Gender Detector
EE 586L DSP Design Laboratory Course Project

Dichen Zhang Divya Ramesh Xin Li

May 12, 2014

Acknowledgments

We are extremely grateful to our professor, Dr. Panayiotis Georgiou for his support and guidance. We have benefited enormously from the weekly meetings at his office in which he would often give us various pointers and insights, encouraging us to think of simple out-of-the box solutions to seemingly difficult engineering problems.

We would also like to thank Xiang Fu, the teaching assistant for this course for his patience and time. He would often sit with the teams for hours thinking about ways to solve issues with the DSP board.

Finally, we thank all our fellow classmates and friends for all their insights and suggestions. There would always be a team that was facing a similar problem, and discussions with them have always been exciting and useful. The non-technical chit-chats with members of other teams also helped refresh our minds when stuck on a certain problem for too long.

Abstract

Gender recognition from faces is a popular problem in computer vision as it is useful in many applications such as biometric authentication, high-tech surveillance and security systems, criminology, inspection, augmented reality, etc. Our system first detects the faces appearing in front of the camera using a Viola - Jones object detection framework, and then performs gender classification using a Support Vector Machine classifier that is trained on Local Binary Pattern features. Our system has been implemented on the DaVinci DSP DM6437 from Texas Instruments. We found that we could detect upto 5 faces at a time with an accuracy of 100%. Restricting the facial region to a pre-determined bounding box produces the right gender result with an accuracy of 80%, and an initialization time of 10 frames.

Contents

Chapter 1

Introduction

Gender Recognition is an interesting problem in the field of computer vision, not only due to the numerous interesting applications, but also because of the number of approaches present to solve the problem. Today, this problem can be solved using structure from facial images, texture from facial images, gait recognition based methods, and can even extend to the speech and audio domain to use the voice recognition based methods. In this project, we have attempted to use texture based features from facial images in order to detect the gender of a subject in real-time.

1.1 Description of the system

Our system consists of an offline training phase and an online testing phase. In the training stage, we apply the algorithms for detection and classification on a standard facial image database. The images are first pre-processed, passed on to the feature extraction stage, and the extracted features are used to train the classifier. For the testing phase, we apply skin detection as a pre-processing procedure on the incoming video frames in order to locate the possible facial regions, and then detect & track the face in subsequent frames. In order to ensure the good image quality of the facial region, a few image processing tasks must be executed. The trained classifier then works on the extracted texture features to perform gender classification. Fig. 1.1 shows a high-level block diagram of our system.

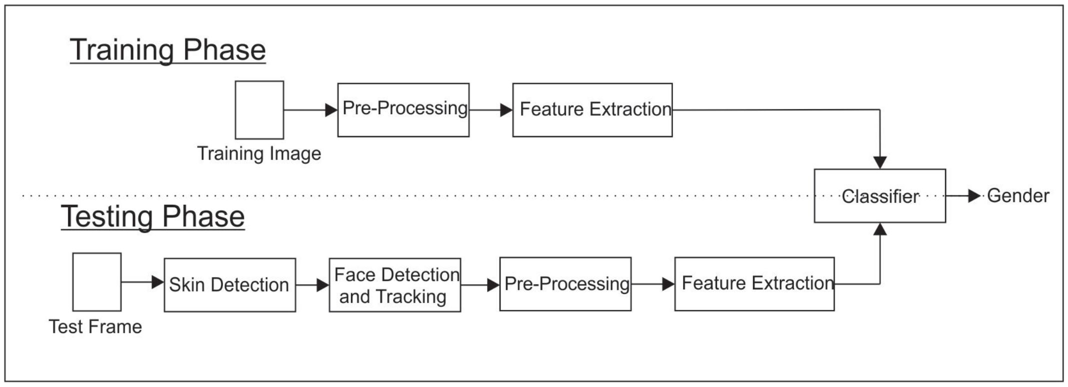


Figure 1.1: A Block diagram of the proposed system.

1.2 Literature Survey

The project roughly consists of the following stages:

- Detection of Human Faces in the input video stream
- Pre-Processing steps to account for variation in lighting, illumination, shading, background etc.
- Texture Feature Extraction
- Classification of extracted features

A discussion of the existing algorithms for each of the stages is carried out below:

1. **Face Detection**: To identify the faces from the input video we intend to employ the robust face detection framework by Viola and Jones[1]. This method has 3 main characteristics: Detection of Haar-like features using integral images, learning with an AdaBoost classifier and an attentional cascade structure that make it possible to run face-detection in real-time.

2. **Pre-Processing**: The variations in lighting, illumination, shading, background, occlusions, noise etc., need to be handled before extraction of features for gender classification. This problem can be handled by using techniques such as contrast stretching, histogram equalization, filtering etc.

3. **Feature Extraction for Gender Classification**: There are various types of feature extraction methods that have been employed in literature. Some of the popular ones include: Fiducial distances, such as the distance between the various features of the face[2], pixel intensity values[3, 4, 5], Local Binary Patterns (LBP)[6, 7, 8], Wavelet and Cosine Transforms[9, 10] and Scale-Invariant Feature Transform[11]. The results of these methods vary significantly depending on the pose variations found in the training image set. Our choice of the feature detection algorithm was guided by the training database used for this purpose. In general, the Fiducial distances and pixel intensity variations work well on frontal images, while LBP, wavelet and cosine transforms perform better on images with pose variations.

4. **Classification**: Support Vector Machines (SVM), AdaBoost algorithms, Neural Networks and Bayesian Classifiers are the commonly used classifier algorithms. While SVM and AdaBoost classifiers are the most popular methods for gender detection from faces, Neural Networks and Bayesian Classifier are more popular for gender recognition based on gait[12].

