x|$ and $d(x) = x^2$ are metrics, which implies that they satisfy the triangle inequality

$$d(x+y) \leq d(x) + d(y). \tag{3}$$

If we rearrange the terms in equation 1 we get

$$\Delta_H = \sum_{j=1}^{n} \left| (\alpha_j - \alpha'_j) + (\beta_j - \beta'_j) \right| .$$

Applying the triangle inequality we have

$$\Delta_H \leq \sum_{j=1}^{n} \left| (\alpha_j - \alpha'_j) \right| + \left| (\beta_j - \beta'_j) \right| = \Delta_G .$$

While our method for comparing two CCV's is based on Δ_G, we also add a normalization step. Without normalization, the distance between the coherence pairs $(0,1)$ and $(0,100)$ is as large as the distance between $(9000,9001)$ and $(9000,9100)$. This is counter-intuitive, because the second difference is fairly small given the number of pixels.

The normalized difference between G_I and $G_{I'}$ is

$$\Delta \quad = \quad \sum_{j=1}^{n} \left| \frac{\alpha_j - \alpha'_j}{\alpha_j + \alpha'_j + 1} \right| + \left| \frac{\beta_j - \beta'_j}{\beta_j + \beta'_j + 1} \right| . \tag{4}$$

The denominators normalize these differences with respect to the total number of pixels. The factor of $+1$ is used to avoid division by zero when the α's or β's are zero.

3.3 Related work

Recently, several authors have proposed algorithms for comparing images that combine spatial information with color histograms. Hsu *et al.* [6] attempts to capture the spatial arrangement of the different colors in the image, in order to perform more accurate content-based image retrieval . Smith and Chang [13], in contrast, are interested in answering queries that combine spatial information with color. While these authors share some motivation with our work, their approaches are technically quite different from ours. We will discuss each in turn.

Hsu begins by selecting a set of representative colors from the image. Next, the image is partitioned 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 querying a database with 260 images, which show that the integrated 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. Our approach can be done 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.

Smith and Chang's algorithm also partitions the image into regions, but his approach is more elaborate. They allow a region to contain multiple different colors, and permits a given pixel to belong to several different regions. Their computation makes use of histogram back-projection [15] 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.

Both of these authors spend considerable effort on selecting a good set of colors. For example, Hsu assumes that the colors in the center of the image are more important than those at the periphery, while Smith and Chang use several different thresholds to extract colors and regions. In contrast, for our experiments we have simply used the RGB colorspace. It is possible that our results could be improved by a more sophisticated choice of colorspace, but we have obtained good results with our straightforward scheme.

4 Performance

We have implemented color coherence vectors, and have used them for image retrieval from a large database. We begin with a query image I, and rank the images in the database in order of decreasing CCV similarity to I. For comparison, we have also implemented color histograms (including several common variations).

Our database consists of 14,554 images, which are drawn from a variety of sources. Our largest sources include the 1,440 images used in QBIC [3], the 11,667 images used in Chabot [9], 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 using the method described in [17]. We have added one or two images from each scene to the database, totaling 349 images (this process is sometimes called "storyboarding" or "keyframing"). The image database thus contains a wide variety of imagery. Image sizes range from 156×88 to 735×478 pixels. In addition, some of the imagery has undergone substantial lossy compression via JPEG or MPEG.

We have compared our results with a number of variations of color histograms. We have experimented with both the L_1 and L_2 distances, with $n = 64$ color buckets. In addition, we have implemented Swain's opponent axis colorspace [15], and used the color discretization scheme he describes. Finally, we have used a finer version of the RGB colorspace, with 3 bits per color (a total of $n = 512$ buckets). We added a small amount of smoothing before discretization, because it improved the performance of these algorithms on our data.

Like Hsu [6] we executed a number of queries where there are "right" answers in the image database. For example, figure 3 shows six images taken from a single MPEG sequence via keyframing. Ideally, when we query the image databases for images that are similar to the first of these images, the remaining five images from this movie should appear near the top. We will call these *category queries*.

We have performed 6 category queries on our database. The categories are listed in table 1. Some examples from another category query are shown in figure 4.

