Facial Expression Recognition for a Sociable Robot

by

Wing Hei Iris Tang

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2007

Author ....
Department of Electrical Engineering and Computer Science
August 31, 2007

Certified by ....
Rodney Brooks
Professor
Thesis Supervisor

Accepted by ....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Facial Expression Recognition for a Sociable Robot

by

Wing Hei Iris Tang

Submitted to the Department of Electrical Engineering and Computer Science
on August 31, 2007, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science and Engineering

Abstract

In order to develop a sociable robot that can operate in the social environment of humans, we need to develop a robot system that can recognize the emotions of the people it interacts with and can respond to them accordingly.

In this thesis, I present a facial expression system that recognizes the facial features of human subjects in an unsupervised manner and interprets the facial expressions of the individuals. The facial expression system is integrated with an existing emotional model for the expressive humanoid robot, Mertz.

Thesis Supervisor: Rodney Brooks
Title: Professor

Acknowledgments

To my advisor, Rodney Brooks, who has been tremendously supportive and patient with me. Thank you for giving me the many opportunities and resources to complete this thesis.

I would like to thank Lijin for her guidance and mentorship over the years. I have been very fortunate to be able to work on Mertz. Thank you to my lab mates, Eduardo, Juan, and Varun, who have been supportive of me during this time. I would also like to thank Tyler for helping me get started on this project and pushing me along the way.

Most of all, I would like to dedicate this to my mom, who did not have a chance to see this thesis to completion, but is the very reason for my being here.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Robotics has emerged as a field that can transform how technology is used in our society. Because of their ability to interact with us physically, robots can be integrated into the human environment in ways that traditional computers and other technologies have been unable to achieve. The 2001 US Census Bureau report on disabilities stated that 34.9 percent of Americans aged 80 and older needed personal assistance to perform daily activities [17]. Robotic technology can fulfill the needs of the elderly by fetching objects that are difficult to reach, helping them in and out of bed, and alerting medical personnel to an emergency. These robotic applications would not only benefit the elderly, but they would also have the potential to be ubiquitous in every part of society.

Even with devices such as the Roomba being welcomed into many households, the presence of robots in our everyday lives is still limited. There are many challenges to overcome before robots can interact with humans in a natural way and operate autonomously within the human space. One of the greatest challenges for a sociable robot is to be able to interact with people, to interpret the emotions of the person and to respond accordingly. This ability is important for day-to-day human-robot interactions, especially when the human is not a trained user, such as the elderly, and is being assisted by the robot in home tasks.

The goal of this thesis is to build a facial expression recognition system that interprets the facial expressions of individuals in an real time. The facial expression system is then integrated with an existing emotional model for the social robot, Mertz. This thesis takes a step to understanding the emotions of the person by identifying five basic facial expressions (happy, surprised, sad, angry, and neutral). These expressions are then integrated with an emotional model to generate emotional responses for the robot.

In this thesis, we attempt to bridge between two different technology directions. First, in the field of social robotics, the research is focused on facial animation and the emotion system. The facial animation research investigates how to create robotic expressions that are realistic [3]. The emotion system research focuses on high-level conditions such as the presence or absence of a desired stimulus, as the motivation for the robot's emotive responses [3, 23]. Second, in the field of computer vision, the research is focused on automated facial expression detection, identification, and analysis [14].

Chapter 2

Related Work

2.1 Robot Platform

This thesis is designed for a humanoid robot, Mertz [2]. Mertz is an active-vision head robot with thirteen degrees of freedom (DOF). Its face has actuated eyebrows and lips, capable of generating facial expressions. It is able to vocalize through a phoneme-based speech synthesizer. To capture visual input, the robot has two digital cameras, one mounted in each eye. The cameras produce 640 x 480, 24-bit color images at the rate of 30 frames per second. It also has a set of voice array desk microphones to gather audio input.

The robot is capable of tracking faces, motion and color segments. It has a multi-

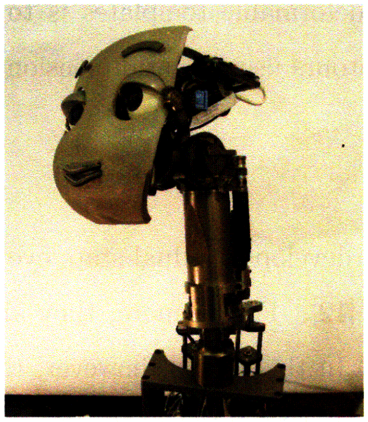

Figure 2-1: Mertz, an active-vision humanoid head robot.

modal attention system that determines where the robot should focus its attention among all the different stimuli within the robot's sensory ego-sphere.

2.2 Emotional Model

This thesis is integrated with the emotional model designed for Mertz [22]. The emotional model processes the face inputs of the robot, maintains an internal model of emotions, and expresses the appropriate emotional responses for the robot. In generating emotion values, the model uses a three-dimensional affect space of arousal, valence, and stance [3]. It is the goal of this thesis to provide the facial expression analysis as inputs to the emotional model, thereby enriching the outputs of the model.

2.3 Facial Features Detection

There have been many approaches in research in tracking specific facial features.

Lip Tracking

One common way of lip tracking is through the use of templates [10, 13]. These templates often use two parabolic arcs to model the lip shape. Tian et al. combined the template method with color and motion information in their lip tracking research. Another variation of using deformable templates is to combine it with the snake algorithm, coding the lip contours using multidimensional vectors [4].

Eye Tracking

For eye tracking, Tian et al. developed a dual-state eye model that detects whether the eyes are open or closed [12]. Furthermore, they also combine the model with eye corner and iris tracking. In this model, however, they assumed that the initial location of the eye is given in the first frame of the image sequences. This assumption is not possible for this thesis because our system needs to operate in an online manner since the robot operates continuously and autonomously in environments with various different people.

2.4 Facial Expression Analysis

There has been many ongoing research on recognizing facial expression from video sequences. Cohen et al. used naive Bayes classifiers to learn different facial motion features and a hidden Markov model (HMMs) to automatically segment and recognize facial expressions [6]. Gokturk et al. used a 3D model-based tracker that tracks both the pose and shape of the face [9]. The expression classification is then performed by a support vector machine.

Instead of recognizing a set of prototypic expressions, Tian et al. developed an automatic face analysis system that recognizes facial expressions into action units of the Facial Action Coding System (FACS) [14]. The action unit recognition is performed by a three-layer neural network. Lucey et al. used the Active Appearance Model (AAM) to recognize the FACS action untis [16].

Chapter 3

Facial Features Detection

The facial feature detection system accepts three types of inputs: a real-time camera capture, a sequence of images, or a single image. With a camera capture, the system samples at every 40 milliseconds. We captured three datasets, Dataset A, B and C, at this rate (more details in Section 5.1).

3.1 Face Detection

3.1.1 Modified AdaBoost Face Detector

Detection of faces in the image was implemented using a modified Viola-Jones AdaBoost face detector, which uses a cascade of boosted classifiers working with Harr-type wavelets as weak feature detectors [24, 15, 18]. We use a trained cascade classifier and object detector provided by the OpenCV library [21]. The object detector outputs a number of rectangular areas each containing the detected face.

We improve upon on the speed and detection rate of the face detector by reducing the size of the input image by a factor of three and run face detection on the image. If no face is detected at the original orientation, we rotate the image 15° and -15° and try face detection on each of the rotated images, in case the subject has tilted his or her head. Once we have detected a face, we smooth the location of the rectangular area by averaging the location of the newly detected face with the location of the last

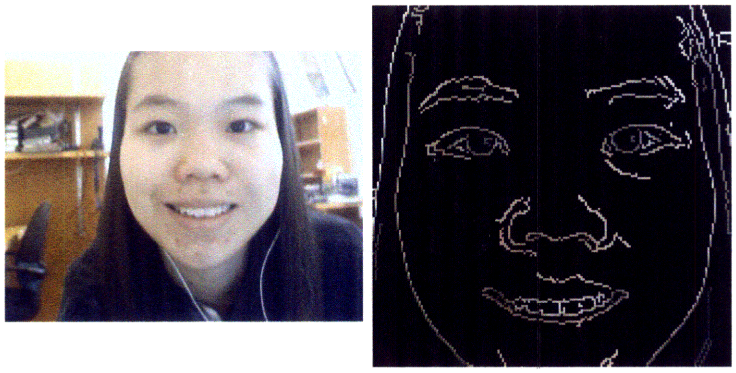
Figure 3-1: An example of edge detection from Dataset A.

detected face.

The face detection takes an average of 14.21 milliseconds per image. This rate is fast enough to allow for face detection on a live video stream in real time.

3.1.2 Canny Edge Detector

Once we detect a face using the face detector, we pass the detected rectangular face area through a Canny edge detector provided by the OpenCV library [5, 21]. The output from the edge detector is used to improve on the eye and mouth detection algorithms. We choose a threshold value of 35 for the edge detector in an attempt to balance between having too many and too few edge details on the dataset faces (see figures 3-1 and 3-2). The threshold parameter is highly sensitive to the quality of the face images. As a further improvement, we would have liked to adapt the threshold parameter to faces of each individual, thus providing the optimal edge details on the individual's face.

3.1.3 Masking the Detected Face Image

We tried to improve upon the detected rectangular face area by adding a mask function that blacks out the non-face portions of the image rectangle, such as the neck and the space beyond the contours of the face. We search from two thirds of the height down along the y-axis of the edge detected image. At each y-coordinate, we search from the outside inwards along the x-axis, looking for the first edge pixel. We have

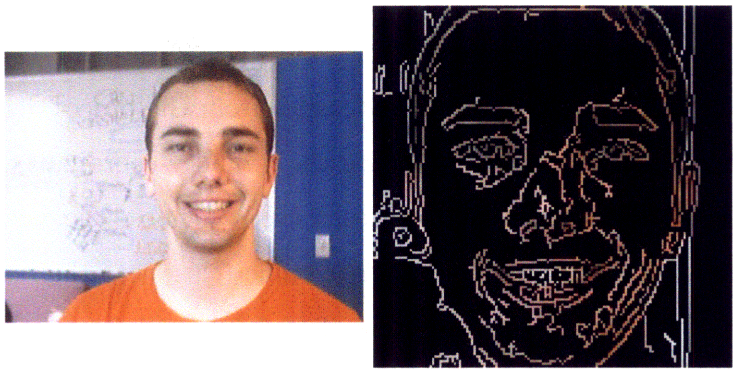
Figure 3-2: An example of edge detection from Dataset B.

two bounding lines ($x = w/h * y - w/2$ and $x = -w/h * y + 3w/2$) for the search area. This will either find the contour of the face within the search area or assume that the contour lies outside of the search area. Ultimately, we abandoned the use of this masking function because of the reliability of the edge detection and the variability of the face contours. The edge detector does not always capture the face contour, which lead the masking function to black out necessary parts of the face. Also, parts of the facial contour may be well outside the detected rectangular face area that the bounding lines will cut off parts of the mouth. If we were able to smooth out the face contour and use that as our bounded search area, then this masking function would be more successful.