We have attempted to identify 3 main tasks in the project. They are:

1. Face Detection

2. Face Tracking

3. Gender Detection

The rest of the report discusses the above tasks in detail. It is organized as follows: Chapter 2 is on Face Detection, Chapter 3 on Face Tracking, Chapter 4 on Gender Detection. We present the details of our Experiments and Results in Chapter 5, and the Conclusion, Future Work and Work Division in Chapter 6.

Chapter 2

Face Detection

The first step in the process flow is to detect faces from the captured video stream. This is a difficult problem as there can be a lot of variations in poses, illumination, occlusion, lighting, background, etc. The application we targeted required the use of frontal face images only, and hence two methods of face detection - localization based on skin and Viola - Jones object detection framework were tested for our application. These two methods have been discussed in the following sections.

2.1 Face Region Localization using Skin Segmentation

In this method, the color of the skin is used as a cue to identify the potential face regions and eliminate the unwanted background regions in the scene. The skin color has better concentration in the YCbCr space, and hence if the extracted frame is in the RGB space, it is first converted to the YCbCr space [13]. A pixel is said to be skin colored if it satisfies the the conditions [80<Cb<140] & [135<Cr<165] in the YCbCr space. Else, it is regarded as a non-skin colored pixel.

The various optional steps to be followed in this method are as shown in Fig. 2.1 and are also described below.

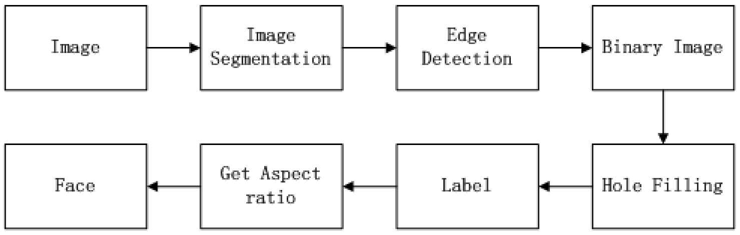


Figure 2.1: Face Localization using Skin Segmentation

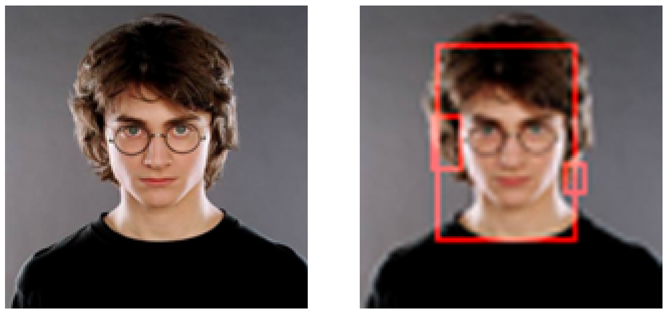
1. **Image Segmentation**: This step aims at smoothing the original image by determining the joint spatial-region domain where the distance is no more than 'hs' in the spatial domain, and no more than 'hr' in the region domain. A recursive labeling function has been implemented for this purpose. First, the original image is extended by 3hs pixels. A label map of the size of the extended image is then set up and initialized to zero. The extended image is traversed pixel by pixel, and if the value in the label map of that position equals to zero, a label function with the position parameters 'i', 'j', and a label 'l' is called. This label function assigns a label 'l' to the pixel at the position specified and creates a search window of (6hs+1)×(6hs+1) centered at (i,j). If the intensity in region domain satisfies the condition $\|x(i) - x(i')\| < hr$, the label function is recalled with the parameters i, j and l. In the first recursion, all pixels in the same cluster will be labeled '1'. The value of this label is then incremented and the recursion is repeated until all the pixels have been successfully labeled. The mean intensity of each cluster is chosen as the intensity of each cluster, and clusters with less than 'M' pixels are eliminated.

2. **Edge Detection**: The Robert's cross difference operator is used for edge detection [14]. An edge gradient map and a histogram of the gradients are created. Using the histogram, appropriate thresholds are determined to classify 10% of the pixels as edge pixels and the remaining 90% as non-edge pixels.

3. **Binarization**: The skin-segmented image is then converted to a binary image.
4. **Hole Filling and Labeling**: The holes in the binary image are to be filled, and the connected regions are labeled. Care is taken to ensure that different sets of connected regions are assigned different labels.
5. **Aspect Ratio Determination**: The aspect ratio for each of the connected regions is determined. The object is classified as a face if the aspect ratio R $\in$ [1, 1.8].

This method although simple, is not robust as it is easily affected by changes in illumination. The range of values for skin varies with the races of the people. Thus, an algorithm that works very well with people of a certain race may not work well with all other races. Therefore, even under controlled and constrained environmental conditions, the algorithm does not always give the best results.

Fig. 2.2 shows the results of this method. This method took 0.124 seconds for execution when executed on Microsoft Visual Studio.



(a) Test Image (b) Localized Face Image

Figure 2.2: Results of Face Localization using Skin Segmentation

2.2 Face Detection using Viola-Jones Object Detection Framework

The Viola-Jones Face Detection framework introduced by Paul Viola and Michael Jones[1], is a feature based approach that achieves competitive detection rates on account of the following key ideas:

- Integral images for fast feature evaluation
- Boosting for feature selection
- Attentional cascade for fast rejection of non-face windows

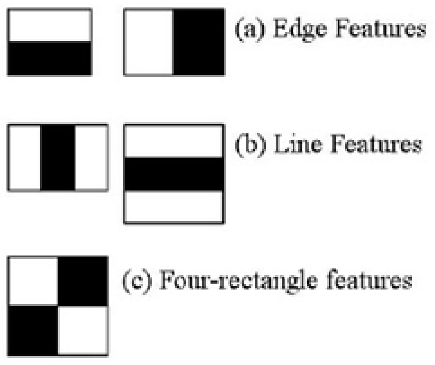
2.3 Feature Extraction

The feature extraction uses the Haar feature extractor that is shown in Fig. 2.3. This feature extractor works by finding the difference between the sum of the pixels in the white regions and that of the black region. Once the Haar features are extracted, the distribution of these features is plotted to obtain the mean and standard deviation of the feature set. This helps to select the useful features that are used to build a set of weak classifiers.

