Feature-Based Face Recognition for Vending Machine vs. AMBER Alert

Adriana Kovashka
Pomona College

Margaret Martonosi
Princeton University

Abstract

Face recognition systems are an important field in computer vision and are currently used to monitor for dangerous persons and track criminals. A face recognition system uses a database of images and compares another image against those to find a match, if one exists. We implemented an original face recognizer in Java and tested it for recall and accuracy with three image sets. For each facial image, we created a fingerprint of 18 features, such as the RGB values for the eye color, the width and height of the face, various ratios etc. We utilized WEKA machine learning to determine which features are most important and give appropriate weights. We found that the distances between the eyes, nose, and mouth were not useful as they vary little between people. Overall, our method achieved very good results. For scenarios like surveillance which require low false negatives, our accuracy rate was 75% (and 2.3% false negatives). For vending machine authentication where low false positives are needed, we had 49.3% accuracy (and 1.5% false positives). Our system often outperformed WEKA since it uses a more flexible classification rule in the form of a similarity score rather than a binary decision tree.

1 Introduction

Face recognition is one of the most exciting topics in computer vision today. It has numerous applications, ranging from security and surveillance to entertainment websites. Face recognition software is commercially available from various companies which advertise it as useful in banks, airports, and other institutions for screening customers. Some countries, such as Germany and Australia, have deployed face recognition at borders and customs for automatic passport control [4]. Face recognition systems have been used to monitor for dangerous persons during large events such as SuperBowl and to (attempt to) track down abductors, for example during the disappearance of three-year-old Madeline McCann [4].

There are two main stages in the process of face recognition. The first is face detection, or the ability of a computer to independently discern the feature of a face in an image. The second part of the process involves the preparation of a fingerprint for the image and then a comparison of that fingerprint to the fingerprints of other images in a database. Different algorithms use different facial features for a fingerprint, and they also differ fundamentally in their approach to comparing images.

Many challenges exist for face recognition. In some cases, depending on the different methods implemented to create a fingerprint for an image, the robustness of the system can be obstructed by humans who alter their facial features through wearing colored contact lenses, growing a mustache, putting on intense make-up, etc. Various ethical concerns are also related to the process of recording, studying, and recognizing faces. Many individuals do not approve of surveillance systems which take numerous photographs of people who have not given their permission for these photographs to be taken. Furthermore, since skin color and other physical features of the face related to race are used in the fingerprint of a person, face recognition might be accused of facilitating racial discrimination.

Our implementation of face recognition uses a fairly simple fingerprint which includes eye and skin color, ratios of distances between prominent facial features, and absolute and relative values of width and height of the face and the eyes. We tested our system in two ways, both of which assume a specific application of the program. The first is the identification of people so that they can purchase items from a vending machine using their previously created accounts, without having to pay the machine directly with cash. The second application is the identification of people who are suspected of abduction, with the possible inclusion of an improved and more robust version of this program in an AMBER-alert-type system.

The following section of this paper presents the previous work done by other researchers which relates to our project. In Section 3, we discuss the specific implementation of our face recognizer. Sections 4 and 5 analyze the the methods and results for our work. Finally, Section 6 lists the conclusions we made and possible future work, while the subsequent sections present our acknowledgements and the resources we used.

2 Related Work

Our work is part of the bigger "Space Aware and Resource Aware Dynamic Network Architecture" (SARANA) joint project of Princeton University and Rutgers University. One goal of the SARANA project is the development of an AMBER alert emergency system [11]. AMBER alert, named after 9-year-old Amber Hagerman who was abducted in 1997, is a system in which messages are displayed on highways asking for people to respond if they have information about the individual suspected of an abduction. The SARANA AMBER alert system "exploits the existing infrastructure of stable and dynamic nodes within a physical space" to allow people to respond and the suspect to be identified. In particular, this system relies on cell phone messages and dispersed surveillance web cameras. The photographs that these cameras take, along with photographs that cell phone owners can take with their phone cameras, could be put through a face recognition system so that the suspect may be found.