We measure performance using two standard metrics from the Information Retrieval literature [12], namely *recall* and *precision*. If we define the images in a given query category as the relevant images, then recall and precision are defined as follows.

$$\begin{aligned} \text{recall} &= \frac{\text{relevant retrieved}}{\text{all relevant}} \\ \text{precision} &= \frac{\text{relevant retrieved}}{\text{all retrieved}} \end{aligned}$$

Query	Category	# Images
1	Owls	22
2	Bird in Cage	10
3	Foxes	17
4	Movie Scene	6
5	Moving Car	6
6	Raging Stream	7
–	Distractors	14,486

Table 1: Images used for category queries

For example, suppose we perform a query with the first movie image from figure 3 as our query image. Using CCV's, the other five images from the category appear as numbers 1, 2, 3, 4 and 6 (i.e., as the most similar image in the database, the second most similar image in the database, etc). Using color histograms, the other five images from the category appear as numbers 20, 74, 184, 632, and 1340. For this query category a recall level of .2 corresponds to retrieving 1 of the 5 images in the category. At this recall level, CCV's yield a precision of 1.0, while color histograms yield a precision of 0.05.

When comparing algorithms for image retrieval, the higher the precision the better. Table 2 shows the performance of CCV-based image retrieval on our six category queries. All the variants of color histogramming gave similar performance. We show results for the L_1 distance and for the opponent colorspace, since these gave the best performance on our data. The highest numbers in the table are shown in bold. Except for Query #3 at a high level of recall, where the opponent colorspace performs better, CCV's give the best results.

We also measure performance with a new metric, which captures how many correct matches are in the set of images the user is willing to browse. For example, it is unlikely the user will consider more than 100 potential images as correct answers. We define *scope* as the n most similar images the user commits to manually searching after placing a query. For a given scope, we can consider the fraction of correct matches the user will find (recall).

Table 3 shows the performance of CCV-based image retrieval on our six category queries, in terms of scope. Again, several variants of color histograms are shown for comparison. By this measure, CCV's are give the best results for all queries.

4.1 Examples

Figure 1 shows an example of a query where color histograms and CCV's give very different results. Figure 1(a) shows the query image. The most similar image to this in our database, according to color histograms[1] is shown in figure 1(b). The image with the most similar CCV (i.e., the smallest value of Δ) is shown in figure 1(c). Note that while this image shows the same object as the query image, the camera position has changed substantially. The image shown in figure 1(c) is ranked #141 by color histograms. All three of these images

[1]in this example, the L_2 distance

Recall	Precision								
	Query 1			*Query 2*			*Query 3*		
	CCV	Hist	Opp	CCV	Hist	Opp	CCV	Hist	Opp
0.25	**1.0**	.043	.625	**.667**	.023	.250	**.571**	.235	.108
0.50	**1.0**	.018	.052	**.078**	.022	.005	**.471**	.034	.104
0.75	**.600**	.020	.010	**.081**	.008	.003	.019	.015	**.051**
0.90	**.514**	.021	.011	**.044**	.009	.004	.007	.006	**.017**

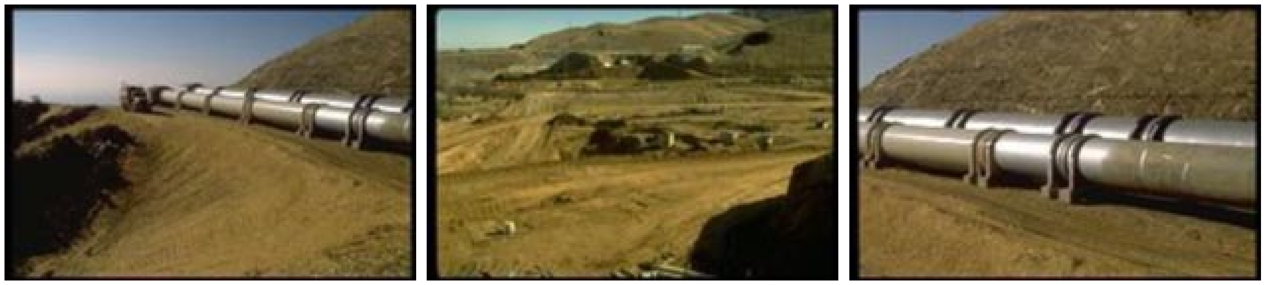
Recall	Precision								
	Query 4			*Query 5*			*Query 6*		
	CCV	Hist	Opp	CCV	Hist	Opp	CCV	Hist	Opp
0.25	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	.333	**1.0**	.500	**1.0**
0.50	**1.0**	.111	**1.0**	**1.0**	.017	.014	**1.0**	.010	.014
0.75	**1.0**	.043	**1.0**	**1.0**	.025	.012	**.500**	.002	.002
0.90	**1.0**	.027	.029	**.667**	.007	.003	**.250**	.002	.002

