Image Recognition – Pictures at an Exhibition

Roger Hsiao
Electrical Engineering Department
Stanford University
Stanford, CA 94305
rhsiao@stanford.edu

***Abstract* – With the proliferation of cell phone cameras, new applications are being sought out by cell phone service providers. We explore a possibility for one such application that will provide image recognition for digitally captured images. Specifically, we investigate the applicability of eigenimages for the recognition of pieces of art that were captured with the Nokia N93 camera-phone.**

1. INTRODUCTION

Image Recognition has always been natural to the human eye and mind, but somewhat less intuitive implemented on a computer. The complexity of images oftentimes make them unwieldy to compare with each other using the standard brute-force method of comparing pixels. We will investigate the use of eigenimages to perform fast image recognition .

The images that we will be analyzing are taken using the Nokia N93 camera phone, with the subjects of the images being various works of art on display in the Cantor Arts Center at Stanford University. The images vary in terms of angle, illumination, and sharpness, which is typical considering the acquisition method (a cell phone camera is difficult to hold steady). Our database consists of 33 distinct works of art, with the algorithm designed to read in an unknown image and specify which of the 33 known images the unknown image is most likely to be.

Preliminary tests show that the algorithm is successfully able to identify at least 80% of the test images. The average run time of the algorithm in Matlab was 52.84 seconds.

2. TRAINING ALGORITHM

A set of 3x33 test images (3 of each individual art piece) were supplied to train the algorithm. These images were each of size 2048x1536, and contained 3 images of varying illumination, orientation, etc. for each painting. Given that these training images were all different, a new set was created to train the algorithm specifically for a certain configuration. Since the purpose of this new set was to train for a certain orientation, I took the clearest image from each of the painting sets and manipulated it within Adobe Photoshop to be fitted to a 1024x768 frame. In doing so, some skewing of the images occurred, as images taken in the vertical configuration were stretched horizontally so that the edges of the frame corresponded to the edges of the actual image. An example of an original training image and the corresponding new training image are provided as Figures 1 and 2 respectively.

Figure 1: Original Training Image

Figure 2: Modified Training Image

Thus, a completely new training set was created to base all the comparisons off of.

The creation of the actual eigenimages was based on the method used to generate eigenfaces. I began by first converting the images from RGB to grayscale (using MATLAB) and then reshaping each training image from 1024x768 into 1x786432 column vectors. I then concatenated all the column vectors to form a large 33x786432. From this new matrix, I calculated the mean 1x786432 image of the 33 columns, and subtracted it from each column to produce a matrix S.

To determine the most significant eigenimages, I created a new matrix by taking the product S^TS, which yields a much smaller matrix than the covariance matrix SS^T. By finding the eigenvectors of S^TS, called $\vec{v}_i$, I was able to find the eigenvectors of the covariance matrix SS^T, called $\vec{w}_i$ by performing the operation (the Sirovich and Kirby Method[1]):

$$\vec{w}_i = S\vec{v}_i$$

With the set of basis eigenimages, $\vec{w}_i$, the set of training images can be mapped into the image space represented by $\vec{w}_i$ by performing the operation:

$$\Omega_k = U^T\left(\vec{i}_k\right)$$

where $\vec{i}_k$ is the original training image k represented as a column vector and Ω_k is the training image k represented in the new image space.

With these basis vectors, Ω_k, the task with an unknown image results in mapping the unknown image into the same image space by performing the operation:

$$\Omega_{unknown} = U^T\left(\vec{i}_{unknown}\right)$$

where $\Omega_{unknown}$ is the representation of the unknown image in the $\vec{w}_i$ image space, and $\vec{i}_{unknown}$ is the unknown image in the original basis represented as a column vector.

The unknown image is then the corresponding training image, k, that minimizes ε in the following equation:

$$\varepsilon = \sqrt{\left(\Omega_{unknown} - \Omega_k\right)}$$

3. FRAME DETERMINATION

With the setup of the most efficient basis image space complete, the next task is to map the unknown images to the same orientation and size as the modified training images.

The most intuitive idea would be to find the corners of the frame, and then map them into the extreme corners of the new 1024x768 image and crop any outlying points. However, this is easier said than done.

The first attempt was to use the Harris Corner Detector, but through experimentation, it was discovered that a single mistaken point could be absolutely devastating to the image recognition program. One wrong corner and the entire image is changed drastically. Also, given the inherent unpredictability in the images, spurious corner detections not corresponding to the frame were very common. Thus, a more robust frame determination method was needed.

The method that seemed to work the best was to create a mask of the painting and frame. This was done by using the Robert's edge detection algorithm (provided by the MATLAB built in function 'edge()'). It was noted that the edges of the frames had the most definite pattern of points, but that they were not equally spaced, nor were they appearing in a consistent, predictable manner (the edge points appeared almost stochastic, but with a higher affinity towards the frames and actual edges). Thus, the first step was to eliminate singular spurious edge points with the use of a 2x2 median filter. However, it was noted that this did not completely eliminate all spurious points, as there were instances where multiple spurious points would be grouped together.

