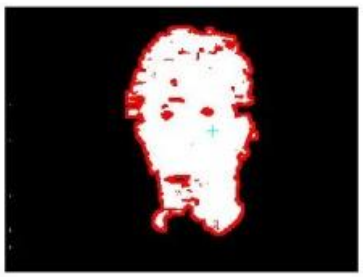
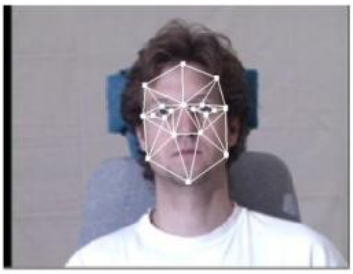
18-551, Fall 2007
Group 5 Final Report

TAKE ME AT FACE VALUE
Pose Estimation and Active Shape Models

Sheethal Bhat, Abhay Mavalankar, Chandni Shah
{sbhat, amavalan, csshah}@andrew.cmu.edu





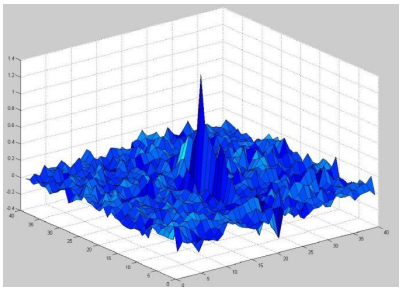


Table of Contents

Introduction

The Problem

There are many applications for facial recognition systems. Security is one of the most important and obvious applications for facial recognition. After recent terrorist attacks across the world, security in airports, subway systems, and all modes of transportation has become of utmost importance. Highly sophisticated Facial Recognition systems can detect blacklisted individuals, thieves, or even high profile individuals without the use of identification cards. Facial Recognition systems can also reduce the use of fake identification cards in bars and casinos and thus reduce underage drinking and gambling. They can also replace keys for secure access areas such as bank vaults or even homes. In order to do this we must have a fast and effective method that can quickly process an image and identify the individual.

Common facial recognition methods are applied to the frontal images of the face. This may not always be practical. When you consider that most CCT cameras are installed in places that require these systems be placed at a height, we get side or top views of the faces at best. We need a way to be able to extract meaningful features of the face and then be able to process the data. Therefore, we need a system that can handle variation in angle.

The Solution

We propose the following solution

1. Face Detection – Finds the region of the face in an image using Skin Segmentation and Blob Extraction.

2. Active Shape Models (ASM) – Uses the information from Face detection to intialize the ASM and iteratively fits the face to obtain shape information. ASM allows for varying angle and expression in an image.

3. Pose Estimation – Uses shape information from the ASM and a 2-D MACE correlation filter to determine whether a person is facing Left, Straight, or Right. We can also detect in between poses to a certain degree of accuracy.

4. Facial Recognition – Uses shape and angle information from the ASM and Pose Estimation to recognize the person. (We did not implement this due to time constraints)

Previous Work

This is the first time Active Shape Model (ASM) has been implemented in 18-551. A similar algorithm called Active Appearance Model (AAM) was attempted by 'Big Brother and His Shifty Eyes,' Group 11 in Fall 2006. They failed to implement their algorithm on the DSK. AAM gives texture and shape information while ASM only give shape information of an image. We believe that since ASM is less computationally expensive than AAM it will be more easily implemented on the DSK.
For face detection we used skin segmentation and implemented our own algorithm for the morphological operations, which allowed for a faster implementation. 'Face Detection for Surveillance,' group 2, Spring 2002 also did face detection, however they used Skin segmentation and performed erosion and dilation as their morphological operations. They implemented a subset of the Schneiderman and Kanade algorithm. 'I Spy and Follow,' group 2 from Fall 2005 did face detection, however they used the Central Slice Theorem to help them detect an object.,
'I Spy and Follow,' group 2 from Fall 2005 used MACE filters for identification of their target objects such as cars and tanks. We used a 2-D MACE correlation filter in combination with the ASM information to determine Pose Estimation. Pose Estimation by ASM and correlation filters has never been done in 18-551 before.

Database

We will use the MULTI-PIE Database[1]. This Database has 68 people with 13 pose variations which is required for training the Model. We need to hand label all the images we need for the training. Test images need not be labeled.
For ASM testing we will use:

1. 20 people with three pose variations for each person, two images of each pose for training.
2. 4 people with three pose variations for each person
3. We will also choose 3 images with in between poses (Poses other than extreme Left, extreme Right, and Straight) for in-between pose estimation.

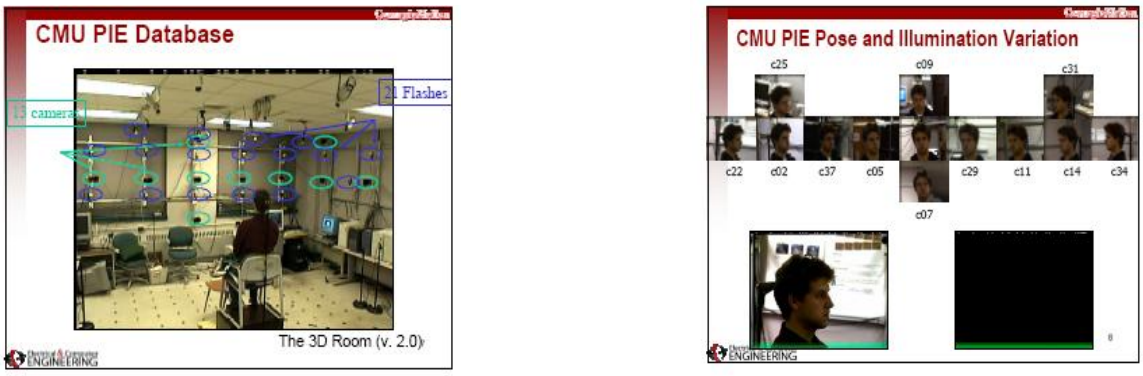


Fig 1: (a) Room where the images were taken, (b) Example set of images

[1] Baker, Simon. "Pie Database." The Robotics Institute. Carnegie Mellon University.
http://www.ri.cmu.edu/projects/project_418.html

Face Detection

The face detection is the initial step of our project. The purpose behind applying face detection on the DSK is to provide an initial position to the ASM. We start out with an image containing faces. The face detection attempts to find out the area containing the faces in the images. The coordinates of the face which is detected are passed to the next step as the initialization coordinates for the ASM.

The face detection algorithm uses the skin detection as the first step of the algorithm. It uses particular thresholds for the R, G and B components of the image. The thresholds are normalized with the sum of the R, G and B components to allow for different types of illumination. Next a series of morphological operations are executed to remove the small blobs and fill in the holes in the image. A unique kind of morphological operations have been used which can give huge flexibility and control to the user. Once the morphology has been performed the blob detection finds the largest blob in the image and finds its dimensions. These dimensions are then checked to see whether they make a face. If thus validated, then the coordinates are given as the initialization parameters for the ASM. The block diagram below gives main steps of the face detection algorithm. The blob detection is an iterative process and will continue to execute till there are faces in the image.