Table 2: Recall *versus* precision results for category queries. Larger numbers indicate better performance.

Scope	Recall								
	Query 1			*Query 2*			*Query 3*		
	CCV	Hist	Opp	CCV	Hist	Opp	CCV	Hist	Opp
10	**.476**	.143	.238	**.222**	.111	**.222**	**.294**	.118	.176
25	**.714**	.190	.333	**.222**	.111	**.222**	**.529**	.294	.176
50	**.905**	.190	.429	**.333**	.111	.222	**.529**	.294	.294
100	**.952**	.190	.429	**.444**	.222	.222	**.588**	.294	.529

Scope	Recall								
	Query 4			*Query 5*			*Query 6*		
	CCV	Hist	Opp	CCV	Hist	Opp	CCV	Hist	Opp
10	**1.0**	.200	.600	**.800**	.200	.200	**.667**	.167	.167
25	**1.0**	.400	.600	**.800**	.200	.200	**.833**	.333	.167
50	**1.0**	.400	.600	**.800**	.200	.200	**1.0**	.333	.167
100	**1.0**	.600	.600	**.800**	.200	.200	**1.0**	.333	.167

Table 3: Scope *versus* recall results for category queries. Larger numbers indicate better performance.



(a) Query image (b) Histogram best match (c) CCV best match *Histogram rank = 141*

Figure 1: Sample query results

have similar color compositions; they are mostly brown, with some blue and some silver.

Figure 2 shows another such example. Figure 2(a) shows the query image. The image with the most similar color histogram in our database is shown in figure 2(b), while the image with the most similar CCV is shown in figure 2(c). This is a picture of the same object translated downward; it is ranked #17 by color histograms. The false match found by color histograms (figure 2(b)) is again of similar colors, but the spatial layout is quite different. For example, in images 2(a) and 2(c), the black pixels are part of large regions, while in figure 2(b) they are scattered.

We have provided a number of additional examples where CCV's perform better than color histograms. Figures 5, 6 and 7 show these examples. In each case, we display the query image, the image with the most similar color histogram (L_2 distance), and the image with the most similar CCV. We also indicate where color histograms would rank the CCV image.

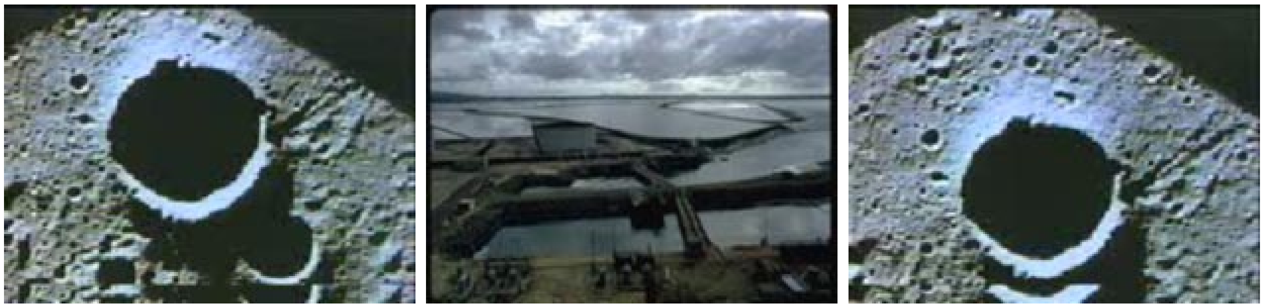
Color images, unfortunately, do not show up very well when printed. The digital images used in these queries, as well as additional examples, can be found from the Web in `http://www.cs.cornell.edu/home/rdz/ccv.html`.

