RECOGNITION AND AGE PREDICTION WITH DIGITAL IMAGES OF MISSING CHILDREN

CS 297 Report
by
Wallun Chan

Advisor: Dr. Chris Pollett
Department of Computer Science
San Jose State University
May 2005

1. INTRODUCTION

The purpose of this report is to provide a synopsis of the results achieved in CS 297 for Fall 2005. The results culminate in four deliverables that demonstrate research and implementation work during the course of the semester. These deliverables provide support for image retrieval and preprocessing, and initial attempts at image recognition and reconstruction. Also, the work performed provides the initial groundwork for further research and experimentation with advanced feature-based methods of extraction, analysis and processing of faces, and reconstruction of age-predicted face images.

2. DELIVERABLE 1 - Face Recognition / Reconstruction with Eigenfaces

The purpose of Deliverable 1 is to verify the feasibility of using the eigenface approach for image recognition and reconstruction. Once successful demonstration of this is achieved, age prediction could be attempted with this technique. The eigenface approach is based on using Principal Component Analysis (PCA) to select patterns that best represent correlations between images thereby reducing redundancy. Given N points in D-dimensional space, the eigenvectors of the covariance matrix for the data set forms the principal axes of the D-dimensional data points. In other words, the image data can be expressed more efficiently in terms of D orthogonal eigenvectors that form a new coordinate system. This is done by projecting the data points onto a vector passing through the mean of the data points in the direction of the eigenvectors. The direction of the new principal axes (eigenvectors) is chosen such that the scatter is maximized for each axis. A fraction of the eigenvectors will typically dominant in this scatter. The measure of scatter is the covariance matrix multiplied by N - 1. The covariance matrix generalizes variance for a random scalar variable to a random vector variable. The

elements of the covariance matrix are the covariances of all pairs of scalar elements of the vector variable. The amount of data may be reduced by eliminating those eigenvectors that do not contribute much to the scatter. This is the basis for image compression. Also, it can be observed that for any set of images, although differences exist, redundancy may still be reduced by taking advantage of the correlation between images. The covariance matrix as described, allows for a transformation that de-correlates a set of data in a least squares sense.

PCA determines and uses the eigenvectors to express the variation between images. Each eigenvector (eigenface) represents a principal component such that any original training image may be reconstructed with a weighted linear combination of the eigenvectors. Each eigenvector contributes more or less to each original image. To reconstruct the each training image from the eigenvectors, the right proportions or weights must be determined. This is done by projecting each training image onto the dominant eigenvectors to form a weight vector. Each training image will have its own characteristic weight vector. In addition, these weights can be used not only for image reconstruction, but also for image recognition. This is done by first projecting a test image to form its weight vector. The test weight vector is then compared with the weight vectors of each training image. The comparison criteria used is the Euclidean distance between the weight vectors. The smallest Euclidean distance found of the weight vectors between the test image and all the training images is that of the recognized image.

2.1 Implementation and Functionality of Application Program

The application program for Deliverable 1 was written and compiled in Visual C++. A screenshot of the program is shown in Figure 1. The application program uses a

specific directory structure in which each person (class) is assigned a directory containing a number of images of that person. In turn, each of these class directories are stored under a root directory for the training and test set of images.

