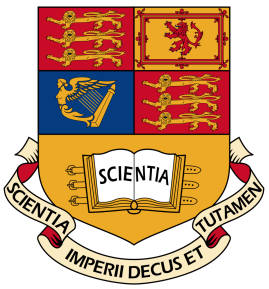


Imperial College London

Final Year Project Report

Unified framework for face image analysis

Author:
Yijie Ge

Supervisor:
Dr T-K. Kim
Second marker:
Dr C. Bouganis

This report is submitted in fulfilment of the requirements
for the degree of *MEng Electronic and Information Engineering*
in the
Department of Electrical and Electronic Engineering
Imperial College London

June 2014

Abstract

This project concerns the research of human face image analysis. The main emphasis of this project is on using the state of the art machine learning algorithm to implement tasks such as head pose estimation for depth images and face landmark detection for RGB images. Random forest will be the method of choice here because it ability to handle large datasets as well as the better prediction performance. As far as head pose estimation is concerned, since human head can be modelled as a rigid object, head pose can be represented by the three angles known as yaw, roll and pitch. Such angles as well as the position of the nose in 3D space will be estimated at a relatively fast speed. In the case of face feature detection, given a RGB image, face features such as eyes, mouth and lips will be automatically and detected robustly with acceptable accuracy.

Acknowledgements

First of all, I would like to thank Dr. Tae-Kyun Kim for provide the opportunity to work on this topic.

Secondly, I would also like to thank my co-advisor, Xiaowei Zhao, Ph.D for the help and advice I received from him throughout the year.

Finally, I would like express my gratitude to Hesam Ipakchi and HongCheng Guo for their support along the way.

Chapter 1

Introduction

Computer vision, a relative new area of research has risen with the growth of technology for the past decade. It is an emerging science which involves teaching machines or computers to see and make decisions and judgements. As discussed in [1], goals of computer vision can be divided into different categories as engineering and way to understand intelligence. Real world applications like surveillance, photography and navigation have been developed and put into use gradually. By applying machine learning techniques, applications can be designed and built to meet varies needs. Although computer vision has a lot of potential applications and is much more efficient than human as the digital data size grows tremendously every year, it has been considered difficult because it frequently fails in accuracy to human visual system by comparison.

For instance, a human can distinguish between body parts of different people under almost any circumstances which are not too extreme. However, a computer may not be able to easily achieve this kind of task due various conditions such as light illumination, viewpoints and different gestures. Therefore, how to represent human knowledge in computers and carry out real time computation efficiently and accurately have become the biggest challenges in the area of computer vision. Moreover, vision techniques are very domain dependent. Some approaches may fail in one specialized application but work well in other specified areas. [1].

Face is a key subject in the area of object detection. Especially face recognition, which is an technology used to verify people's identity is very widely used in various fields such as security authentication systems, search potential criminals in urban area and unlocking smart phones. And research conducted on this subject is getting more and

more important. In this project, we will try to address some sub-problems in facial recognition.

This report will emphasis on the state of the art machine learning algorithm known as random forest, which will be mentioned in detail in section 2.4 and will be used to tackle problems such as head pose estimation in section 3 and face feature points detection in section 4 respectively.

Chapter 2

Background

In this section, the main algorithm used for this project will be mentioned as well as preliminaries regarding to computer vision.

2.1 Computer Vision Preliminaries

Before I go in to the details, let me introduce some necessary fundamentals.

2.1.1 Image Format

As far as this report is concerned, both RGB and depth images are used. RGB images have three channels, namely red, green and blue. Each of which has a value ranges from 0 to 255 which represents the respective intensity. The combination of the three channels will represent a pixel in a digital image. Unlike RGB, Grayscale (also know as black and white) images have only one channel and only possess the intensity information of a pixel. [21]

The conversion from RGB to grayscale by the Luminosity method is :

$$Grayscale = 0.21 \times Red + 0.72 \times Green + 0.07 \times Blue \tag{2.1}$$

Examples of a RGB image and it's corresponding Grayscale are shown in Figures 2.1 and 2.2

Figure 2.1
RGB

Figure 2.2
Gray

FIGURE 2.3
Depth and Colour

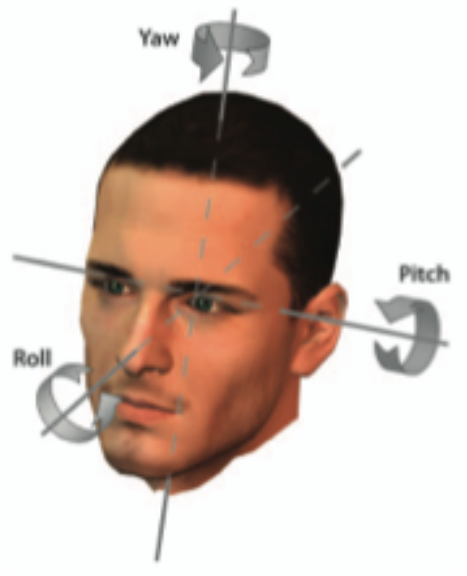


FIGURE 2.4
"The three degrees of freedom of a human head can be described by the egocentric rotation angles pitch, roll, and yaw " [3]

In addition to 2D images, due to the rise of depth sensor such as Microsoft Kinect. Depth images become available. Each pixel of a depth image contains the information of the distance from an object in a 3D scene to the viewpoint. An example of a depth image with its colour version is shown in Figure 2.3.

2.1.2 Euler Angles

Euler Angles are the most commonly used rotational coordinates. As far as this project is concerned, Euler angles are used to classify the head rotations. As shown in Fig 2.4

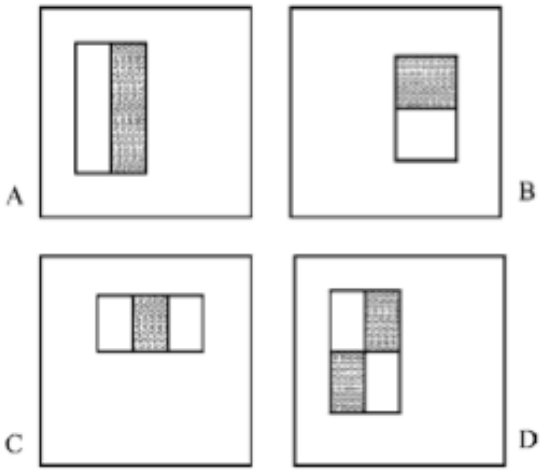


FIGURE 2.5
A, B show the two rectangular feature whereas C and D show three rectangular feature and four rectangular feature respectively. The sum of the pixel values in the white rectangle is subtracted from the sum of the pixel values in the grey one. [23]

2.2 Haar-Like Features

The use of Haar-Like feature (as shown in Fig 2.5) plays a big role in the scene of object detection. As mentioned in [23], feature based systems perform better than pixel base systems. It is also known as rectangular features. In this project, only two rectangle feature is used which computes the difference between the sum of the pixels of two rectangular regions. One may notice, as the size of the rectangular region grows, the computation time will increase dramatically. Integral image is introduced to overcome this problem. [23]

2.2.1 Integral Image

Feature computation is an important and necessary step in face image analysis but it could also be very time consuming as the size of the rectangular regions increase. As a result, integral image is used which enables fast computation. An integral image at location x, y contains the sum of the pixels above and to the left of x and y.(See Fig

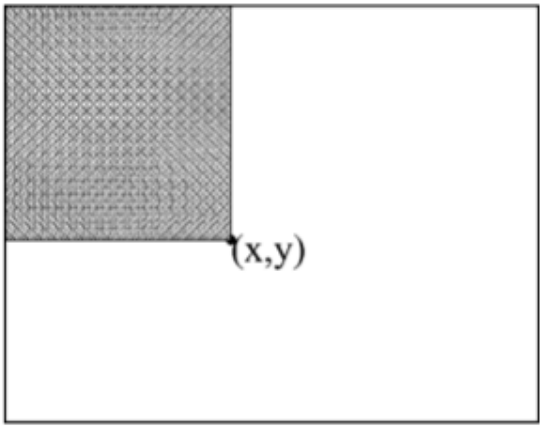


FIGURE 2.6
The value of the integral image at point(x,y) is the sum of all the pixels above and to the left [22]

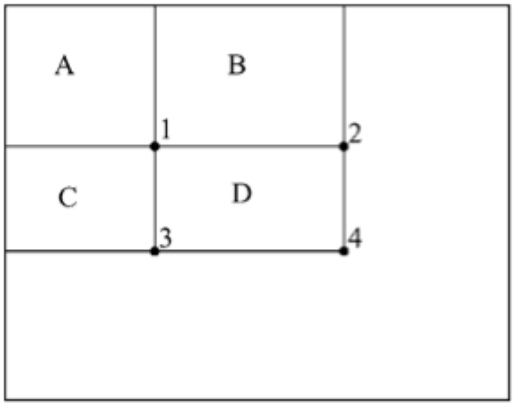


FIGURE 2.7
If we are required to computer the sum of all the pixel values in D, the only 4 points we need to know is 1,2,3,4 since the value at 1 is A, the value at 2 is A + B, the value at 3 is A + C and the value at 4 is A + B + C + D, with a bit of algebra, it is not difficult to see that D = 4 + 1 - 2 - 3. [22]

2.6 [22] Analytically :

$$sum(x, y) = \sum_{a \leq x, b \leq y} i(a, b) \tag{2.2}$$

Where $sum(x, y)$ is the pixel value at position (x,y) of the integral image and $i(a, b)$ is the pixel value at position (a,b) of the original image. Once integral images are created, sum within any rectangle can be computed by only referencing four points in the integral image. As shown in Fig 2.7.