4.2 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, the images must be sorted in order of decreasing similarity. Note that most methods for content-based indexing have these distinct phases, including color histogramming and the methods of Hsu *et al.* and Smith and Chang.

We ran our experiments on a 50 MHz SPARCstation 20, and provide the results from color histogramming for comparison. We have expended a reasonable amount of effort on the implementation of CCV's and of color histograms. Although the implementations could be improved (for example, by using a real database instead of simply storing images and indexes in the filesystem), we have done all the simple optimizations we could.

The following table summarizes the costs for color histogram comparison and for CCV-



(a) Query image (b) Histogram best match (c) CCV best match *Histogram rank = 17*

Figure 2: Another query result

based comparison, both when an image is inserted into the database and when a query is performed. The images used for benchmarking are 232×168.

Method	Insertion	Querying
Color histograms	66.7 images/sec	21940 comparisons/sec
Color coherence vectors	5.03 images/sec	7746 comparisons/sec

5 Conclusions

There are a number of ways in which our algorithm could be extended and improved. For example, the color constancy problem [7] causes objects of the same color to appear rather differently, depending upon the lighting conditions. It is possible that a more sophisticated colorspace, such as HSV, could alleviate this limitation.

Our approach can also be extended to quantities beyond color and coherence. We are investigating *histogram splitting*, in which the pixels of a given color bucket are subdivided and compared based on particular features (coherence, texture, etc.). We are also interested in extending our work to include local queries, such that we can search for images in which a subset of another image (e.g. a particular object) appears.

For sufficiently large imagebases, any single measure is unlikely to succeed. This suggests a more interactive approach to database querying. For example, after the initial query, the user could label the images within his scope as “good” or “bad”. The search engine could then dynamically reorder the queries, based on a growing stock of information about the user’s needs.

Acknowledgements

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, and Justin Miller has been supported by a grant from the GTE Research Foundation.

References

[1] Farshid Arman, Arding Hsu, and Ming-Yee Chiu. Image processing on compressed data for large video databases. In *ACM Multimedia Conference*, pages 267–272, 1993.

[2] M. G. Brown, J. T. Foote, G. J. F. Jones, K. Sparck Jones, and S. J. Young. Automatic content-based retrieval of broadcast news. In *ACM Multimedia Conference*, 1995.

[3] M. Flickner *et al.* Query by image and video content: The QBIC system. *IEEE Computer*, 28(9):23–32, September 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] Arun Hampapur, Ramesh Jain, and Terry Weymouth. Production model based digital video segmentation. *Journal of Multimedia Tools and Applications*, 1:1–38, March 1995.

[6] 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.

[7] E. H. Land and J. J. McCann. Lightness and Retinex theory. *Journal of the Optical Society of America*, 61(1):1–11, 1971.

[8] Akio Nagasaka and Yuzuru Tanaka. Automatic video indexing and full-video search for object appearances. In *2nd Working Conference on Visual Database Systems*, October 1991.

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

[10] K. Otsuji and Y. Tonomura. Projection-detecting filter for video cut detection. *Multimedia Systems*, 1:205–210, 1994.

[11] Alex Pentland, Rosalind Picard, and Stan Sclaroff. Photobook: Content-based manipulation of image databases. *International Journal of Computer Vision*, to appear.

[12] Gerard Salton. *Automatic Text Processing.* Addison-Wesley, 1989.

[13] John Smith and Shih-Fu Chang. Tools and techniques for color image retrieval. *SPIE proceedings*, 2670, February 1996.

[14] 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.

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

[16] Patrick Winston and Berthold Horn. *Lisp.* Addison-Wesley, second edition, 1984.