Face recognition is not a new field, and an early prototype of a feature-based approach was developed over a hundred years ago. In the nineteenth century, Alphonse Bertillon invented anthropometry, a system of measurements of the physical features of criminals by which they could be identified [1]. Bertillon took photographs of the eyes, noses, foreheads, chins, profiles etc. of individuals and stored them in his "database." While this database was examined by humans and not by computers, it relied on the same biometric approach as modern feature-based face recognizing systems.

Many researchers have worked with feature-based methods. One example of a face recognizer that relies on the use of facial features is described in the paper by Cox et al. [3]. Cox, Ghosn, and Yianilos used a mixture-distance technique which performs at 95% rate using distance records of 30 features whose locations were selected manually. The features include width and height of the eyes, nose, mouth, face, and differences between these from various points.

Feature-based methods are also used in object detection. Viola and Jones present a feature-based method in their paper [13]. Their system, which relies on basic features rather than separate pixels and uses a training set of 4916 facial images labeled manually, performs 15 times faster than all previous face detection algorithms and has a high rate of correctness. Another paper [5] presents a different approach which uses images from Google image search as a training set for learning categories. Kremer of Rutgers University developed a face recognition system which relies on the distances between pixels in two superimposed images [9].

In his 2007 Ph.D. thesis, Wang studied the learning algorithms provided by WEKA [6] to determine relationships between the parameters in data records. He also reviewed WEKA's InfoGainAttributeEval attribute selector which ranks features based on their importance for information gain as part of data mining. We used his work [14] as reference in our own use of WEKA's algorithms.

3 System Overview

3.1 Face Detection

In our work, we implemented an original face recognizing system in Java. For the face detection phase, we used the program Visage, whose code was made available for use on Sourceforge.net [10]. Visage is a system written by Restom which, when connected to a web camera, detects the face in the video stream and tracks the eyes and nose. The original code also allows the option of using winking with the eye as a mouse click, and moving the nose to move the cursor. We only use the part of the code which performs face detection, with multiple modifications.

The altered Visage code reads a JPEG image from a specified file and converts it to an array of pixels (represented by their RGB color) using classes and methods provided by the Java Media Framework (JMF) [7]. After that, Visage uses various algorithms to detect a face. It superimposes a 3x2 matrix in the upper part of an image and searches for dark regions and a brighter one between those two, following certain criteria which can be found in the paper accompanying Restom's work [10].

Our Face Recognizer code makes use of some features which Visage detects in the face, such as the eye pupils and the nose tip. Additionally, we use a method in Visage which determines whether a pixel lies in a skin region of the face or not, using statistics for various skin types. We use this information when creating a face mask whose coordinates help us calculate the width and height of the face.

3.2 Face Recognition

As mentioned before, there are many methods for performing face recognition, and the one we chose is feature-based. For each face, we create a fingerprint which includes the following 18 features: (a) red, green, and blue values for the eye color; (b) ratios between these values denoted as RG, GB, and RB; (c) the width and height of the eye and the ratio between these values; (d) the ratio between the distance between the two eyes and the distance between the eye-line and the nose tip; (e) the width and height of the face and the ratio between these two values; (f) the RGB values of the skin color; (g) the angle of the chin; (h) the number of lines (as detected by Hough transformation) passing around the chin. These were chosen by inspection of facial images and noting the differences between them, analyzing the feasibility of extracting these features using the face detection available from Visage, and comparing our intuition about which features should be used with other feature-based face recognition methods.

3.2.1 Extraction of Values and Database Preparation

We compute the values for the above features, and record them in a database. We use Visage to determine the location of the two eye pupils and extract the eye-color by computing the average red, green, and blue values of each pixel in a determined area encompassing the eye pupil, excluding pixels which represent a skin color. We also calculate the ratios between the red and green, green and blue, and red and blue values and record these. Using the location of the eye pupils, we check a small rectangular region surrounding the eye pupil for the outermost left, right, bottom, and top pixels which do not represent a skin color. We sum up the widths and heights of the two eyes and divide by two to obtain the average width and height of the eye, as well as the ratio between these.