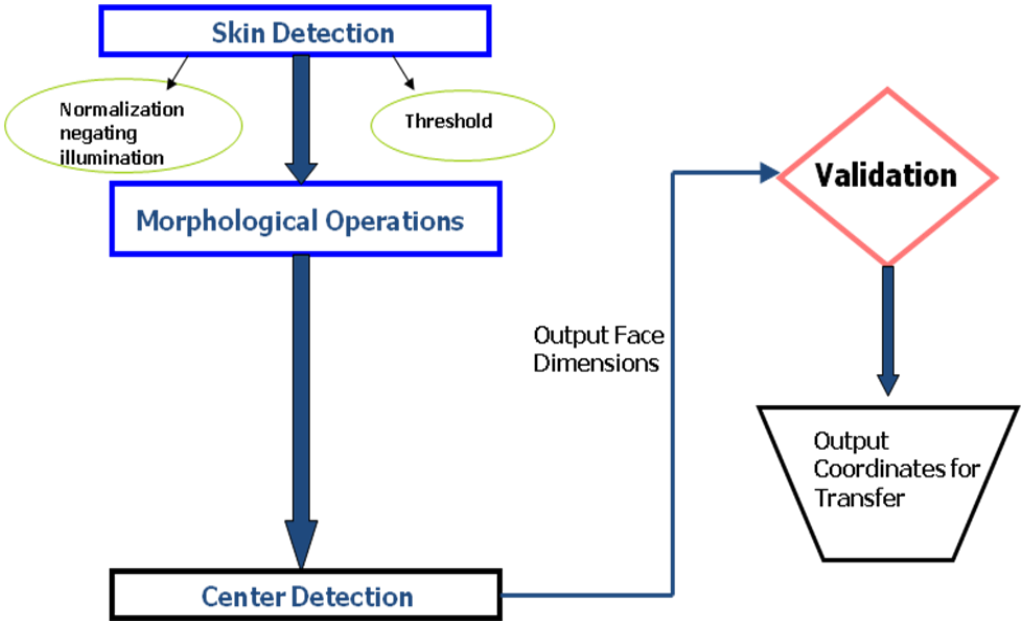


Fig 2: Skin Detection Flow Chart

Skin Detection

An RGB histogram model with 256 bins per channel has around 16.7 million degrees of freedom[2]. We can construct a general color model from the generic training set using a histogram with 256 bins per channel in the RGB color space. The histogram counts are converted into a discrete probability distribution P (.)

[2] Statistical Color Models with Application to Skin Detection Michael J. Jones and James M. Rehg , Cambridge Research Laboratory Compaq Computer Corporation

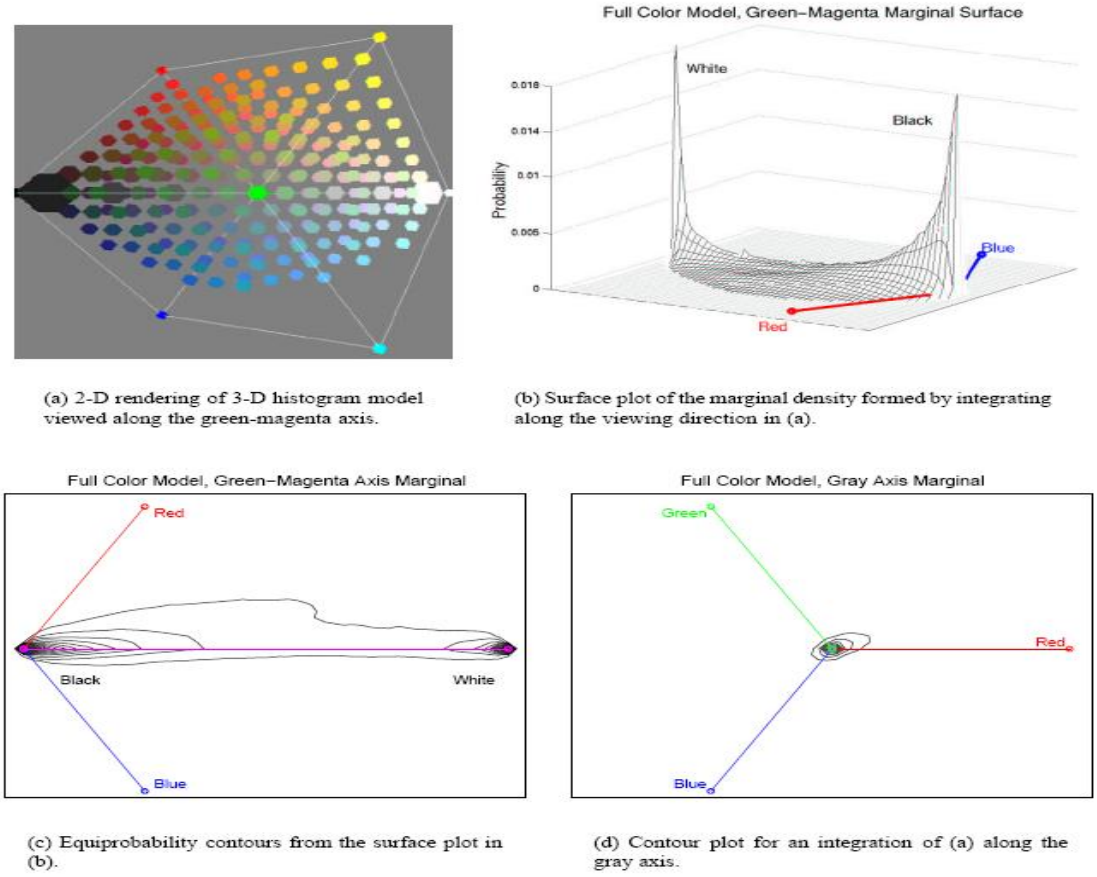
To visualize the probability distribution, a 3-D model in which each bin is rendered as a cube whose size is proportional to the number of counts it contains. The color of each cube corresponds to the smallest RGB triple which is mapped to that bin in the histogram.
Figure 3(a) shows the histogram. This rendering uses a perspective projection model with a viewing direction along the green-magenta axis which joins corners (0, 255, and 0) and (255, 0,255) and in color space. The viewpoint was chosen to orient the gray line horizontally. The gray line is the projection of the gray axis which connects the black (0, 0, 0) and white (255,255,255) corners of the cube. Down-sampling and thresholding the full size model makes the global structure of the distribution more visible.

By examining the 3-D histogram from several angles its overall shape can be inferred. Another visualization of the model can be obtained by computing its marginal distribution along a viewing direction and plotting the resulting 2-D density function as a surface. Figure 3(b) shows the marginal distribution that results from integrating the 3-D histogram along the same green-magenta axis used in Figure 3(a). The positions of the black-red and black-green axes under projection are also shown. The density is concentrated along a ridge which follows the gray line from black to white. White has the highest likelihood, followed closely by black. Additional information about the shape of the surface in Figure 3(b) can obtained by plotting its equiprobability contours. These are shown in Figure 3(c). It is useful to compare Figure 3(c) with Figure 3(a) as they are drawn from the same viewpoint. This plot reinforces the conclusion that the density is concentrated around the gray line and is more sharply peaked at white than black[3].

An intriguing feature of this plot is the bias in the distribution towards red.
This bias is clearly visible in Figure 3(d), which shows the contours produced by a different marginal density, obtained by integrating along the gray axis. The distribution shows a marked asymmetry with respect to the axis of projection that is oriented at approximately 30 degrees to the red line in the figure. This bias is due largely to the presence of skin in Web images.

In summary, the generic color model built from Web images has three properties:
1. Most colors fall on or near the gray line.
2. Black and white are by far the most frequent colors, with white occuring slightly more frequently.
3. There is a marked skew in the distribution toward the red corner of the color cube.
Also 77% of the possible 24 bit RGB colors are never encountered (i.e. the histogram is mostly empty)[3].

[3] M. J. C. Van Gemert, Steven L. Jacques, H. J. C. M. Sterenborg, and W. M. Star. Skin optics. IEEE Trans. On Biomedical Engineering, 36(12):1146–1154, December 1989.



(a) 2-D rendering of 3-D histogram model viewed along the green-magenta axis.

(b) Surface plot of the marginal density formed by integrating along the viewing direction in (a).

(c) Equiprobability contours from the surface plot in (b).

(d) Contour plot for an integration of (a) along the gray axis.

Fig 3:(a), (b), (c), (d) **Ref:** [2]

The color of skin in the visible spectrum depends primarily on the concentration of melanin and hemoglobin [2]. The distribution of skin color across different ethnic groups under controlled conditions of illumination is quite compact, with variations expressible in terms of the concentration of skin pigments (see [4] for a recent study). However, under arbitrary conditions of illumination the variation in skin color will be less constrained. This is particularly true for web images captured under a wide variety of imaging conditions. However, given a sufficiently large collection of labeled training pixels we can still model the distribution of skin and non-skin colors accurately.

