Real Time Facial Landmark Prediction with Regression Trees and Online Multiple Instance Learning

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT

FOR THE AWARD OF DEGREE

OF

MASTER OF TECHNOLOGY
IN
SIGNAL PROCESSING & DIGITAL DESIGN

Submitted by

SUMANA BHANDARI
2K14/SPD/501

Under the supervision of

Sh. Rajesh Rohilla
(Associate Professor, Department of ECE)

Department of Electronics & Communication Engineering
Delhi Technological University
(Formerly Delhi College of Engineering)
Delhi-110042

2014 – 2017

DECLARATION

This is to declare that the dissertation with the title "Real Time Facial Landmark Prediction with Regression Trees and Online Multiple Instance Learning" is the result of authentic work under the guidance of Mr Rajesh Rohilla, Delhi University of Technology. I have followed the instructions as required by theses writing rules. This project report is submitted in partial fulfillment of all the requirements for the degree of Master Of Technology in Electronics and Communication Engineering at Delhi Technological University, Delhi for the 2014-2017 academic session. The project experiments included in this thesis are performed by me and not procured from any other source. Any reference for theoretical purpose are duly acknowledged and the respective original publication are mentioned.

Sumana Bhandari (2K14SPD501)

CERTIFICATE

This is to certify that the dissertation titled **"Real Time Facial Landmark Prediction with Regression Trees and Online Multiple Instance Learning"** submitted by **Ms. Sumana Bhandari**, Roll. No. 2K14/SPD/501, in partial fulfilment for the award of degree of Master of Technology in Signal Processing & Digital Design at **Delhi Technological University, Delhi**, is an authentic record of student's genuine work carried out by her under my supervision and guidance in the academic session 2016-17.

Rajesh Rohilla
Supervisor
Associate Professor
ECE Dept., Delhi Technological University

Acknowledgment

I express my gratitude and appreciation towards my supervisor **Mr. Rajesh Rohilla, Associate Professor** Department of Electronics and Communication, for his constant support and amazing encouragement which helped in successful completion of my thesis.

I am also thankful to **Dr. S. Indu,** Head of Department (Electronics & Communication Engineering), towards all the faculty and staff of Electronics & Communication Engineering and friends for their ideas and discussion which were useful in completion of this **"Thesis"** work.

I wish to indebt my deep sense of appreciativeness towards family and friends which were great source of encourage who bestowed upon me their grace and were source of my inspiration and hard work.

Sumana Bhandari
M.Tech (SPDD)
2K14/SPD/501

Dedicated to my loving Parents, Guide, Friends

And

The Almighty…

TABLE OF CONTENTS: Page

LIST OF FIGURES

ABSTRACT

Detection of Facial feature landmark is implemented by the proposed and efficient algorithm which works absolutely in real-time. Recent algorithms detecting facial features on 2D images requires good quality images. The method demonstrates how an Ensemble of Regression trees with Online Multiple Instance Learning (MIL), can be used for the estimation of face's landmark positions, thus achieving actual real time speed and performance with the accurate predictions. Based on this, a general structure/framework is presented based on gradient boosting on learning the regression trees ensemble. This procedure optimizes the desired metric namely, sum of square error loss, along with instance label classifiers helping in achieving superior results with real-time performance. Thus significant improvement is achieved over state of the art methods. Lastly, we analyze the how appropriate priors helps with the selection of efficient features in the structure of image data.

INTRODUCTION

Real-time face detection and facial landmark prediction are very important to monitor human facial variations, e.g. drowsiness detection, monitoring eye blinks, head pose estimation, face alignment, human computer interface that eases communication in case of disabled people and for anti-spoof protection. Facial Landmarks localizes and represents salient regions of the human face, such as:

- Right Eye
- Left Eye
- Nose
- Jaw
- Right Eyebrow
- Left Eyebrow
- Mouth

Facial Landmarks prediction is the subset of shape prediction method. In context of facial landmarks, we detect the key facial structures on face using the methods of shape prediction.

The method starts as follows:

Using Regression Trees to estimate the facial landmark positions directly from the sparse subset of pixel's intensities. Thus, Gradient Boosting framework is used for learning the regression trees ensemble that optimize sum of squared error loss, therefore naturally handling missing and partial labelled data. Using appropriate priors exploits the image data structure that efficiently helps with feature selection. We demonstrate the same on real-time head pose detection and estimation along with face alignment.

From thereon, we propose an efficient and simple algorithm to monitor and analyze facial expressions by using the distribution of facial features as input of a classification system in order to recognize expressions from video sequences.

1.1 CHALLENGES

There has been a number of issues that has been addressed in the proposed solutions:

1. The facial features from a face image, extracted in the vector form can vary greatly due to deformation in shape and changing conditions in illumination. That makes feature and shape estimation accuracy difficult. This presents a dilemma as reliable features are needed to predict shape accurately, and on the same lines, an accurate shape estimate is needed to extract the reliable features. Hence, an iterative approach is taken to deal with the problem.
2. Landmarks detection is complicated if unique identity in face appearance, position of the faces in image, rigid transformations variability in individual's expression and partial occlusion by other objects on the faces greatly complicates the task.
3. Usually there are implicit and stringent requirements on setup, relative head-orientation, image resolution, motion dynamics, illumination, etc.

1.2 CONTRIBUTION

In focus of the mentioned challenges, the following contributions are made:

1. The semantic information viz. facial landmark key points are denoted as shape vector S, viz. (x_1, y_1, x_2, y_2 … x_n, y_n) where (x_i, y_i) represents the landmark within the image and n represents the landmarks which is needed to detect. Shape Invariant feature selection based on ensemble of regression trees minimizes the loss function during training time.

2. To achieve good and measurable performance, the feature points are applied on a tracking method called Online Multiple Instance Learning (MIL) which further improves the estimation the facial landmarks position.

3. To achieve efficient performance, important parameters are identified.

4. Results, