Experiments with Image Signatures

Viresh Ratnakar

August 29, 1996

1 Signatures used

In every case, the image was either grayscale, or converted to grayscale from RGB (which is, approximately, $0.3R + 0.6G + 0.1B$). Conversion to grayscale simplifies the problem, and we haven't really lost much information as our goal is only to come up with a rather *general* image-similarity measure. I tried several DCT-based and color-indexing based variants. Wavelets need some thought/discussion, as there are some problems there, but we definitely need to try them too, later.

1.1 DCT-based signatures

All of these are based on distributions of DCT coefficients over all the 8×8 non-overlapping blocks in the image. Further, the coefficients within each block are ordered in the standard zig-zag order (that approximates "importance") used in JPEG/MPEG. Let $C(b, n)$ denote the value of coefficient number n $(0 \leq n \leq 63)$ in block number b (that's about the only notation you'll really need to remember, to understand the signature methods). Each kind of DCT-based signature has 64 numbers, one each for each coefficient, only the first few are actually used to compute distance between signatures.

Ratio

This is the first idea I had proposed. For each coefficient, I capture the distribution with the fraction of blocks in which the coefficient value is above a threshold. Specifically, the n^{th} component of the signature is given by:

$$Ratio(n) = \frac{\#\text{ of blocks in which } |C(b,n)| > H(n)}{\#\text{ of blocks in which } |C(b,n)| > L(n)}$$

Here $L(n)$ is a "significance" threshold—the idea is to ignore coefficients with very low values. I chose $L(n)$ as the number below which most JPEG compressors change the n^{th} coefficient to zero. $H(n)$ was set to $2.5 \times L(n)$. Like I pointed out during our discussions, the distribution of DCT coefficients is nearly Laplacian, and in that case $Ratio(n)$ fully describes the distribution.

StdDev

Uri had wondered about the round-about way the "Ratio" signature captures the distributions, and said that if indeed we are trying to capture the essence of Laplacian distributions ($\frac{\lambda}{2}e^{-\lambda|x|}$), why not use the Laplacian parameter λ itself! This, the "StdDev" signature, uses the standard deviation of each coefficient distribution, and is the usual method for estimating λ. Note that for the Laplacian distribution, standard deviation is $\sqrt{2/\lambda}$.

$$StdDev(n) = \sqrt{\frac{\sum_b C(b,n)^2}{\text{Total } \#\text{ of blocks}}}$$

CutStdDev

To ignore coefficients with very low values, even when using the "StdDev" signature, I modified it a bit as follows: For the n^{th} coefficient, only the blocks where its value is $> L(n)$ are considered, where $L(n)$ is the same threshold that's used in the "Ratio" signature. Call a block b with $|C(b,n)| > L(n)$ as *significant* for coefficient n. Then,

$$\mathit{CutStdDev}(n) = \sqrt{\frac{\sum_{b \text{ significant}} C(b,n)^2}{\text{Total \# of significant blocks}}}$$

My original idea was to ignore the DC coefficient ($n = 0$) in the signatures, in order to ignore the overall brightness level (DC coefficient is just the mean of all pixels in a block). In any case, I tried both leaving it in and leaving it out. Thus, the signatures used in the experiments (listed completely in Figure 1), use coefficients l through h, where l is always either 0 or 1. Distances between DCT-based signatures are just the Euclidean distances.

1.2 Color Indexing (CI)

I read the paper on Color Indexing by Swain and Ballard, from International Journal of Computer Vision, 1991. Their idea is to measure histograms of occurrences of colors, grouped into bins. "Closeness" of two image histograms is measured as the sum of the smaller count in each bin. To compare with the DCT signatures, I wanted to somehow come up with only a few (≤ 64) numbers. Further, I wanted the signature to be such that I could try using only a part of it (say, the first 10 numbers). Note that I anyway have only 256 colors, since I have converted RGB images to grayscale.