3.2 Eye Detection

3.2.1 Modified AdaBoost Eye Detector

We use the same modified AdaBoost object detector provided by the OpenCV library [21] to detect eyes. We use a trained cascade classifier for the eye which has been provided by the Watson library [19]. Because we can rely on robust output from the face detector, we input only the top half of the detected face image to the eye detector. As a refinement to increase the number of successful eye detections, we try the object detector at most three times with increasingly accurate but slower parameters: ($min\_neighbors = 2$ & $scale\_factor = 1.1$, $min\_neighbors = 1$ &

$scale\_factor = 1.01$, $min\_neighbors = 0$ & $scale\_factor = 1.001$).

Once we receive the candidate eye rectangles from the object detector, we sort the candidates into left and right eyes using the x- and y-coordinates. Afterward, if we still have more than one candidate for each eye, we choose the candidate with, in order of preference, the greatest y-coordinate value (closest to the bottom) and the smallest width. Choosing the highest y-coordinate value minimizes the likelihood of the rectangle containing parts of the eyebrow, since this issue arises often with this cascade classifier. We choose the smallest width because we want to minimize the non-eye area of the rectangle.

3.2.2 Refining the Detected Eye with Eye Corners

We further improve upon the detected eye area by eliminating non-eye areas using the Canny edge detection output of the face. For each candidate eye, we start half way down the height of the eye and search along the y-axis. At each y-coordinate, we search for the first edge pixel (x_min) from the outside inwards toward the center of the eye, updating only when the x-coordinate differs from the very first edge pixel found at the half-height ($x\_firstMin$) by more than 10. This helps identify the two eye corners of each eye, so we can refine the eye-region to span only from one corner to the other (see figures 3-3 and 3-4). We start the search from the middle of the eye because the corners of each eye are closest to the halfway point. When we previously searched from the top down, we often received mistaken results, usually because parts of the eyebrow would be the furthest points out, thus preventing us to find the actual eye corners.

After doing this analysis for both candidates, we compare the widths and heights of the two eyes. If they vary by more than the thresholds (5 pixels for width, 2 pixels for height), we correct for the differences by using the smaller width or height.

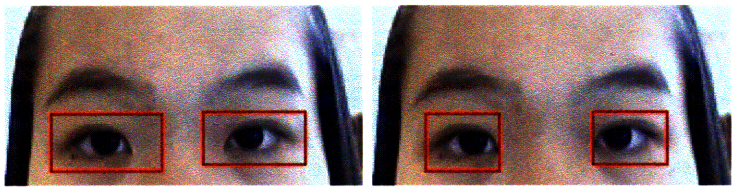
Figure 3-3: An example of eye corner refinement from Dataset A.

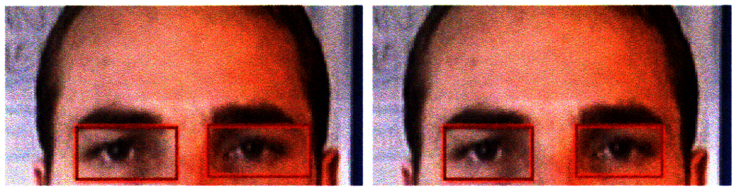
Figure 3-4: An example of eye corner refinement from Dataset B.

3.2.3 Eyebrow Detector

To find the eyebrows, we search within the rectangle defined by the width of the eye, and the height from the top of the eye to half way to the top of the face bounding rectangle. While this bounded search area may cut off the ends of the eyebrow, it also lowers the probability of including the hair line and other lines that result from, for example, furrowing of the brows. We use the Canny edge detection output during our search, assigning all the edge pixels we found as eyebrow pixels.

We improve upon our search results for each eyebrow by eliminating outliers from the set of points that define the eyebrow (see figure 3-5). In our algorithm, we eliminate points that are outside of the range of $(x\_mean \pm 1.8 * x\_std, y\_mean \pm 1.2 * y\_std)$. The parameters are chosen because we want to be more tolerant of variations in the x-coordinates since the eyebrow can be pretty wide, but we want to be less tolerant of variations in the y-coordinates since the height of the eyebrow is generally uniform.

3.2.4 Finding the Eyebrow Angles

Once we have the set of points defining each eyebrow, we fill in the polygon and find the line through it using the Hough transform [7]. We use the implementation provided by the OpenCV library, decreasing incrementally from a threshold of 40

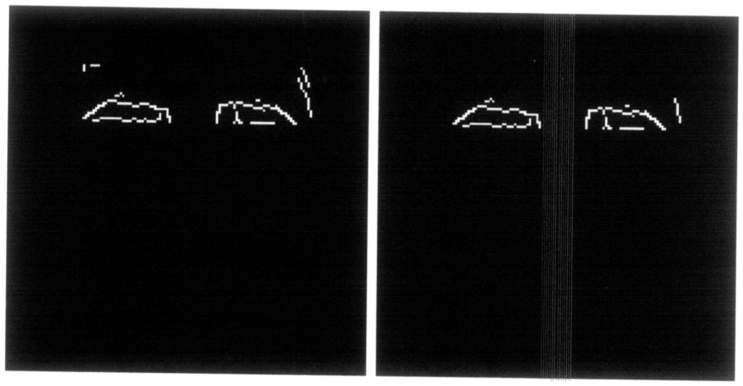
Figure 3-5: An example of eyebrow detection and refinement from Dataset B.

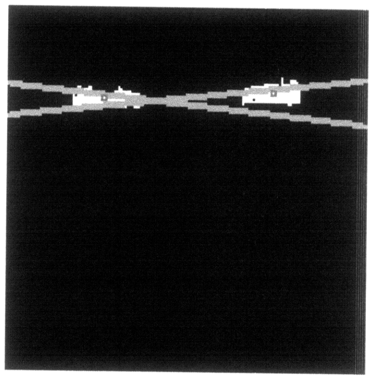
Figure 3-6: Eyebrows of 6° and 6° from Dataset A.

until a set of lines is found [21]. We average the results of the Hough transform and measure the angle from the center of the x-axis, positive toward the top of the face and negative otherwise (see figure 3-6).

3.2.5 Finding the Distance between the Eyebrows and the Eyes

With the detected eye image, we attempt to locate the iris by looking for the darkest part of the eye. We first tried looking for a 3x3 square with the lowest RGB sum, but often this mistakenly identified the corners of the eye instead of the iris. We then modified the search shape to a 5x5 hexagon without the center pixel, but again it located the darker corners of the eye instead of the iris. We now calculate the central moment of the eye image, with far better results. We also calculate the central moment of the eyebrow and find the vertical distance between the two moments.

3.3 Mouth Detection

3.3.1 Modified AdaBoost Mouth Detector

We train a cascade classifier for the mouth using a set of mouth samples provided by the AVCSR library [20]. Because the mouth detection rate is low, we try two different object detectors to look for the mouth. For both detectors, we use the bottom third of the face image as input. First, we try the OpenCV object detector at most three times with increasingly accurate but slower parameters: ($min\_neighbors = 2$ & $scale\_factor = 1.1$, $min\_neighbors = 1$ & $scale\_factor = 1.01$, $min\_neighbors = 0$ & $scale\_factor = 1.001$). If this detector fails to identify any candidate mouths, we then try an object detector provided by the AVCSR library, again with the same three sets of parameters. The AVCSR object detector differs from the OpenCV object detector in that it restricts the width of the detected object to the range [min_width, max_width], instead of to a minimum window size. The AVCSR object detector provides an additional layer of detection because changing widths is suitable for the search for the mouth.

Once we receive the candidate mouth rectangles from the object detector, we choose the candidate with the smallest width as our detected mouth.

3.3.2 Analyzing the Mouth State

We analyze whether the detected mouth is open or closed by combining the following two sets of information.

Using the Color Information

First, we analyze the color information of the detected mouth. We check the RGB values of each pixel within the detected mouth and find the percentage of white and black pixels over the sum of red, white, and black pixels. The reason is that when someone's mouth is open, the white pixels represent his teeth and the black pixels represent the inside of the mouth, with the red pixels representing the lips. By

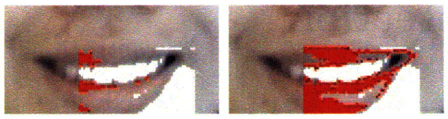
Figure 3-7: An example of increased number of correctly identified red pixels by switching from an absolute scale to a relative scale in red classification from Dataset A.

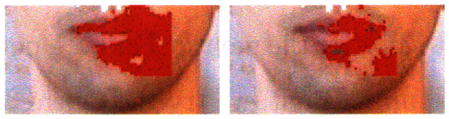
Figure 3-8: An example of increased number of correctly identified red pixels by switching from an absolute scale to a relative scale in red classification from Dataset B.

checking the ratio of these three colors, we can estimate whether the mouth is open.

We discover that the RGB values vary greatly among the different datasets. So in order to classify whether a pixel is red, we have to use a relative scale based on the average RGB values of the whole detected mouth instead of an absolute scale (see figures 3-7 and 3-8). The detection of red pixels improve the most by switching to the relative scale. We keep the absolute scale for the white and black pixels.

We also make another refinement to the red, white, and black pixel classification of the detected mouth. We notice that, within the detected mouth rectangle, the corners of the rectangle would get misclassified as white, thus throwing off the color ratio. We know that when the mouth is open, the white pixels representing the teeth should be within the red pixels representing the lips. So we eliminate all the white pixels that fall outside of the range of $(x\_mean_{red} \pm 3 * x\_std_{red}, y\_mean_{red} \pm 1 * y\_std_{red})$, where the range is calculated from the set of red pixels (see figure 3-9). We choose these parameters because we want to be tolerant in the x direction since the mouth can be pretty wide, but less tolerant in the y direction since the teeth should be tightly confined within the lips.

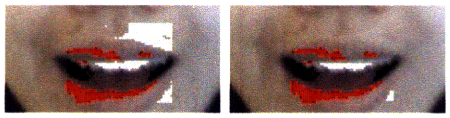
Figure 3-9: An example of eliminating white pixel outliers from Dataset A.

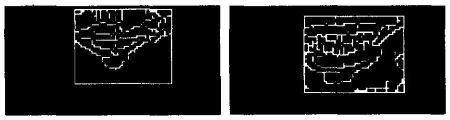
Figure 3-10: A 45% increase of edge pixels in the detected mouth from Dataset B.

Using the Edge Detection

Second, we analyze the percent change of the number of edge pixels in the detected mouth from that of the previously detected mouth. We note that when the mouth switches from closed to open, the number of edge pixels increases because the teeth are captured in the edge detection (see figure 3-10).

