Modified Eigenimage Algorithm for Painting Image Retrieval

Qun Feng Tan and David Lau
Department of Electrical Engineering, Stanford University

***Abstract* — This paper presents a fast and viable method of painting image retrieval. The painting is first isolated using edge detection and region counting. We use perspective transform and eigenimage analysis to select the desired image.**

I. Introduction

THIS project provides a reliable technique which recognizes paintings, on display in the Cantor Arts Center at Stanford University, based on snapshots taken with a camera-phone. Such a scheme would be useful as part of an electronic museum guide: users would point their camera-phones at a painting of interest and would hear commentary based on the recognition result. The method we are providing strikes a balance between efficiency and accuracy, and is an attractive alternative which is specifically tailored to the image set we are given.

II. Experimental Setup

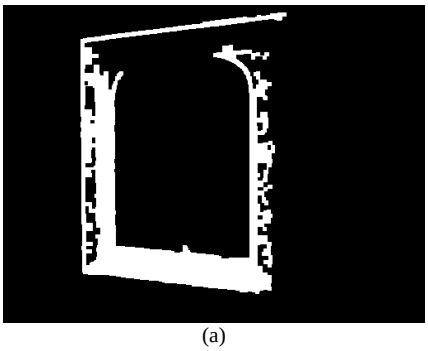
We are given 33 painting photos from the Cantor Arts Center. These 2048x1536 24bits-per-pixel RGB jpeg encoded photos were based on snapshots taken on with a camera-phone. We know several things about the images from the start:

1) The frames on the paintings will be included in the photograph, which can provide us a benchmark for edge detection.
2) There are only these 33 images in the database. This gives us a very restricted set by which we can do some tweaking to customize our algorithm to this small dataset.
3) The desired painting may not be the only feature in the photo. There maybe objects at the side, for example, snippets of other paintings or text box descriptions. We need a way to extract the painting and filter out all extraneous features.

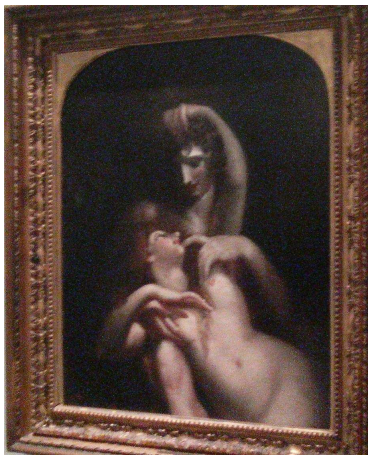
III. Algorithm Description

A. Edge Detection and Corners Detection

Edge detection and corners detection methods have similarities: both make use of discrete differentiation operators



(a)



(b)

(c)

Fig. 1. Result of our edge detection, region labeling and isolation algorithm (a), then overlaid and cropped with the original (b), and the original photo is shown in (c). Note: these images are not to scale.

to calculate local gradient changes. Thus, in order to maximize the program efficiency, one option is to merge edge detection and corners detection.

First we blurred the input images (and also the training images) with a Gaussian filter in order to remove some of the noise. We then applied a Roberts edge detector, which gives us binary images with the detected image. The edge detector mainly detects the frame, since it is the most contrasted part that distinguishes the painting from the wall. We then dilated and eroded the image to connect the disjointed line-segments together.

After we get our edges our largest region algorithm is applied by using region counting and region labeling techniques. We assume that the painting will be the most significant object in the photo and thus have the largest area. After doing this, we will basically get a quadrilateral frame, and then we did corners detection on this (see Fig. 1).

Our corner detection algorithm takes the point with the smallest "city block" distance, which is vertical distance plus horizontal distance, to each of the four matrix corners. We did note that the absolute x-y distance is incorrect in certain perspective cases, thus we chose to minimize the "city block" distance. There are cases where two corners may be closer to one of the matrix corners, but these are a highly distorted that we assume will not occur.

We first calculate our corners, then contracted the image to contain just the frame. After extracting the relevant image and corners, we did a perspective transformation on it (see Fig. 2). In our function *getcorners*, we obtain the four corners of the painting. Then we do the following mapping:

Top Left Corner → (0,0)
Top Right Corner → (1000,0)
Bottom Left Corner → (0,1000)
Bottom Right Corner → (1000,1000)

The projective transformation is defined as

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = T^{-1} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Now we make a few assumptions:

$$T^{-1} = \begin{bmatrix} t_{1,1} & t_{1,2} & t_{1,3} \\ t_{2,1} & t_{2,2} & t_{2,3} \\ t_{3,1} & t_{3,2} & t_{3,3} \end{bmatrix}$$

$$u = \frac{u'}{w'}$$

$$v = \frac{v'}{w'}$$

Fig. 2. Result of our perspective transform on our isolated picture (see original image in Fig 1c).

Solving the above we would get

$$u = \frac{t_{1,1}x + t_{1,2}y + t_{1,3}}{t_{3,1}x + t_{3,2}y + t_{3,3}}$$

$$v = \frac{t_{2,1}x + t_{2,2}y + t_{2,3}}{t_{3,1}x + t_{3,2}y + t_{3,3}}$$

which would yield the new points after the transform.

B. Eigenimage Analysis

The eigenimage technique is based on linear projection of the image to a low dimension feature face [1]. It uses principal components analysis, which is basically eigenvector decomposition and then discarding the insignificant eigenvalues. Since the size of the matrix is too big for efficient eigen-decomposition, we use the Sirovich and Kirby method for decomposition. For our project, we used a "modified eigenimage method." We use 3 images for each of the 33 classes. The first image is the painting without the frame, the second image is the painting with the full frame, and the third image is the painting with a partial frame. This ensures maximum correlation with the desired image should our corners algorithm not get the full frame. The theory is as follows:

We need the eigenvectors of the correlation matrix SS^H, however we only need 33 X 3 = 99 linearly independent vectors to form our space. Instead of considering SS^H, we consider S^HS which has rank 99. At the end, we simply

multiply S to the eigenvectors obtained to get the eigenvectors of SS^H. Although the number of eigenvectors we get is not the complete set, we will not need all of them, but just the significant ones to create our projection space (eigenspace).

When we get our image, we do the painting extraction described earlier and then project the result on our projection space. We then sum the coefficients corresponding to images in the same class. We would then compare the projection coefficients we got with the summed coefficients of the 33 images. We used the L_2-norm comparison (Euclidean distance).

In order to speed up execution of our code, we stored the projection space we created with the 99 training images into a MAT-file and load it when we need it. The MAT-file storing the projection space vectors have a relatively large size, which may be a drawback in a real application. However, there is an easy way to overcome this. We can use the DCT or the DFT to dimensionally reduce the pictures before performing the eigenimage test. However, for accuracy in this project, we decided to use the images in its entirety.

C. Difficulties and Our Solutions

Shadows: This is the biggest problem that we faced. Many shadows were dark enough to contrast sharply with the colors of the wall. The edge detector might pick it up and our program might misinterpret it as part of the frame itself. We overcame this problem and vastly increased accuracy by including three sets of pictures in eigenimage analysis: picture alone with no frame, with the full frame without shadow, and the full frame with shadow. Extreme shadow cases, such as long shadows with sharp edges, made the projection values between different images closer, but our algorithm still picked the correct image due to the size of our training set.

Dilating and eroding: in some cases, the edge detection yields faint edges, and thus our structuring element would be too small to connect the edges leading us to get a single edge of the painting. Big structure elements, however, do not work for overly contrasted images that may have more noise near the edges. These images would have their frames in addition to addition nearby unwanted objects connected to them due to the big structuring elements. Thus, we did "dynamic structuring element resizing." When the dilation fails to connect parts of the painting frame together, we would get only part of the frame, which would mean that the size of our extracted image would have a "bad" aspect ratio. By "bad" we mean that the image extracted may be either too flat or elongated, and usually, this will happen when only part of the frame is extracted. We manually found an aspect-ratio threshold of [0.5 1.9] for our training set. Anything that falls outside this range would be continually passed through edge detection with increasing structuring element size until it falls within this threshold.

D. Attempted Alternatives

Various Resolutions: Using better resolution pictures in our eigenimage training set: We tried using higher quality training images taken with a digital camera. However, this yielded less accurate results since spectral decomposition is actually dependent on image quality too, so it would make more sense to use image of the same quality in our training set.