I created 64 bins (so a pixel with value x goes to bin number $x/4$), and measured the histogram (normalizing counts to lie in [0..1] by dividing with total number of pixels). I sorted these fractional counts in descending order. Let $b(n)$ denote the bin number that occurs at rank n in this order. The signature consists of these sorted counts, along with their bin numbers:

$$\mathit{CI}(n) = (b(n),\ \text{count in bin number } b(n))$$

Distance between signatures cannot be the simple Euclidean one, as the bin numbers may be different (for example, if we use only $\mathit{CI}(0)$ through $\mathit{CI}(10)$). Here is the distance function, when the signature consists of the top k counts, $\mathit{CI}(0)$ through $\mathit{CI}(k-1)$: Find the list of all the bins involved in the two signatures. Some of these will be common to both, while some won't. For the common bins, add the squared difference between the counts. For bins that occur in only one signature, add the square of the count for the signature in which that bin appears (effectively assuming the count for the other signature to be zero, as it is low enough not to have figured in the top k counts). Divide by the total number of bins in the list, and take square root. Hope I haven't made it sound more complicated than it really is!

Figure 1 shows the list of signature variants I actually used. The first column gives each method a unique number, in the order in which I tried them. This number needs to be used to view the results (sigh) as described later. The second column identifies the kind of signature (Ratio, StdDev, CutStdDev, or CI). The third and fourth columns, respectively, identify the start and end indices of the coefficients (or counts, in case of CI) used.

2 Test images

I started with a set of about 60 images, all grayscale or converted to grayscale. I expanded this set to 110 images, by applying some operations on some of the images. In each case, only one operation was used, i.e., I never used combinations of operations. The operations are:

1. Bright
2. Crop - cut a border off
3. Dim

Number	Kind	Start	End
0	Ratio	0	9
1	Ratio	1	10
2	StdDev	0	9
3	StdDev	1	10
4	CutStdDev	0	9
5	CutStdDev	1	10
6	CI	0	9
7	Ratio	1	15
8	Ratio	1	20
9	StdDev	1	15
10	StdDev	1	20
11	CutStdDev	1	15
12	CutStdDev	1	20
13	CI	0	14
14	CI	0	19
15	Ratio	1	5
16	Ratio	1	8
17	Ratio	1	9

Figure 1: Summary of signature methods used

4. Jpg - compress to JPEG
5. Rot - rotate
6. HistEq
7. Rvid - like taking a negative
8. Enlarge
9. Reduce
10. Sharp
11. Frame - overlay a black border

Almost all of these operations are provided by `xv`. All the images are in the directory `/p/qclic/Code/Signature/Tests/Images`. The images created by the above operations are named as `<original-name>+<operation>`.

For evaluating the various signature methods, I clustered the images together as follows: Each **root** cluster consists of an original image along with all its variants created by the above operations. Each **group** cluster consists of all **root** clusters of original images that are similar subjectively. Basically the **group**s only consist of images from the same source (typically, the same video sequence), except for the largest group, into which I threw in all the facial images.

3 Viewing the results

For each method in Figure 1, you can view the best `N` matches for any image as follows. Go to the directory `/p/qclic/Code/Signature/Tests`. Use the command `ShowImageMatches m id N`, where,

- `m` is the method number (0-17) from Figure 1,

- `id` is either the image number (0-109) or the name of an image (`ls Images/*` will list all possible image names), and,
- `N` is the number of matches you want to see.

A window will pop up, showing the best `N` matches. For example, `ShowImageMatches 1 lena.pgm 20` will show the best 20 matches for lena.pgm, using method number 1 (Ratio, coefficients 1 through 10).

In addition, you can view the results after birch clustering. In this case you are not restricted to just the 18 methods listed in Figure 1. But color indexing results cannot be seen this way, as I don't know how to use birch with non-Euclidean distance measures. The command is `ShowClusters m f l n`, where,

- `m` is the method name: either Ratio or StdDev or CutStdDev (just the first letter in upper or lower case will do),
- `f` is the starting coefficient number: use 0 or 1 (1 if you want to ignore the DC coefficient),
- `l` is the ending coefficient number (less than 64), and,
- `n` is the number of clusters you want birch to create.

If birch has been run with these parameters before, `ShowClusters` will simply read the cluster file, otherwise it will first run birch to create the cluster file (it's very fast—only 110 points). After reading the cluster file, it will, for each cluster, list the names of images in that cluster, and then prompt you to ask whether you want to see the cluster (in which case a window will pop up showing all the images in the cluster).