Combining the Information Sets

In doing this analysis, we now take into account the location and state of the previously detected mouth. We are able to take advantage of this information because we are observing a continuous sequence of images, as in the situation of human-robot interactions. We use this additional information in several ways. First, if the mouth detector fails to detect any candidates in the current face image, we use the previously detected mouth location for the rest of our analysis. This was a necessary improvement because the mouth detection rate is much lower than the face and eye detection rates. Second, we use the pixel and state information from the previous mouth to strengthen our analysis.

```
if(lastMouthState == MOUTH_CLOSED && change > .3)
{
  if(ratio < .05)
    isMouthClosed = true;
  else
    isMouthClosed = false;
}
else if(lastMouthState == MOUTH_OPEN && change < -.3)
{
  if(ratio > .5)
    isMouthClosed = false;
  else
    isMouthClosed = true;
}
```

```
else if(ratio > .15 && ratio < .7)
{
  if(lastMouthState == MOUTH_CLOSED && (abs(change) < .1 || change < -.4))
    isMouthClosed = true;
  else
    isMouthClosed = false;
}
else if(ratio >= .7)
{
  isMouthClosed = false;
}
else //ratio <= .15
{
  if(lastMouthState == MOUTH_OPEN && (abs(change) < .15 || change > .4))
    isMouthClosed = false;
  else
    isMouthClosed = true;
}
```

In our algorithm, if the mouth was previously closed and the percent edge pixel change is above a threshold of 30%, we determine that the mouth is now open. In the opposite case, a previously open mouth and a percent change of less than -30% will yield a now closed mouth. In both of these cases, we safeguard against our assumptions by also checking whether the color information is congruent. For example, if the ratio of white and black pixels is very low, we reject the assumption that the mouth has opened.

In the fourth case, the ratio of white and black pixels is very large, so we determine that the mouth is open. In the third case, the ratio indicates that the mouth is likely to be open. In the fifth case, the ratio is very small, so the mouth is likely to be closed. But to safeguard against our assumptions, we check with the pixel information to see whether there is a large change and also whether the change is in the right direction. For example, when the ratio is small but the previous mouth is open and the percent change is small or is largely positive, the mouth is more likely to have remained open. In these cases, the thresholds are chosen to favor determining that the mouth is open because the color ratio is unreliable.

Chapter 4

Facial Expression Analysis

4.1 Parameters to the Facial Expression Recognition System

Upon passing the input image through the facial features detection system, four parameters ($mouthState, browAngle, browIrisDistance, deltaDistance$) are generated from the image as inputs to the facial expression recognition system. These parameters are the mouth state (open or closed), the average of the left and right eyebrow angles, the average of the left and right distances between the central moment of the eyebrow and the central moment of the eye, and the percent change of the current $browIrisDistance$ from the $browIrisDistance$ of the previous image.

We try two versions of calculating the $browIrisDistance$ from the input image. At first, we use directly the average distance measured in each image. However, we discover that this parameter varies significantly depending on the individual (21.92 for Dataset A, 14.94 for Dataset B, 11.65 for Dataset C). So in our second attempt, we normalize this parameter with the mean and standard deviation of each dataset ($(browIrisDistance - mean)/std$). The normalized parameter yields better results than the unnormalized parameter, as discussed later in Section 5.2. However, the normalized parameter also requires a different set of assumptions about the inputs presented to this facial expression recognition system. First, because our datasets are collected according to the individual, we have the labels necessary to determine the mean and standard deviation for each individual. Since we do not have facial recognition in our facial expression recognition system, we have to assume that "person" label would be provided to us as an input. In the case of Mertz, there is an unsupervised incremental face recognition framework that can provide this labeling to our system [2]. The second concern is over calculating the mean and the standard deviation over the whole dataset. To align with the incremental framework of Mertz, it would be more appropriate to build the two measures incrementally as well. But performance would likely go down and we would also need to store and revisit the entire history of the parameter upon each new input.

4.2 Facial Expression Training and Recognition

We conduct experiments on classifying and recognizing facial expressions using two different methods: a naive Bayes classification and a self organizing map. We use a training set of 118 images (31 from Dataset A, 55 from Dataset B, 32 from Dataset C). We also manually code the facial expressions of these images with four possibilities: happy, surprised, sad, and angry. The results of these experiments are presented in Chapter 5.

4.2.1 Naive Bayes Classification

The first method we use on our training set is a naive Bayes classifier, which is probabilistic model based on Bayes' theorem. The trainer is presented with four-dimensional feature vectors from the training set. During the training, we compute the parameters of the model by counting the occurrences of the training data feature vectors in each feature bin. For *mouthState*, there are only two bins since the value is binary. For each of the other three features, we choose five bins that are equally spread over the range of the feature from the training data.

We make a simplification in our classification by limiting the number of expression types to two, combining happy and surprised to a "positive" expression, and sad and angry to a "negative" expression. We than use the probabilistic model built from this classification to recognize expressions on future inputs.

4.2.2 Self Organizing Map

The second method we use to classify the facial expression training set is the self organizing map [8, 1, 11]. Self organizing map is a neural network that uses unsupervised learning to allow visualization of data by reducing the dimensions of the training samples. In our case, we reduce the four dimensions of our dataset to a two-dimensional self organizing map. We chose a map of size 5x5, initializing each node's weight vector with random values between 0 and 1. We find the best matching unit (BMU) to the current input vector by calculating the euclidean distance between each node's weight vector and the input vector. Because the range of our four parameters are quite different, we normalize the euclidean distance by the standard deviation of each parameter. Upon finding the best matching unit, we update the weight vectors of the nodes that lie within the BMU's neighborhood. The neighborhood starts off with a radius of 2.5, shrinking according to an exponential decay function as time and the number of iterations increase. For each node that lies within the the neighborhood, its weight vector is updated proportional to the distance the node is from the BMU with a learning rate that decays over time.

While a self organizing map provides a way to classify the training set in an unsupervised manner, we still need to apply the expression labels to the nodes of the map in order to use the map to recognize expressions from other data images. We use the expression data coded for the training set to label each node, with probabilities for each expression calculated based on the percentage of each expression is found within the node. When we receive an input image for recognition, we find the best matching node to the input and output the expression with the highest overall probability. The probability is calculated similar to the neighborhood used in training; we use not only the probabilities associated with the matching node but also those within the node's neighborhood with each weight proportional to the distance from the matching node.

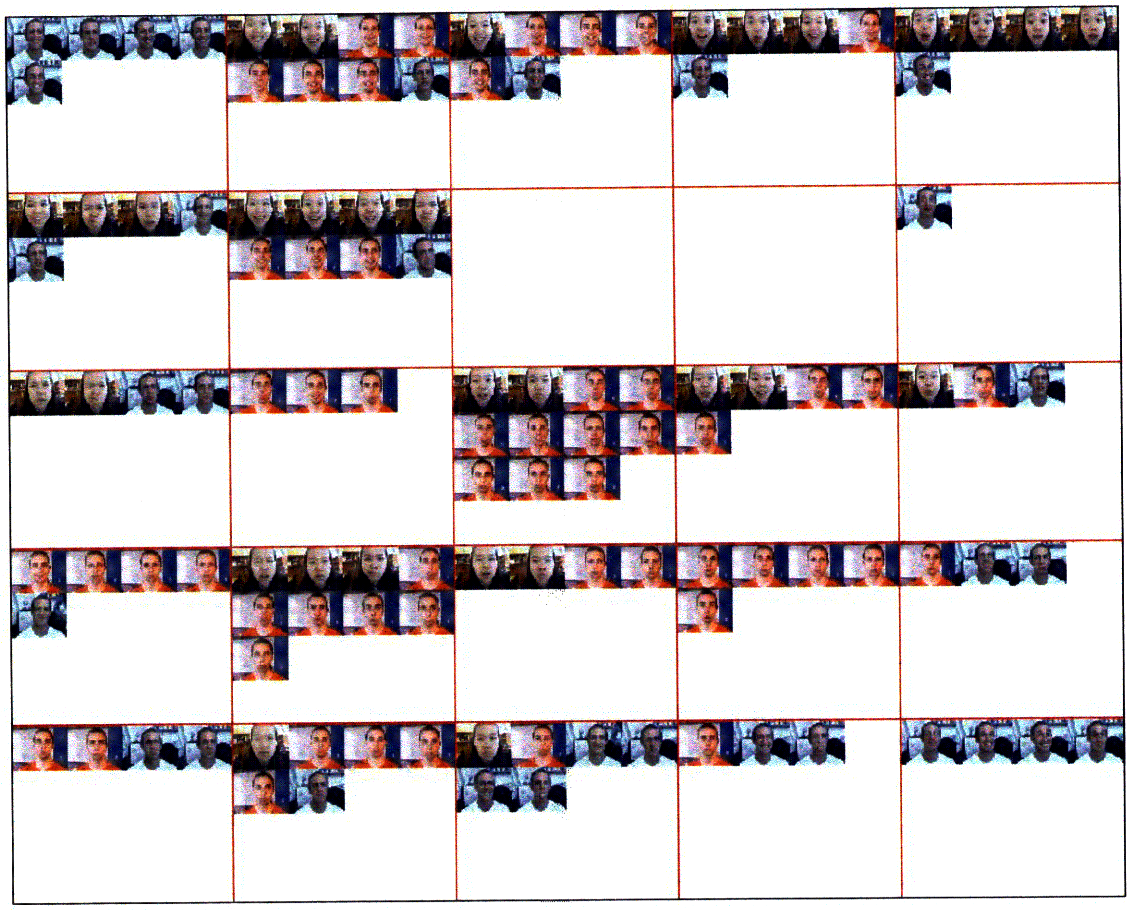

Figure 4-1: An example of a self organizing map classifying facial expressions.

Chapter 5

Results

5.1 Datasets

We collected 320 x 240 color frontal facial images from three individuals in an indoor environment. We did not control for illumination or background. The images were captured at every 40 milliseconds or 25 frames per second. There are 84 images in Dataset A, 154 images in Dataset B, and 131 images in Dataset C. Datasets A and C were captured with a Logitech QuickCam Zoom camera and Dataset B was captured with a Logitech Quickcam Pro 3000 camera. The quality and brightness of the images vary among the datasets, yielding to different performance results.

5.2 Facial Features Detection

Face Detection

Face detection rate is 99% for Dataset A and 100% for Datasets B and C (see figures B-1, B-2, B-3). The face detection takes an average of 14.83 milliseconds for Dataset A, 15.01 milliseconds for Dataset B, and 12.87 milliseconds for Dataset C.

Eye Detection

The detection rate for both eyes and eyebrows is 93% for Dataset A, 97% for Dataset B, and 72% for Dataset C (see figures B-4, B-5, B-6). The lower performance of Dataset C is due to the less precise output from the modified AdaBoost eye detector. The matching rectangle for the eyes tend to be much bigger then the area of the actual eye, thus often including the eyebrow inside the rectangle. Thus, it is much harder to detect the eyebrow.