The next issue was to actually separate the frame edges from all the other spurious points. A pixel by pixel analysis would fail here because finding extreme points that were deemed edges would pick up the spurious points over the actual frame edges. Thus, the next step was to dilate the image with a structuring element in the shape of a diamond (to accentuate the horizontal and vertical components of the frame). The size of the diamond was experimentally determined to be best when set at 20. The dilated image was then eroded, to eliminate the possibility of multiple spurious points combining with the actual frame and creating an amorphous blob that contained the entire image.

The next step was to use the MATLAB command imfill() to fill up the 'holes' in the image. Since the dilation of each of the points on the edge of the frame generally created a closed polygon, the inside portion of the frame would be filled. However, an issue that was discovered through experimentation was that oftentimes, there would be spurious frame edges where the dilation/erosion technique had actually combined spurious points with the frame points. These elements were cut off from the main frame structure by erosion, followed with dilation of the image with long linear structuring elements at 0 degrees and 90 degrees. The length of the lines could be extremely large (in this case, 300) due to the fact that the inside of the frame would be filled with all white, and thus, the erosion/dilation procedure would not significantly affect the frame itself. However, in instances where there were smaller artifacts attached to the frame, this set of erosions/dilations would, in effect, separate them. Also, the erosion/dilation would remove many of the smaller artifacts spread around the image.

The final steps were to actually isolate the main frame mask, which was easily done by utilizing the MATLAB command regionprops(). Since the erosion/dilation by the long lines had eliminated many of the smaller structures, the regionprops() command generally had to only calculate the properties of less than three structures, making it operate relatively quickly. The structure with the largest area was deemed to be the frame mask, and every other structure was cleared, yielding a mask highlighting only the picture and frame in the original image. A bounding box was determined, and the corners of the

bounding box were mapped using to the extreme corners of the final 1024x768 image with all other portions of the image being cropped out. Thus, we were left with a close-up view of strictly the picture and frame.

Examples of the original image, the mask, and the final image are provided as figures 3, 4, and 5 respectively.

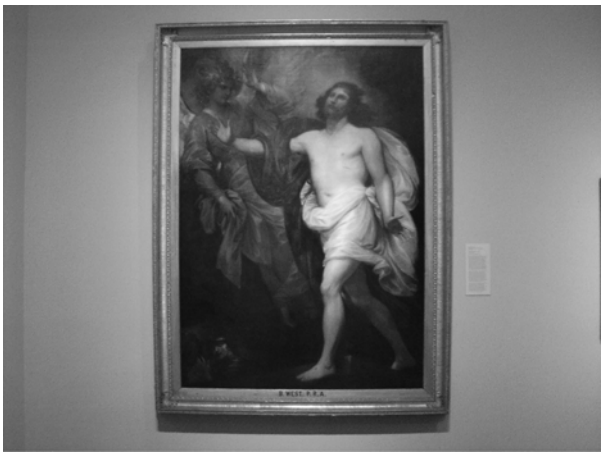

Figure 3: Original Grayscale Image

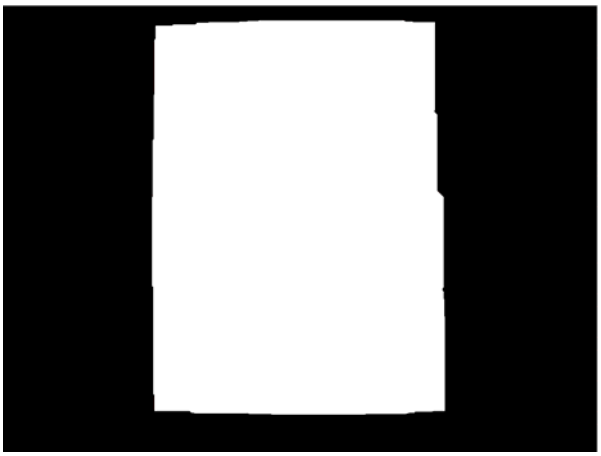

Figure 4: Logical Mask

Figure 5: Final image to be processed

Experimentation was done with using the 'orientation' property returned by regionprops(), but this was eliminated as the recognition algorithm seemed to be able to resolve the angular differences, and the run time of the algorithm was becoming too long.

The final resized and reoriented image was then mapped into the $\vec{w_i}$ image space and compared with the known images as described in the previous section.

4. RESULTS

Since the training images that were used were actually not entirely from the set of 3x33 training images that were provided, testing was performed on the original given images. It was observed that the algorithm performed very well in determining certain images, but poorly on images with complex frames or when there were significant objects in the background (such as the frame of another image). However, the algorithm was able to identify approximately 80% of the test images in the training set.

The algorithm run time was a serious issue, averaging 52.84 seconds for each image. This was greatly effected by the large data structures that were being loaded from the training data (the matrices U and Ω). It was unknown whether the run time would be significantly changed on the SCIEN machines, but since it was under 60 seconds, and no obvious optimization could be performed, it was left alone.

4. CONCLUSION

The image recognition algorithm proved to be relatively reliable. The eigenimage approach to image recognition seems to be very robust, as it was able to compensate for the angle displacement and slightly different orientations. This demonstrates that image recognition can be largely based on the eigenimage approach, but must be experimented and tweaked with (as many of the parameters were determined through experiments).

4. REFERENCES

[1] L. Sirovich and M. Kirby, "Low-dimensional procedure for the characterization of human faces,"
Journal of the Optical Society of America A, 4(3), pp. 519-524, 1987.