Next, we make a "face mask" by locating the nose and a rectangular shape in which, for every pixel, we check whether it represents a skin pixel. In this case, we use a modification of the original isSkinPixel method from Visage, in which the range allowed for a pixel to be recognized as a skin pixel is smaller since we want to avoid picking up hair color as skin as this would disrupt the accuracy of our face bounding box. Our modified method omits some skin pixels, but the neighboring ones make up for these omissions. We also want to have an elliptical mask since the face is elliptical, so in the rectangular shape described above, we check whether each pixel is at a specific distance from the nose (shorter near the eyes, larger near the chin and forehead). If so, and this pixel is also a skin pixel, we include it in the face mask. In

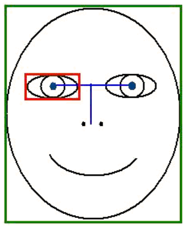

Figure 1: Some measurements of facial features

order to avoid inefficiency, we do not store the actual pixels in an array. Since we are only interested in finding the most extreme points of the bounding box for the face, we keep track of the rightmost, leftmost, bottom, and top pixels of the face mask only. We then compute the width and height of the face, as well as the ratio between these, and record it in the database. The bounding box for the face mask is shown in Figure 1. Using the original isSkinPixel method, which omits less pixels than our method, we compute the red, green, and blue values for each skin pixel and calculate their average, which we also record in the database.

Finally, we use Hough Transform code by Matthews available on [8] (and also used by the original Visage code) to detect all regions in the image which resemble straight lines. We use the Geom class (used by Visage) to determine the angle of each line. We eliminate lines which do not pass through the central bottom part of the image and lines which have have too large or too small slope and angle as these are not likely to pass near the face. For the remaining lines, we check through what elements each one passes, and we exclude those lines for which half of the pixels coincide with or are less than one pixel apart from skin pixels. We do this because we are only interested in lines around the face, not through the face. We then group the left-going and right-going lines, compute the average angle of each group, and sum up the absolute value of the averages to obtain the chin angle, which we record in the database.

Each record is given a name, which corresponds to the person whose face is in the image. The format of the record database is a TXT file.

For testing purposes, we also developed a Viewer class, in which we display the JPEG image and highlight various features of the face with different colors.

3.2.2 Reading the Database and Checking for Matches

After preparing a database with images, we test our database with the same images, recognizing each one by checking for matches with existing records in the database. We compute a fingerprint in the manner described above, and then, for each record in the database, we compute a score describing the similarity between the currently checked image and that record. As described in Section 4.1, we use machine learning from WEKA [6] to determine what factors are most important when computing this score. In particular, we compute the differences for each factor (various RGB values, sizes, ratios, etc.) between the current image value and the database record value. We use the ranks provided by WEKA's InfoGainAttributeEval attribute selector as weights, and we multiply the differences by these weights, then sum up these products to arrive at the similarity scores. Thus, a low score means high similarity with the record. (When testing with images already in the database, this score is zero when the image is compared with the record describing the same image.) The similarity scores are entered in an array and sorted, excluding scores which are greater than the score threshold. This threshold is smaller for the vending machine application and larger for AMBER alert, with current sample values of 75 and 150 respectively. After sorting, the top five matches are recorded in a separate file. The program checks whether a specified number (which we call match threshold) of these top five matches has the same person's name, and if so, outputs this result. The match threshold is different for the two different applications of our Face Recognizer. It is higher for the vending machine application (currently 3 or 4) and lower for AMBER alert (currently 2), for reasons described in Section 5.

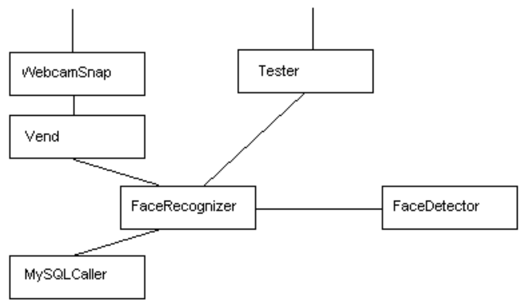

Figure 2: Order of execution for classes which use the Face Recognizer

3.3 Deployment

One application of our face recognition code is for identification of customers who wish to purchase items from a vending machine. Our system provides code which allows the capturing of an image through a webcam and the analysis of this image through our face recognizer. If a match is found, the customer is allowed to purchase the item using funds from their previously set-up account. The sum of money in their record is updated accordingly.