[17] Ramin Zabih, Justin Miller, and Kevin Mai. A feature-based algorithm for detecting and classifying scene breaks. In *ACM Multimedia Conference*, pages 189–200, November 1995.

[18] HongJiang Zhang, Atreyi Kankanhalli, and Stephen William Smoliar. Automatic partitioning of full-motion video. *Multimedia Systems*, 1:10–28, 1993.

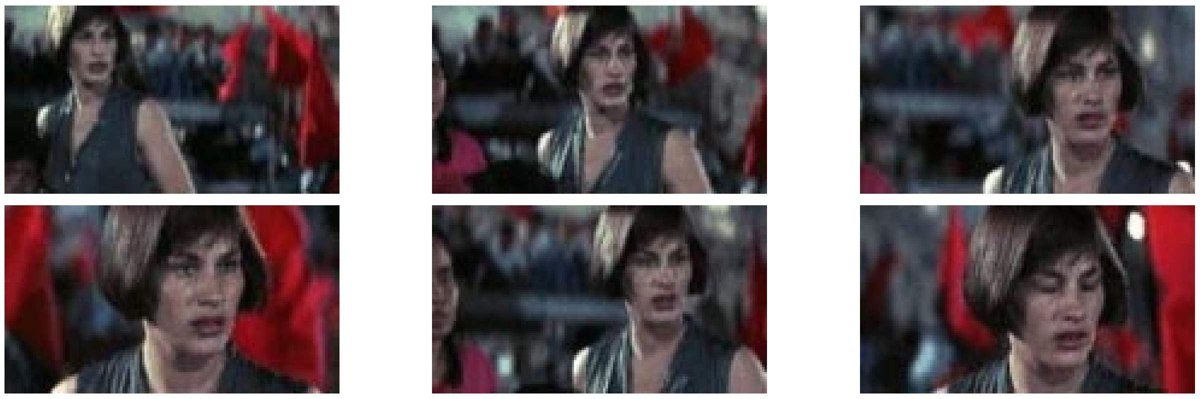


Figure 3: Images from the *Movie Scene* query category.

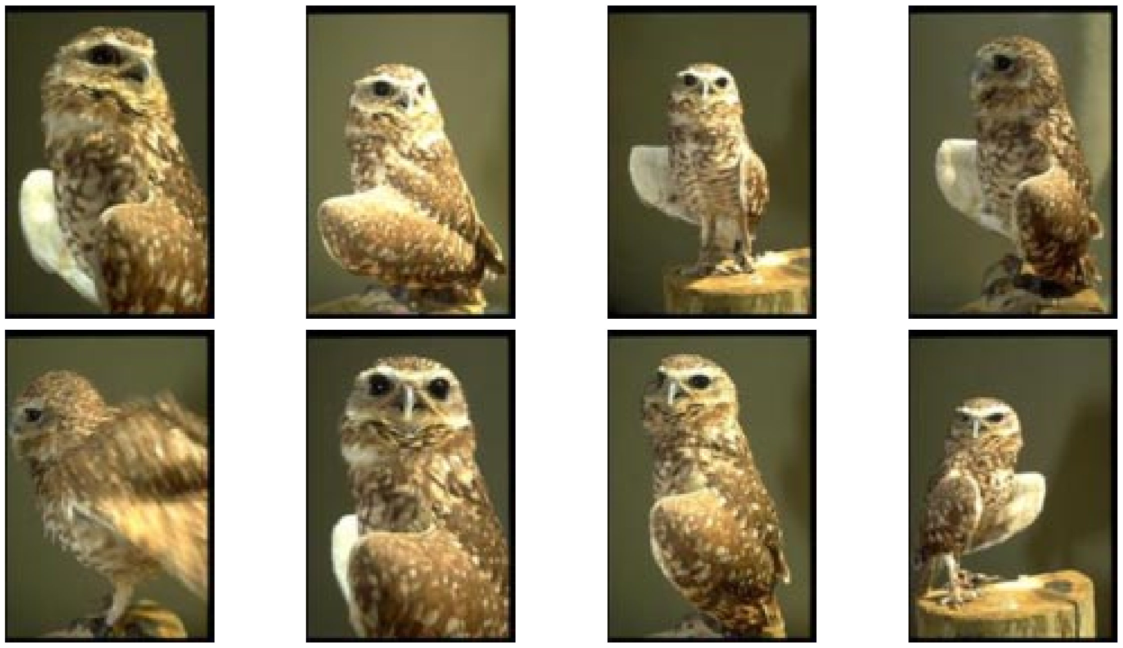


Figure 4: Selected images from the *Owls* query category.