Given skin and non-skin histograms we can compute the probability that a given color value belongs to the skin and non-skin classes:

1. P(rgb|skin) = s[rgb]/ Ts
2. P(rgb|~skin) = n[rgb]/Tn

where s[rgb] is the pixel count contained in bin [rgb] of the skin histogram, n [rgb]is the equivalent count from the non-skin histogram, and Ts and Tn are the total counts contained in the skin and non-skin histograms, respectively [1].

[2] Statistical Color Models with Application to Skin Detection Michael J. Jones and James M. Rehg , Cambridge Research Laboratory Compaq Computer Corporation
[4] Symon D'Oyly Cotton and Ela Claridge. Do all human skin colors lie on a defined surface within LMS space? Technical Report CSR-96-01, School of Computer Science, Univ. of Birmingham, UK, Jan 1996.

The skin and non-skin color models can be examined using the same techniques we employed with the full color model. The contour plot marginalizations are formed by integrating the distribution along two orthogonal viewing axes. These plots show that a significant degree of separation exists between the skin and non-skin models. The non-skin model, is concentrated along the gray axis, while the majority of the probability mass in the skin model lies off this axis. This separation between the two classes is the basis for the good performance of our skin classifier, which will be described in the following sections. The only difference in the construction of these two models is the absence of skin pixels in the non-skin case. The result of omitting skin pixels is a marked increase in the symmetry of the distribution around the gray axis. This observation suggests that although skin pixels constitute only about 10% of the total pixels in the dataset, they exhort a disproportionately large effect on the shape of the generic color distribution for Web images, biasing it strongly in the red direction. This effect results from the fact that the skin class occurs more frequently than other classes of object colors (52 % of our images contained skin).
Given skin and non-skin histogram models we can construct a skin pixel classifier.

Thresholds
The threshold are based on two parameters the R component and the sum of the R,G,B values.
Below is the pseudocode of the threshold pattern used.

```
sum =(R +G+B ) ;
        if (sum ~=0)
                r1= R / (sum);
                g1= G / (sum);
                b1= B / (sum);
        else
                r1=0;
                g1=0;
                b1=0;
        end
        if( r1> .38)
                gg = g1 / r1 ;
                bb =b1 / r1 ;
        else
                gg=0;
                bb=0;
        end
        if ( (gg  < .85) && (gg > .2) && (bb < .85) && (bb > .2))
                facedetect
                area=area+1;
        end
```

Thus by using the sum we make the thresholds resistant to the illumination variations. i.e for different amount of illumination we ensure that the thresholds remain the same.

Morphology
We have implemented a simple and unique method of erosion and dilation ,which is much faster than the current method. Though simple this method gives a lot of flexibilty and further gives the

user the ability to perform various operations ranging from erosion ,dilation to edge detection. All with the same algorithm but changing the parameters.
The following is the pseudocode of the algorithm

```
for x=1:1:n-1  // column
    for y=1:1:m-1  // row
        neighsum=0;
        if ( (out(y,x) ==1) ) // it is a potential skin pixel.
              neighsum=sum of the 8 surrounding pixels;
              if neighsum>5
                make the pixels which are at a distance of 2 pixels (x or y ) as skin pixels
              elseif neighsum > 3
                make the pixels which are at a distance of 1 pixel (x or y) as skin pixels.
              Else
                Make the selected pixels as a non skin pixel
              end
        end      // %end of the if loop, checking if skin pixel
     end     //% end of the for loop incrementing the rows
end       //  % end of the for loop incrementing the columns
```

Once this is done we can run the entire sequence for different parameters thus giving us the flexibilty that we desire for the purpose of eroding and dilating of the image.
In our program we have used this sequence twice for different parameters of the neighsum.Since this does not involve the multiplication operations associated with the correlation of the structuring element of the erosion and dilation operatrions we can save a lot of cycles.Also here we essentially vary the structuring element size depending upon the probability that the associated pixel is a skin pixel.

A pixel which is a skin pixel will have neighbours which are skin pixels too. If there are more than 5 neighbouring pixels skin pixels then we can say with a high degree of confidence that the associated pixel is a skin pixel .Thus we use a bigger structing element an d thus make the pixels in the periphery of 2 pixels to be skin pixels. Similarly If there are more than 3 neighbouring pixels skin pixels then we cannot be sure weather the associated pixel is a skin pixel. But if we do nothing then we wont be able to fill the holes in the face.

Thus we adopt a strategy to minimize our possible error by reducing the structuring element size and making the pixels in the periphery of one pixels to be skin pixels.Now if these were truly skin pixel then this means that we are increasing the probability of these probable skin pixels to connect with more neighbors. Thus the reason for the second run of the algorithm with slightly stricter parameters. Thus if the pixels get connected in the first round then in the second round we can be certain that they are skin pixels and then proceed to use a larger structuring element

Blob Extraction

This is the part of the algorithm which extracts the blobs after the morphology is performed on them. We scan the image and find the maximum run of horizontal skin pixels and then scan it again to find the maximum run of vertical skin pixels .For the maximum run of horizontal skin pixels, the column number is stored . Similarly for the maximum run of vertical skin pixels, the row number is stored[5].

Thus the potential centre of the largest blob in the image is obtained by using these coordinates. But since we need to specify a region of interest to pass on to the ASM, we compute the size of the rectangular block containing the blob and then pass these coordinates to the ASM . From which the centre pixel is once again computed. The idea behind is to correct the centre in case the blob which requires to be extracted exceeds the image dimensions. Thus doing this automatically corrects the centre in case it is not exactly in the centre of the blob .

Results

The following are some of the results of skin detection algorithm and morphological operations They are results for groups of images. Since the morphological operations are of the same size some faces have got megred. The solution is to scale the structuring element accordingly for smaller sizes of faces.

Fig 4: (a) Test image, (b) Skin detected, (c) Morphological operated

The test image in Fig 4a shows the detections of a group of people. By setting relatively strict thresholds one can quite accurately determine the skin pixels. The upper part of the image there are a lot of falsely detected pixels but overall the algorithm manages to detect the faces in the picture thus giving the initial coordinates to the ASM. The image in Fig 4a brings out the algorithm advantage as it does not detect the woman's brown-yellow shirt even though it is quite close to the skin color

Fig 5: (a) Test image, (b) Skin detected, (c) Morphological operated

The test image in Fig 5a shows the detections of a group of people who are at a distance. By setting relatively strict thresholds one can quite accurately determine the skin pixels. Note that in

[5] CMU Avinash Baliga, Dan Bang, Jason Cohen, Carsten Schwicking, Face Detection for Surveillance,18551 spring 2002 project ,ECE, Cannegie mellon univeristy

the center part of the image there are a lot of falsely detected pixels representing the pant of the person as it is quite close to a skin color but overall the algorithm manages to detect the faces in the picture thus giving the initial coordinates to the ASM. The detected trousers will be eliminated by the validation part of the algorithm where the dimensions of the detected blob are checked. The image brings out the algorithm advantage as it does not detect the t-shirt of boy on the right even though it is quite close to the skin color.

Fig 6: (a) Test image, (b) Skin detected, (c) Morphological operated

The test image in Fig 6a is used to show the ability of the algorithm to detect people of different races. Note since some of the clothes worn by the people are quite close to skin color these pixels are also getting detected. But this image also brings out one possible negative of this algorithm. This is susceptible to shiny surfaces which are close to skin color as demonstrated by the detection of pixels of the person 's black shiny leather jacket with the metallic trinkets.

The next section shows the face pixels detected for the images of the multi PIE database at different poses. The centre is marked with a cyan colored cross. The portion in red shows the part which has been added due to the morphological operations. The image 7b is the skin detection where some portion of the chin is not detected hence leading to the offset of the centre which is marked in blue. Here the image has been smoothened and the holes have been filled up so that it can be detected as a blob.