The Face Recognizer package includes a class that provides connectivity to a MySQL database which is different from the image database described in Section 3.2 and is used to store accounts. The four methods in this class set up the database, insert records in it, and update those records by adding or subtracting a specified sum of money.

The performance of the face recognizing system can be tested with a supplied image set (or sets). The order in which the two applications run and use face recognition is illustrated in Figure 2 above.

4 Experimental Methods

4.1 WEKA

WEKA [6] provides data mining algorithms which allow the extraction of important relationships between the data which are done using machine learning. We utilize WEKA to determine which features are most important in order to give according weight to the differences between the examined image and the database records. We input the records we have collected in our database, convert the record file to the ARFF format which WEKA uses, and run one of WEKA's attribute selection methods, InfoGainAttributeEval. The Ranker method gives the features some rank, which we use as the weight of that feature in our similarity score.

The purpose of WEKA is to classify input data and generate a decision tree which determines under what conditions (feature values) is another condition true. WEKA's 10-fold test separates the data into 10 groups and performs 10 trials, in which 9 groups serve as the training set and 1 as the testing set. This test is used to generate a correctness score for WEKA's classification. While we do not use the generated decision tree, we use this test to obtain a general idea about the quality and reliability of the data.

4.2 Image Sets

During the core part of our research process, we used three different image sets, each of a different size. At first, while only using some of the features of the final version of our Face Recognizer, we used 480 of the hundreds of images provided by Oxford University [12]. These were close-up images of 12 women and 12 men on light green background, with 20 photographs per person. Our Face Recognizer worked well with this database of images.

The second image set we used consisted of 450 images and was provided by [2]. Of these 450 we only used 445, as the remaining 5 were single images, and 2 were hand-drawn. This set consisted of a varying number of images per person, with under 31 individuals total (of which we used 26). The images showed the persons in different locations, with background which included numerous colorful elements. The Face Recognizer did not perform as well with this database.

The third set we used included images of people in the Electrical Engineering department, taken in one location, with a web-camera. It consisted of 132 images of 9 individuals, with a varying number of images per person.

5 Results and Discussion

5.1 Performance

We tested our Face Recognizer with the three datasets described above, and with different values for the thresholds which our program uses (primarily the score and match threshold described above). We also compared our performance with that of WEKA.

We computed several scores. The first is a recall score, which measures how many times there was a match found, regardless of the correctness of the match. The second four scores are true positive, false positive, true negative, and false negative scores. The true positive score measures how many of the total matches are correct matches, and the false positive one expresses the difference between the total number of matches and that of the correct ones. The true negative score represents the number of images for which no record was made as a face was not detected. We called this "true negative" since the failure to recognize a face was not caused by faults in the face-recognizing part of our system. Finally, the false negative score describes how often we failed to find a match when one could have been found. When calculating the percentages for TP/FP rates, the total number of correct matches was divided by the total number of images we tested the system with, including ones for which we obtained no record due to failure of the face detection part. The scores for WEKA have been re-calculated to reflect the lack of record for some of the photos.

5.1.1 Overview of Results for First and Second Image Sets

For the first image database, our system achieved a correctness score of 92.5%, while WEKA performed with 90% correctness. For the second image database, our classifier had 20.2% recall, while WEKA recognized 21% images correctly, as shown in Figure 3. When examining the database records, we established that the vast difference between these two pairs was caused by the face detector's failure to detect a face correctly. In fact, in images from the second database, the face detector often picked up various dark areas as the eyes since the background was very complex, unlike the background in the first database.