Mouth Detection

We manually coded the mouth states of the images in the three datasets in order to check the performance of our mouth detection algorithm. We find that the algorithm correctly identified the mouth states at a rate of 80% for DatasetA, 77% for Dataset B, and 84% for Dataset C (see figures B-7, B-8, B-9).

5.3 Naive Bayes Classification and Facial Expression Recognition

Using the training data with the normalized *browIrisDistance* parameter, the naive Bayes classification yields a 77% rate of matching the manually coded expressions of the training data. Using the training data without the normalized *browIrisDistance* parameter, the naive Bayes classification yield a 80% rate of matching the manually coded expressions of the training data.

5.4 Self Organizing Map and Facial Expression Recognition

Using the training data with the normalized *browIrisDistance* parameter, the 3x3 self organizing map classification yields a 58% rate of matching the manually coded expressions of the training data to four dimensions and 79% to two dimensions (see figure B-13 and table A.2). Using the training data without the normalized *browIrisDistance* parameter, the 3x3 self organizing map classification yields a 57% rate of matching the manually coded expressions of the training data to four dimensions and 80% to two dimensions (see figure B-12 and table A.1).

We create a test image set using images from Dataset B that were not used in the training set, throwing out blurry images. The test image set consists of 94 images. We use the 3x3 normalized self organizing map to recognize the test data (see figure B-14). We also manually coded the expressions associated with the test data set. The recognition yields a 24% rate in four dimensions and a 26% rate in two dimensions.

5.5 Discussion

The performance of the face and eye detections is good across the three datasets. The mouth and mouth state detection performance could benefit from the use of a deformable template.

As for the classification, it is surprising to see that the naive Bayes classification performs at the same level as the self organizing map classification for the two expression dimensions. The self organizing map in this case has the advantage of being unsupervised. It is also surprising to see the unnormalized parameter success rate to be slightly higher than the normalized parameter success rate. In the self organizing maps though (see figures B-12 and B-13), we can see that in the unnormalized version, the cells are populated more according to the individuals. In the normalized version, most cells contain a mixture from the three datasets. This question of whether to use normalization will need to be explored with a bigger sample size.

The facial expression recognition performance was disappointingly low. I believe the experiment suffers from the following problems. First, the test data were a poor representation of the four expressions we are trying to recognize. The manual coding was difficult because many of the data does not really fit in any of the four categories. I think we can improve our performance by adding a neutral expression. It would also be beneficial to have a test data set that consists of images outside of our three datasets, such as images from the Cohn-Kanade Facial Expression Database. Second, the manual coding of the expressions is unreliable. It was done by one person, without any objective measures. We can improve this source of error by having multiple people code the data and choose the group consensus. Alternatively, we can use a more objective system such as the Facial Action Coding System (FACS).

Appendix A

Tables

Happy	0.33	0.33	0.67
Surprised	0	0.67	0.08
Sad	0.56	0	0.08
Angry	0.11	0	0.17
Happy	0	0.25	0
Surprised	0	0.25	0
Sad	0	0.50	0
Angry	0	0	0
Happy	0.60	0.71	0.11
Surprised	0.20	0.29	0.11
Sad	0	0	0.48
Angry	0.20	0	0.30

Table A.1: Facial expression probabilities associated with Figure B-12.

Happy	0.50	1.00	0.72
Surprised	0	0	0.07
Sad	0.25	0	0.07
Angry	0.25	0	0.14
Happy	0.08	0.33	0.29
Surprised	0.08	0.67	0.71
Sad	0.54	0	0
Angry	0.31	0	0
Happy	0.63	0.21	0
Surprised	0.25	0.12	0
Sad	0	0.49	0.50
Angry	0.13	0.19	0.50

Table A.2: Facial expression probabilities associated with Figure B-13.

Appendix B

Figures

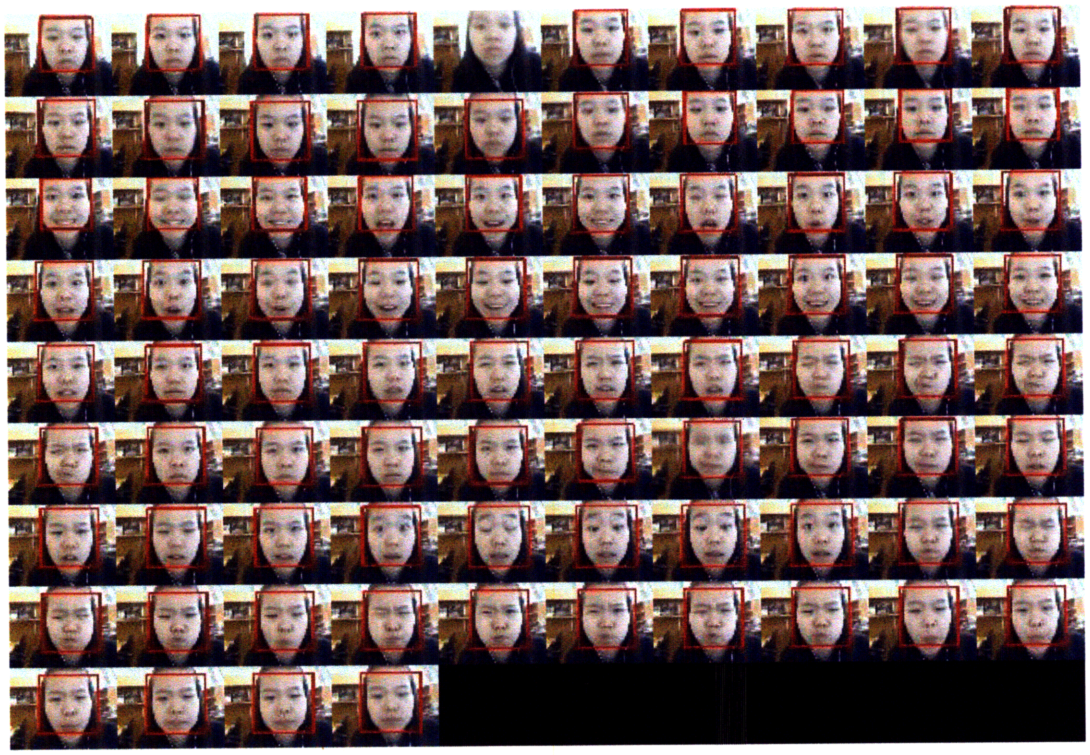
Figure B-1: Face detection results from Dataset A.

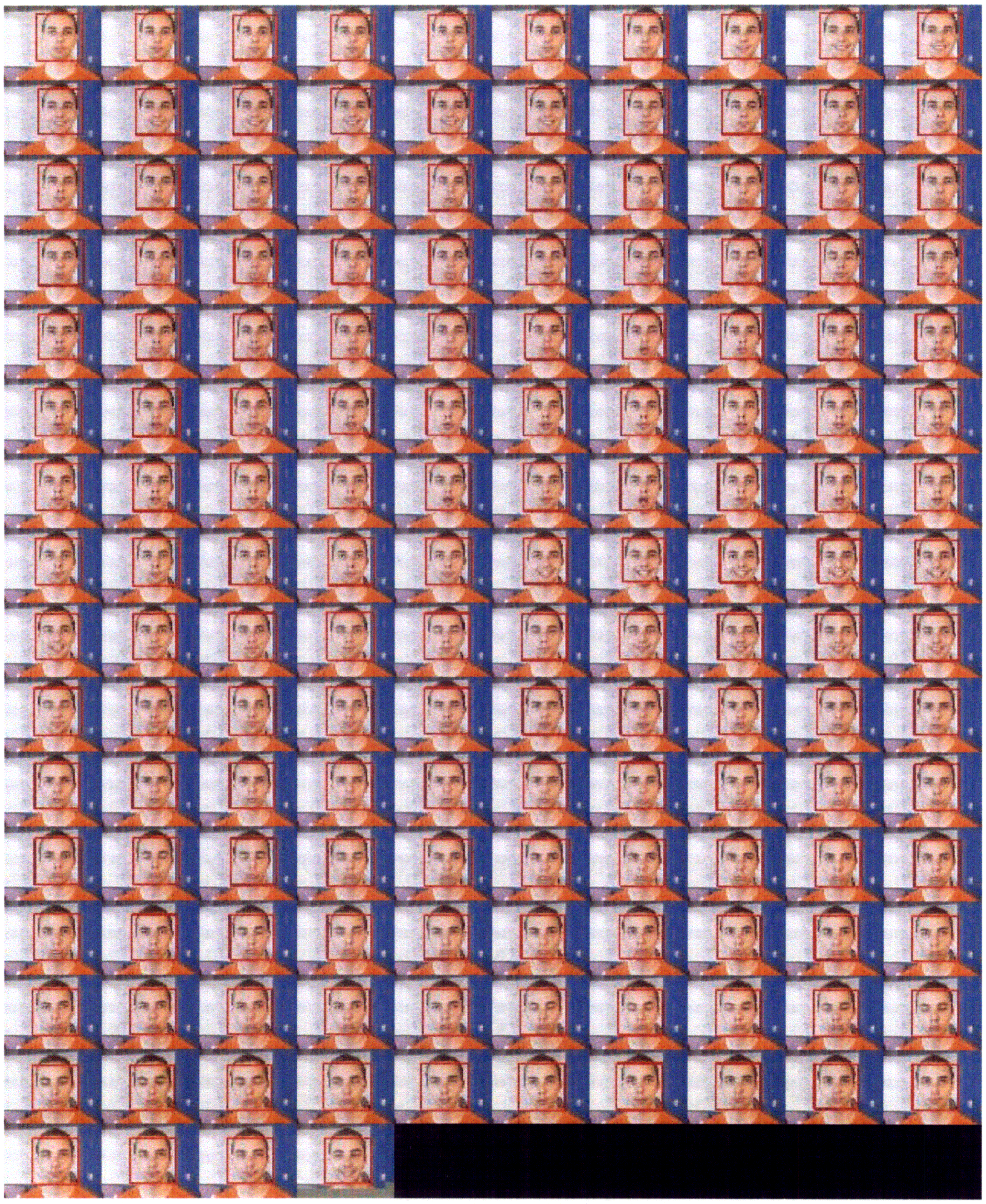

Figure B-2: Face detection results from Dataset B.