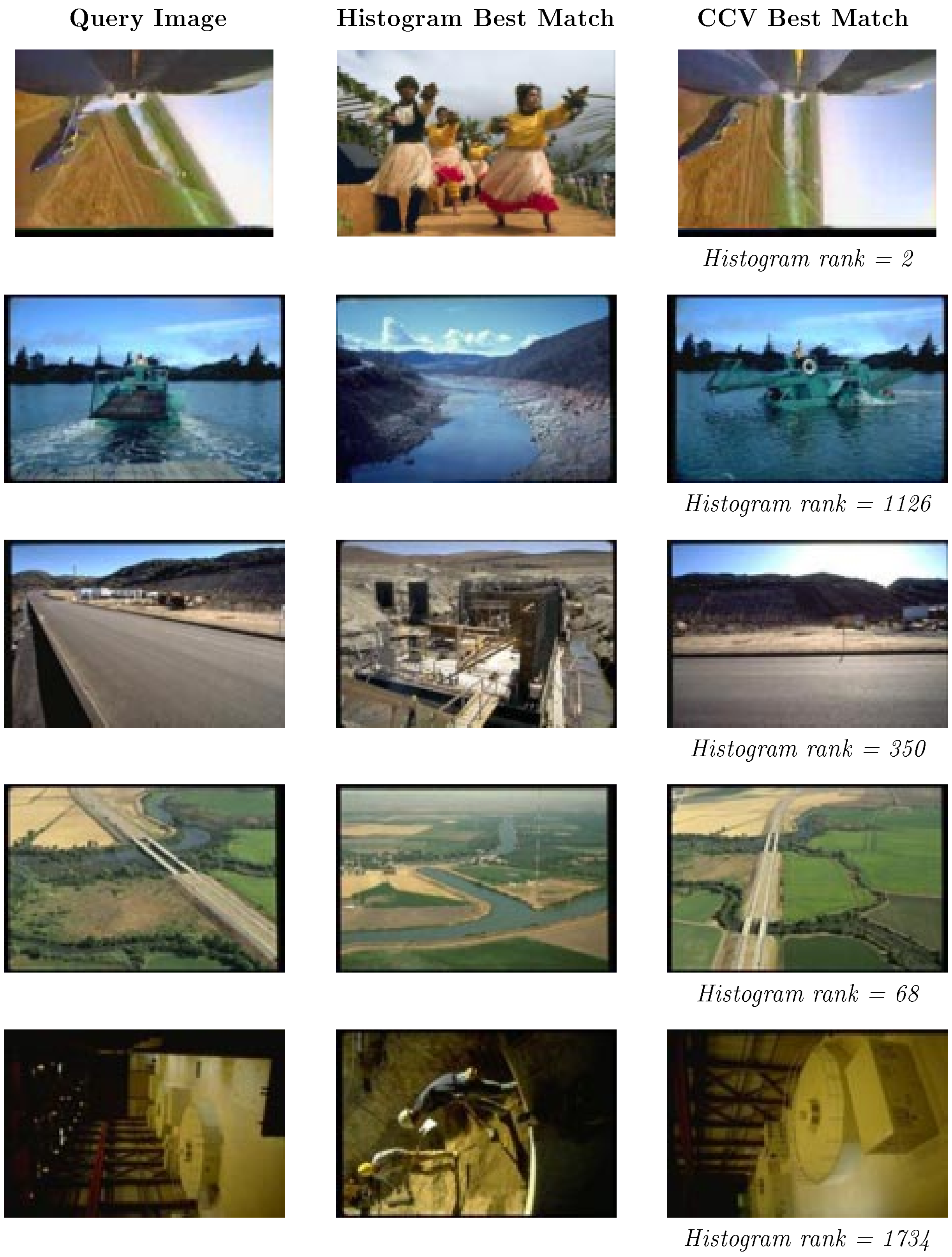


Figure 5: Query results