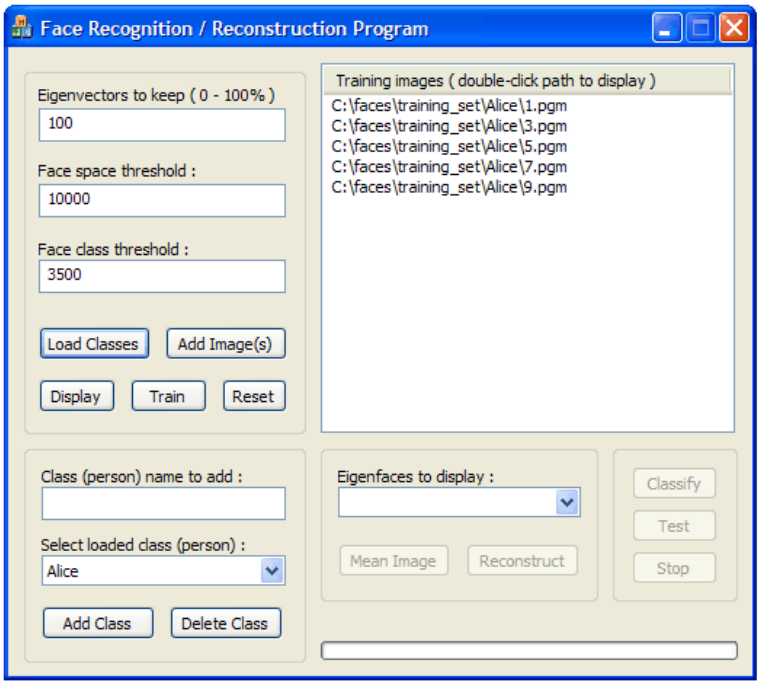

Figure 1: Face recognition / reconstruction program

2.2 Testing the Application Program

A sub-goal of Deliverable 1 is to determine how well PCA performs on unseen images. That is, given a set of training images of various people, how well does the classifier perform on images that it was not trained on. Therefore, to test the recognition performance of the application program, grayscale .pgm images were downloaded from [14]. These images represent 10 various facial poses from 40 subjects for a total of 400 images. The size of each image is 92 by 112 pixels, at 8-bits grayscale. The faces vary with respect to the lighting, facial expressions like closed or open eyes and mouths, and face accessories like glasses. The background features are eliminated, leaving just details of the head. In-plane and out-of-plane head rotation also vary.

The images are separated into training and test sets as noted above. Each set consists of 40 directories, each containing 5 images for a particular class (person). The test set consists of the same 40 classes as the training set, but with different facial poses as that of the training images. By loading the training images into the application program, training the classifier, and generating recognition performance statistics for the test images, we have the results as shown in Figure 2.

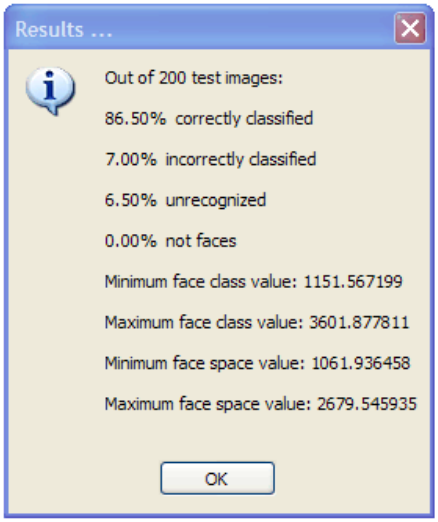

Figure 2: Recognition results for 200 test images

The results above show a maximum face class value of 3601.88. It also indicates that 6.5% of tested images are unrecognized. This margin may be decreased by increasing the face class threshold to 3602.0 to force classification at the expense of increasing the number of incorrectly classified images. The maximum face space value is noted to be 2679.54. But since we know *a priori* that we are testing images, we set the face space threshold to a large number so that all images are considered faces. In general, given unknown images, the face space threshold must be specified (usually with trial and error) in conjunction with the face class threshold to classify images accordingly.

2.3 Image Reconstruction

The main goal of this application program is to determine how well images can be reconstructed. More specifically, how effective is the eigenface approach in reconstructing recognizable new images given known input images. The "Reconstruct" button reconstructs an original or new image. Figure 4 shows the reconstruction results for a training image from [14] with different percentages of eigenvectors retained. As eigenvectors are truncated, the reconstructed image approaches the mean image.

Figure 3: 100%, 75%, 50%, and 25% of eigenvectors retained; mean image (last)

3. DELIVERABLE 2 - Python Webcrawler and Image Preprocessor

The purpose of this deliverable is to implement Python scripts that enter a missing children's website, downloads all relevant images by using the web-page metadata, preprocesses the images in terms of color, size, scale, orientation, and sets up a directory structure to organize and store the images. The implementation consists of three Python modules, webcrawler.py, face_ops.py, and image_ops.py, and a C++ helper program "trackface.cpp".

3.1 Python Script 1 - webcrawler.py

The function of this script is to go through a website and download all children's images for a particular state. After the images are downloaded, a specified destination directory structure is created, and the images are stored in the file format as downloaded. This script communicates with a servlet object to obtain the needed data by sending the

appropriate parameters:

http://www.missingkids.com/missingkids/servlet/PubCaseSearchServlet?act=usMapSearch&missState=CA&searchLang=en_US).