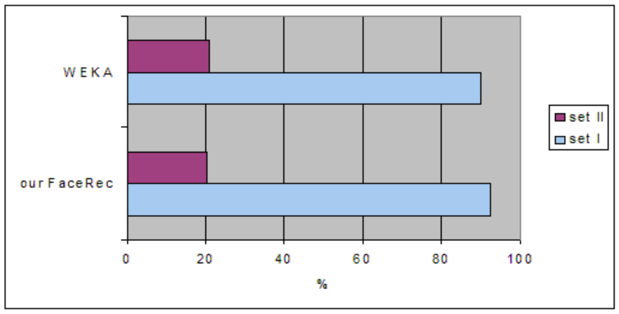

Figure 3: Performance for two very different image sets

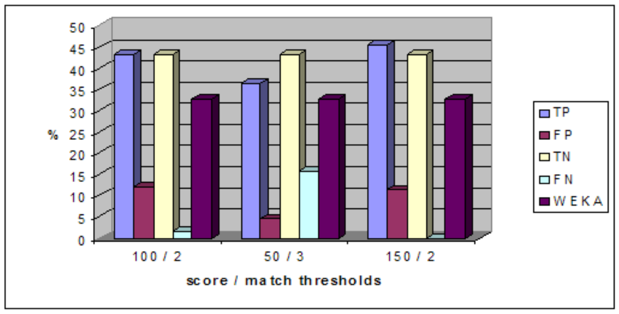

Figure 4: Performance of Face Recognizer with original images and three different threshold configurations

5.1.2 Results for the Third Image Set

We populated our image database in consecutive steps, with images of different people added at different times. We also implemented the extraction of various features at different times. In the figures below, we present some of the results we obtain in the second half of our research process.

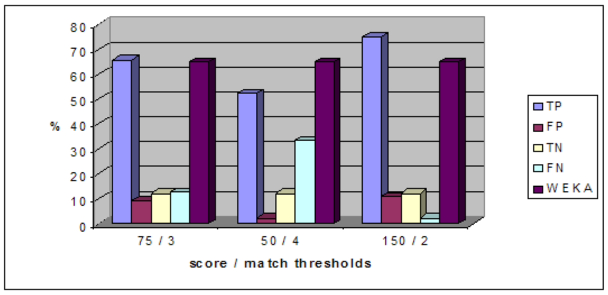

Figure 5: Performance of Face Recognizer with images modified in Picasa and three different threshold configurations

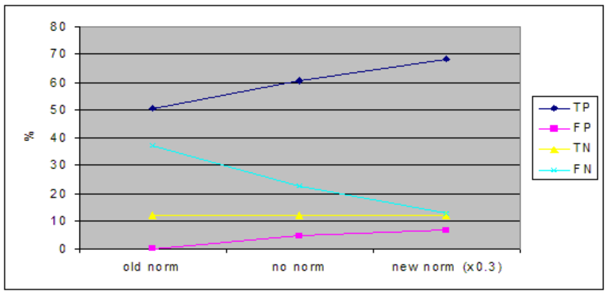

Figure 6: Role of normalization scheme

Figure 4 describes the performances of our system with 132 images taken with a web-camera, none of which were modified after their taking. The chart shows the true positive, false positive, true negative, false negative, and WEKA scores as described above.

Figure 5 above illustrates the Face Recognizer performance with the same 132 whose contrast has been modified using the "I'm Feeling Lucky" option of Google's Picasa program. As seen from the chart, the performance of the system is significantly improved since the images are clear, with higher contrast, and features are more likely to be discerned and extracted correctly.

The importance of a good normalization scheme is evident from Figure 6. Such a scheme is crucial as the features in the fingerprint have highly varying ranges. For example, the RGB values for color are in the range 0 to 255, while the values for various ratios are between 0 and 1.

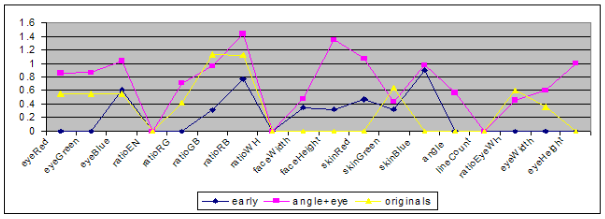

Figure 7: Weights for features as assigned by WEKA

The chart in Figure 7 compares the values which WEKA assigned to the facial features at different times of the development of our face recognition system: early stage with fewer images, addition of measurements for angle of chin and eye height and width, and original images which have not been modified through Picasa.