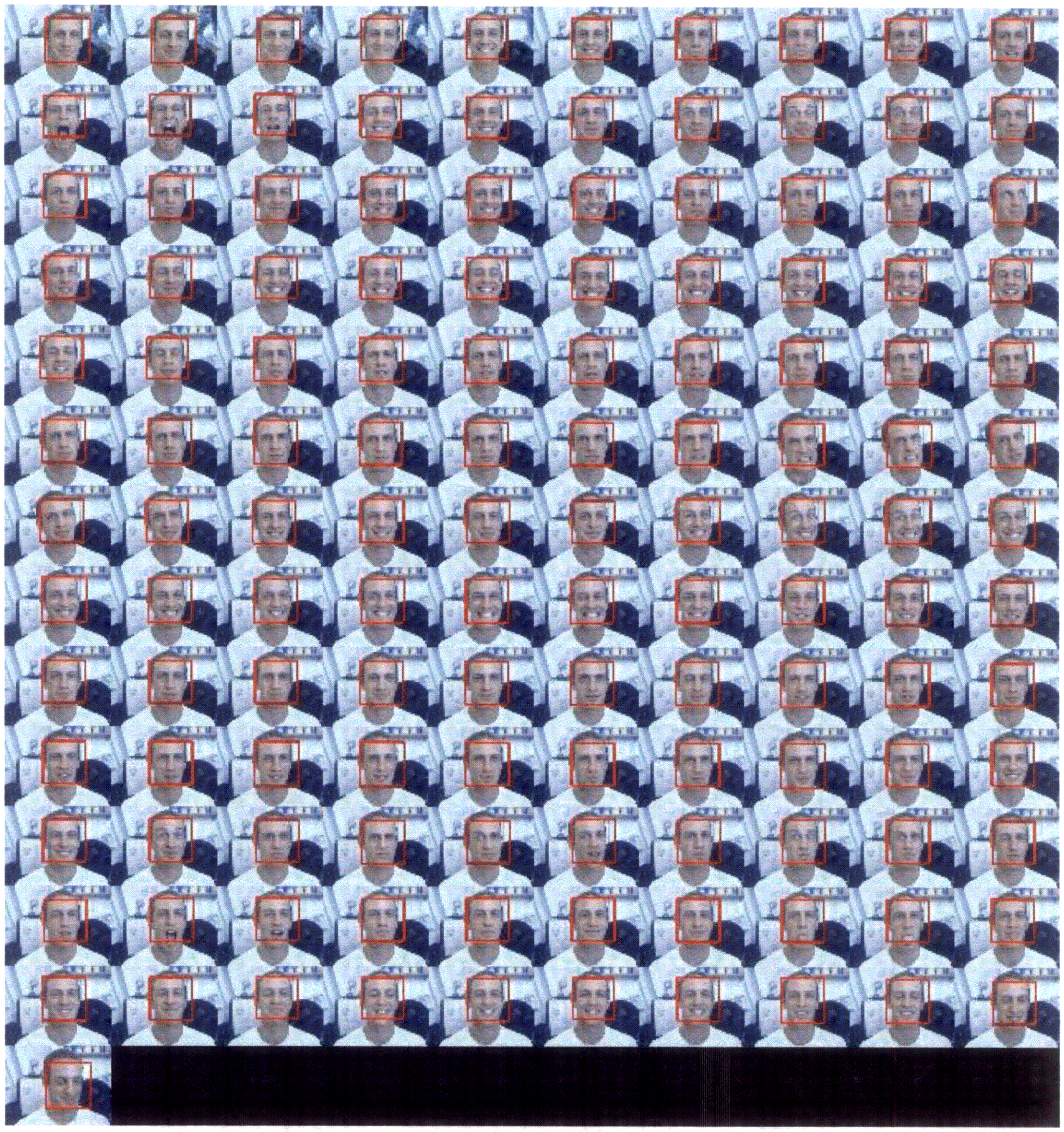

Figure B-3: Face detection results from Dataset C.

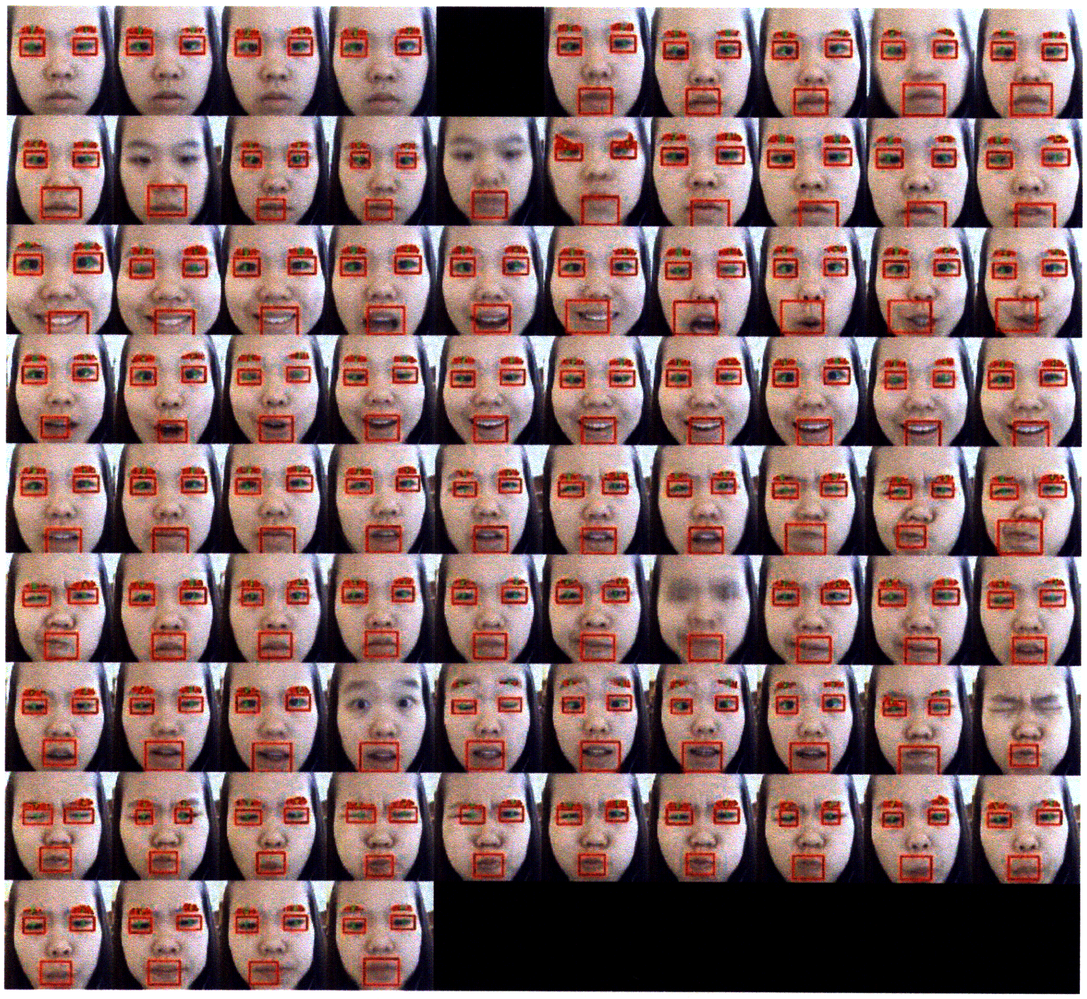

Figure B-4: Eye, eyebrow and mouth detection results from Dataset A.

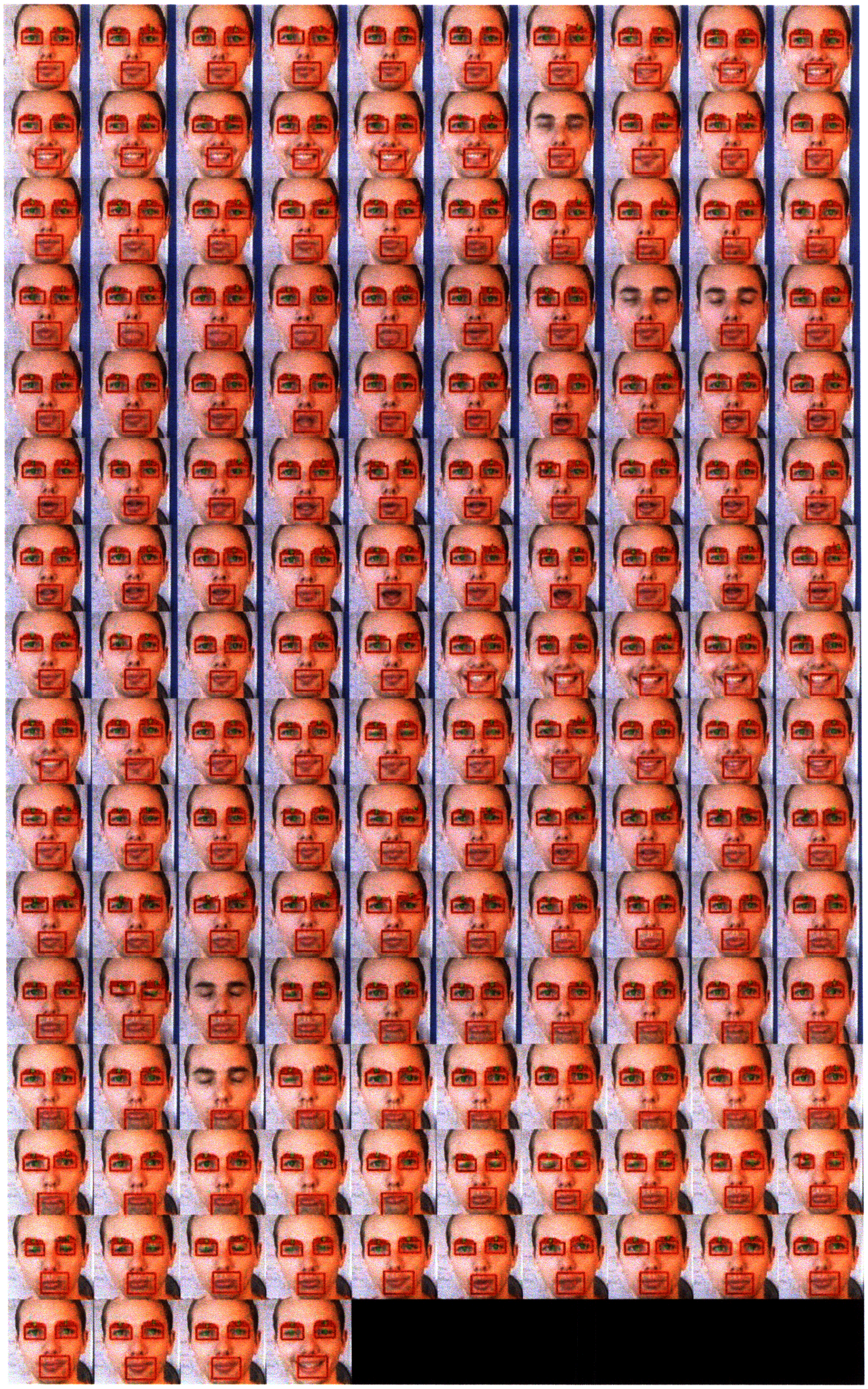

Figure B-5: Eye, eyebrow and mouth detection results from Dataset B.