2.3 AdaBoost

In general, AdaBoost is an algorithm used to construct a strong classifier by combining number of weak classifiers as a weighted sum:

$$F(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \tag{2.3}$$

Where $F(x)$ is the final strong classifier and $h_t(x), alpha_t$ are weak classifiers and their respective weights. The general Adaboost algorithm for a two class classification problem can be interpreted as follow:

Algorithm 1 Adaboost

Given training data points, $(x_1, t_1)..(x_N, t_N)$ where t_n is the target value of x_n, and $t_n \in \{-1, 1\}$

Initialise data weight $\{w_n\}$ by $w_n^{(1)} = \frac{1}{N}$ for $n = 1...N$

for For $m = 1...M$ **do**

(a) Learn a classifier $y_m(x)$ that minimises the weighted error:

$J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(X) \neq t_n)$

Where I is the impulse function which is 1 when $(y_m(X) \neq t_n)$

(b) Evaluate $\epsilon = \frac{\sum_{n=1}^{N} w_n^{(m)} I(y_m(X) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}}$, and set $\alpha_m = ln\{\frac{1-\epsilon}{\epsilon}\}$

(c)Update the data weights by $w_n^{(m+1)} = w_n^{(m)} exp\{\alpha_m I(y_m(X) \neq t_n)\}$

end for

Make predictions using the final model by: $Y_M(x) = sign(\sum_{m=1}^{M} \alpha_m y_m(x))$

Adaboost is used for real time face detection proposed by Viola and Jones which is the prior step for face feature point detection (discussed in detail in section 4.1)

2.4 Random Forest Framework

In this section, Random Forest will be mentioned in detail since it is the algorithm used for this project. Random forest is first mentioned by Breiman in 2001 [7]. And it had been widely used and explored in many applications since then. Before we discuss how to create a forest, let's look at how to build a decision tree first.

2.4.1 Decision Trees fundamentals

"A tree is a set of nodes and edges organized in a hierarchical fashion" as can be seen in figure 2.8 [4, 9]. This structure is faster when it comes to searching the data space. In contrast to graphs, loops does not exist in trees. In this project, only binary trees are considered. Intuitively, a decision is a tree which can make decision about the incoming data and it is built in order to decompose complex problems into smaller pieces which are easier to solve. For example, we are given an image and asked to tell whether there is a snake inside that image. The constructed tree from the prior data set (different images with and without presence of snakes) will tell us the probability of a snake being in the image with the image given. And depending on the size of the training set and the parameters applied to build the decision trees, the results might be different. In a decision tree, leaf nodes contain either the probability distribution of the which class the incoming data belongs to (classification) or the value the incoming data should approximately take(regression) and non-leaf nodes contains stores binary tests which are applied to incoming data. To perform a test on unseen data, data is passed from the root of the tree and follows the test at each intermediate node until the leaf node is reached [5,14,15].

2.4.2 Tree training

Building a tree is a supervised learning problem which requires the training data to be annotated with labels on the target value. For example, input data is in the form of $\{(\mathbf{x}_i, y_i)\}$, where $\mathbf{x}$ is in $2\mathbf{D}$ space, i is data label from 1 to N and y is the target value for $\mathbf{x}$.We want to construct the tree which can give the best representation for the associated data. Every data point is sent to the root node and the split function is chosen so that the information gain (discussed later in section 2.4.3) of that split is maximized.(as shown in Figure 2.9)This process is repeated after each split with the data at each child node until a leaf node is created according to some predefined stopping criteria such as the maximum depth of the tree,etc. [6]

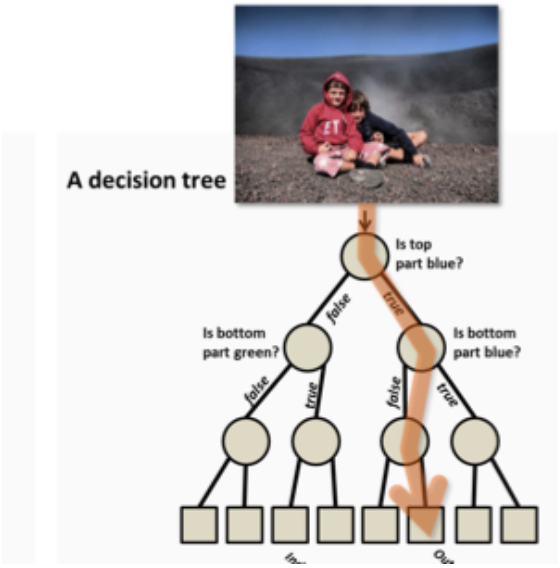


FIGURE 2.8
A decision tree that determines if the picture is taken outdoor or indoor. [4]

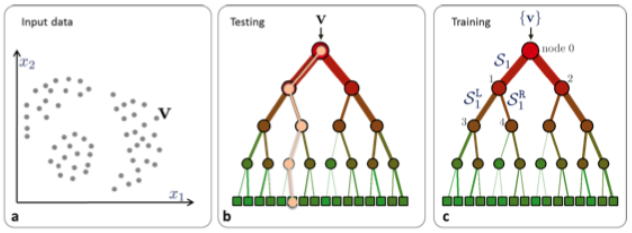


FIGURE 2.9
Decision tree for 2D data clustering [4]

2.4.3 Entropy and information gain

$$I = H(S) - \sum_{i \in \{1,2\}} \frac{|S^i|}{|S|} H(S^i) \tag{2.4}$$

Equation (2.4) shows the relationship between entropy and information gain when building a decision tree. Where I is the information gain, $H(S)$ is the entropy of the parent node, $\sum_{i \in \{1,2\}} \frac{|S^i|}{|S|} H(S^i)$ is the expected entropy of its children. As shown in figure 2.10. **a** is the data set before first split, **b,c** are the data set after two different splits. And the split in **c** has a higher information gain than the split in **b**. This is the most general version of the information gain. However, the measures of entropy in different problems could vary a lot (discussed in later chapters).

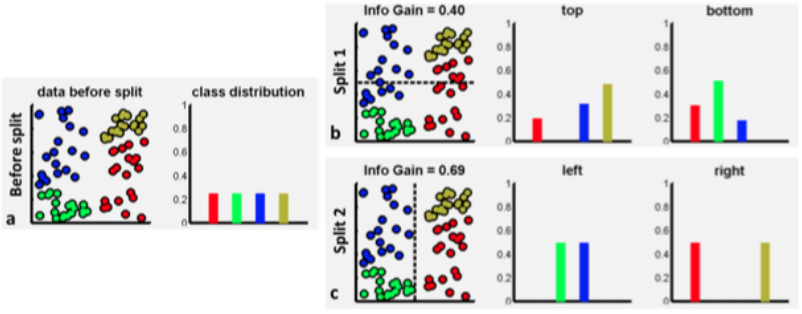


FIGURE 2.10
Information Gain for discrete, non-parametric distributions [4]

2.4.4 Random Forest

A random forest an ensemble of randomly trained decision trees with randomized collection of features at each split [4].

In general, random forest is distributed algorithm which means that trees are trained in parallel. Compared to other methods such as boosting, random forest has the advantage of searching only in a subset and because of the reduced size of the tests set for each tree, faster training time can be obtained and the collection of the trees at testing time is proved to be more accurate in prediction. Trees in a random forest are learned from randomly sampled subsets of data which reduces over-fitting compared to training on the whole data set. [2,5,7] The other randomness which is often used during the training stage of the random forest is a random subset of splits for optimizing the information gain at non-leaf nodes [2]. And overfitting can be avoided by setting the maximum depth of the tree at the beginning.

At testing time, unseen data is pass to all trees in the forest during which binary test at each node of every tree is performed until the leaf node is reached. After the process, the forest will gather all class distributions of the leaves and produce an average value for the estimation as described in (2.5). Where $\Pr(P)(c|\mathbf{v})$ is the average probability of data point $\mathbf{v}$ belonging to class $\mathbf{c}$ and $\mathbb{P}_t(c|\mathbf{v})$ is the probability of data point $\mathbf{v}$ belonging to class $\mathbf{c}$ in one tree.

$$\Pr(P)(c|\mathbf{v}) = \frac{1}{T}\sum_{t=1}^{T}\mathbb{P}_t(c|\mathbf{v}) \tag{2.5}$$

This the most general version of random forest. There are variations regarding to this report such as the goal of the forest where not classification but regression is needed. These variations will be discussed along with the implementations in the later chapters.