Fig 7: (a) Test image, (b) Skin detected, (c) Morphological operated

The next image below is that of the left view of the person. Many spots in the intermediate picture of the skin detected pixels have been eliminated while the holes have been filled up.

Fig 8: (a) Test image, (b) Skin detected, (c) Morphological operated

The Fig 9a is a right view of a person. Many of the falsely detected isolated skin pixels have been eliminated. In Fig 9b and 9c some part of the shoulder is also detected (because of the presence of the red and change of illumination at those areas).

Fig 9: (a) Test image, (b) Skin detected, (c) Morphological operated

Timing estimates

All timing estimates computed are for images which are 240 *320 in size.

Skin Detection

1. 353*240*320 = 27,110,400 (no optimization)
2. 8,678,400 (parallelization ,op level-3)
3. Theoretical values assuming maximum parallelization as possible in the program
4. 17 external memory accesses * 5.625* 320*240= 7,344,000 (external memory)
5. Considering implementation using DMA ,we have to consider data in internal memory
6. 17 *1.5*320*240= 1,958,400 (op-1)
7. Math operations assumed to be running in parallel with the fetches because the fetch timings are higher.

Morph

1. 43,238,400 (no optimization)
2. 21,619,200 (parallelization, op level-3)
3. Theoritical = 42 accesses * 5.625 * 76800=18,224,640
4. Considering implementation using DMA ,we have to consider data in internal memory
   42 *1.5*320*240= 4,838,400 (op-1)

Blob Extraction (EXTERNAL MEMORY)

Assuming only one blob of 100x100 extracted

1. access per pixel=56250 cycles
2. scans of entire image each with 2 external memory accesses =209850
   i. 5,22,240 (no optimization)
   ii. 2,61,120 (parallelization ,op level-3)

Comparision with group 2 ,spring 2002timing estimates. [f4d]

1. Circular structuring element, unoptimized: overflow in cycle counter
2. Circular structuring element, square-skip, (dilate-erode, erode-dilate) to dilate-erode:
3. 673 Mcycles
4. Square structuring element, optimized:
5. 563 Mcycles
6. Virtual square structuring element, optimized & inlined:
7. 547 Mcycles
8. Virtual square structuring element, SBSRAM, DMA transfers:
9. 527 Mcycles

Active Shape Models

We used Active Shape Models to find the shape information of the face. This is later used for more efficient pose estimation.

Algorithm

Active Shape Model is a statistical approach to modeling shapes and appearances, which are used to represent objects in images. A model is trained from a set of images annotated by a human expert. By analyzing the variations in shape and appearance over the training set, a model is built which can mimic this variation. An instance, X of the statistical model is created by defining position, orientation, and scale of the picture. Then, we take an iterative approach to improving the fit of the instance X[6],[7],[8].

Since we have 3 poses we will build 3 ASM models, one for each pose. For a given test image we apply all the 3 ASM models and determine the best fit using texture features determined from correlation filters which will be explained at a later section.

Training

We have M training images and n points on each image.
Each point has an x and y coordinates, so we have a vector of size 2n x 1

Hence the training matrix X is of size 2n x M where we have M samples of 2n data.
Each pixel location x and y is considered as data and the M images provide the samples for this data.

$$x_i = [\, x_{1i} \;\; x_{2i} \;\; x_{3i} \;\ldots\; x_{2ni} \,]^T \quad \ldots\ldots(1)$$

1. PCA is done on this data to reduce the dimension. So we have a set of eigenvectors and eigenvalues. The eigenvalues are used to weight the movement of the points.

   $$x = \bar{x} + P * b \quad \ldots\ldots(2)$$

   where $\bar{x}$ is the mean of all the input shapes
   and $P = (p_1|p_2| \ldots |p_t)$ contains t eigenvectors of the covariance matrix
   and b is a t dimensional vector given by

   $$\mathbf{b} = \mathbf{P}^T(\mathbf{x} - \bar{\mathbf{x}}) \quad \ldots\ldots(3)$$

   The vector b defines a set of parameters of a deformable model.

[6] T. F. Cootes, C. J. Taylor, D.H. Cooper, and J. Graham, "Active Shape models – Their training and application," In Proceedings of Computer Vision and Image Understanding, Vol. 61, pp. 38–59, 1995.
[7] Ghassan Hamarneh, Rafeef Abu-Gharbieh and Tomas Gustavsson, "Active Shape Models – Part I: Modeling Shape and Gray Level Variations"
[8] Rafeef Abu-Gharbieh, Ghassan Hamarneh and Tomas Gustavsson, "Active Shape Models – Part II: Image Search and Classification"

2. With the landmarks over M training images we know the neighboring pixels around each landmark. We also take some pixels above and below each point. This number of above and below points is chosen by us. The neighbors are not chosen in the x and y coordinates directly. The neighbors are chosen perpendicular to the edge of the point as seen in Fig 10.

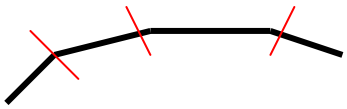


Fig 10: Path along which the model will search for the best fit.

These neighbor points are stored for later comparison with the test images. We take the mean derivative of these points and normalize it.
This Matrix is now called MnDrPoints and is of size
((num of points above landmark) + (num of points below landmark) + 1) * 2n
where n is the number of landmarks we have labeled.

Testing

For a test image we do not have the landmarks. We have the general location of the face.
We start from the mean shape of the ASM.

1. For each landmark we check the neighboring pixels and compare with trained data in MnDrPoints to find the new location of each pixel. Once we have the new location we apply some constraints on the shifts dx to synchronize the movement of all the points so that the shape is smooth
   Note – this constraint can be any thing and is decided based on the type of data

With the new location we get new PCA weights. b = eigenvectors/shape
Use the new value of b for the next iteration to find the next shape. We do this for a number of iterations till the shape converges.

The PCA and the intensity values around each landmark are both used to find the exact shape of the object in a test image.

ASM in Action

Initial Pose After 5 Iterations Convergence

Fig 11: Example of ASM fitting from Tim Coote's paper

Results

Examples of the the Correct ASM model fitting the face.

LEFT ASM Straight ASM Right ASM

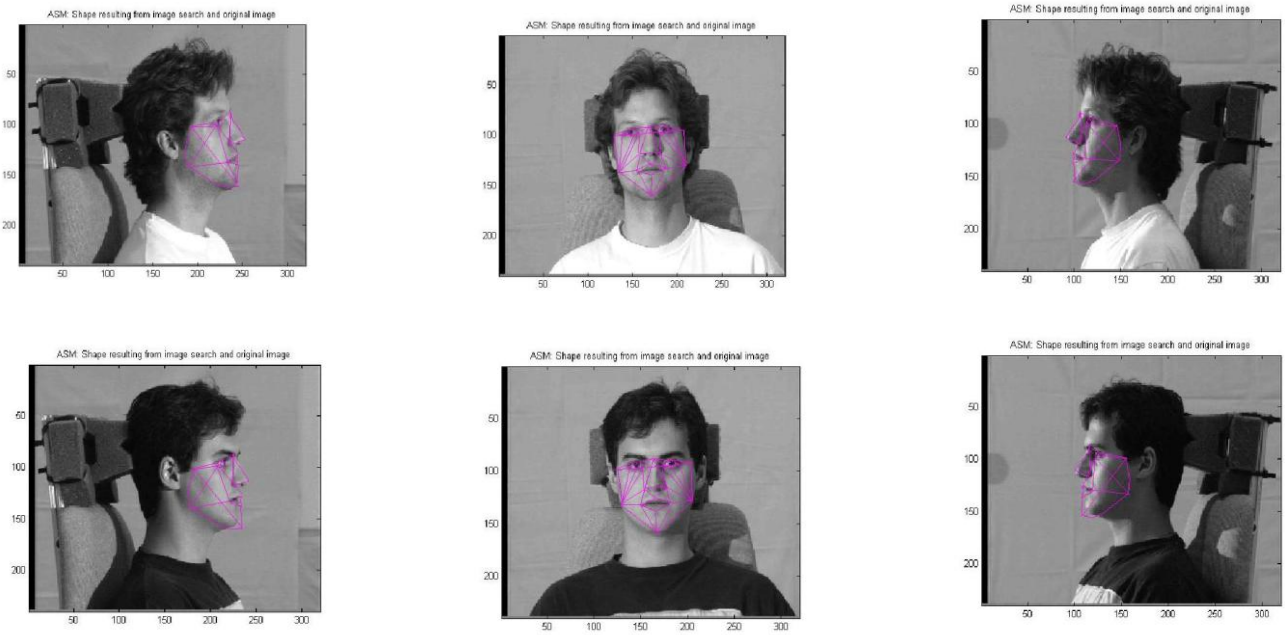


Fig 12: (a)Left ASM on a Left face, (b)Straight ASM on a Straight Face, (c) Right ASM on a Right Face

Example of the wrong ASM fitting the face.