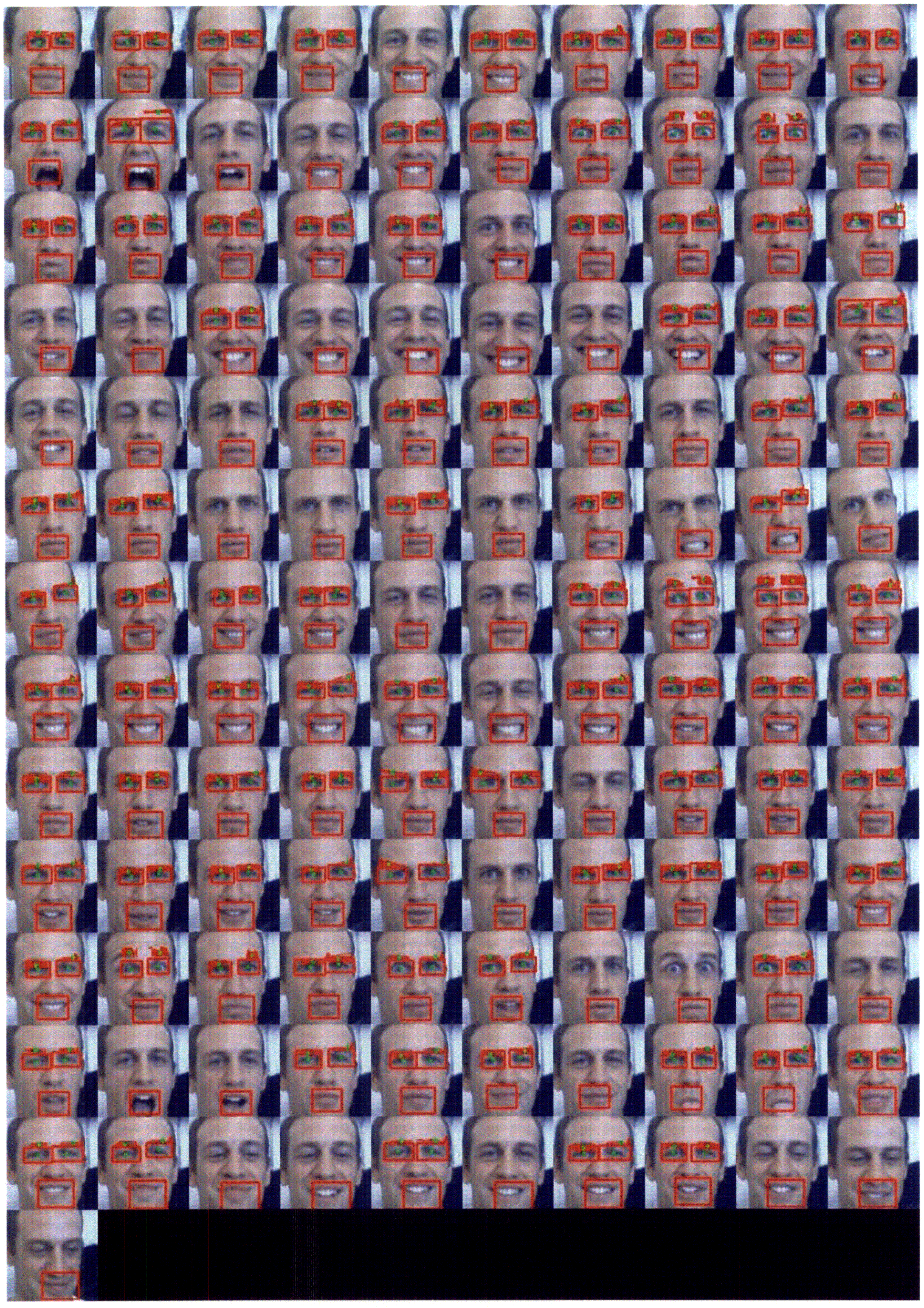

Figure B-6: Eye, eyebrow and mouth detection results from Dataset C.

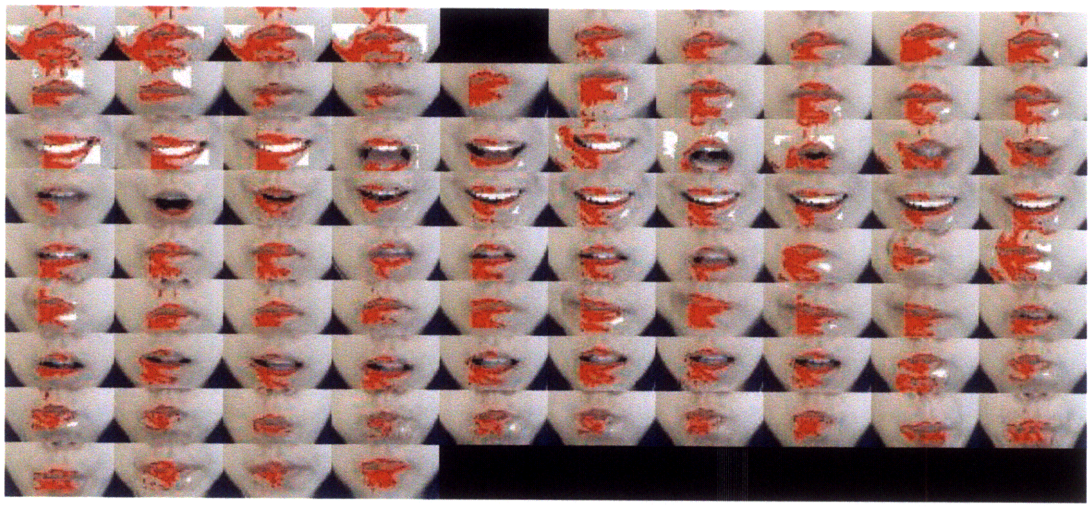
Figure B-7: Mouth analysis results from Dataset A.

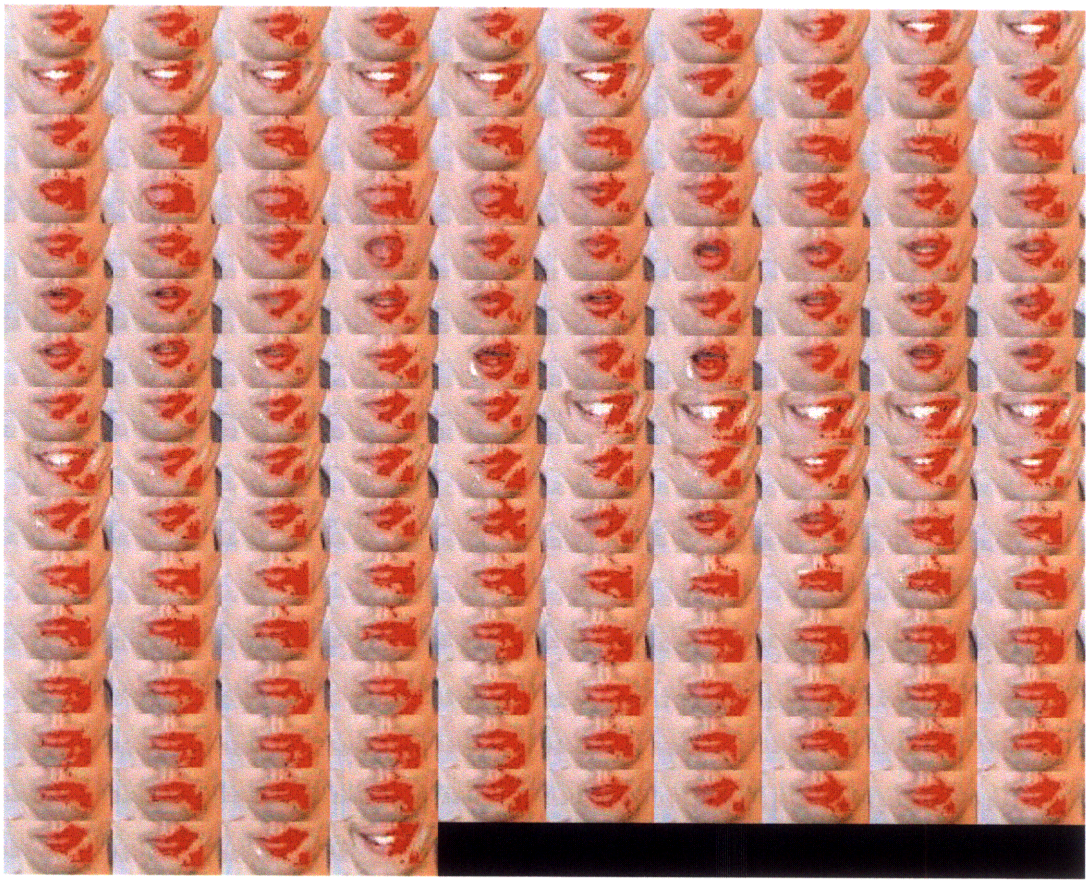
Figure B-8: Mouth analysis results from Dataset B.

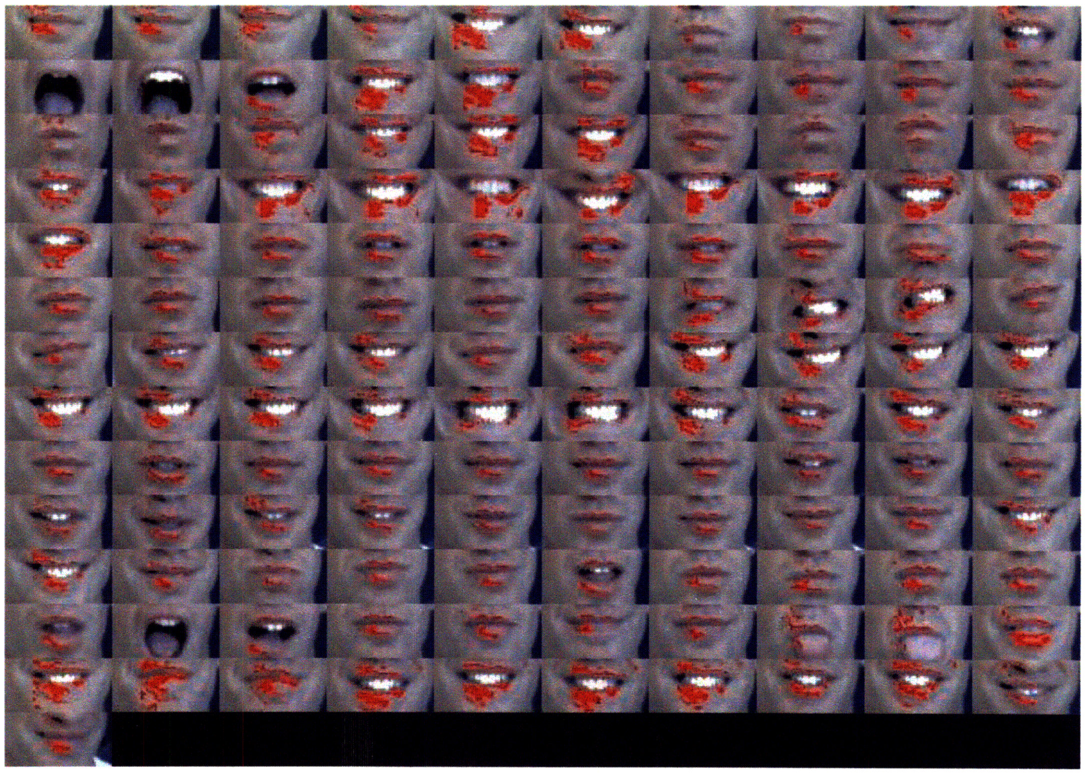
Figure B-9: Mouth analysis results from Dataset C.