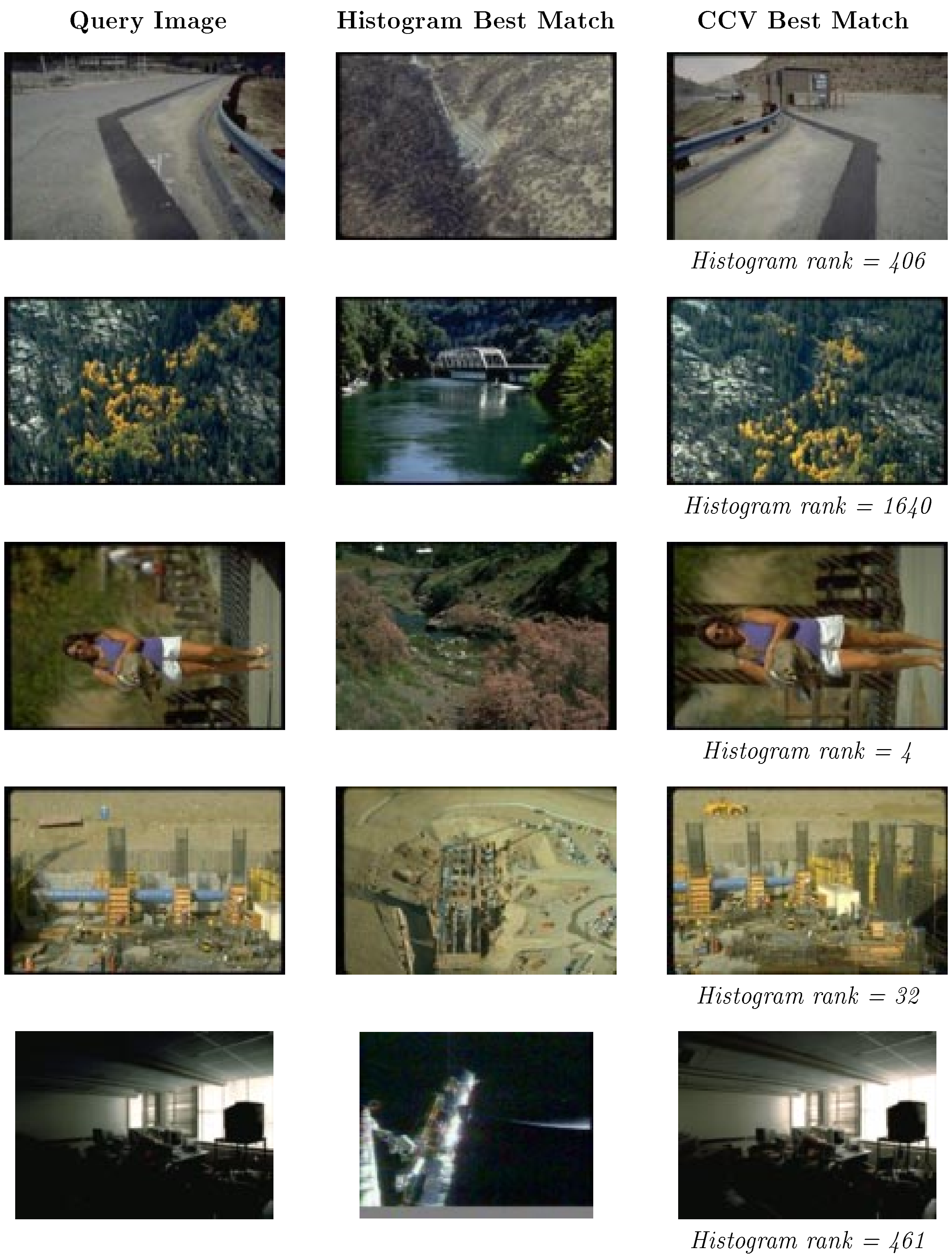


Figure 6: Query results (continued)