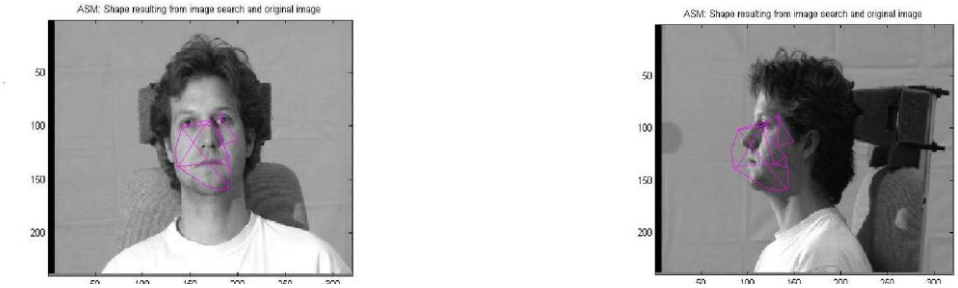


Fig 13: (a)Left ASM on Straight Face (b)Left ASM on Right Face

Discussion of Results

The fitting on the face is dependant on a number of design parameters. Some of them are

1. Initialization – The quality of the ASM fitting is heavily dependant on knowing where the face is. Hence we need a face detection module for our project.
2. Number of search points – The asm fitting depends on the number of points above and below the landmark that we train on and also the number of points below and above the landmark that we search during the fitting. In our project we chose 5 points above and below for landmarking and 12-15 points above and below for searching. The performance will obviously improve is we increase the number of points in our training stage.
3. Multiple of Standard Deviation – The movement of the ASM algorithm is controlled by the multiple of the standard deviation that is specified as an input to the algorithm. Higher this number, greater is the degree of freedom to the ASM. Typical values vary around thrice the standard deviation. In our project, we allowed the user to specify this as an input to the GUI. This allowed us to see the effect of the multiple on the ASM fitting for the same initialization.

Limitations

ASM is particularly useful for segmentation and tracking kind of applications.

There is some evidence of using ASM for recognition[9]. This greatly depends on the choice of landmarks. We have to be sure to choose the landmarks such that the variation is different across different classes. This will lead to weights that are different across classes and we can classify objects based on these weights. Face Recognition using ASM is still a research topic.

ASM can be extended to Active Appearance models (AAM) where we use both shape and texture information to model and recognize a face. This is computationally more intensive and gives much better results. The texture features are also used in the iterative procedure. Note that in ASM we also use the gray level appearance in some way to determine the new location of the pixels, but AAM works by reconstructing the face from the shape vectors and comparing that to the actual trained data to determine the convergence. Hence the complexity is significantly greater. There has been significant work on recognition, tracking using AAMs. AAMs on face is a popular area of research and has proved to give good recognition rates. This is a good area of further work in 18-551

[9] Yu Yuan and Kenneth Barner, "An active Shape Model Based Tactile Hand Shape Recognition with Support Vector Machines"

Timing Estimates

Theoretical estimates
Image access
The image is accessed
(numSeacrhPointsBelow * numSearchPointAbove * numLandmarks) per iteration.
So the access times for the image access is
12 * 12* 20 * 5.6 per iteration.
= 16128 cycles per iteration.

This is because the image is in external memory. Since we don't know exactly which pixel will be accessed each time and considering the overhead of transferring the area of the image to internal memory each time, we felt it was better to have the image in external memory.
Explanation : The pixels that are accessed depends on the deformation and given low deformation constraints the points can move significantly. This means we cannot exactly estimate before hand which part of the image has to be stored in internal memory. To do the transfer for each iteration , in our opinion increased the overhead.

Inverse matrix of 4x4 matrix
This was an external piece of code and not optimized. This took approx 10,000 cycles per inverse. The results of the inverse were compared with matlab for many matrices to ensure the accuracy of this piece of code. Except for precision errors the results were accurate.
= 11,324 cycles per iteration

Operations
There are many functions that are called multiple times. Each of them access various 40x1 vectors to perform operations on them. While it is hard to keep track of all the operations and how many times each one is called. We can make a theoretical estimate for each function per iteration and check against the practical values.
All the variables are in internal memory. Only the image is in external memory.

ASM functions
AlignShapeToShape
(11 multiplies, 10 adds, 21 memory access) * 20 times
= 21/2 *1.5* 20 = 330 cycles
(3 memory access, 1 add, 1 multiply)*14*4 times + 16 memory access
= 168 + 12 = 180 cycles
inverse migs = 11,324 cycles
(3 memory access, 1 add, 1 multiply)*16*4 times + 16 memory access
= 204 cycles
(3 memory access, 1 add, 1 multiply)*16 times + 4 memory access
= 204 cycles
(4 memory access) * 20 times
= 60 cycles
Total = 12,302

ScaleRotateTranslate

(4 multiplies, 4 adds, 8 memory accesses) * 20 times

= 120 cycles

Total = 120 cycles

LimitTheJump

40 memory accesses

Total = 40*1.5 = 60 cycles per call

Find_dx

40 times*(3 memory access, 1 adds)

= 40*2*1.5 = 120 cycles

scalerotatetranslate( )

40 memory access

= 40*`1.5 = 60 cycles

scalerotatetranslate( )

40 times*(3 memory access, 1 adds)

= 60 cycles

Total = 480 cycles per call

LimitTheB

(2 accesses)* 40 times

numEigenvalues memory accesses, numEigenvalues multiplies

= 120*10*1.5 = 1800 cycles

Total = 1800 cycles per call

NormalizeShape

40 memory access = 60 cycles

Scalerotatetranslate()

(2 memory access, 2 adds) * 20 times

= 1.5* 20 = 30 cycles

(4 memory access, 2 adds) * 20 timess

= 2*1.5*20 = 60 cycles

Total = 270 cycles per call

GetBeforeAfterPoints

(2 memory access, 2 adds)

= 2*1.5 = 3 cycles

Total = 3 cycles per call

GetMatchingPosition

20 memory accesses

(2 adds, 3 memory access)*24*20

= 20*1.5 + 2*1.5*24*20 = 1470 cycles

Total = 1470 cycles per call

GetNormalAngle
(4 memory access)
= 4*1.5 = 6 cycles
Total = 6 cycles per call

Find_db
(2 memory access)*40*numEigValues
= 1200 cycles
(1 add, 1 multiply, 4 memory access), 40*numEigValues times
= 2400 cycles
Total = 3600 cycles per call

GetLineCoorsThruPoint
(3 memory access, 2 adds, 1 multiply)*24 times
= 3*1.5*24 = 75.6 cycles per call
Total = 75.6 cycles

Total number for all ASM functions
= 12,302+75.6+3600+6+1470+3+270+1800+480+60+120
= 20179 minimum cycles per iteration

Practical
Find_db = 4276 cycles
GetNormalAngle = 13 cycles
LimitTheJump = 4670 cycles
ScaleRotateTranslate = 1730 cycles
InverseMatrix[10] = 12586 cycles
LimitTheB = 2124 cycles
GetBeforeAfterPoints = 43 cycles
AlignShapeToShape = 12,672 cycles
GetLineCoorsThruPoint = 125 cycles
GetMatchingPosition = 1632 cycles
NormalizeShape = 338 cycles
Find_dx = 493 cycles

Total = 28193 minimum cycles per iteration

Pose Estimation

We used a 2D Mace Correlation filter on the nose region of the face. The nose region was extracted by shape information given from the ASM. Our correlation was an invariant filter and could be moved around the face**.**

[10] Inverse matrix computer Program, written by Tao Pang in conjunction with "An Introduction to Computational Physics", published by Cambridge University Press in 1997

Algorithm

MACE Filter

The Minimum Average Correlation Energy (MACE) Filter is designed to minimize the average energy E in the correlation plane or Average Correlation Energy (ACE). The effect of minimizing the ACE is that the resulting correlation planes would yield values close to zero everywhere except at the location of a trained object, where it would produce a strong peak. The Resulting MACE filter is as follows in a vector form[11].

hMACE = **D**-1 X (**X** + **D**-1 X) -1 c…….. (4)

If we have N training images with each image having d pixels.
X in equation 4 is d x N. C has the prespecified correlation peaks from the N images and D is the average power spectrum density. D is calculated by taking the average magnitudes squared of vector X.