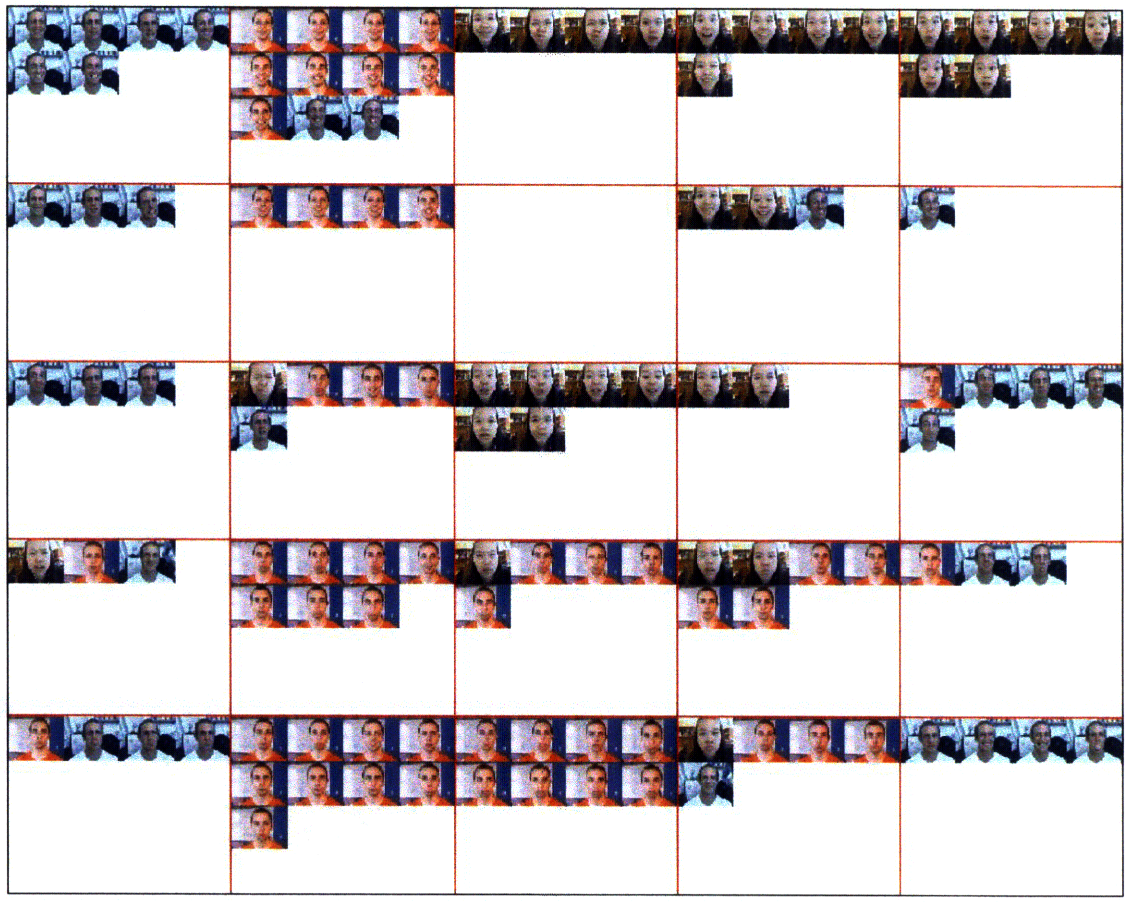

Figure B-10: A 5x5 SOM with 5000 iterations on the training set.

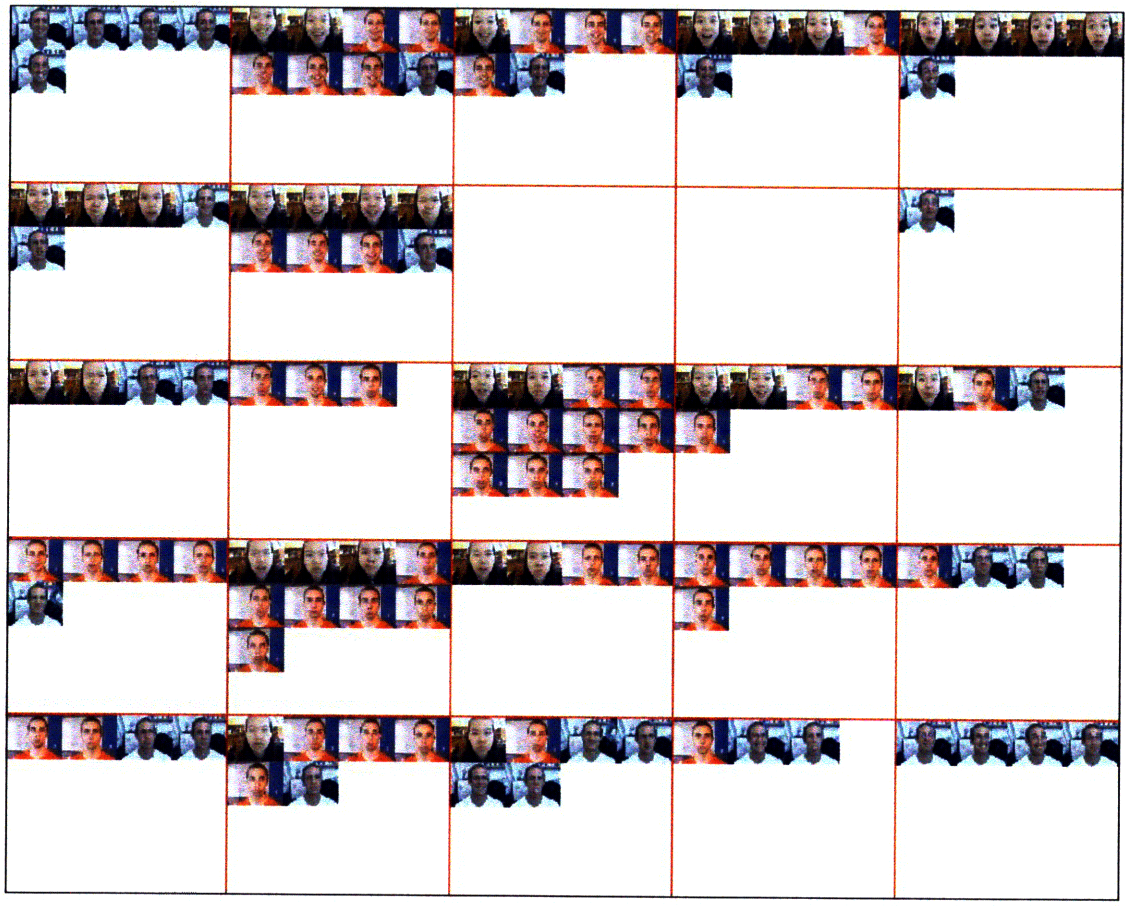

Figure B-11: A 5x5 SOM with 5000 iterations on the training set with the normalized *browIrisDistance* parameter.

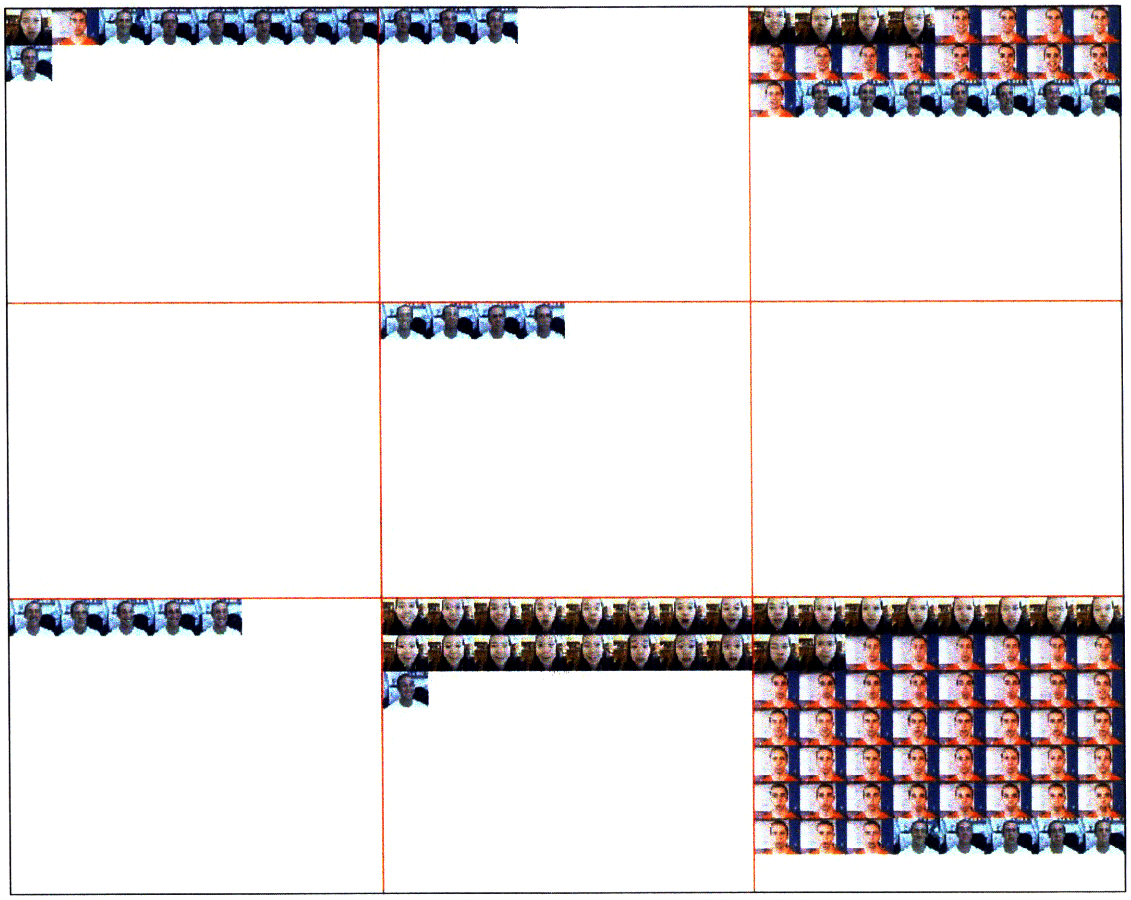

Figure B-12: A 3x3 SOM with 5000 iterations on the training set.