Chapter 3

Head Pose Estimation

Efficient and accurate head pose estimation can be very useful to many application, but it is a very challenging problem. By using the approach proposed by Fanelli [2], accurate pose estimation is achieved. Random regression forest is the algorithm of choice since its ability to handle big datasets. In this chapter, I will discuss the training procedure of such forest in detail in 3.1. The evaluation is done on a synthetic dataset of 3D images. And the performance of such forest will be discuss in 3.2.

3.1 Random Forest Training

A tree in the forest is trained from annotated patches randomly extracted from a subset of the training images. Each tree is assigned a maximum depth which it is allowed to grow to. The depth of the trees is relative to the performance of the forest during the testing time. Beginning at the root, every tree is build recursively by applying binary test at each non-leaf node. These tests will decide which direction will each patch be sent. i.e. left or right child of that node. Binary tests used are also randomly selected from a test pool. There is a trade-off between how many tests to be done at each node and how much time to be spent at each node. Generally the more tests to be done the more accurate split will be achieved. The goal of all the binary test is to find best one that maximizes the information gain. [2] The problem is essentially an optimisation problem. Analytically, the objective function is:

$$\theta^* = \arg\max_{\theta} IG(\theta) \tag{3.1}$$

$$IG(\theta) = H(P) - \sum_{i \in \{left, right\}} w_i H(P_i(\theta)) \tag{3.2}$$

Where $w_i = \frac{|P_i(\theta)|}{|P|}$ which is the proportion of patches each child has after every split. $H(P)$ is the entropy measurement. (discussed in 3.1.3) This process will continue until the maximum depth is reached. Once the maximum depth is reached, the nodes are classified as leaf nodes and necessary information will be store at such nodes.

3.1.1 Training Data

Training data is depth images annotated with the 3D locations of the tip of the nose together with the three head rotational angles, namely pitch, yaw, roll. Square patches with fixed size are extracted from each depth image. Each patch is annotated with two vectors: $\alpha^1 = (\alpha_x, \alpha_y, \alpha_z)$ is the 3D offset vector between the centre of the patch and the position of the nose. And $\alpha^2 = (\alpha_{pitch}, \alpha_{yaw}, \alpha_{roll})$ is the head orientation as Euler angles. Since when patches are sampled, they are sampled not only from the face but also from other body parts like torso. Every patch is also given a class label $c_i \in \{1, 0\}$ as positive (near the face) or negative (far from face) respectively. As a result, each training patch

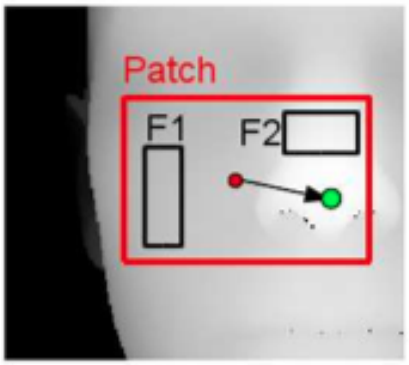


FIGURE 3.1
Example of a training patch (the red box), rectangles F1,F2 are randomly chosen within the patch, red dot is the patch centre and the green dot is the location of the nose [2]

is therefore in the form of $P_i = (I_i, c_i, \alpha^1, \alpha^2)$ Where I_i is the feature obtained from each patch, in this case it is the original depth values. [2]

3.1.2 BinaryTest At Non leaf nodes

The binary test at each node is defined as follows:

$$|F_1|^{-1} \sum_{q \in F_1} I(q) - |F_2|^{-1} \sum_{q \in F_2} I(q) > \tau \tag{3.3}$$

Where F_1 and F_2 is two rectangles within the patch and τ is a threshold. What the above equation essentially represents is the difference between the average values of two rectangles which is very similar to the Haar-like feature mentioned in 2.2. [2] As shown in Figure 3.1. All the best tests found will be store at every non-leaf node.

3.1.3 Measure of Entropy

At each node, vectors α^1, α^2 can be modelled as realisations of multivariate normal random variable. [6] And by definition, the differential entropy for a random variable is:

$$h(x) = -\int_X f(x) log f(x) dx \tag{3.4}$$

The probability density function for a k-dimensional normal is :

$$f_{\mathbf{x}}(x_1,,,x_k) = \frac{1}{\sqrt{(2\pi)^k|\mathbf{\Sigma}|}} exp(-\frac{1}{2}(\mathbf{x}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)) \tag{3.5}$$

Applying both above equations with k = 3, we find the differential entropy is :

$$H(P)^n = \frac{1}{2} log((2\pi e)^3|\mathbf{\Sigma}^n|) \tag{3.6}$$

where $n \in (1,2)$. However, from equation 3.6, we can deduce the regression measure is:

$$H_r(P) = \sum_n log(|\mathbf{\Sigma}^n|) \propto H(P)^n \tag{3.7}$$

Now we can substitute equation 3.7 into 3.4, maximising equation 3.4 favours binary tests chosen earlier which minimize the determinant of the covariance matrix $\mathbf{\Sigma}$and reduces the regression uncertainty. [2, 6]

At testing time, once a patch is passed down to the forest, the forest is used not only to give probabilistic votes in a continuous space but also to determined the patch is eligible to cast a vote. Therefore a measure of the classification uncertainty at training time is needed in this case. [2, 6]

$$H_c(P) = -\sum_{k=0}^{K} p(c=k|P) log(p(c=k|P)) \tag{3.8}$$

Where K = 1 meaning the patch is either positive or negative. Now we have two measures 3.7 and 3.8 and we combine them as a weighted sum of two as proposed by Okada [2, 24]:

$$H_c(P) + \alpha max(p(c=1|P) - t_p, 0) H_r(P) \tag{3.9}$$

When minimising the above, the second term is zero until the proportion of the positive patches reaches a threshold. Then the second term comes into play. The choices of t_p, α will be mentioned in 3.2

3.1.4 Information to be stored at leaves

Once the patches get to the maximum depth, leaf nodes will be formed and few bits of information will be stored. Firstly, since patches are mark as positive or negative at the early stage, the percentage of positive patches reach at the leaves is recorded which

is essentially the class probability $p(c = k|P)$. And the mean vectors and traces of the covariance matrices computed from α^1 and α^2 of all the patches arrived at each leaf node. In this way, a combination of classification and regression forest is built ready for the experiments.

FIGURE 3.2
positive patch in green, negative patch in red [2]

3.2 Evaluation

3.2.1 Dataset

The experiment is conducted on Biwi Kinect Head Pose Database which is a synthetic database and contains over 15000 images capture from 20 people, all of which are annotated with the 3D nose position, the head rotation angles and have resolution 640×480 . The yaw, pitch and roll angle are within the range of $\pm 75 \pm 60 \pm 50$ respectively. Each depth image contains not only the head but the other body parts with the background removed (value of each pixel that does not belong to the person will be zero). Head mask which can be used to sample positive patches are also provided. [2]

3.2.2 Experiments

As far as training is concerned, image 15 subjects out of 20 are used and the remaining 5 people are used for testing. In the For each tree, 2000 images are randomly sampled from the image pool. For each image, 40 patches are sampled , 20 positive and 20 negative patches as shown in Fig 3.2. Positive patches are sample using the masks as shown in Fig 3.4. Negative patches are sample from other body parts.

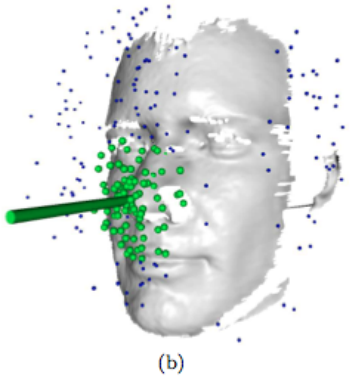


FIGURE 3.3
blue dots are the outliers and green dots are the votes used to give the actual estimated [2]

Every patch occupies an area of 80×80 pixels. When growing each tree, maximum depth is set to 15 and at non-leaf nodes, 500 random pairs of rectangles F_1, F_2 in Fig 3.1 and 20 random thresholds in equation 3.3 are selected from the pool. Therefore in total 10000 tests are carried out for each incoming training patch at each non-leaf node. If the number goes beyond this threshold the training time takes too long and the performance does not scale very much. As mentioned in 3.1.3, parameters α, t_p are set to 1.0 and 0.7 as suggested by [24].

During the testing time as illustrated in Fig 3.6, since the data is provided with the background removed, bounding boxes around the actual data are extracted first from test images and patches are sampled within such bounding boxes. Each patch is passed down the tree and then directed by the binary test store during the training until it reaches a leaf node where the patch centre in 3D representation will be added onto the mean 3D offset vector stored at leaf node earlier to give an estimated of the nose position. Integral image is used when computing the binary tests therefore extra computation time is ignored.

A patch is allowed to cast a vote only if the positive class probability at that leaf node is 1 and the trace of the covariance matrix of the nose position is below a threshold. Since leaf nodes with high variance at the nose position are not informative and will affect the result. The lower the threshold is ,the more votes will be filtered before the final estimation which could speed up the estimation process and decrease the uncertainty of the estimation.The threshold of the trace of the covariance matrix is set from 300 to 2000 to observe the corresponding performance and plots are shown in Fig 3.5. As we can see from the plot, overall accuracy decreases as the maximum variance allowed for a patch to cast a vote and the good accuracy is achieved around 400.