For research purposes, at one stage of the development of our project we experimented with splitting the third image set (described below) into two parts, one that we only used for training and the other only for testing. For the 20 images in the testing set, we obtained a true positive score of 80% and false positive of 10%. In comparison, when the 20 images were also part of the testing set, the score under the current version of the program was 85% true positives and 0% false negatives. Thus, while the performance rate decreases for unknown images, this decrease is small.

As expected, by varying the values for the match and score thresholds, we obtained different results. In particular, by increasing the match threshold (thus requiring a higher number of the same name to appear in the top five matches) and decreasing the score threshold (or allowing matches to be only slightly different from the examined image), we achieved low false positive scores but higher false negative scores. This threshold scheme is more appropriate for a vending-machine type identification, since we would like to be sure that we are not using the account of person B to pay for the products which person A wants to purchase. On the other hand, by decreasing the match threshold and increasing the score threshold, we achieve lower false negative and higher false positive scores. In this AMBER-alert type case, it is important not to miss any potential criminals, and we are willing to flag some individuals for further examination even when these are not actual suspects in order to minimize the chance of missing dangerous suspects.

Our Face Recognizer system often performs better than WEKA's classifier. This observation could be explained through the nature in which both systems find matches. WEKA uses a decision tree, and an image has to satisfy a set of conditions in order to be recognized as a proper match. However, if it fails just one condition, the match is not found correctly. On the other hand, our face recognition system uses a similarity score where each feature has a weight rather than a system of yes/no conditions. Even if an image fails to meet one of WEKA's conditions, it would still be recognized properly by our classifier as long as the difference between its value that has been rejected by WEKA and the database record is not too large.

5.2 Observations

Throughout the course of our research, we made numerous observations about factors that affect the performance of our face recognizer and the process of face recognition as a whole. We also implemented the extraction of some some feature which did not prove to be useful in our work.

5.2.1 Features Useful and Not Useful in Final Recognizer

At the beginning of our work, we used a different scoring scheme, which assigned a similarity score of 1 (closest match), 2, 3, or 4 (not likely to be a match) to every record in the database, using different thresholds for the differences between the features of the examined image and the database records. However, since we could not calculate perfectly what these thresholds must be, this scoring scheme only resulted in moderate success.

We found that the distances between the eyes, nose, and mouth in a face were not as useful as one might suspect. For the first image set, we wrote a method which found the location of the mouth by looking for a darker region in a specified range underneath the nose tip. While the extraction and use of this feature in the scoring scheme was useful for this image set, it was not useful for the second two image databases as a mouth could often not be found. However, we discovered that this failure would not affect the performance of our Face Recognizer significantly since the weights which WEKA assigned to the ratios between the eyes / nose / mouth distances were very small. By examining these ratios, we found that they were often the same for all people. The ratio of the distance between the eyes and that between the eye-line and the nose was often 2.0, that of the distance between the eyes and that between the eye-line and the mouth was 2.5, and the ratio of the eyes to nose versus eyes to mouth distance was 0.8. Thus, we found that these ratios would not be useful in our code. In fact, in the later part of our work in which we discarded the extraction of the mouth location and only computed the first of the above-mentioned ratios, WEKA assigned a weight of 0 to this ratio.

The skin color was normally assigned a high weight by WEKA, often higher than the weight for the eye-color, and it proved to help our face recognizer perform with better accuracy.

5.2.2 Problems Faced and Comparison with Other Recognizers

The three main problems for our Face Recognizer were the lighting, background, and color of skin. Unlike some other face recognition systems, ours relies on the use of color, and thus the RGB values we extracted were highly dependent on the amount of light in the location where the pictures were taken. In fact, while testing the recognizer with two different image sets and adding the source as a parameter, we discovered that this parameter was detected by WEKA as one of the most important ones and was placed near the top of the decision tree which WEKA generated.

The background in the images was also one of our concerns due to the likelihood that the face detecting algorithm would detect a false face. Thus, we tried to use a background which included as few features as possible. Unfortunately, the face detection algorithm often worked incorrectly when we tried to detect the face of a person with very dark skin.