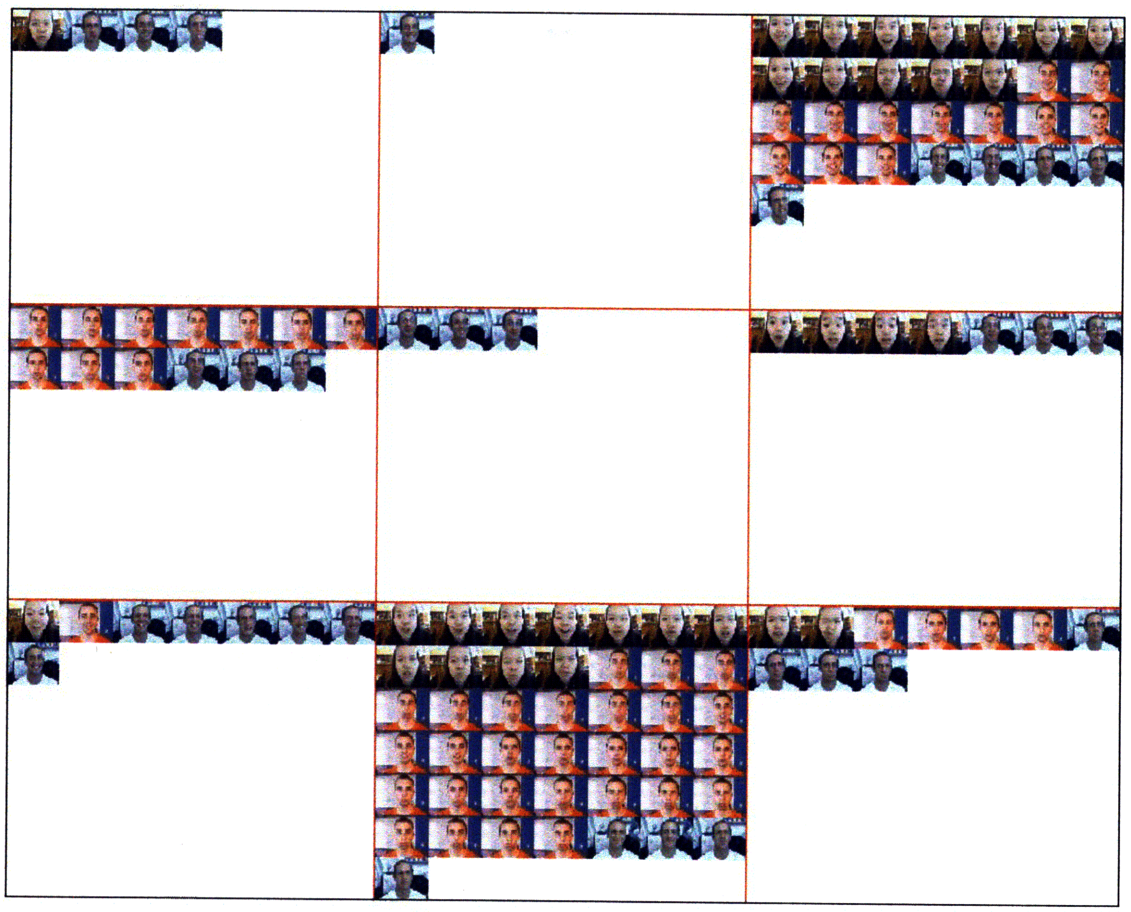

Figure B-13: A 3x3 SOM with 5000 iterations on the training set with the normalized $browIrisDistance$ parameter.

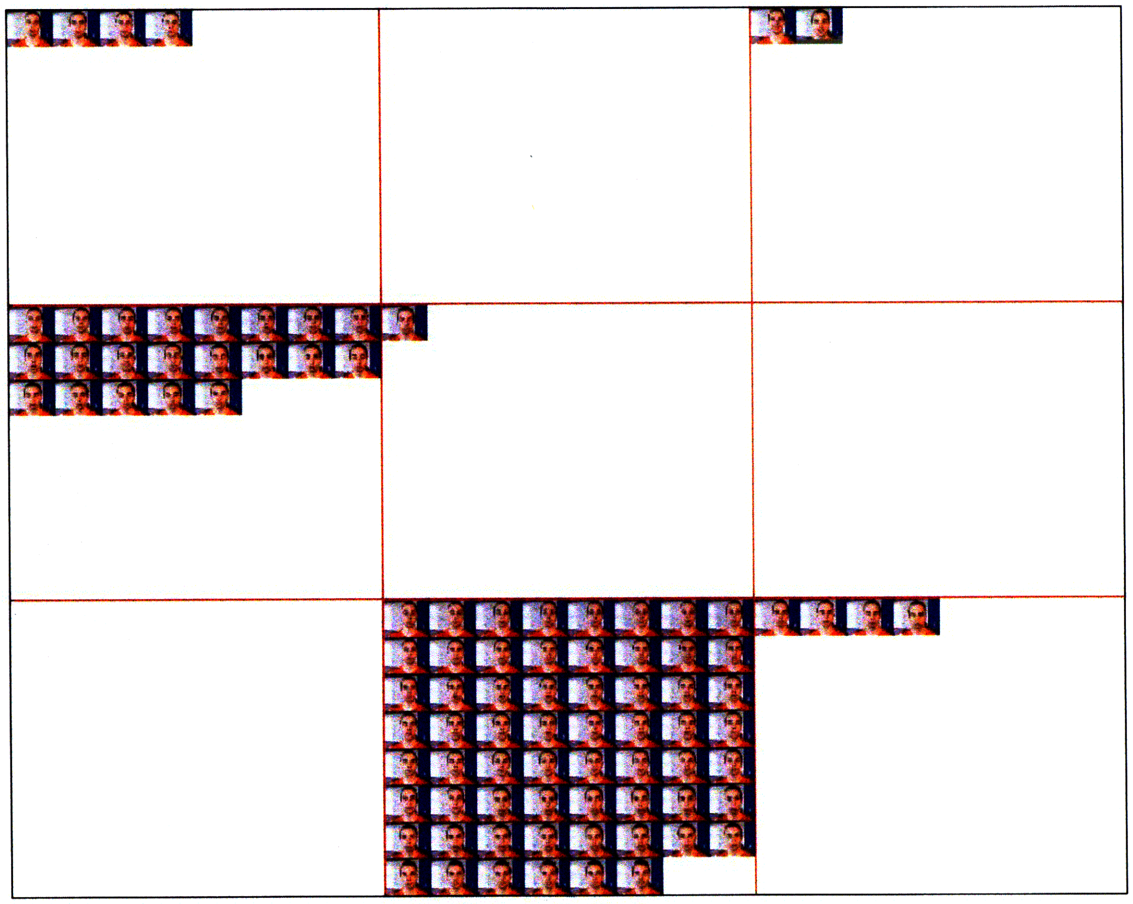

Figure B-14: Facial expression recognition results using the normalized 3x3 SOM.

Bibliography

[1] *Kohonen's Self Organizing Feature Maps Tutorial.* Available: http://www.ai-junkie.com/ann/som/som1.html.

[2] Lijin Aryananda. *A Few Days of a Robot's Life in the Human's World: Toward Incremental Individual Recognition.* PhD thesis, Massachusetts Institute of Technology, 2007.

[3] Cynthia Breazeal. *Designing Sociable Robots.* The MIT Press, 2002.

[4] Christoph Bregler and Yochai Konig. Eigenlips for robust speech recognition. *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1994.

[5] John F. Canny. A computational approach to edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.

[6] Ira Cohen, Nicu Sebe, Ashutosh Garg, Lawrence S. Chen, and Thomas S. Huang. Facial expression recognition from video sequences: Temporal and static modeling. *Computer Vision and Image Understanding*, 2003.

[7] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Comm. ACM*, 15:11–15, 1972.

[8] Tom Germano. *Self Organizing Maps Tutorial*, 1999. Available: http://davis.wpi.edu/ matt/courses/soms/.

[9] Salih Burak Gokturk, Jean-Yves Bouguet, Carlo Tomasi, and Bernd Girod. Model-based face tracking for view-independent facial expression recognition. *Proc. of the 5th International Conference on Automatic Face and Gesture Recognition*, 2002.

[10] Marcus E. Hennecke, K. Venkatesh Prasad, and David G. Stork. Using deformable templates to infer visual speech dynamics. *Signals, Systems and Computers*, 1994.

[11] T. Kohonen. *Self-Organizating Maps.* New York : Springer-Verlag, 1997.

[12] Ying li Tian, Takeo Kanade, and Jeffrey F. Cohn. Dual-state parametric eye tracking. *International Conference on Face and Gesture Recognition*, 1999.

[13] Ying li Tian, Takeo Kanade, and Jeffrey F. Cohn. Robust lip tracking by combining shape, color and motion. *Proc. of ACCV, Taipei, Taiwan*, 2000.

[14] Ying li Tian, Takeo Kanade, and Jeffrey F. Cohn. Recognizing action units for facial expression analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23:97–115, 2001.

[15] Rainer Lienhart and Jochen Maydt. An extended set of haar-like features for rapid object detection. *IEEE ICIP*, 1:900–903, September 2002.

[16] Simon Lucey, Iain Mathews, Changbo Hu, Zara Ambadar, Fernando de la Torre, and Jeffrey Cohn. AAM derived face representation for robust facial action recognition. *Proc. of the 7th International Conference on Automatic Face and Gesture Recognition*, 2006.

[17] Jack McNeil. Americans with disabilities: 1997. *U.S. Census Bureau Current Population Reports*, pages 70–73, 2001.

[18] Gerard Medioni and Sing Bing Kang, editors. *Emerging Topics in Computer Vision.* Prentice Hall, 2005.

[19] Louis-Philippe Morency. *WATSON: Real-time Head Tracking and Gesture Recognition Library.* Available: http://groups.csail.mit.edu/vision/vip/watson/.

[20] Ara V. Nefian, Lu Hong Liang, Xiao Xing Liu, and Xiaobo Pi. *Audio-Visual Continuous Speech Recognition Library.* Available: http://sourceforge.net/projects/opencvlibrary/.

[21] Intel Research. *Open Computer Vision Library.* Available: http://sourceforge.net/projects/opencvlibrary/.

[22] Wing Hei Iris Tang. Emotional model for a humanoid robot, mertz. MIT Undergraduate Advanced Project, 2006.

[23] Juan D. Velasquez. *When Robots Weep: A Computational Approach to Affective Learning.* PhD thesis, Massachusetts Institute of Technology, 2007.

[24] Paul Viola and Michael J. Jones. Rapid object detection using a boosted cascade of simple features. *IEEE CVPR*, 2001.