3.2 Python Script 2 - face_ops.py

This script takes as input the directory structure created by webcrawler.py. It traverses the directory structure and crops all faces in the images to maximize face space. Images are also converted from .jpg to grayscale .pgm format. A new directory structure is created and copied from the original. In addition, in-plane rotation is attempted to correct for head tilts. And lastly, a non-aged and aged face image is compared to see if person is facing the same direction. If not, one of the faces is mirrored. This is done by comparing the weighted average of the image difference between the non-aged and aged image, and between the non-aged and mirrored aged image. Otherwise, images are not cropped and converted to grayscale .pgm format only. Either way, images are resized to specified dimensions and stored in the intended destination paths.

3.3 Python Script 3 - image_ops.py

This script takes as input the output directory structure created by face_ops.py and produces another directory structure with resultant images from the subtraction of non-aged and aged images, concatenation of non-aged and aged images, concatenation of images with a blank image, and concatenation of images with themselves. Specifically, the output from face_ops.py should be such that corresponding non-aged and aged images are stored in the same directory.

3.4 Helper Program - trackface.cpp

The purpose of "trackface.cpp" is to process a grayscale .pgm face image for the pixel locations of the face boundaries and eye locations. The face boundary locations are used to crop the faces accordingly, and the eye locations are used to correct for in-plane face rotations, in essence straightening out each face, if needed. "face_ops.py" calls this program to generate a log file that contains the face boundary and eye locations of a face. "face_ops.py" then parses this log file for the data, and uses it to crop faces and straighten out any in-plane rotation.

4. DELIVERABLE 3 - Image Alignment / Feature Location with Shape Contexts

This deliverable explores the feasibility of using the idea of shape contexts to provide a robust means of quantifying and representing the shape of an object. This representation is then used along with several other techniques to support image alignment and feature location of an image. The idea is to manually select features on a template image, and then align a test image to the template image thereby locating the features in question on the test image.

4.1 Description of Shape Context

Shape context is characterized by the spatial relationship between a point and all other points in a shape. Specifically, it is defined by a set of points sampled from the internal or external edges of a shape. The edges may be obtained by using an edge detection algorithm. This application uses the Boie-Cox edge detector [12] which exhibits excellent characteristics in detection (high signal-to-noise ratio), localization (accurate pixel marking), and unique response (low sensitivity to spurious signals). For a given sampled point, a shape context descriptor is defined by determining the set of

vectors from this point to all other sampled points on the shape. Specifically, the shape context for a point is a log-polar histogram that sorts all vectors for a given point by relative distance and angular orientation. Figure 4 from [1] illustrates this.

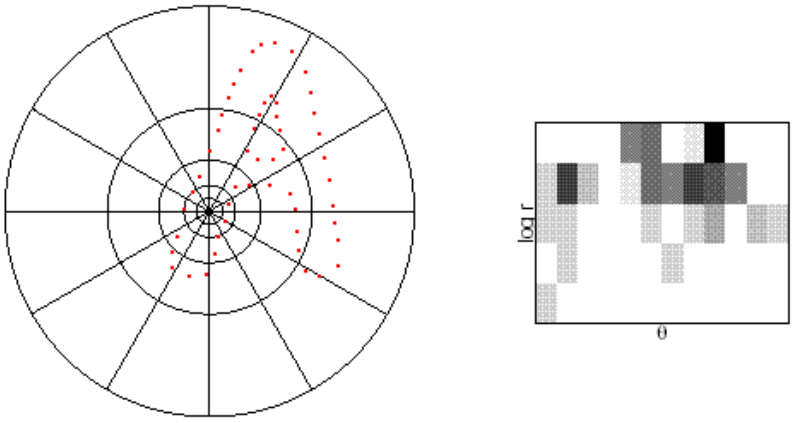

Figure 4: Log-polar bins; corresponding flattened histogram

Qualitatively, the shape context for each point gives a precise description of the relative position of a point to all other points. Thus, it can be used as an effective measurement of shape similarity for a corresponding point on another shape to be compared. This implies that there must be a full point-to-point correspondence between points on two shapes. To obtain point-to-point correspondence, the log-polar histogram representing the shape context for each point on a shape may be used to calculate the measurement cost between points on two shapes. Given two shapes each with n sampled points, a cost matrix representing all point pairs of size n by n is set up where each matrix element is the χ^2 cost between points on two shapes. The cost equation is given below, where C_{ij} represents the cost between points p_i and p_j on different shapes, and $h_i(k)$ represents the k^{th} bin of the normalized histograms at points p_i and p_j.