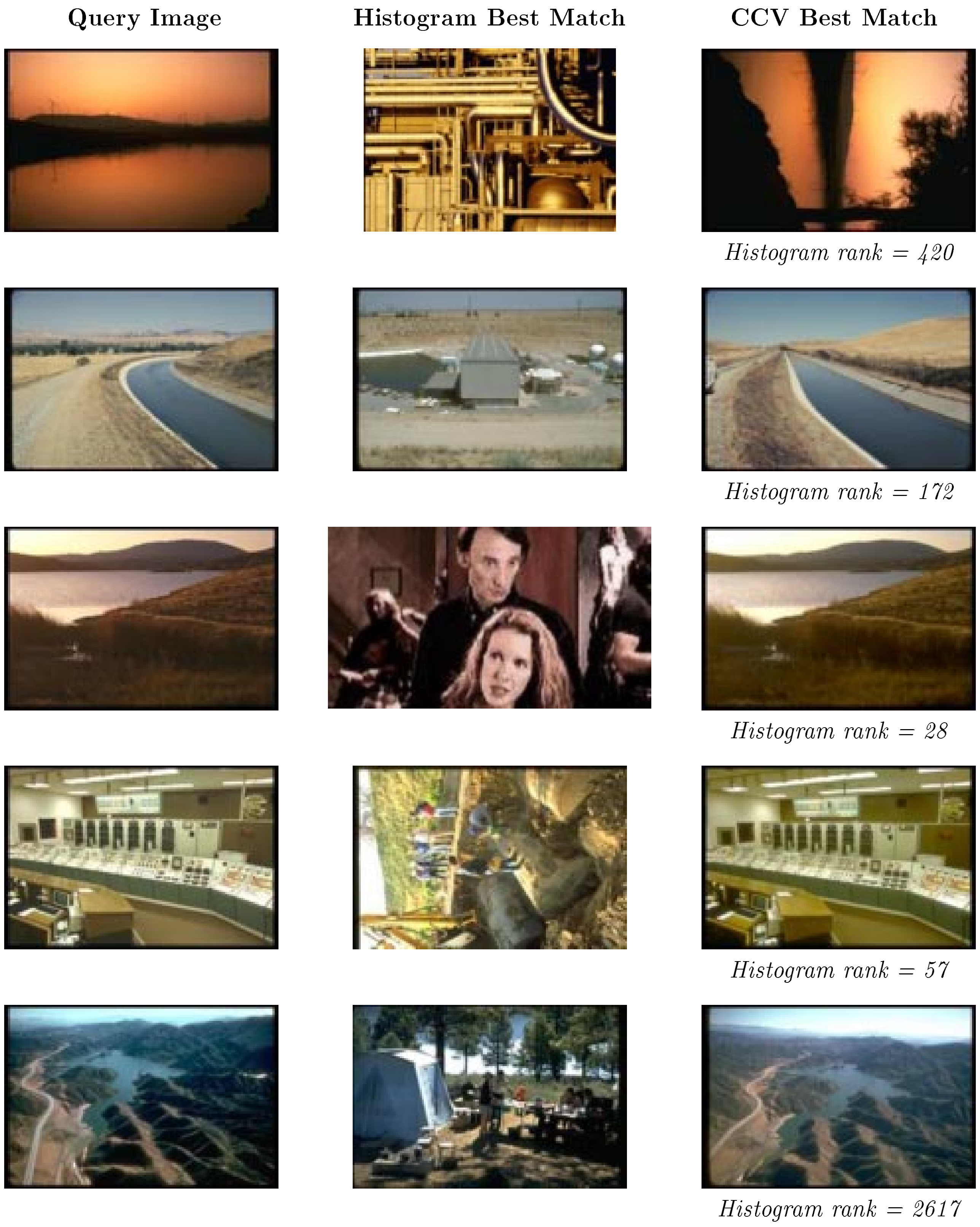


Figure 7: Query results (continued)