Implementation

We first implemented this in Matlab with various filter sizes. We then decided to make it part of the user input. Hence we allow the user to choose the filter size over which the correlation has to be performed. The correlation filter is generated dynamically in matlab and stored to be sent to the DSK. The overlap and add is done over a larger image area than the filter size. This is different from Lab 3 as we do not have a 3x3 kernel but overlap and add over the size of filter window using the entire 40x40 or 100x100 window as a kernel. Since we are doing correlation we apply this window over a larger area. For example we apply the 40x40 window over a 60x60 image area that is given by the ASM.
The Peak of this correlation plane is found and then maximum value is sent back to the PC for comparison.

Results

An example of the MACE correlation filter is shown in Fig 14a and an example correlation output is shown in Fig 14b.

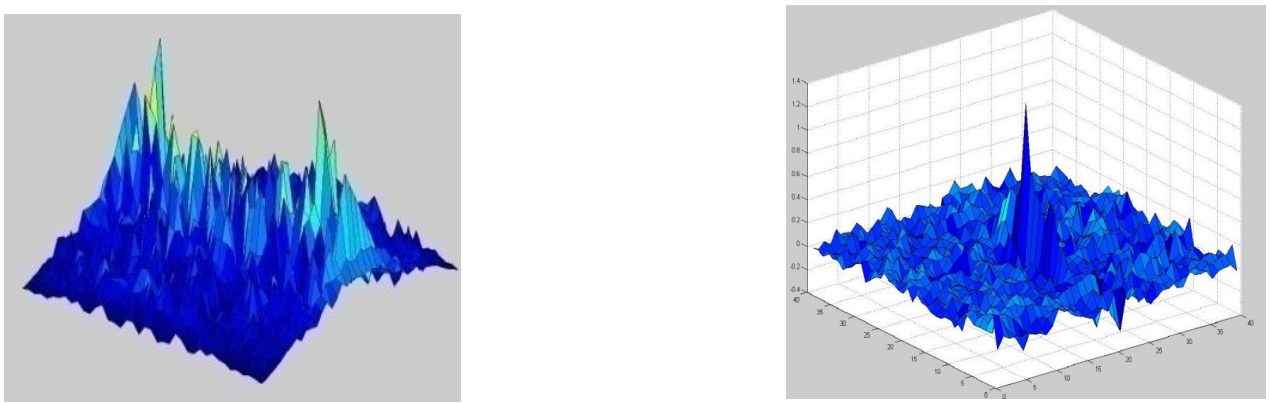


Fig 14:(a) example of MACE filter simulated in Matlab, (b)Correct Correlation output on a test image simulated in Matlab

[11] Savides, Marios, B.V.K Vijaya Kumar, and Pradeep Kholsa. "Face Verification Using Correlation Filters." Carnegie Mellon University – Electrical and Computer Engineering. http://www.ece.cmu.edu/~kumar/Biometrics_AutoID.pdf

In Fig 14b we notice the high peak and the low side lobes. This is an example of how the MACE filter works. It minimizes side lobes to zero and maximizes correlation with the trained object.

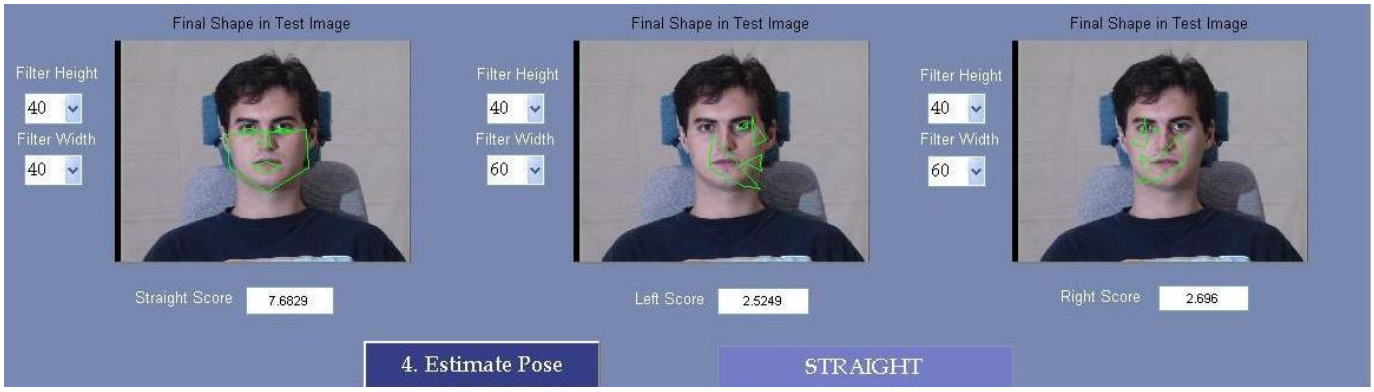


Fig 15: GUI output for an example of Pose Estimation working on a Straight Face.

We tested the pose estimation over a set of images and documented the results as shown in the following tables.

These are tables for the extreme left, straight, and right images. Ten test images were used.

Table 1: Zero shift initialization

Image Angle	**# of Images Tested**	**Left**	**Straight**	**Right**
Left	10	80%	20%	0
Straight	10	0	100%	0
Right	10	0	0	100%

Table 2: +5 shift initialization

Image Angle	**# of Images Tested**	**Left**	**Straight**	**Right**
Left	10	70%	30%	0
Straight	10	20%	80%	0
Right	10	0	10%	90%

Table 3: +10 shift initialization

Image Angle	**# of Images Tested**	**Left**	**Straight**	**Right**
Left	10	70%	20%	10%
Straight	10	0	70%	0
Right	10	0	10%	90%

Table 4: +20 shift initialization

Image Angle	# of Images Tested	Left	Straight	Right
Left	10	50%	40%	10%
Straight	10	30%	70%	0
Right	10	0	40%	60%

Table 5: Zero shift initialization

Image Angle	# of Images Tested	Left	Straight	Right
Straight (22^o Left)	10	20%	80%	0
Straight (22^o Right)	10	0	100%	0

Table 1 represents perfect initialization of the ASM, it reads as follows: A left image was estimated as left 80% of the time and 20% of the time it was estimated as straight. A straight or right image was estimated as straight and right respectively 100% of the time. Table 2 and 3 follow in a similar manner.

Table 5 depicts the results of a straight image looking 22 degrees to the left and straight image looking 22 degrees to the right. As you can see the results are quite accurate.