4 Evaluating the methods

I gave each method a score as follows: for each image, I scan the ordered list of matches until all the images with the same **root** have been seen. Count the number of images *not* in the same **root** but appearing before the last of the **root** -images have been seen. Further, count images with same **group** separately from images in a different **group** . Add 1 to the score for each **group** image appearing before some **root** image, and 2 for each non-**group** image appearing before some **root** image. A low score implies all the images with the same **root** are very close, while a high score says that lots of other images are closer to the image than images with the same **root** . Add the scores for all images to calculate the method score.

For example, let A, B, C, D, A+jpg, A+crop, C+jpg, and D+crop be the images, with **root** clusters being

- A, A+jpg, A+crop,
- B,
- C, C+jpg, and,
- D, D+crop,

and **group** clusters being

- A, A+jpg, A+crop, B,
- C, C+jpg, and,
- D, D+crop,

If the ordered matches for image A+crop are,

1. A+crop
2. C (add 1 to non-**group** count)
3. B (add 1 to **group** count)

Rank	Number	Kind	Start	End	**group** Count	Non-**group** Count	Score	Eq-Wt Score	Eq-Wt Rank
1	1	Ratio	1	10	197	1312	2821	1509	1
2	0	Ratio	0	9	211	1339	2889	1550	2
3	4	CutStdDev	0	9	170	1396	2962	1566	3
4	17	Ratio	1	9	211	1377	2965	1588	4
5	16	Ratio	1	8	205	1424	3053	1629	5
6	2	StdDev	0	9	174	1466	3106	1640	7
7	7	Ratio	1	15	182	1464	3110	1646	8
8	14	CI	0	19	152	1482	3116	1634	6
9	8	Ratio	1	20	200	1498	3196	1698	10
10	13	CI	0	14	152	1527	3206	1679	9
11	6	CI	0	9	155	1553	3261	1708	11
12	10	StdDev	1	20	223	1527	3277	1750	12
13	12	CutStdDev	1	20	165	1602	3369	1767	13
14	15	Ratio	1	5	222	1598	3418	1820	14
15	9	StdDev	1	15	219	1609	3437	1828	15
16	11	CutStdDev	1	15	167	1697	3561	1864	16
17	3	StdDev	1	10	217	1724	3665	1941	18
18	5	CutStdDev	1	10	175	1755	3685	1930	17

Figure 2: Evaluation of signature methods

4. D+crop (add 1 to non-**group** count)
5. A
6. C+jpg (add 1 to non-**group** count)
7. A+jpg (last image with same root as A+crop)
8. D

So the total score for this image is (**group** count) + 2*(non-**group** count) = 1 + 2*2 = 5. The overall score for a method is the sum of its score for all images.

Figure 2 shows the scores for all 18 methods in Figure 1. The first column gives the rank, second column is the method number. Third, fourth, fifth columns are method name, start index, end index respectively (repeated from Figure 1 for convenience). Sixth column gives the total **group** count for all images, while seventh column gives the total non-**group** count. The total score is listed in the eighth column. Since the clustering into **group**s was fairly arbitrary, column nine gives the total score giving equal weight to **group** count and non-**group** count. The last column gives the rank of the method with this scoring (the ranking doesn't really change much).

Method number 1, which is DCT Ratio for coefficients 1 through 10, seems to be the winner with this scoring. I also evaluated the methods from the point of view of each *operation* separately. That will add too many tables here, so I'll just report the winner(s) for each operation in Figure 3. Detailed scores for each method for each operation can be found in the files `/p/qclic/Code/Signature/Tests/Scores/sortedopscores.<op>`.

The color indexing methods (6,13,14) do very well, as expected, with operations that do not change the histogram much (like rotate). Overall, especially for similarity between two images that are not necessarily transformed versions of the same image, method number 1 (Ratio 1-10) seems to give the best performance subjectively. It also has the best overall score.

Of course, we need to discuss these results in greater detail. Just in case we don't meet until October, this is food for thought.

Operation	Winning method number(s)
Bright	0 1 3 5 9 10 11 12 15 16 17
Crop	14
Dim	12
Jpg	2 3 9 10
Rot	6 13 14
HistEq	14
Rvid	1 2 3 5 7 8 9 10 11 12 15 16 17
Enlarge	6 13 14
Reduce	6 13
Sharp	14
Frame	9

Figure 3: Evaluation of signature methods for operations