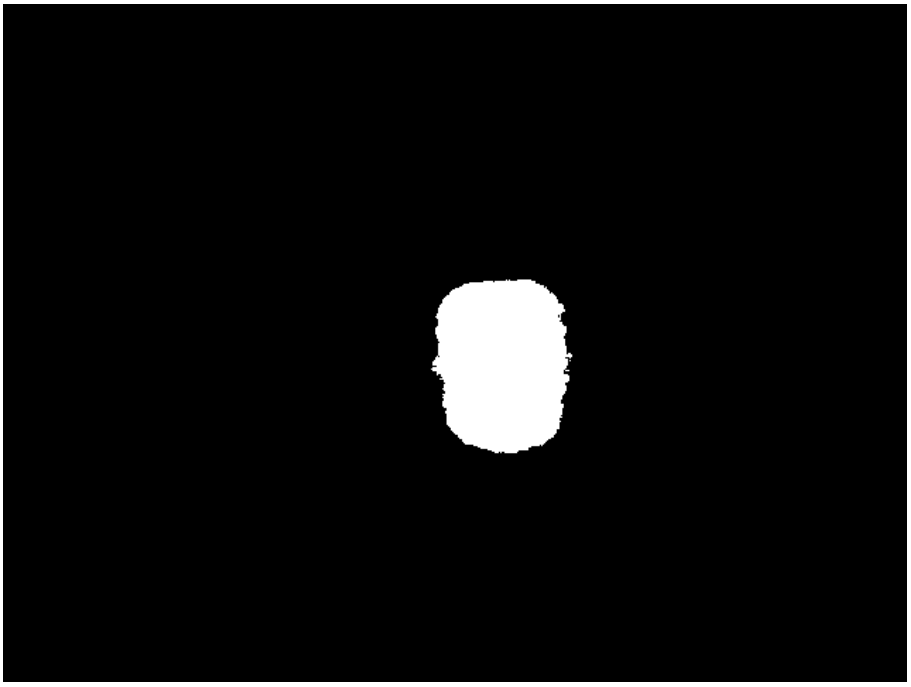


FIGURE 3.4
White pixel area represents the head position, positive patches will be sampled around this area

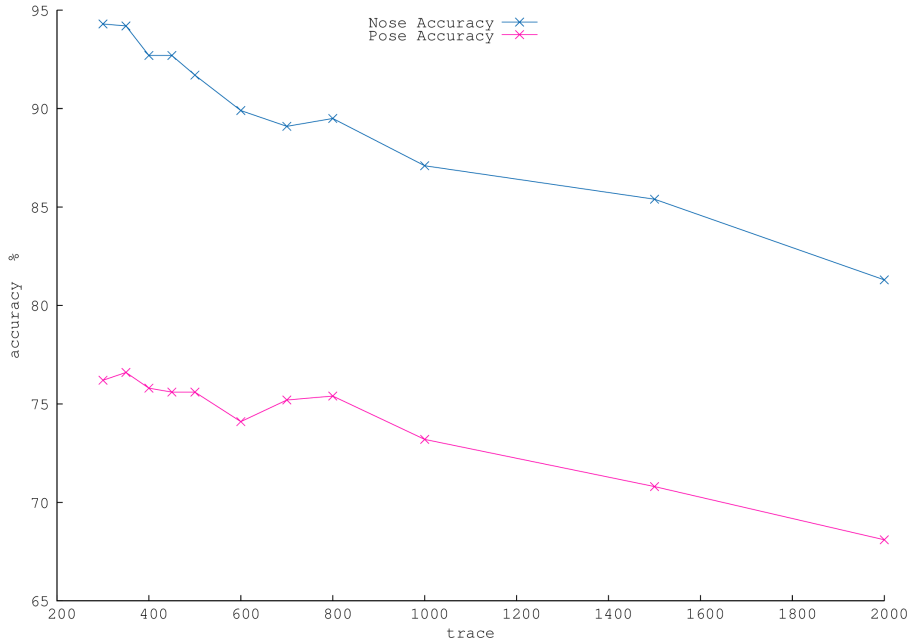


FIGURE 3.5
Accuracy for nose and pose estimation vs maximum variance, nose in pink and pose in blue

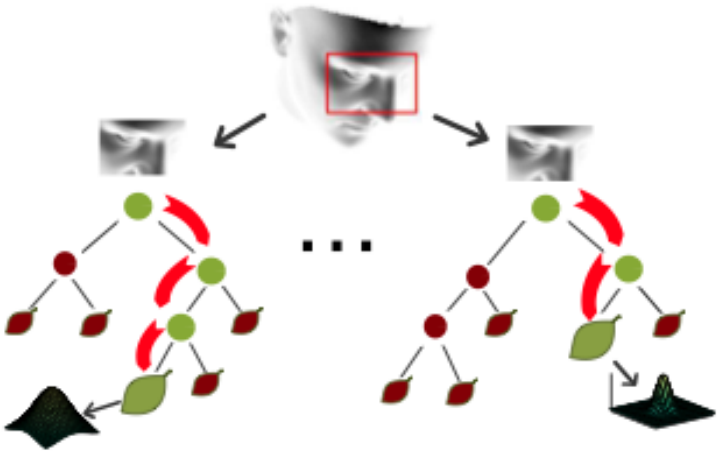


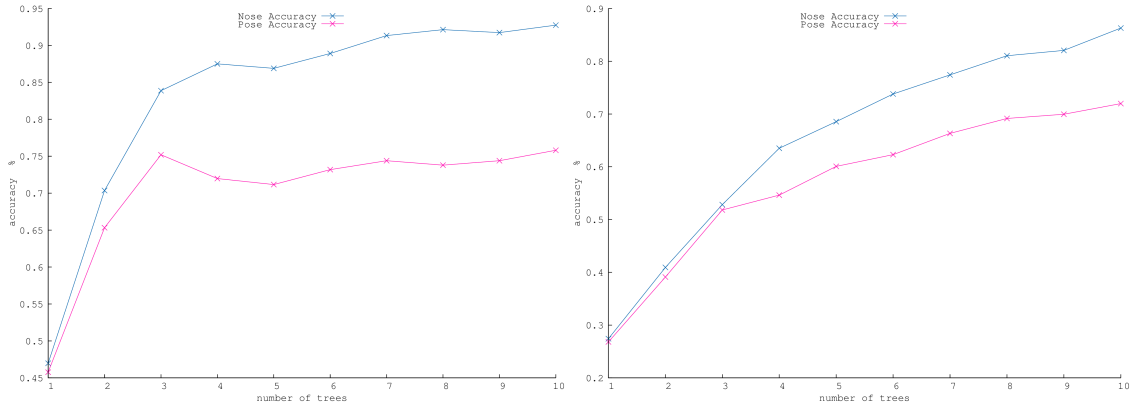
FIGURE 3.6
testing patches are passed down to each tree to get a possible vote at the arrived leaf node [2]

After every test patch is processed, there is a huge pool of candidate votes and there will be outliers. (see in fig 3.3) And it turned out to be essential to remove these outliers before moving on to the next step. These outliers are removed by mean shift. [2] Then the mean of the remaining votes is computed again to be the final estimated.

How densely the test patches will be sample will directly affect the result as well. There is a tradeoff between how much time to be spent and how much accuracy to get. In this case, since the synthetic data is with the background removed, patches only needs to be sampled within the window that covers the person's body but not the whole image. Where the accuracy denotes the success rate among test images. A successful detection is achieve if the nose error is below 20mm and the angle error is below 15 degrees. [2]

The accuracy for both nose the angles are plotted in figure 3.7.The test is performed on 500 test images with various density that patches are sampled.As can be seen from the plots, number of trees loaded for the estimation affect the result more in the early stage but as more trees are loaded, the performance improves in a more gradual manner and finally converges. And as more patches are sample for each test image, the accuracy could improve from 70% to almost 90%.

The comparisons between the estimated and ground truth values for Euler angles versus each frame are plotted in figures 3.8, 3.9, 3.10. As can be seen from the plots, the overall estimation of yaw and roll angles seems to be better than the pitch angle.