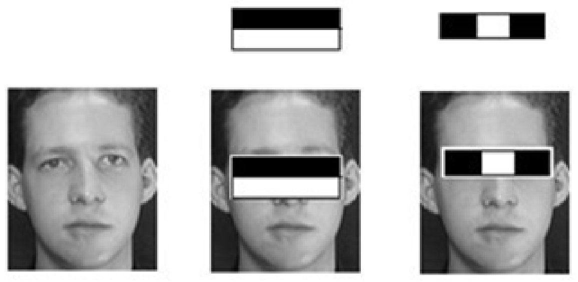
For the training samples of faces, if more than 95% of values are concentrated in region about the mean value of that feature for face training images, and if more than 50% of the values are concentrated in the same region for both face and non-face training images in the same region, then that feature is defined as a useful feature.

2.4 AdaBoost Classification

If (x_i, y_i), $i \in [1, N]$ represent the training images with $y_i \in [-1, 1]$ being the correct class for x_i, the initial weights can be assigned as $D_1(i) = \frac{1}{N}$.



(a) Haar Patterns



(b) Feature Extraction

Figure 2.3: Haar Feature Extractor

After obtaining the useful features, the weight of each classifier is updated as:

$$\alpha_i = \frac{1}{2} ln(\frac{1-\epsilon_i}{\epsilon_i})$$

The distribution for each image will then be updated as

$$D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_i y_i h_i x_i)}{Z_i}$$

The useful features are sorted in the increasing order of the error and only the first few (190 in our case) features are used to form Adaboost classifier. The Adaboost classifier is shown in Fig. 2.4. In each stage, start and end refer to the starting and ending index of the sorted feature obtained from training, and value means the value of the stage to which sum of features in each stage is compared.

There are a total of 10 stages. A search window of varied sizes and positions is used to traverse the image. The search window is first normalized the by the standard deviation of pixels in the window, and the size of the search window. If the search window can pass all the stages, the object under consideration is a face, else, it regarded as a non-face object.

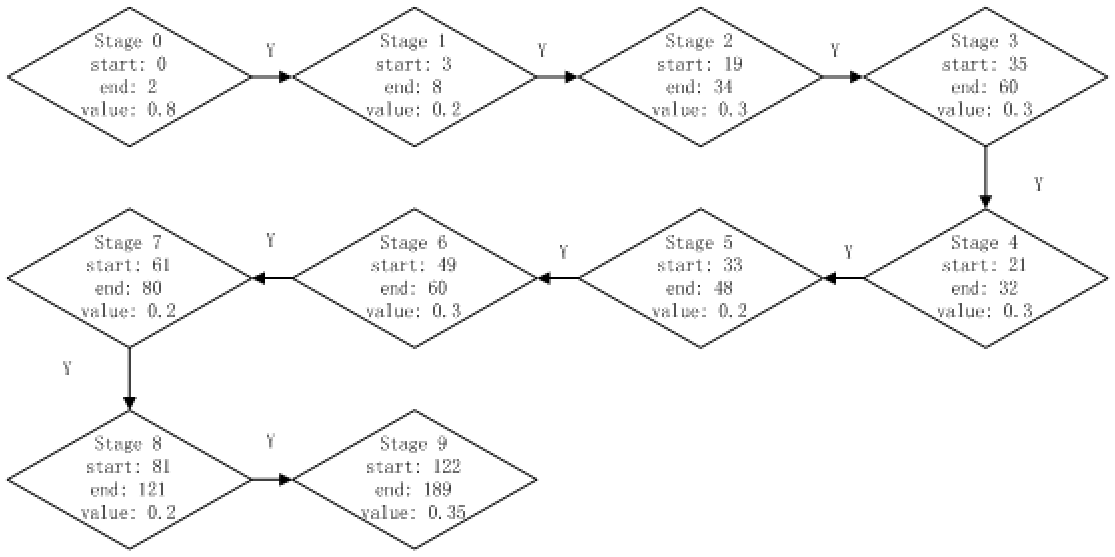

Stage 0 start: 0 end: 2 value: 0.8
Y
Stage 1 start: 3 end: 8 value: 0.2
Y
Stage 2 start: 19 end: 34 value: 0.3
Y
Stage 3 start: 35 end: 60 value: 0.3
Y
Stage 4 start: 21 end: 32 value: 0.3
Y
Stage 5 start: 33 end: 48 value: 0.2
Y
Stage 6 start: 49 end: 60 value: 0.3
Y
Stage 7 start: 61 end: 80 value: 0.2
Y
Stage 8 start: 81 end: 121 value: 0.2
Y
Stage 9 start: 122 end: 189 value: 0.35

Figure 2.4: AdaBoost Classifier.