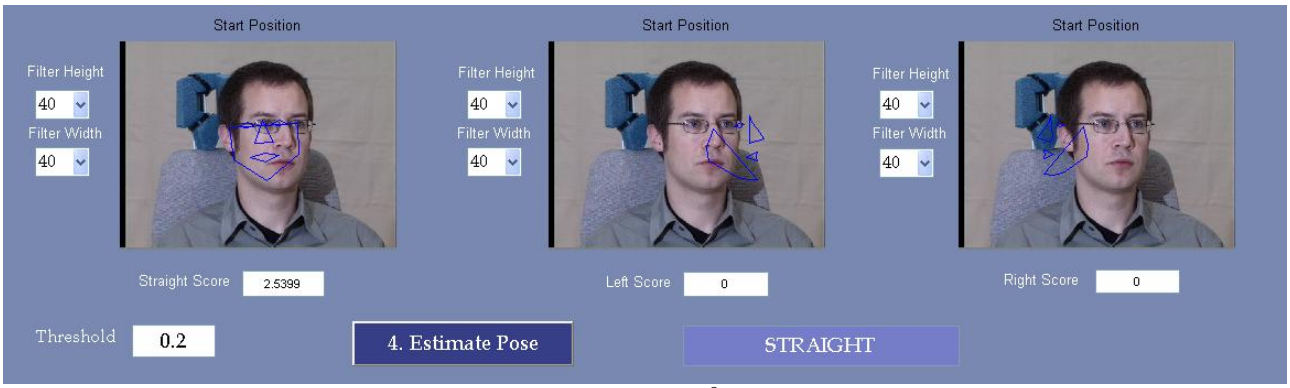


Fig 16: Example of a straight image looking 22^o to the left, it is identified as straight

Discussion of Results

The above results show that the pose estimation varies with the degree of initialization. The further the initial ASM is from the exact position the worse the estimation becomes.

However an average accuracy of about 75% remains up to +10 off the accurate position. The accuracy nears 100% at exact initialization and down to about 50% at +20 initializations. Also a straight image that is 22^o off to the left or right is generally depicted as straight. i.e Fig.16. When we were testing this we saw that the second highest correlation output to a straight image 22^o to the left or right was the left or right ASM fitting respectively. Therefore, the results were generally quite accurate.

In addition, the Correlation output results vary with the size of the filter, as expected. Since the region over which we apply the correlation filter is a function of the filter size, we see the pose estimation is better with some filter sizes on particular test images vs. others. We are certain there are different ways of estimating the pose, either through neural network or some other parametric method. This could be a good area of further work in 18-551.

Limitations

The pose estimation is initialization sensitive. When developing the pose estimation we assumed that the ASM would fit the face perfectly even with a starting position that was not very close to the centroid of the face. This is a key assumption for our pose estimation which may not always hold true due to the error rate of the ASM fitting.

Trouble Shooting

Before, we decided to use a correlation filter we tried many alternate algorithms for pose estimation:

1. Discriminant analysis on the landmarked points as recommended by the original Tim Cootes paper. We transformed the test image landmarks to an LDA subspace and tried to use those coefficients for classification. This did not work as well as expected, because of the poor discriminability of the landmarks.
2. Initial SVM but were unable to pursue it by varying the landmarks and the number of landmarks due to time constraints. We think with the right SVM model and better training we might be able to increase the performance of the pose estimation. We were unable to test this Hypothesis.
3. Minimum distance classifiers in the PCA and the image subspace for recognition. For a given pose we got the coefficients, and found the minimum distance to the coefficients of each training image. We hoped to be able to perform face recognition by this method. This did not give very good results. The coefficients of the particular test face were closer to the wrong face than the correct one.
4. Minimum distance classifier comparing the resultant shape from the ASM fitting with the training image shapes. This led to the same results as before.

Graphical User Interface

We built a GUI in Matlab. It displays results for Face Detection, Active Shape Model, and Pose Estimation and allows for easy execution of these codes from one window.

How it works

1. The GUI loads the image, saves the 3 channels as binary files. It also reads in all the user inputs required for the Face detection and the ASM. It then stores all the data in files. The GUI then executes the ASM PC exe code that reads in all the data from the files and sets up the socket for transferring this data to the DSK.

2. The DSK code is executed separately, and the DSK code reads in the data from the PC code to perform the face detection first , the ASM of the 3 poses next and the pose estimation for each pose next. The results that are returned to the PC are the score of the test image against each pose. This maximum of this score gives us the estimated pose.

3. The Face detect debug intermediate images are displayed in the GUI. The ASM also sends the intermediate shapes to the PC. This happens multiple times with every converging iteration. These steps show us how the shape deforms as it searches for the best fit in the test Image.

GUI

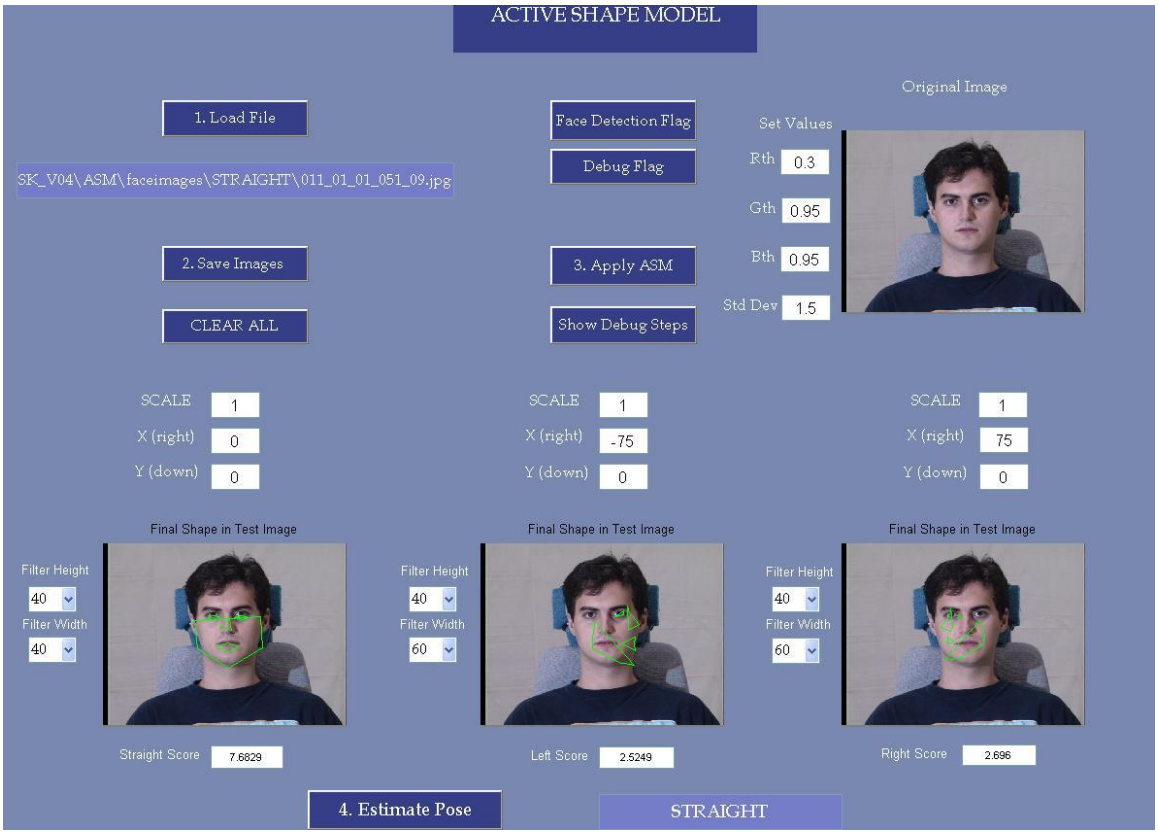


Fig 17: GUI of our Pose estimation Project

Process flow

1. The skin segmentation algorithm, finds an approximate location of the face.

2. The 3 ASM models are applied one by one on the test image. Once the shape converges to the best fit, the ASM coordinates are used to find the central region of the face. Typically this would be the T–section of the face.

3. We compare using simple dot products around this region, with mace filters that we built from all the training images. The mace filters are pre-computed from the same region of the faces in Matlab and sent to the DSK for calculating the score.

The Process flow in the form of a flow chart is shown below in Fig 11.