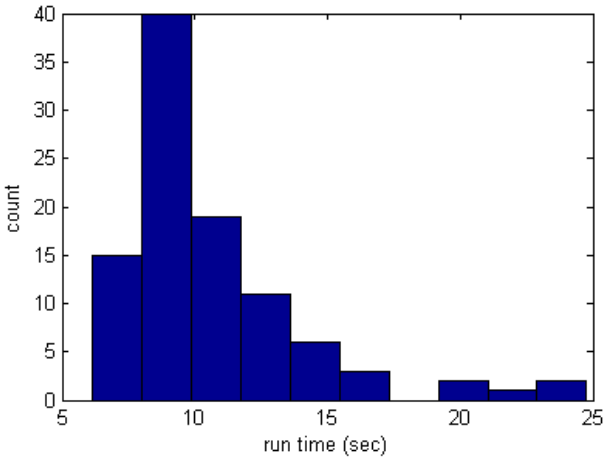


Fig. 3. Distribution of run times from our given 99 training images.

Haralick Corners Detector: We programmed a Haralick corners detector function and ran it on our training set. However, the results were not reliable. This is because what the Haralick corners detector does is to use a gradient operator to calculate the horizontal and vertical gradient and then compute the normal matrix to determine if there is a sharp edge in a certain window. However, this is hard to control well as there may be locations where gradient changes are steep but no corners are present.

Radon Transform: Once we obtained the frame, we could use the Radon transform to try to obtain the equations of the edges as lines. Using these, we could get the corners based on the edges which would be quite robust. Deciding which of the lines was which edge, however, was not trivial. Simply taking the biggest values of our Radon space may yield multiple lines for one edge. Other methods that we tried for isolating the four edges were complex or unreliable.

Color Histograms: As a preliminary identification step, we wanted to use the fast and relatively scale invariant method of histograms with color coherent vectors (CCV) [2], [3]. CCVs divide each RGB color pixel into two groups: those that are "coherent" meaning that they are in a 8-neighbor region greater than area t, and "incoherent" pixels that are not. This was not as reliable for a top 1 answer primarily due to drastic changes in illumination and varying sizes of the captured frame make it. From our experiments with the training set, we can get up to 78% accuracy for a top 1 answer, and 98% accuracy for the top 7 list. However, as we improved our eigenimage analysis, this step was not necessary.

Scale Invariant Features: SIFT or SURF would be the most robust and reliable way to approach this problem, but we abandoned the idea because the complexity and computational intensity were unnecessary. There are only 33 paintings with restricted features and so in order to get time savings, we found it more expedient to just tailor our algorithm to the 33

images. [4], [5]

E. Performance Analysis

Our algorithm's runtime is fairly consistent with a mean run time of 10.6 sec and a standard deviation of 3.5 sec. There are some outliers with greater than 20 sec run times, which are due to longer perspective transforms (see Fig. 3).

The main bottleneck is in the perspective transform. This portion took about 80% of the processing time. The second to fourth longest-running functions were edge detection, and image resizing, and eigenimage analysis respectively.

Our algorithm's scoring has a wide distribution in projection values. This can be attributed to varying levels of confidence and differences in each painting's eigenspace "uniqueness." A lower projection value indicates a better match for the resultant answer from our algorithm. One painting that was difficult to analyze consistently had a high projection value. We also plotted the difference in projection value between the top two answers produced from our algorithm. This can be considered another measure of how confident the algorithm is in its answer (see Fig. 4). The greater the difference, the more sure we are that our answer is correct.

We ran our algorithm on the training set of 99 images given, and we obtained 100% accuracy. We also ran it on several images we created, and most were successes. Those that failed were the ones that had extreme perspective transforms or contrast levels, which would never occur with a reasonable user or museum environment.

IV. Future Work

With the limited amount of time to implement this project, many other topics still can be explored. Below are possible experiments or topics that we wanted to investigate:

1) To make edge and corner detection more robust, we could use color information, spatial location, and Bayesian probabilistic methods to remove shadows. By removing shadows, we can more accurately pick corners and improve distinct projection values.
2) We can add more images to our classes to improve the accuracy of the detection. Perhaps images of different illuminations, contrasts, or frame sizes.
3) Joint histograms are actually multidimensional histograms which uses other picture features besides colors, like gradient magnitude, texture etc. We may be also consider using this to capitalize more on the color information provided by the paintings. [6], [7]
4) We can also improve the speed of our program by algorithm improvement. For example, we can use DCT or DFT to reduce the number of coefficients needed, so that comparison can be speeded up.