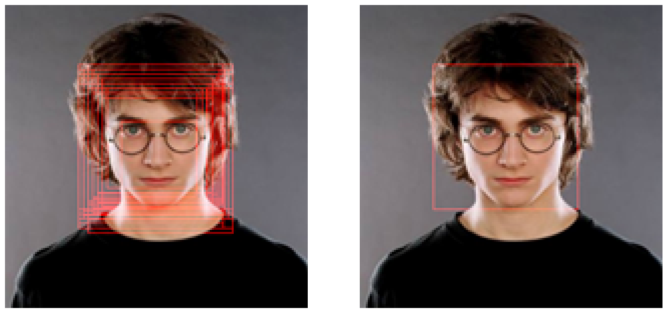
2.5 Attentional cascade for fast rejection of non-face windows

The pseudo code below shows the first stage (Stage 0). There are three trees in this stage. Each tree consists of two or three rectangles. The first two values in the rectangle describe the coordinate of the top-left point in the search window, the third and forth values describe the length and width of the feature, and the last value describes the weight of the current rectangle. Each tree gives one value. If the value is larger than the tree value, the right value is calculated, else, the left value is calculated. After calculating each tree in the stage, compare the sum of the value with the stage value. If the value is larger than the stage value, enter the next stage; otherwise, define as a non-face.

```
<stages>
    <_>
      <!-- stage 0 -->
      <trees>
        <_>
          <!-- tree 0 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>3 7 14 4 -1.</_>
                <_>3 9 14 2 2.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>4.0141958743333817e-003</threshold>
            <left_val>0.0337941907346249</left_val>
            <right_val>0.8378106951713562</right_val></_></_>
        <_>
          <!-- tree 1 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>1 2 18 4 -1.</_>
                <_>7 2 6 4 3.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>0.0151513395830989</threshold>
            <left_val>0.1514132022857666</left_val>
            <right_val>0.7488812208175659</right_val></_></_>
        <_>
          <!-- tree 2 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>1 7 15 9 -1.</_>
                <_>1 10 15 3 3.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>4.2109931819140911e-003</threshold>
            <left_val>0.0900492817163467</left_val>
            <right_val>0.6374819874763489</right_val></_></_></trees>
      <stage_threshold>0.8226894140243530</stage_threshold>
      <parent>-1</parent>
      <next>-1</next></_>
    <_>
```

2.6 Post Processing to Combine Rectangles

In the Viola Jones algorithm, the same face may be detected multiple times. Hence after getting all the face regions, the overlapped rectangles are combined by considering the overlapping area from larger rectangles to smaller rectangles. If more than 80% of the area has been detected by other rectangles, the current rectangle is omitted. The results of the detection after combining overlapped rectangles are as shown in Fig 2.5.



(a) Before combining overlapping boxes.

(b) After combining overlapping boxes.

Figure 2.5: Post Processing.

Chapter 3

Tracking

Tracking can be defined as estimating the trajectory of a moving object in a plane as it moves around in the scene. It is an important computer vision task, especially in those cases that need to detect the presence of a specific object in every frame of a certain video sequence. Tracking objects can be complex due to [17]:

- Loss of information caused by projection of the 3D world on a 2D image
- Noise in images
- Complex object motion
- Nonrigid or articulated nature of objects
- Partial and full object occlusions
- Complex object shapes
- Scene illumination changes
- Real-time processing requirements

The tracking process can be simplified by placing constraints on the objects, in addition to having prior knowledge of the object being tracked.

For our application, we used the feature-tracking method, which is the Optical Flow based Kanade-Lucas-Tomasi tracker scheme. The Kanade-Lucas-Tomasi (KLT) feature tracker is based on the Lucas Kanade method of Optical Flow tracking. It is a combination of the work in [15] and [16].

In the sections below, we give a brief description of the algorithm.

3.1 Feature Extraction

The first step of the tracking method is to extract good features. As suggested in [15], the Hessian Features are extracted from the region of interest, which is the bounding box containing the detected face in this case. The hessian matrix for each is calculated as:

$$H(x,\sigma) = \begin{bmatrix} I_{xx}(x,\sigma) & I_{xy}(x,\sigma) \\ I_{xy}(x,\sigma) & I_{yy}(x,\sigma) \end{bmatrix}$$

where $I_{xx}(x,\sigma)$, $I_{xy}(x,\sigma)$ and $I_{yy}(x,\sigma)$ are the second derivatives. The points at which the determinant of this matrix is maximum are chosen as the good features to track. This is done by calculating the Eigen values, and choosing a certain number of points that correspond to having the highest eigen values.

3.2 Computing Optical Flow Vectors

If h is the displacement between two images $F(x)$ and $G(x)$, where $G(x) = F(x+h)$, then an approximation is made as

$$F'(x) = \frac{G(x) - F(x)}{h}$$

This approximation only holds for small displacements. Besides the approximation to h depends on the location x. Thus, various values of h are averaged by using a window approach. This average can be improved by using a weighting function, where the weights are proportional to the distance of the pixels from the pixel at location x. Upon obtaining the estimate, $F(x)$ can be moved by the estimate of h. The procedure is applied repeatedly, yielding a type of Newton-Raphson iteration. The sequence of estimates will ideally converge to the best h, and this is the optical flow vector. The iteration can be expressed by:

$$\begin{cases} h_0 = 0 \\ h_{k+1} = h_k + \dfrac{\sum_x \frac{w(x)[G(x)-F(x+h_k)]}{F'(x+h_k)}}{\sum_x w(x))} \end{cases}$$

A sample result of tracking a moving object through 20 frames is as shown in Fig. 3.1.

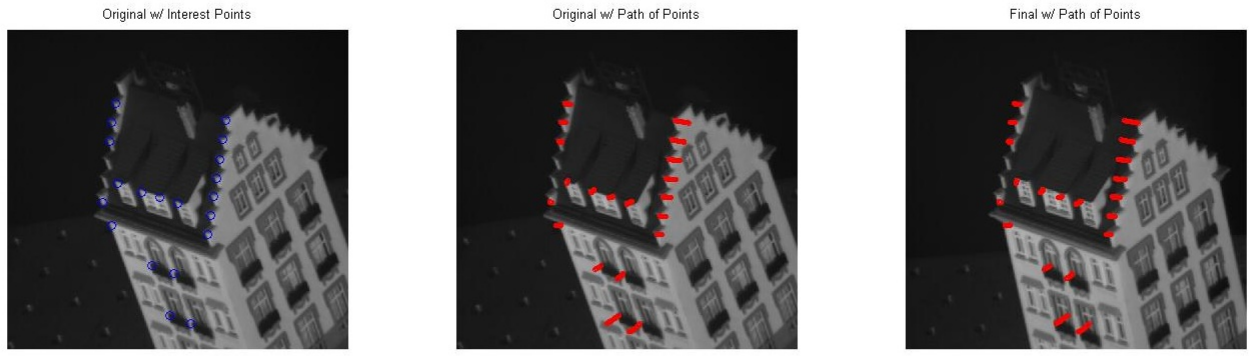


Figure 3.1: A sample tracking result.