One major advantage of our face recognizer in comparison with other system is that the moderate rotation and tilt of the face do not affect performance significantly. While other face recognizing systems use matrices and compare the images on a pixel-by-pixel level, ours extracts features no matter at what angle they are located from other features.

5.3 General Concerns

5.3.1 Feature Altering

There are some features which our algorithm uses that can quite easily be altered. For example, one can wear colored contact lenses and thus decrease substantially the chance that our face recognizer would detect a match. We tested this hypothesis by recording the fingerprint for the same person wearing clear contacts (brown eye color) and green contacts. For each of the photos taken with green contact lenses, the top five matches only included other green-eyed photographs, and the same applied for the brown-eyed photographs. Other features, such as the skin color, can also be altered in order to disable recognition, although not as easily.

5.3.2 Ethical Issues

Face recognition can be used as a surveillance method which many people might criticize. It invades the privacy of the subjects, and it also performs close analysis of their facial features and records this information. While the record database can normally be kept secure, recording private data of this kind without authorization involves a moral decision.

Additionally, the classification and recognition of individuals based on facial features such as the width of their eyes or their skin color requires discrimination based on race. Again, such discrimination presents an ethical concern.

6 Conclusions and Future Work

This paper presented our implementation of a face recognizing system using features of a face including colors, distances, and ratios. Using its two degrees of freedom, our system allows two modes of operation, one that results in very few false positives and another which results in few false negatives. We have demonstrated various concerns related to the face recognition process, such as the lighting and background conditions in which the facial images are taken. Our approach is relatively independent of head tilt and rotation.

Our system could be improved in the future through the development of a face detection algorithm which is less prone to incorrectness and failure, performs well regardless of the skin color, and uses a more complete fingerprint which would help achieve higher true positive and lower false positive and negative results. A more extensive feature set would also prevent the chance of tricking the system through the alteration of facial features.

7 Acknowledgments

We would like to thank the Computing Research Association and its Distributed Mentor Program which sponsored this research. We are also thankful to the members of the Princeton Electrical Engineering department who posed for our camera so we could use their faces in the third image set.

8 Project Webpage

The website documenting our work on this project can be found here:

`http://www.princeton.edu/~kovashka`

This page provides some useful information and links to resources.

References

[1] "Alphonse Bertillon." Wikipedia, The Free Encyclopedia. 11 Jul 2007, 06:24 UTC. Wikimedia Foundation, Inc. 20 Jul 2007. http://en.wikipedia.org/wiki/Bertillon.

[2] "Computational Vision: Archive." 17 Mar 2005. http://www.vision.caltech.edu/html-files/archive.html.

[3] Cox, Ingemar J., Joumana Ghosn, and Peter N. Yianilos. "Feature-Based Face Recognition Using Mixture-Distance."

[4] "Facial recognition system." Wikipedia, The Free Encyclopedia. 25 Jun 2007, 05:53 UTC. Wikimedia Foundation, Inc. 20 Jul 2007. http://en.wikipedia.org/wiki/Facial_recognition_system.

[5] Fergus, R., L. Fei-Fei, P. Persona, and A. Zisserman. "Learning Object Categories from Google's Image Search."

[6] Ian H. Witten and Eibe Frank. "Data Mining: Practical machine learning tools and techniques." 2nd Edition. Morgan Kaufmann, San Francisco, 2005.

[7] "Java Media Framework API (JMF)." http://java.sun.com/products/java-media/jmf/.

[8] "Koders - Source Code Search Engine." http://www.koders.com.

[9] Kremer, Ulrich. "Face-Code" (written in C).

[10] Restom, Adel. "Visage." http://sourceforge.net/projects/visage-hci/.

[11] "SARANA - A Space Aware and Resource Aware Dynamic Network Architecture." http://www.research.rutgers.edu/ũli/Sarana/.

[12] Spacek, Libor. "Face Recognition Data." 21 Jun 2007. http://cswww.essex.ac.uk/mv/allfaces/index.html.

[13] Viola, Paul and Michael Jones. "Rapid Object Detection using a Boosted Cascade of Simple Features."

[14] Wang, Yong. "Situation-Aware Optimizations in Challenging Networks." Princeton University, Computer Science Department. Ph.D. dissertation, August 2007.