V. Conclusion

We went through many algorithms and chose those that were more relevant and reliable. We see that the eigenimage implementation is fast and very accurate. Because of its easy implementation and quick runtime, it can be easily implemented on mobile devices such as handphones without much overhead. Our algorithm can be also expanded to deal with more complex computer vision problems.

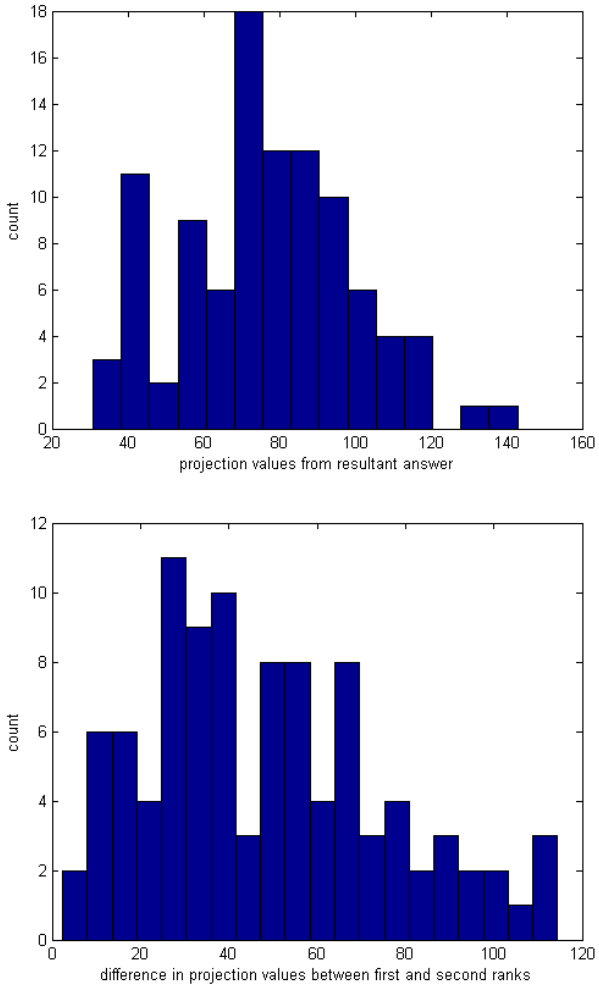


Fig. 4. Distribution of projection values from eigenimage analysis (top) and distribution of the projection value difference between the top match and second match from our eigenimage analysis (bottom).

References

[1] P. Belhumeur, J. Hespanha, and D. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711-720, 1997.

[2] G. Pass and R. Zabih, "Histogram refinement for content based image retrieval," in *Proc. IEEE Workshop on Applications of Computer Vision*, 1996, pp. 96-102.

[3] G. Pass, R. Zabih, and J. Miller, "Comparing images using color coherence vectors," in *Proc. 4th ACM Conf. Multimedia*, Boston, MA, Nov. 1996.

[4] D.G. Lowe, "Object Recognition from Local Scale-Invariant Features," *Proc. Seventh Int'l Conf. Computer Vision*, pp. 1150-1157, 1999.

[5] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision - ECCV 2006*, A. Leonardis, H. Bischof,

and A. Pinz, Eds. of *Lecture Notes in Computer Science*, Springer, 2006, vol. 3951, pp. 404-417.
[6] G. Pass and R. Zabith, "Comparing Images Using Joint Histograms," *Multimedia Systems*, vol. 7, pp. 234-240, 1999.
[7] A. Rao, R. Srihari, and Z. Zhang. Spatial Color Histograms for Content-Based Image Retrieval, *11th IEEE Intl Conf on Tools With AI*, 1999.

DIVISION OF WORK

This section indicates main contributors to specific sections.

A. Qun Feng Tan

- Testing and implementing the eigenimage and Sirovich and Kirby algorithm
- Testing and implementing the perspective transformation algorithm

B. David Lau

- Creating the testing harness and collecting statistical data from test results
- Testing and implementing the CCV algorithm

C. Equal contributions were made to all other parts below

- Edge and corner detection algorithms
- Picture isolation using morphological processing
- Researching alternative algorithms
- Writing and editing the report