Chapter 4

Gender Detection

Of the many approaches to solve the problem of Gender Detection, feature extraction and pattern classification is a simple yet efficient method that suited our application and constraints. The method is based on extracting texture patterns using a feature extractor called the Local Binary Pattern extractor. These features are then passed on the a kernel based Support Vector Machine for classification, and this method works well for both low degree and 30deg faces [18].

4.1 Feature Extraction using Local Binary Pattern (LBP)

The facial image is first divided into 3×3 blocks. A center pixel in each block has 8 neighbor pixels. This center pixel is compared to each of its 8 neighbors (on its left-top, left-middle, left-bottom, right-top, etc.). Following the pixels along a circle, i.e., in a clockwise or counter-clockwise direction, a '1' is recorded in the case that the center pixel's value is greater than that of the neighbor being compared, a '0' otherwise. This gives an 8-digit binary number, which is converted to decimal for convenience. For each block a histogram is then calculated, which instead of containing 256 bins, contains only 59 bins. A local binary pattern is called uniform if the binary pattern consists of at most two bitwise transitions from 0 to 1. For example, the patterns 00000000 (0 transitions), 01110000 (2 transitions) and 11001111 (2 transitions) are uniform, while the patterns 11001001 (4 transitions) and 01010010 (6 transitions) are not. In the computation of the LBP labels, uniform patterns are more important. They are used in such a way that there is a separate label for each uniform pattern, and all the non-uniform patterns possess a single label [19]. After computing the 59-bin histogram for each block, the 3×3 histograms together are then cascaded to form a 531-dimensional feature for the image of the face.

The LBP extractor is as shown in Fig. 4.1.

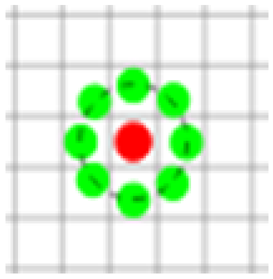


Figure 4.1: Local Binary Pattern Feature Extractor

4.2 Classification using Support Vector Machines (SVM)

Given a set of trained samples, each marked as belonging to one of two categories (such as female or male), an SVM training algorithm builds a model that assigns each new sample to a particular category. An SVM model represents the samples as points in space, mapped in such a way that the two categories are divided by as wide a margin as possible. New samples are then mapped into that same space (feature space) and predicted to belong to a category based on the side of the gap they fall on. In our gender detection case, a kernel function was used to project the input feature data to a higher dimensional feature space in order to find a line separation of the two gender groups as shown in the graph in Fig. 4.2.

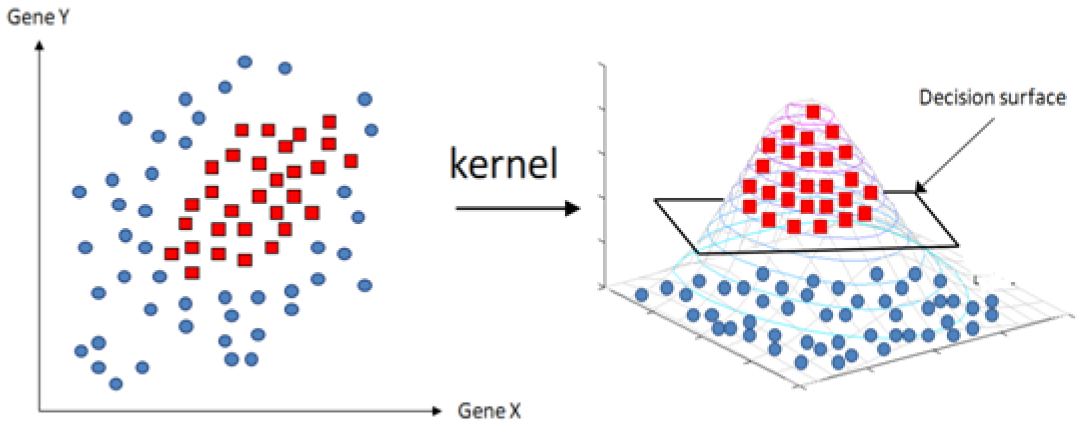


Figure 4.2: A Graph of the Kernel Support Vector Machine Function [20].

A third degree polynomial kernel function defined by the function $K(x_i, x_j) = (1 + x_i^T x_j)^p$, when used was found to improve the result of gender detection.

The SVM model gives the support vectors which are $\vec{x_i}$, and the parameters $a_i y_i$, and b, where i ranges from 1 to N. Therefore, once we have the test data, which comprises of the same LBP features extracted from the testing image, we plug the support vectors and parameters into classification equation in 4.1. The result of this equation will be either 1 or -1, which indicates a male or female respectively.

$$f(x) = sign(\sum_{1}^{N} a_i y_i K(\vec{x_i}\vec{x}) + b) \tag{4.1}$$

A purple bounding box was used to indicate that the subject is a female, while a green bounding box was used to indicate that the subject is a male. A sample result is as shown in Fig. 4.3.