(A) stride 5 (B) stride 10

FIGURE 3.7: accuracy vs number of trees

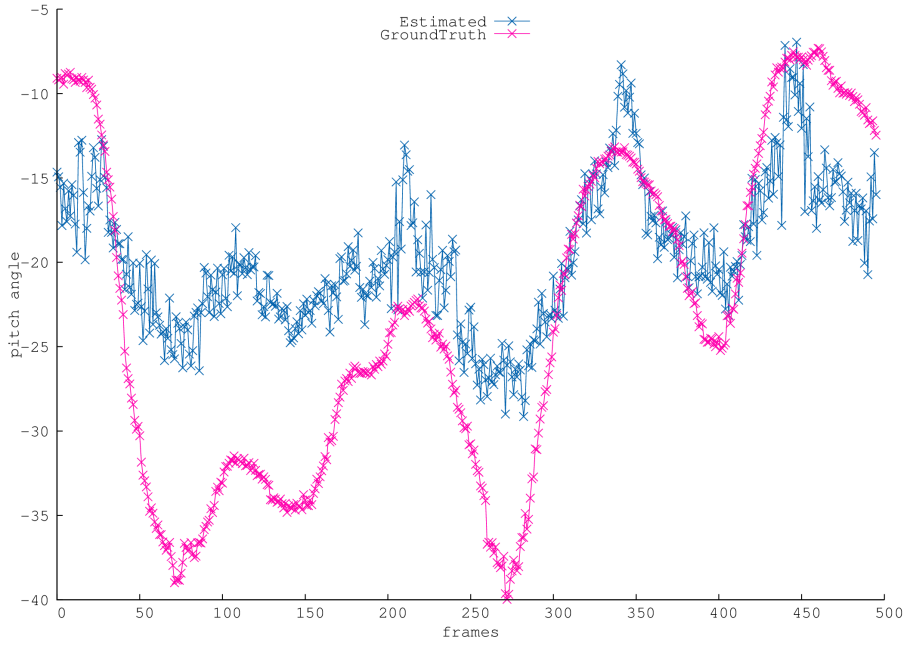


FIGURE 3.8
estimated pitch angle and ground truth

As we increase the tolerance for error, better accuracy can be achieve as can illustrated in figures 3.11 and 3.12. For nose position, the margin of estimated error is extended from 20mm to a range of 15 to 21 mm, the accuracy is increased from below 70% to over 95%. Similarly, the accuracy increases from 45% to 90% as the margin of error increases from 10 to 20.

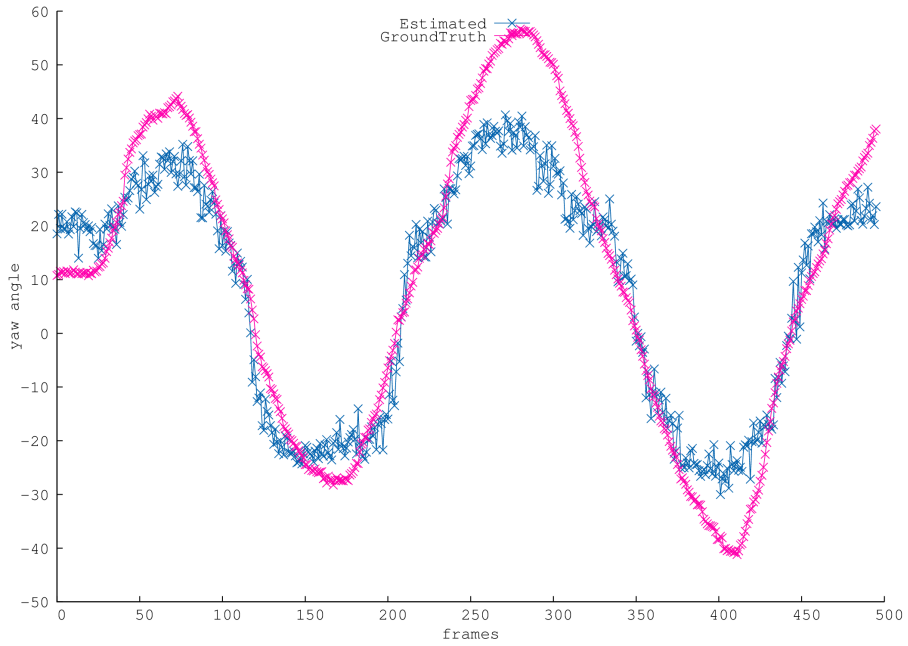


FIGURE 3.9
estimated yaw angle and ground truth

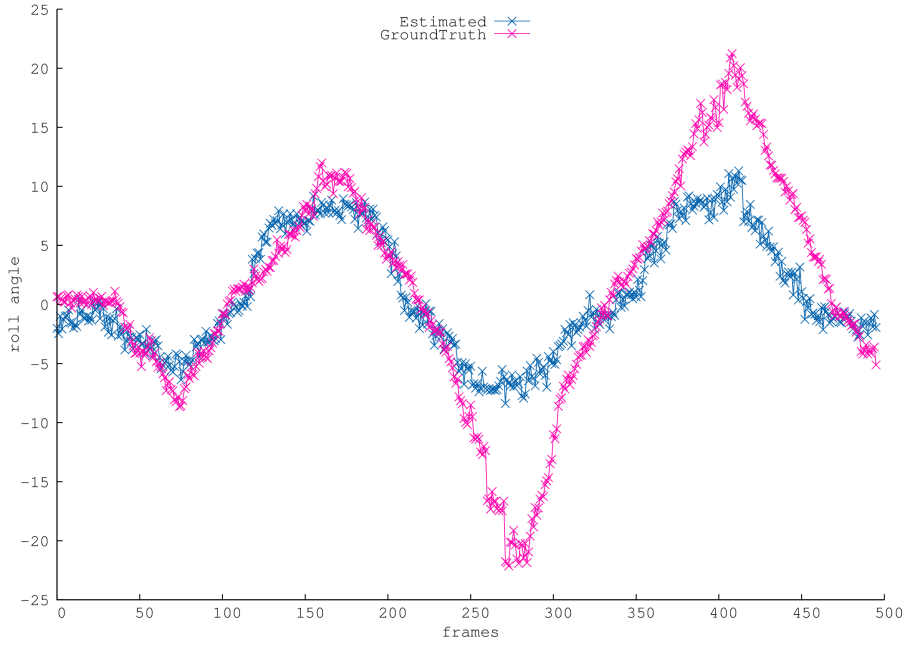


FIGURE 3.10
estimated roll angle and ground truth

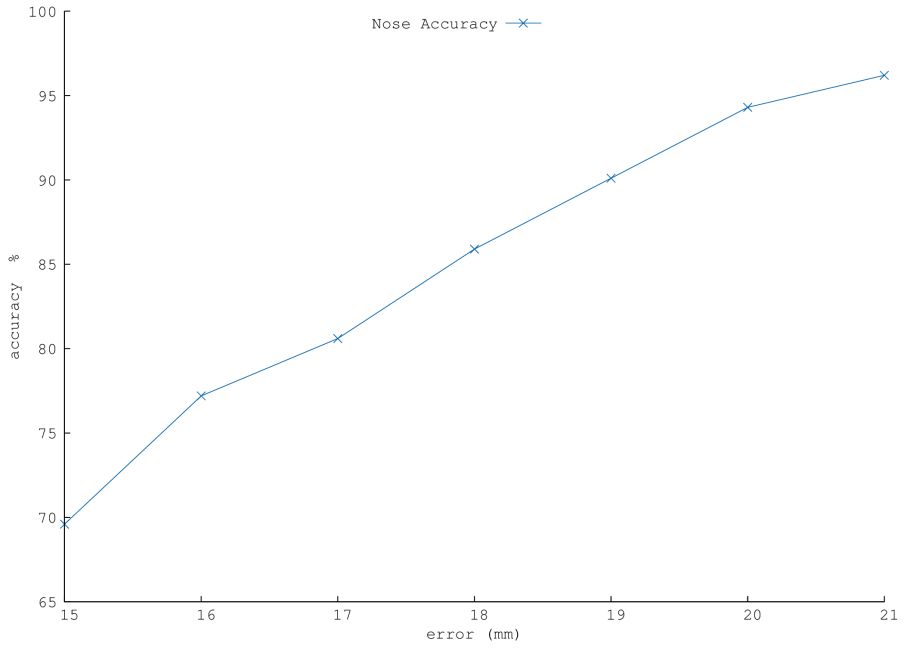


FIGURE 3.11
nose accuracy vs error stride 5

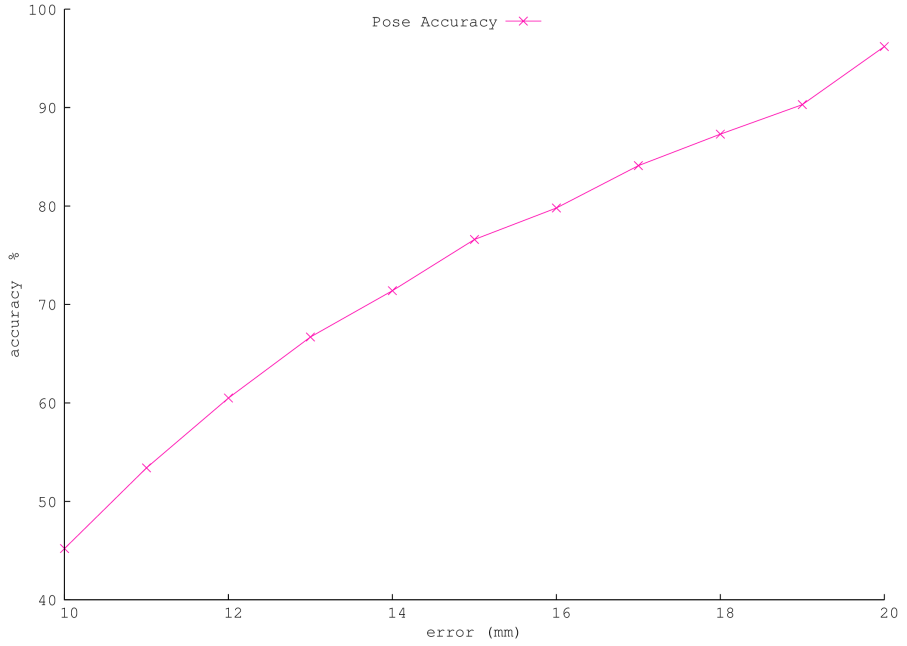


FIGURE 3.12
pose accuracy vs error stride 5

Chapter 4

Facial Feature Points Detection

The second part of this report will focus on robust face feature detection for 2D images. As far as the method is concerned, again a regression forest similar to the one is used for the head pose estimation with a few variations will be built for purpose. In addition, a robust face detector [23] is also used as a prior step for landmark detection. Although, face detection is not exactly in the scope of this project, some of the necessary details for this algorithm will be mentioned in this chapter. Forest training will be discussed in 4.2 and evaluation will be done in 4.3.