$$C_{ij} = \frac{1}{2}\sum_{k=1}^{K}\frac{[h_i(k) - h_j(k)]^2}{h_i(k) + h_j(k)}$$

The cost matrix represents an instance of the weighted bipartite matching problem. In modeling the bipartite matching problem with respect to shape contexts, we wish to minimize the total cost for all point pairs. The bipartite matching problem may be solved in $O(n^3)$ by using the Hungarian algorithm. This application uses a more efficient algorithm provided by [4] and [13] in solving the linear assignment problem.

4.2 Thin-Plate-Spline Point Alignment Method

Having solved the point-to-point correspondence problem, the shapes must be aligned to match features on one image to another. [1] suggests using the thin-plate-spline method of point-set alignment. This technique attempts to model coordinate transformations from one set of points to another by using a weighted combination of thin-plate-splines centered about each control point that allows a mapping function to be interpolated through these points exactly. Figure 5 from [9] illustrates this.

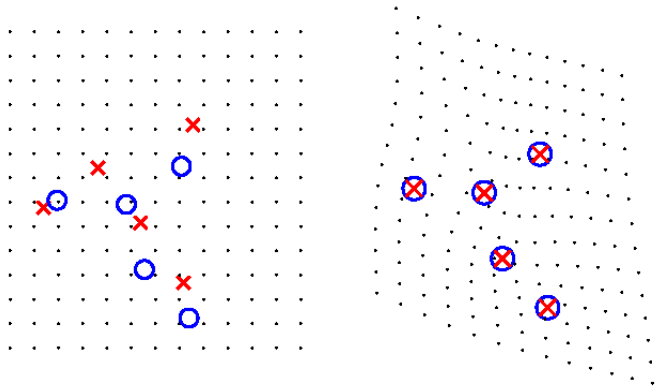

Figure 5: Illustration of thin-plate-spline mapping

4.3 Operations of the Image Alignment / Feature Locate Program

The implementation of this application first performs edge detection to retrieve the edge data of images. These edges are then sampled for both test and target point-sets for two images to be aligned. Point-to-point correspondence is determined between the two point-sets for a minimum overall matching between point-sets. Thin-plate-spline

interpolation is then performed to provide a mapping function from test to target image coordinates. Selected test points are then grouped in target image coordinates. A screenshot of the application program is shown in Figure 6.

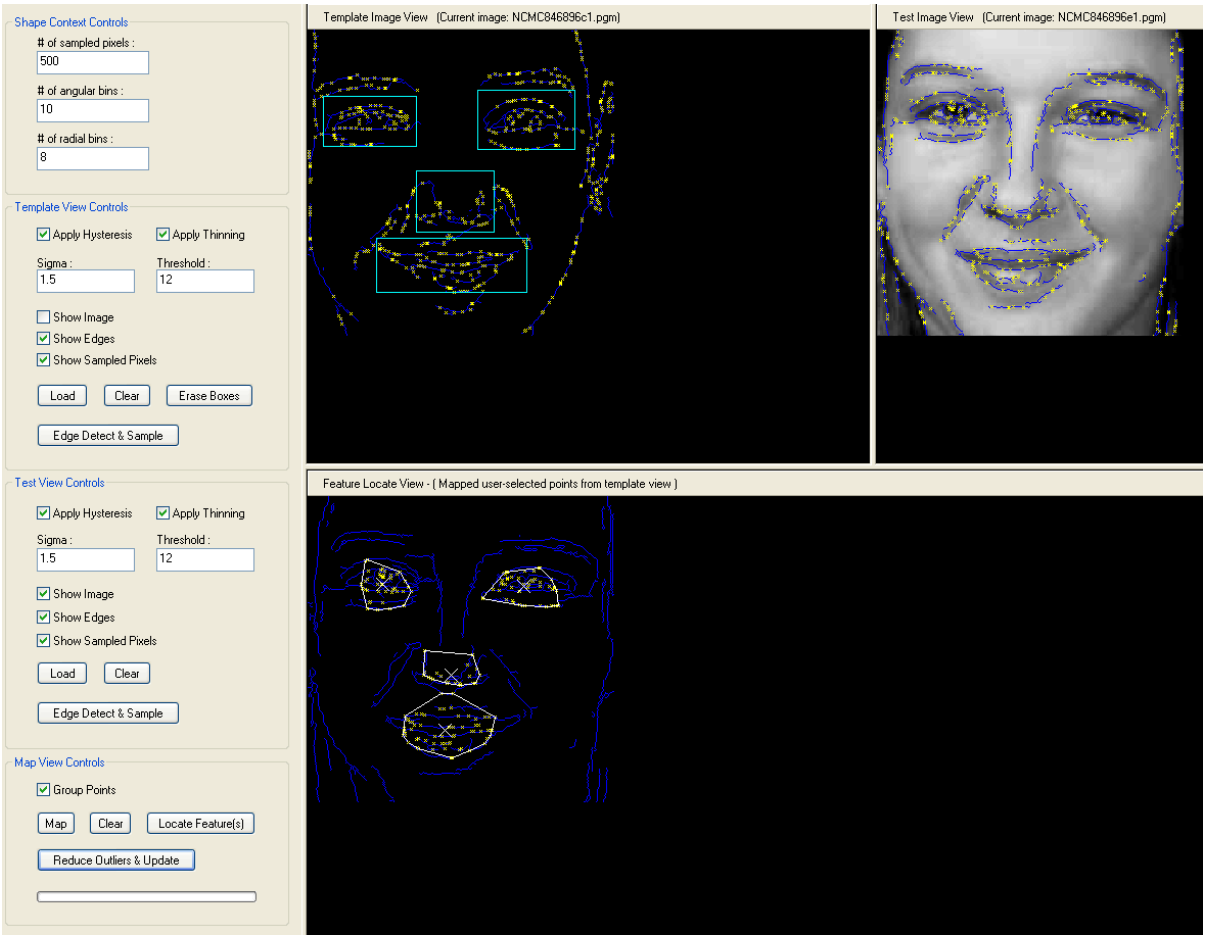

Figure 6: Image Alignment / Feature Locate Program

4.4 Functionality of the Image Alignment / Feature Locate Program

The program consists of four views: the control, template, test, and feature locate views. The control view allows the user to load and display images for the template and test views. The “Load” button reads and displays images in the template and test views. The “Edge Detect and Sample” button performs edge detection on the template and test images and samples points from these edges. The "Map" button applies thin-plate-spline

interpolation to the test and target point-sets. The user may then click and create boxes in the template view to select features to map to the test view, which are then shown in the feature locate view. The "Locate Feature(s)" button takes all the user-selected points from the template view, maps it to the test view, groups them into convex hulls according to user selections, and then displays the centroid of each group in the feature locate view. The "Reduce Outliers & Update" button attempts to reduce the number of outliers during grouping. This is done by removing any points of a group that are greater than a fixed number of standard deviations from the mean radial distance between the group centroid and a point.

5. DELIVERABLE 4 - Age Progression by Reconstructing Concatenated Images

The purpose of this deliverable is to use the tools developed for Deliverables 1 and 2 to produce results for an initial attempt at age progression of images of missing children. Deliverable 1 is an implementation of the eigenface approach for image recognition and reconstruction. Deliverable 2 consists of Python script implementations that download missing children's images, preprocess images in terms of size and file format, and support image cropping, mirroring, and concatenation.

Specifically, we want to create new images that are concatenations of original images with their corresponding enhanced images. We use the face reconstruction program of Deliverable 1 to train on a number of these concatenated images. Then we input into the face reconstruction program a concatenated image that consists of the same original image of a child on both sides of the image. We also input into the reconstruction program a concatenated image that consists of an original image on one side and a blank image on the other side. In both cases, we wish to obtain a reconstructed

image with one side of the image converted into an enhanced (aged) version of the original image.