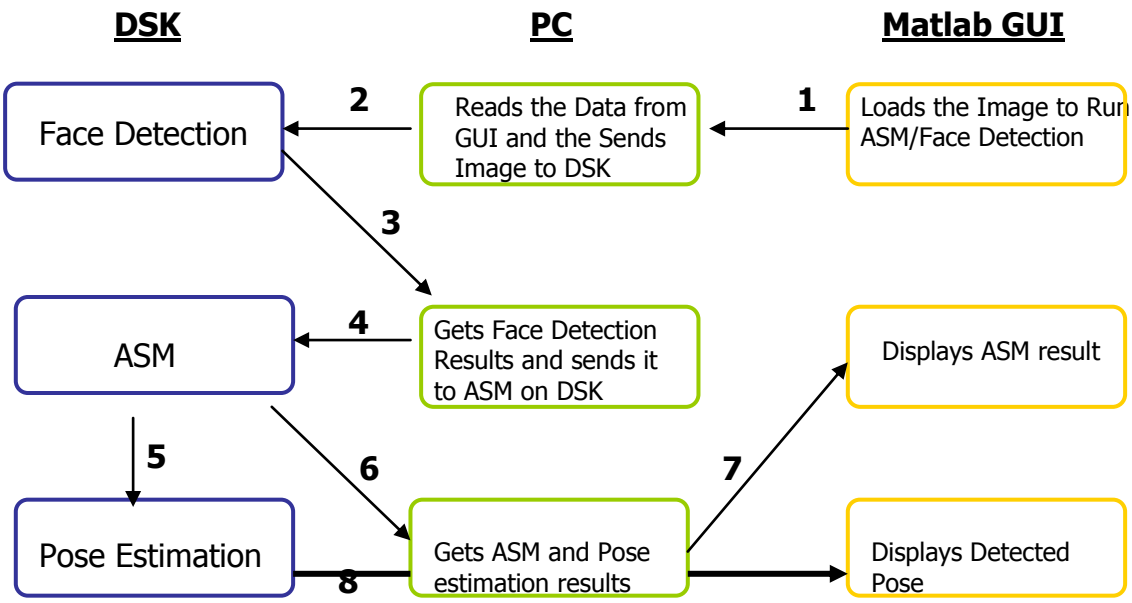


Fig 11. Process Flow

Schedule and Distribution of Tasks

Planned Tasks	Scheduled Start Date	Scheduled End Date	Task responsible
Be familiar with ASM theory and code -Read the papers by Tim Cootes -Project Meeting to understand ASM -Read the matlab code by GHassan	10/01/2007	10/10/2007	Abhay Chandni Sheethal

-Run the examples and try different parameters of data			
Implement Face detection in matlab and test -Understand various methods of face detection -Implement face detection using skin segmentation on matlab and test images	11/22/2007	11/15/2007	Abhay
Implement Face detection on DSK -Understand Schniederman Kanade algorithm -Implement on PC side and test with images	11/16/2007	11/24/2007	Abhay
Convert ASM matlab code to C and test -Start with skeleton code and build functions -Test on PC side -Compare performance with matlab code	10/11/2007	11/15/2007	Sheethal
Get ASM working on DSK for Test Images -Load the code onto DSK -Test the code on the DSK -Compare performance with matlab code	11/16/2007	11/24/2007	Sheethal
Timing Estimates - Report Face Detection ASM	10/11/2007	11/15/2007	Abhay Sheethal
Implement Pose Estimation in matlab -Implement the correlation filter and classification -Test the correlation output	11/01/2007	11/25/2007	Sheethal, Chandni
Implement Pose Estimation in DSK -Implement the correlation filter and classification -Implement the data transfer bw PC and DSK	12/02/2007	12/03/2007	Sheethal, Abhay
Test The Pose estimation – Continously test the Pose estimation to improve the algorithm	11/01/2007	12/03/2007	Chandni
Develop the GUI to load the Face from a file and run the ASM -Prepare GUI requirements -Design the GUI on paper -Develop the base GUI design -Group review of the GUI design	11/18/2007	11/22/2007	Sheethal

-Link the GUI to the DSK/PC code -Test complete GUI that implements all requirements.			
Integration of all modules	12/02/2007	12/03/2007	Sheethal
Testing of all modules	12/02/2007	12/03/2007	Chandni

Implementation Issues

During the initial development and testing phases, several problems became apparent. The most immediate problem was our pose estimation. We had tried multiple algorithms in matlab, but the results of the pose estimation were not accurate enough. As mentioned above we tried svm's, discriminant analysis and minimum distance classifiers using the shape and the coefficients. Finally we used spatial correlation filters to perform a dot product over the region of interest and to determine the pose. This meant the algorithm would run much, much slower.

For the ASM algorithm , it would not help the algorithm to get the image to internal memory as for each iteration we would access only 480 pixels. But we could have moved the image to internal memory for the face detect. This would mean that we use DMA and we would need to get 3 channels into the internal memory + space for the output. We were unable to complete this section of the code due to time constraints. Also since we would need 4 such buffers, the block size would be much smaller than what we used in Lab 3 where we used a block size of 40. We estimated that given our code size we would need at max a block size of 25 rows.

Implementing the ASM and pose estimation accurately took up the entire duration of the course and hence we were unable to optimize the code to a great extent. We were also unable to perform extensive testing of the ASM on a greater number of landmarks, since that would involve a lot of manual landmarking.

Conclusion

We were able to implement everything up to and including Pose Estimation on the DSK. We were unable to find a way to do Face Recognition due to the limits of our implementation of the ASM as well as time constraints. Therefore for future 18-551 groups, either improvement on the ASM behavior or a different algorithm that provides texture information such as AAM should be implemented for Face Recognition.

References

[1] Baker, Simon. "Pie Database." The Robotics Institute. Carnegie Mellon University. http://www.ri.cmu.edu/projects/project_418.html

[2] Statistical Color Models with Application to Skin Detection Michael J. Jones and James M. Rehg , Cambridge Research Laboratory Compaq Computer Corporation

[3] M. J. C. Van Gemert, Steven L. Jacques, H. J. C. M. Sterenborg, and W. M. Star. Skin optics. *IEEE Trans. On Biomedical Engineering*, 36(12):1146–1154, December 1989.

[4] Symon D'Oyly Cotton and Ela Claridge. Do all human skin colors lie on a defined surface within LMS space? Technical Report CSR-96-01, School of Computer Science, Univ. of Birmingham, UK, Jan 1996.

[5] CMU Avinash Baliga, Dan Bang, Jason Cohen, Carsten Schwicking, Face Detection for Surveillance,18551 spring 2002 project ,ECE, Cannegie mellon univeristy

[6] T. F. Cootes, C. J. Taylor, D.H. Cooper, and J. Graham, "Active Shape models – Their training and application," In Proceedings of Computer Vision and Image Understanding, Vol. 61, pp. 38–59, 1995.

[7] Ghassan Hamarneh, Rafeef Abu-Gharbieh and Tomas Gustavsson, "Active Shape Models – Part I: Modeling Shape and Gray Level Variations"

[8] Rafeef Abu-Gharbieh, Ghassan Hamarneh and Tomas Gustavsson, "Active Shape Models – Part II: Image Search and Classification"

[9] Yu Yuan and Kenneth Barner, "An active Shape Model Based Tactile Hand Shape Recognition with Support Vector Machines"

[10] Inverse matrix computer Program, written by Tao Pang in conjunction with "An Introduction to Computational Physics", published by Cambridge University Press in 1997

[11] Savides, Marios, B.V.K Vijaya Kumar, and Pradeep Kholsa. "Face Verification Using Correlation Filters." Carnegie Mellon University – Electrical and Computer Engineering. http://www.ece.cmu.edu/~kumar/Biometrics_AutoID.pdf

[12] Stoytchev, Alexander. "HCI/ComS 575X: Computational Perception." 2007. Iowa State University. http://www.cs.iastate.edu/~alex/classes/2007_Spring_575X/slides/08_SkinColor.pdf