4.1 Face Detection

As a preliminary step for feature points detection, face detection is needed since feature points are local information which can only be found around the face area. Haar feature based classifiers are trained to form a such detector.

Firstly, for training images, both positive and negative images are needed. And referencing to sections 2.2 and 2.2.1, rectangular features are calculated from sub windows extracted from each image. Since this pool of candidate features is very big, Adaboost is used to select the best features. [23]

For testing, the most area of an image is non face, we don't want apply every feature selected by Adaboost on every sub windows from test image. Therefore, Viola and Jones "introduce a method for constructing a cascade of classifiers to get good detection rate and decrease the computation time. The idea is to use classifier with less but efficient features to reject the majority of the sub-window before apply more complex classifiers". [23] The two features they used can be seen in Fig 4.1. With the filtering procedure, time taken for processing each image is lower. The graphical description of such process can be seen from Fig 4.2.

FIGURE 4.1
"The two features are shown in the top row and then overlayed on a typical training face in the bottom row. The first feature measures the difference in intensity between the region of the eyes and a region across the upper cheeks. The feature capitalizes on the observation that the eye region is often darker than the cheeks. The second feature compares the intensities in the eye regions to the intensity across the bridge of the nose." [23]

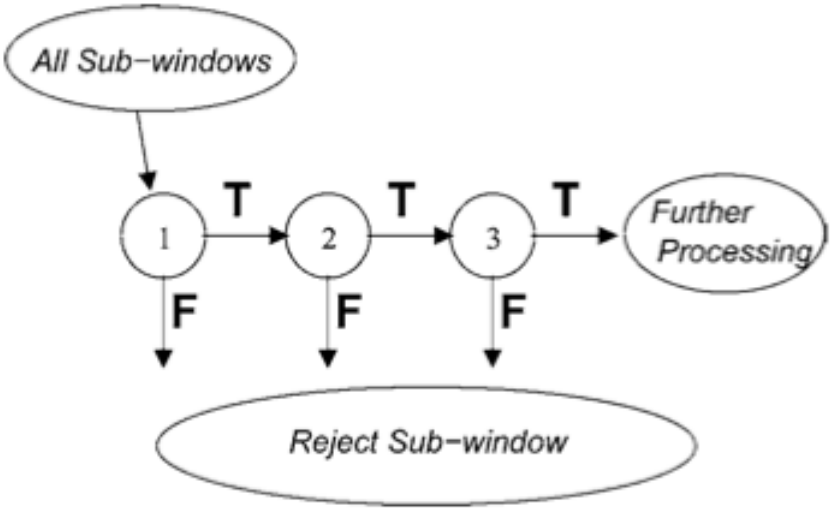


FIGURE 4.2
All sub windows are passed to the cascade of classifiers and in the early stages, the majority of sub windows will be rejected by the initial classifiers and fewer will be left for further processing [23]

4.2 Forest Training for Feature Detection

Similar to the forest built in chapter 3, trees are learned from randomly chosen subsets of all training images. a frontal face detector [23](implemented by opencv as CascadeClassifier) is used to extract the face bounding box from the given image. Squared patches $P = (\alpha^1, \alpha^2, \theta)$ which are again marked positive and negative are randomly extracted within such bounding boxes and outside the box. Where α^1, α^2 are the image features used, which in this case is the grey values of the raw test images and the normalised grey values to compensate for illumination changes respectively as suggested by [9] and θ is the offset from each feature point to the centre of the patch.

A pool of binary tests applied to patches at each non-leaf node is the same as did in section 3.1.2 by using equation 3.3 with different sub rectangular regions and thresholds. And Each node will split patches into two subsets, namely left and right as before.

Maximizing the information gain 3.4 is required in order to find the best split in the pool. The measure of entropy HP would be different from the one used in 3.1.3. Since there are numbers of feature points to be located, $H_r(P)$ could be treat as the class uncertainty measure and it is defined as the following: [9]

$$H_r(P) = -\sum_{n=1}^{N} \frac{\sum_i p(c_n|P_i)}{|P|} log(\frac{\sum_i p(c_n|P_i)}{|P|}) \tag{4.1}$$

$$p(c_n|P_i) \propto exp(-\frac{|d_i^n|}{\lambda}) \tag{4.2}$$

Where $p(c_n|P_i)$ is the probability of a given patch P_i belongs to the feature point n, which is based on the distance between the patch centre and each of the landmark position. [9] And is proportional to $exp(-\frac{|d_i^n|}{\lambda})$. Since we still need to classify each patch during testing at the leaf nodes, a measure of the class uncertainty is required. The measure $H_c(P)$as described in equation 3.8 is used here. Same as the head pose estimation, $H_c(P)$ and $H_r(P)$ are combined by a weighted sum as in equation 3.9. The best test parameters found will be store for each non leaf node.

The objective of forest in chapter 3 is to predict the location of the nose tip, whereas in this case, there are dozens of landmark locations to infer. Therefore at leaf nodes, the

mean and trace of covariance matrices of the offsets to each feature points have to be store.

4.3 Evaluation

The dataset used for this experiment is FaceWarehouse [25]. There are 150+ test subjects and each of which has 20 different images taken with various facial expressions. 74 facial landmarks are annotated with each image. 17 of the facial landmarks such as eye brows, nose, eyes, and mouth were chosen for the purpose of this experiment as shown in Fig 4.3. Image resolution is 640 $\times$ 480. And the faces in this dataset are all frontal faces.

During training, a frontal face detector [23](implemented by opencv as CascadeClassifier) is used to extract the face from the given image. (shown in Fig 4.4). Once the face bounding box is obtained, random patches with size of 80 $\times$ 80 (same size as used for head pose estimation) are extracted according to the bounding box. Number of patches sampled per training image is 80, one quarter of which are negative patches sampled outside the bounding box and the rest is positive patches. The λ in equation 4.2 is set to 0.125 for this experiment as suggested by [9].

During testing, for each input image, face is initially extracted to obtain the bounding box ,in which test patches are densely sampled. Each patch is then passed to the trees in the forest where binary tests are performed at each non leaf node with the parameters stored during training. Once the patch reaches the leaf node, the two necessary conditions for a patch to cast a vote for the location of each feature point are as follows. Firstly, the class probability at that leaf node has to be 1 since with the distortion of negative patches, the distribution stored at leaf node is less informative. Secondly, for every leaf node, trace of covariance matrices for each feature point are stored during training, if such trace exceed certain threshold, the patch will not be allowed to cast a vote for the corresponding feature location. The remaining candidate votes are then filtered by mean shift to remove outliers in order to produce the final estimate.

The successful detection for a feature point is determined by whether the estimated point is within a certain region of the actual point. Such region is define by a circle centred at the ground truth point with a radius of 8 pixels as shown in figure 4.5. Examples of false detection and true detection are shown in figures 4.6 and 4.7.