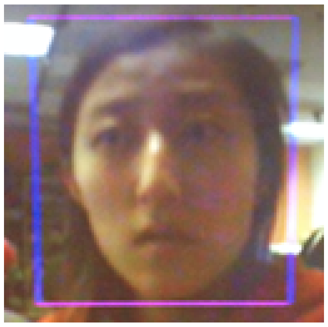


Figure 4.3: A sample result indicating that the subject is female.

Chapter 5

Experiments and Results

All our simulations were first prototyped either in MATLAB and/or Visual Studio and later run on the board using the Code Composer studio software from Texas Instruments.

5.1 Face Detection and Tracking

The database used for training for face detection was the MIT face database[21], which consists of 2429 faces and 4547 non-faces. Each image in the database if of the size of 19×19. Some examples of faces and non-faces are as shown in Fig. 5.1. The five Haar feature extractors resulted in a total of 40977 features, out of which 2044 features were found to be useful. Later, only the first 190 useful features were utilized to construct the AdaBoost classifier.

The optimized Viola-Jones algorithm, when run on the board could detect upto 5 faces at a time within a time frame of 0.252s. Although all faces were detected from the scene, the use of skin-color segmentation in combination with fewer stages of the object detection framework resulted in occasional classification of skin-colored non-face regions as facial objects.

The results of our real-time face detector, implemented on the TI DM6437 DaVinci board are as shown in Fig. 5.2.

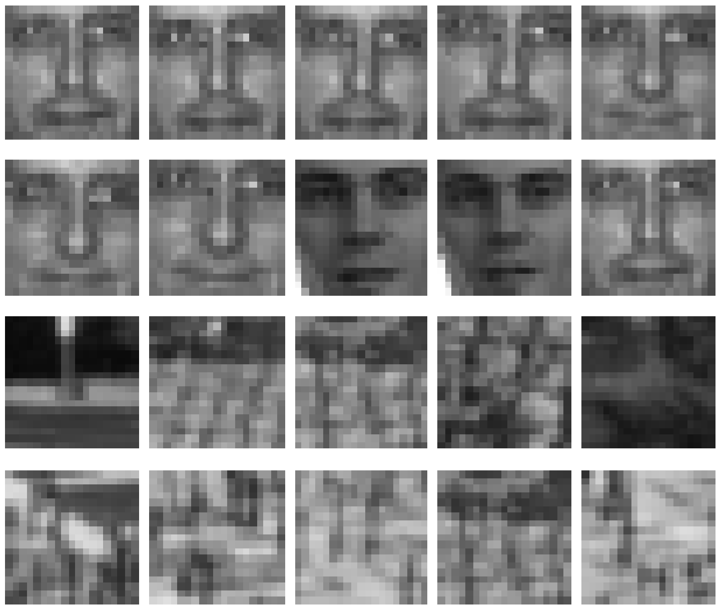


Figure 5.1: Sample images from the MIT database.

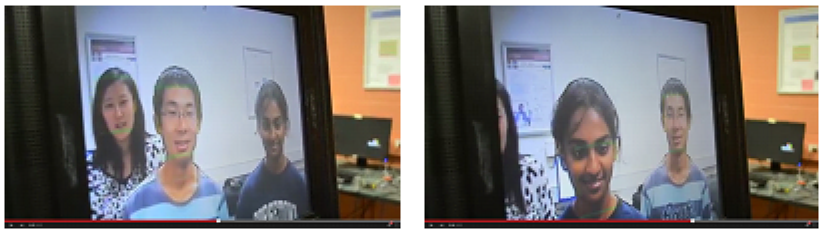


Figure 5.2: Results of real-time face detection on TI DM6437 DSK.

5.2 Gender Detection

Training data for gender detection was obtained from the AR database[22] which contained 55 females and 55 males. Each of these subjects had 14 images with variations in facial expression, illumination conditions and occlusions, thus making it a robust training set. The database also contained people with and without accessories, varied hair length, etc. A few sample images from this database are as shown in Fig. 5.3.

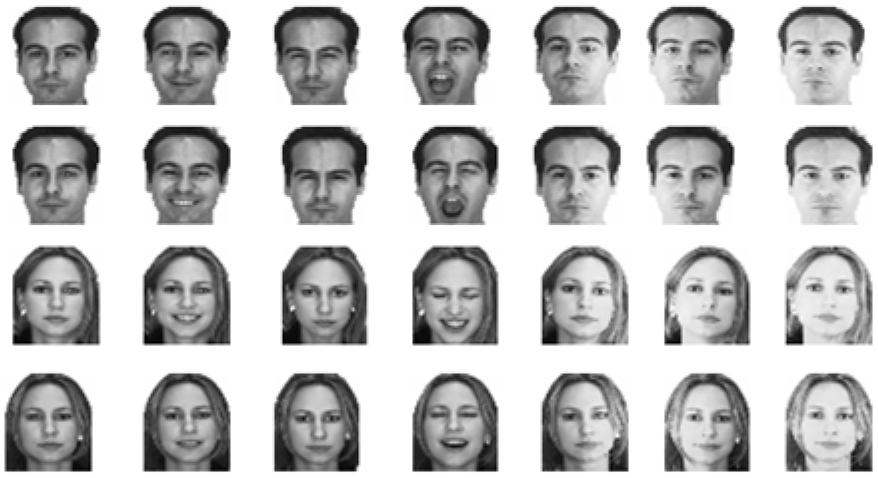


Figure 5.3: Sample images from the AR database.

When the face of the subject is restricted to a pre-defined bounding box region, the gender detector produces the right output in 8 out of 10 cases, on an average. There exists an initial time delay of 10s which is due to the output being determined by a voting mechanism, where in a majority vote obtained from the first 10 frames is used to decide the classification.