5.1 Testing the Python Scripts

We first retrieve a number of missing children's images by using the "webcrawler.py" python script to download from [16]. After the images are downloaded, a specific directory structure is created and the images are stored as downloaded. Next the Python script "face_ops.py" is invoked. This script takes as input the directory structure created by "webcrawler.py". It traverses the directory structure and crops all faces in the images to bring them forward to maximize face space (i.e. reduces pixels covered by hair and background). Images are also rotated to correct for in-plane rotation and converted from .jpg to grayscale .pgm format. A new directory structure is created. We then call the Python script "image_ops.py" to perform further image preprocessing on the output directory structure of "face_ops.py". As previously mentioned, we want to create a set of concatenated images that consist of original images on one side, and enhanced images on the other side. We also want to create a set of concatenated images that consist of the same original image of a child on both sides of the concatenated image. We use one set for classifier training, and the other set for image reconstruction.

5.2 Reconstruction of Concatenated Images

Given the generated sets of concatenated training and input images, we use the reconstruction program of Deliverable 1 to produce age-enhanced images. We train the program and obtain example reconstruction results as shown in Figures 8 and 9.

Figure 7: Concatenated training image

Figure 8: Original on left and right

Figure 9: Original on left and blank on right

5.3 Discussion of Results

Some of the reconstructed results for Deliverable 4 look reasonably legible. However, this simplistic global approach for image reconstruction is limited to a large degree, and paradoxically, by the similarity between the original and enhanced images. The more similar an original and enhanced image is, the better the reconstruction results, which defeats the purpose of reconstructing the image. This approach works reasonably well when the configuration of the overall face and specific features are not drastically different.

6.0 Future Work

Future research and implementation will focus on more flexible feature-based methods of extraction, analysis, and reconstruction. Specifically, work in CS 298 will concentrate on researching and developing methods to analyze and process discernible facial features such as the eyes, mouth, and perhaps the nose. A significant portion of CS 298 will also consist of developing techniques to produce legible age enhanced faces using the processed facial feature data.

Bibliography

[1] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. Technical Report UCB//CSD-00-1128, UC Berkeley, January 2001.

[2] F. L. Bookstein. Principal warps: thin-plate splines and decomposition of deformations. IEEE Trans. Pattern Analysis and Machine Intelligence, June 1989.

[3] Pattern Classification. Richard O. Duda, Peter E. Hart, and David G. Stork. John Wiley & Sons, 2001.

[4] R. Jonker and A. Volgenant. "A Shortest Augmenting Path Algorithm for Dense and Sparse Linear Assignment Problems", Computing 38, 325-340, 1987.

[5] Numerical Recipes in C. William H. Press, Brian P. Flannery, Saul A. Teukolsky, William T. Vetterling. Cambridge University Press, 1988.

[6] Stuart Russell and Peter Norvig. Artificial Intelligence: A Modern Approach. Pearson Education, Inc., 2003.

[7] Practical Algorithms for Image Analysis: Description, Examples, and Code. Michael Seul, Lawrence O'Gorman, and Michael J. Sammon. Cambridge University Press, 2000.

[8] Eigenfaces for Recognition. Matthew Turk and Alex Pentland. Journal of Cognitive Neuroscience, Vol. 3, No. 1, 1991.

Web References

[9] Gianluca Donato and Serge Belongie. Approximation Methods for Thin Plate Spline Mappings and Principal Warps. http://www-cse.ucsd.edu/~sjb/pami_tps.pdf, 2002.

[10] Jarno Elonen. Thin Plate Spline editor - an example program in C++. http://elonen.iki.fi/code/tpsdemo, 2003.

[11] Lindsay I. Smith. A Tutorial on Principal Components Analysis. http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf, 2002.

[12] Ingemar J. Cox. Boie-Cox Edge Detector Code. http://www.ee.ucl.ac.uk/~icox/.

[13] R. Jonker and A. Volgenant. Linear Assignment Problem Code. http://www.magiclogic.com/assignment.html.

[14] Cambridge University Engineering Department Database of Faces. http://www.uk.research.att.com/facedatabase.html.

[15] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Face Detection Source Code. http://vasc.ri.cmu.edu/NNFaceDetector/.

[16] Missing Children's Image Database. http://www.missingkids.com.