For this experiment, 10 trees are trained and as the number of tree loaded, the accuracy improves as expected as can be seen from the plots 4.8,4.9 and 4.10. Overall, success

FIGURE 4.3
Example of an image used for feature points detection, landmarks are marked in red

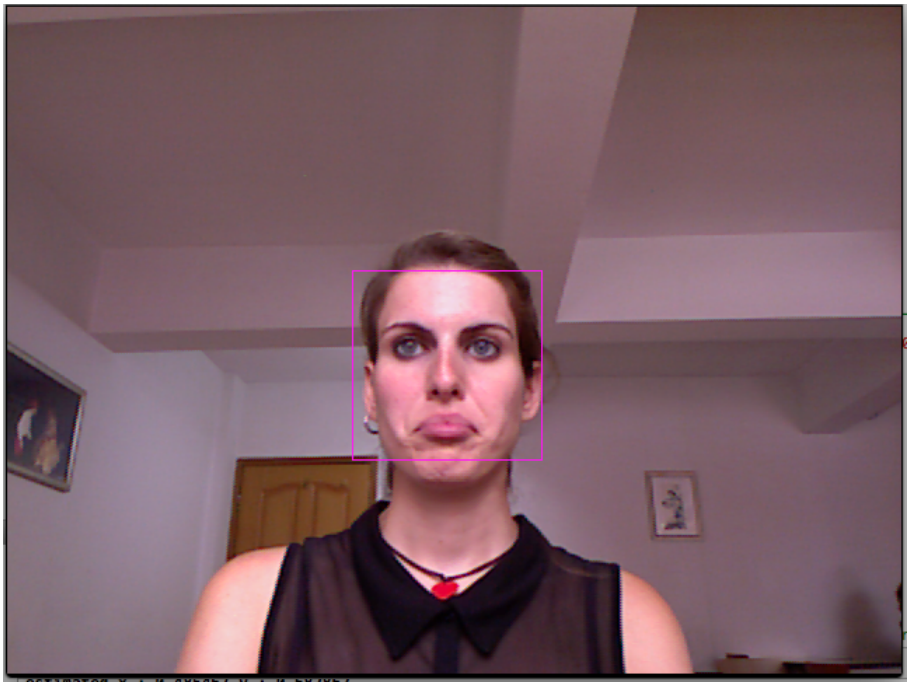


FIGURE 4.4
The region inside the purple bounding box is extracted face

FIGURE 4.5
The red circles are the error tolerance region for each feature point

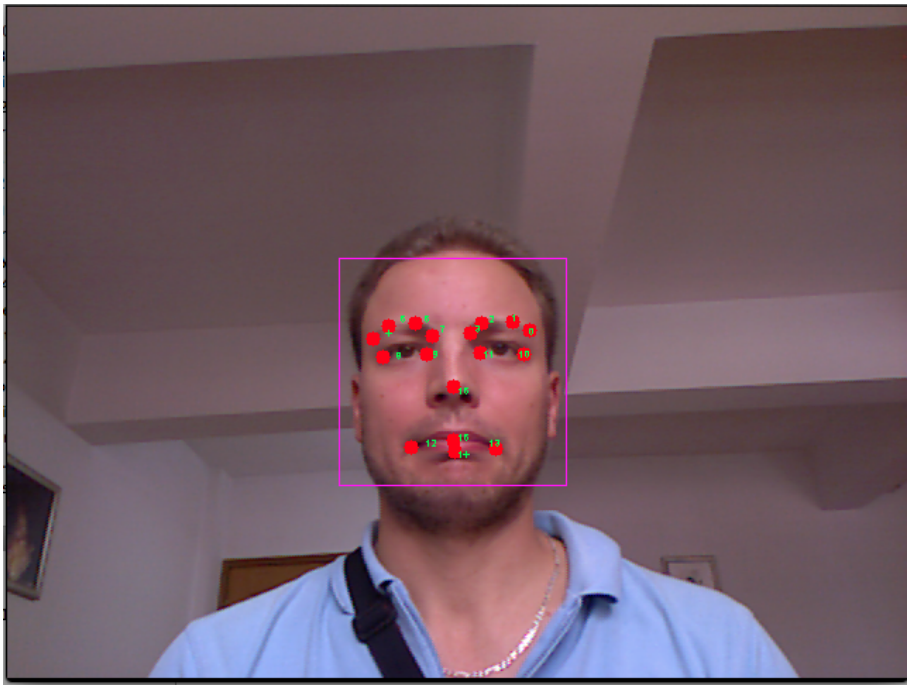


FIGURE 4.6
Example for false detection for some feature points such as points near right eye brow

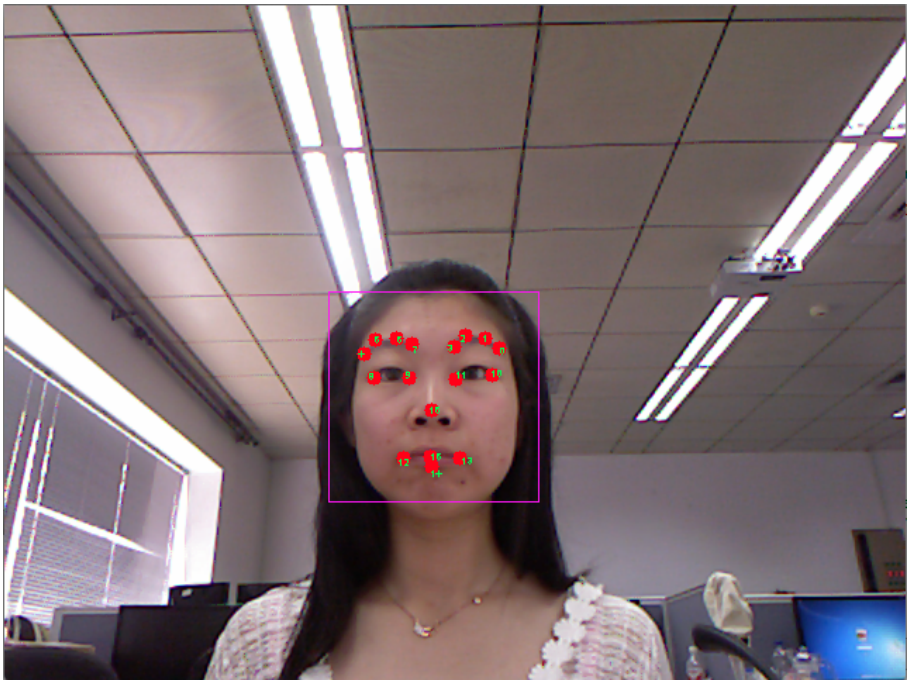
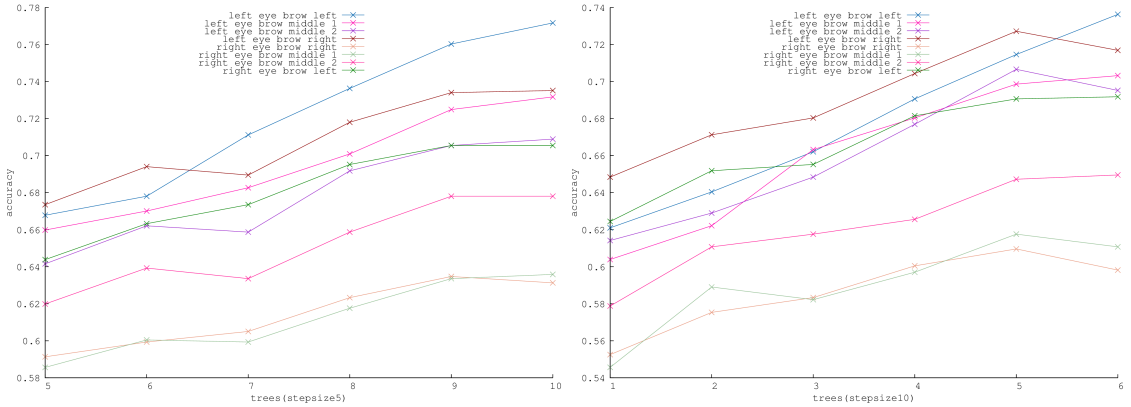