The results of the real-time gender detector, with a localized bounding box region are as shown in Fig. 5.4.

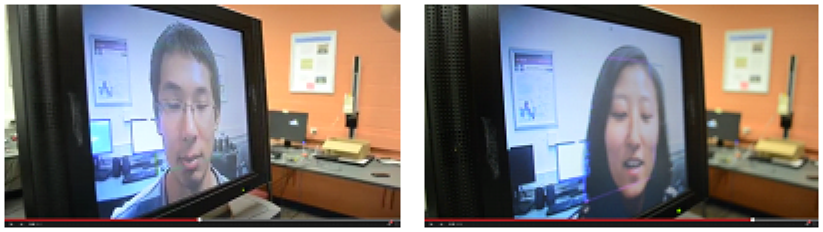


Figure 5.4: Results of real-time gender detection on TI DM6437 DSK.

The results after combining the face and gender detectors, are as shown in Fig. 5.5. The results of of the gender detector whose input is derived from the output of the face detector are not very consistent. This is due to the fact that the training samples for the gender detector consist of faces that are centered, and contain no hair or neck regions. The face detector does not always produce such a centered image. When slightly off centered, the texture features detected from such a bounding box vary greatly thereby leading to inconsistent results.

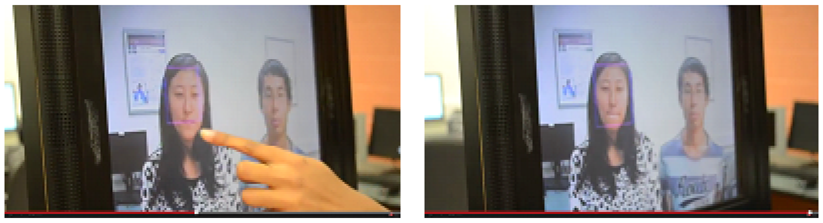


Figure 5.5: Results of real-time face and gender detection on TI DM6437 DSK.

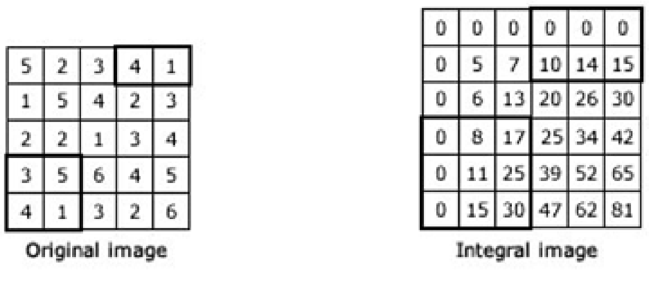
5.3 Issues Encountered and Their Solutions

1. **Face Detection**

   The original unoptimized code was very slow on the board and took upto 25.68s for the detection of a single face from the scene.

   Solutions:

   - **Substituting *float* data type with *int*** The tree and stage values were multiplied by 2^{20} after reading in the *.xml data to convert them from floating data type to integer data type.
   - **Down sampling the original frame and searching alternate pixels** The original image was down-sampled by a factor of 4, and the search for the face regions on this image was carried out on alternate pixels. Also, restricting the search window to a certain size helped to speed up the process of face detection on the DSP board.
   - **Use of Integral Images and Specific Traversal Methods**: The integral image, shown in Fig. 5.6 can be used to calculate the intensity of a rectangular region with just 4 array accesses. By changing the traversal method in these images, we can achieve further acceleration. By keeping the size of image a constant, but enlarging the search window requires us to compute only one integral image which can be reused in further calculations as opposed to the computation of multiple integral images where the image size varies and the size of the search window remains a constant.
   - **Use of Look-Up Tables**: The size of the feature vector is now dependent on the size of the search window. Pre-determining the sizes of the search window enables us to know the size of the feature vectors before hand. Thus, these sizes can be stored in a Look-Up Table and referenced as necessary.
   - **Using Fewer Viola-Jones Stages**: Since the stages are sorted in the decreasing order of error, only the first a few stages will have larger effectiveness, and the later stages do not bear much significance. Thus, only the first few stages of the face detection framework were used.



5	2	3	4	1
1	5	4	2	3
2	2	1	3	4
3	5	6	4	5
4	1	3	2	6

Original image

0	0	0	0	0	0
0	5	7	10	14	15
0	6	13	20	26	30
0	8	17	25	34	42
0	11	25	39	52	65
0	15	30	47	62	81

Integral image

Figure 5.6: An integral image.

These optimizations resulted in almost a 90% decrease in the time for face detection, i.e., from 25.68s on the computer to 2.52s on the board.

2. **Tracking**

   - **Computational Complexity**: If n is the number of warp parameters and N is the number of pixels in the template T, the cost of each iteration is O($n^2N + n^3$). The most expensive step is the computation of the Hessian, which alone takes time O(n^2N).
   - **Speed Issues**: There is a delay of 1 frame from input to output, as the algorithm needs two frames atleast to compute displacement on the previous frame. Futher, the algorithm needs floating point operations and hence takes 1.03 frames per second on a standard laptop computer for an image size of 480×512, when run in the Visual Studio environment.
   - **Issues on the board**: DM6437 optimized for fixed point operations, and not floating point algorithm such as the Lucas Kanade tracker. Further, window based approach of the algorithm requires images to be stored as matrices. Unavailability of dynamic memory allocation functions such as malloc and calloc resulted in occupying huge internal sections in the memory.

     Solution: Skin segmentation was performed to isolate the possible face regions in the video, and then passed to the Viola-Jones object detection framework for classifying as facial and non-facial regions.

3. **Gender Detection**

   - **Memory Allocation**: We need to allocate the arrays with the starting address being able to be divided by four for the reason that, the board operates 4 bytes of data at a time. Therefore, if we use Uint8 arrays we need to pay attention to its size, which is to be a multiple of 4.

     Solution: In order to make sure that we get the correct video frames from the camera, a new position of the data starting from 0x80000000 and having a length of 0x02000000 is defined in link.cmd, which is the "VIDEO_BUFFER". This buffer stores the data for the input and output image, and they use the external memory.
   - **Reading Large Training Data**: After doing the training for gender detection offline, we get a pretty large model file which takes a lot of time to be read by 'fscanf'.

     Solution: Using "fread" function instead of "fscanf" function to read the model one time from the model file into a "char" array, and then allocating the "chars" to different arrays using "atof" and "atoi" functions helped to solve this problem.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This project aimed to detect multiple faces from a video stream and also indicate the gender of the faces detected using colored bounding boxes representative of the gender. The project was implemented on the DM6437 DaVinci development board from Texas Instruments. We were able to successfully detect upto 5 frontal faces at a time from the video stream, in a time of 2.52s using the Viola-Jones face detection framework. The need for tracking of faces was simplified by region localization using skin segmentation. The Local Binary Pattern feature extractor trained on Support Vector Machines (SVM) classifier performed well under constrained environment, such as a pre-defined bounding box region for the face, identifying the gender correctly in 8/10 cases with an initialization time equivalent of 10 frames. However, the gender detector performed poorly when combined with the face detector due to various issues, which we think can be resolved using the methods discussed in the next section.

6.2 Future Work

Currently, the accuracy of the gender detector is heavily dependent on the features extracted from the bounding box region passed on from the result of the face detector. A slightly off-centered bounding box can hence produce highly unstable results. This is due to the texture based feature detection approach used in the project. If we use distance-measure based features, such as the distance between the eyes, nose, mouth, etc., to train the classifier, and if the bounding box extractor of the face detector uses the knowledge of the ratio of the mean width and height of faces, we believe that this problem can be solved.

6.3 Work Division

The project was divided into three parts based on the functionality into Face Detection, Face Tracking and Gender Detection. Dichen Zhang was in-charge of the Face Detection and Code Optimization on the DSP board, Divya Ramesh of the Face Tracking mechanism, and Xin Li of the Gender Detection part of the project.

Chapter 7

Bibliography

[1] V. Paul, and M. J. Jones. "Robust real-time face detection." *International journal of computer vision* 57.2 (2004): 137-154.

[2] Xu, Ziyi, Li Lu, and Pengfei Shi. "A hybrid approach to gender classification from face images." *Pattern Recognition*, 2008. ICPR 2008. 19th International Conference on. IEEE, 2008.

[3] Zafeiriou, Stefanos, Anastasios Tefas, and Ioannis Pitas. "Gender determination using a support vector machine variant." *16th European Signal Processing Conference (EUSIPCO-2008)*, Lausanne, Switzerland. 2008.

[4] Moghaddam, Baback, and Ming-Hsuan Yang. "Learning gender with support faces." *Pattern Analysis and Machine Intelligence*, IEEE Transactions on 24.5 (2002): 707-711.

[5] Phung, Son Lam, and Abdesselam Bouzerdoum. "A pyramidal neural network for visual pattern recognition." *Neural Networks*, IEEE Transactions on 18.2 (2007): 329-343.

[6] Hadid, Abdenour, and Matti Pietikinen. "Combining appearance and motion for face and gender recognition from videos." *Pattern Recognition* 42.11 (2009): 2818-2827.

[7] Alexandre, Lus A. "Gender recognition: A multiscale decision fusion approach." *Pattern Recognition Letters* 31.11 (2010): 1422-1427.

[8] Zheng, Ji, and Bao-Liang Lu. "A support vector machine classifier with automatic confidence and its application to gender classification." *Neurocomputing* 74.11 (2011): 1926-1935.

[9] Scalzo, Fabien, et al. "Feature fusion hierarchies for gender classification." *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on.* IEEE, 2008.

[10] Li, Zhen, Xi Zhou, and Thomas S. Huang. "Spatial gaussian mixture model for gender recognition." *Image Processing (ICIP), 2009 16th IEEE International Conference on.* IEEE, 2009.

[11] Wang, Jian-Gang, et al. "Dense SIFT and Gabor descriptors-based face representation with applications to gender recognition." *Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference on.* IEEE, 2010.

[12] Ng, Choon Boon, Yong Haur Tay, and Bok Min Goi. "Vision-based human gender recognition: A survey." 2012.

[13] Albiol, Alberto, Luis Torres, and Edward J. Delp. "Optimum color spaces for skin detection." *ICIP (1).* 2001.

[14] Pratt, William K. "Digital image processing." *New York 242.* (1978).

[15] Shi, Jianbo, and Carlo Tomasi. "Good features to track." *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on.* IEEE, 1994.

[16] Bouguet, Jean-Yves. "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm." *Intel Corporation* 2 (2001): 3.

[17] Yilmaz, Alper, Omar Javed, and Mubarak Shah. "Object tracking: A survey." *ACM computing surveys (CSUR)* 38.4 (2006): 13.

[18] Lian, Hui-Cheng, and Bao-Liang Lu. “Multi-view gender classification using local binary patterns and support vector machines.” *Advances in Neural Networks-ISNN 2006*. Springer Berlin Heidelberg, 2006. 202-209.

[19] `http://www.scholarpedia.org/article/Local_Binary_Patterns`.

[20] Hardin, Douglas, Isabelle Guyon, and Constantin F. Aliferis. “A Gentle Introduction to Support Vector Machines in Biomedicine”. *World Scientific*. 2011.

[21] `ftp://whitechapel.media.mit.edu/pub/images`.

[22] Martinez, Aleix M. ”The AR face database.” *CVC Technical Report* 24 (1998).