FIGURE 4.7
Majority of the features are detected within the margin of error

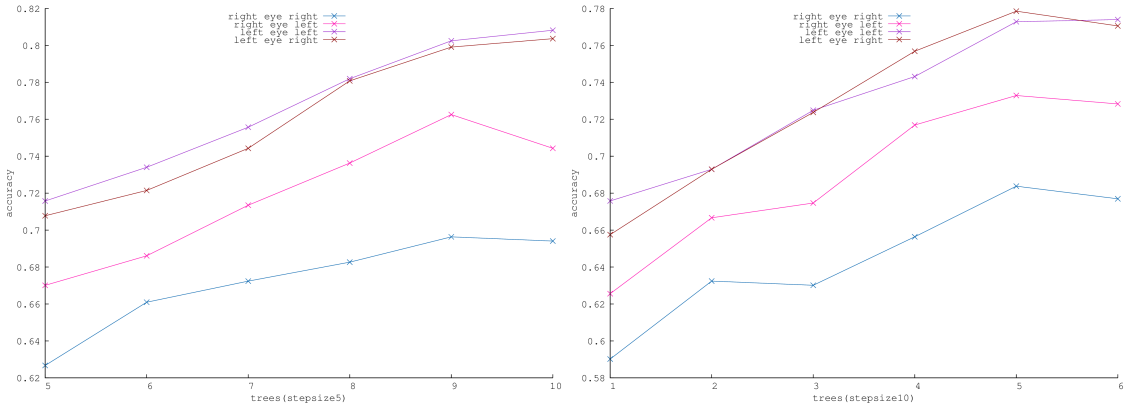


(A) stride 5 (B) stride 10

FIGURE 4.8: Eye brows accuracy vs number of trees

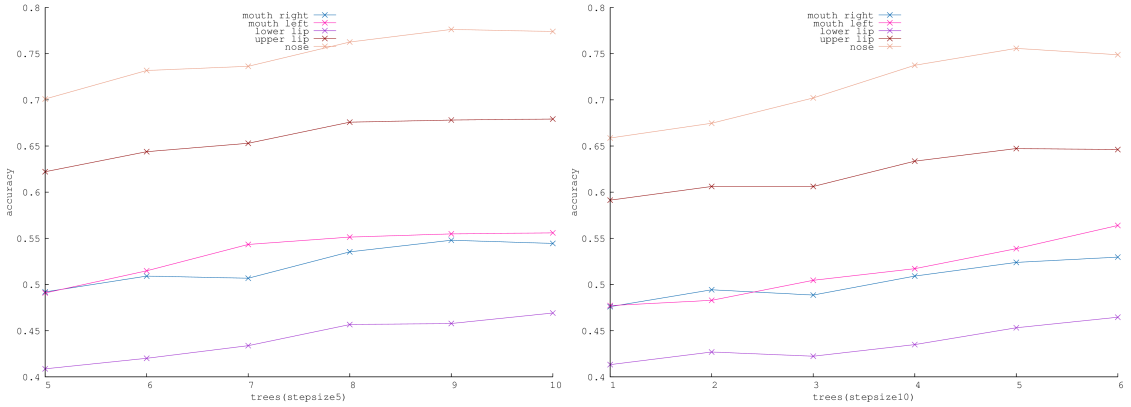
rate of feature points around eyes and nose area is higher than the detection rate around mouth. The reason for such low performance around mouth area might due to the huge deformations. When patches are sampled at a density of 5 stride, the overall accuracy is about 3% higher than when sampled at 10 stride.

In addition, speed is also an important parameter in performance and the time taken to process each image is proportional number of patches sampled. Figure 4.11 shows the relationship between the average time taken to process each image after it has been loaded



(A) stride 5 (B) stride 10

FIGURE 4.9: Eyes accuracy vs number of trees



(A) stride 5 (B) stride 10

FIGURE 4.10: Nose and mouth accuracy vs number of trees

into memory and the number of trees load. When sampling at a density of 5 stride and 10 trees are loaded, the time taken is approximately 0.1 seconds. In other words, at this sampling density, the system can handle 10 frames per second. However, the computation time decreases to only 0.04 seconds (25 frames per second) when patches are sampled at stride 10 which is a 3 % drop in accuracy on average.

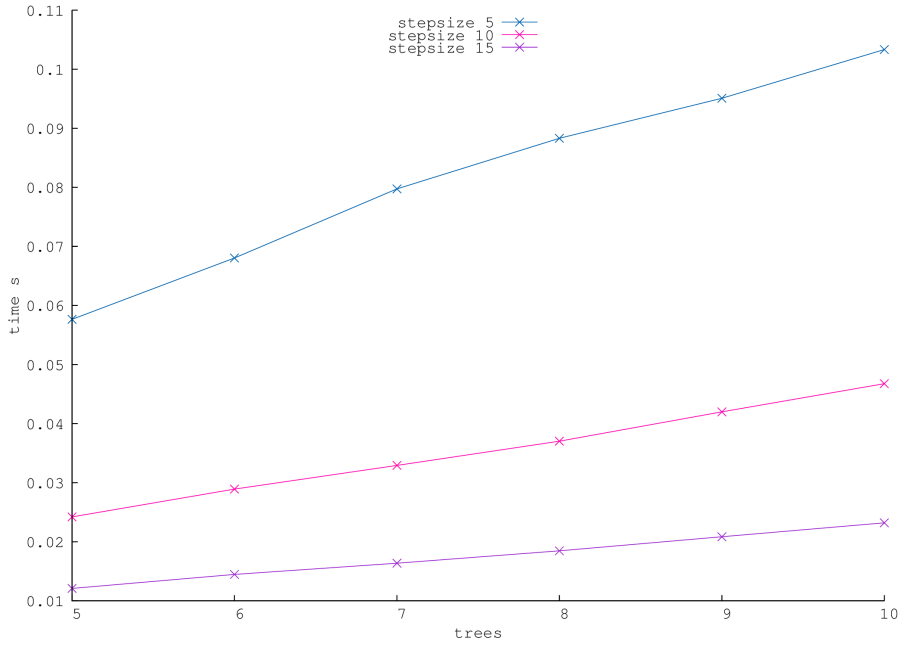


FIGURE 4.11
average processing time for each image(frame)various densities, stride 5,10,15 in blue,pink,purple respectively

Chapter 5

Conclusion and Further Work

Bibliography

[1] Erik G. Learned-Miller, "Introduction to Computer Vision," 2011.

[2] Gabriele Fanelli, Matthias Dantone, Juergen Gall, Andrea Fossati, Luc Van Gool, "Random forests for real time 3D face analysis," International Journal of Computer Vision, 101(3):437-458, 2013.

[3] Erik Murphy-Chutorian, Mohan Manubhai Trivedi, "Head Pose Estimation in Computer Vision: A Survey," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.31, no.4, pp.607,626, April 2009.

[4] A. Criminisi, J. Shotton, E. Konukoglu, "Decision Forests for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning," 2011.

[5] Carla E. Brodley, Paul E. Utgoff, "Multivariate Decision Trees," Journal Machine Learning Volume 19 Issue 1, Pages 45 - 77 April 1995 .

[6] Gabriele Fanelli, Luc Van Gool, Juergen Gall , "Real Time Head Pose Estimation with Random Regression Forests," Proceeding CVPR '11 Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition Pages 617-624 ,2011.

[7] Leo Breiman "Random forests. Machine Learning," Journal Machine Learning Volume 45, Issue 1 , pp 5-32, 2001.

[8] Tin Kam Ho "The Random Subspace Method for Construction Decision Forests,"Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.20, no.8, pp.832,844, Aug 1998.

[9] Matthias Dantone, Juergen Gall, Gabriele Fanelli, Luc Van Gool "Real-time Facial Feature Detection using Conditional Regression Forests ," Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on , pp. 2578,2585, 16-21 June 2012.

[10] Sotiris Malassiotis, Michael G. Strintzis "Pose and illumination compensation for 3D face recognition," Image Processing, 2004. ICIP '04. 2004 International Conference on , vol.1, pp.91,94, 24-27 Oct.2004.

[11] Angela Caunce, David Cristinacce, Chris Taylor, Tim Cootes "Locating Facial Features and Pose Estimation Using a 3D Shape Model," ISVC (1) 2009: 750-761

[12] Guodong Guo, Yun Fu, Thomas S. Huang, Charles R. Dyer "Locally Adjusted Robust Regression for Human Age Estimation,"Applications of Computer Vision, 2008. WACV 2008. IEEE Workshop on , pp.1,6, 7-9 Jan. 2008

[13] Guodong Guo, Yun Fu, Thomas S. Huang, Charles R. Dyer "Head Pose Estimation: Classification or Regression?," Pattern Recognition,ICPR 2008. 19th International Conference on , pp.1,4, 8-11 Dec. 2008

[14] Thomas G. Dietterich "An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization,"Journal Machine Learning,Vol 40 Issue 2, Pages 139 - 157 ,Aug 2000

[15] Paul E. Utgoff "Incremental Induction of Decision Trees," Machine Learning, Volume 4, Issue 2, pp 161-186, Nov 1989

[16] Feng Zhou, Jonathan Brandt, Zhe Lin"Exemplar-based Graph Matching for Robust Facial Landmark Localization," IEEE International Conference on Computer Vision (ICCV), 2013

[17] Clement Creusot, Nick Pears, Jim Austin"3D Face Landmark Labelling,"Proceeding 3DOR '10 Proceedings of the ACM workshop on 3D object retrieval, Pages 27-32, 2010

[18] Stylianos Asteriadis, Kostas Karpouzis, Stefanos Kollias"Head Pose Estimation with One Camera, in Uncalibrated Environments," Proceeding EGIHMI '10 Proceedings of the 2010 workshop on Eye gaze in intelligent human machine interaction, Pages 55-62, 2010

[19] Roberto Brunelli, Tomaso Poggio"Face Recognition: Features versus Templates," Journal IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 15 Issue 10, Page 1042-1052, Oct 1993

[20] Xiaowei Zhao, Shiguang Shan, Xiujuan Chai, Xilin Chen"Cascaded Shape Space Pruning for Robust Facial Landmark Detection," nternational Conference on Computer Vision (ICCV2013), Sydney, Australia, December 1-8,2013

[21] Tarun Kumar, Karun Verma"A Theory Based on Conversion of RGB image to Gray image,"International Journal of Computer Applications (0975 – 8887) Volume 7- No.2 Sep 2010

[22] Simone Frintrop,Maria Klodt,Erich Rome "A Real-time Visual Attention System Using Integral Images," 2007

[23] Viola, P. and Jones, M. J. Robust Real-Time Face Detection. International Journal of Computer Vision (IJCV) 57 (2, 2004) 137–154.

[24] Okada, R.: Discriminative generalized hough transform for object detection. In: International Conference on Computer Vision (2009)

[25] Chen Cao, Yanlin Weng, Shun Zhou, Yiying Tong, Kun Zhou, FaceWarehouse: a 3D Facial Expression Database for Visual Computing

[26] Robert E. Schapire, Explaining AdaBoost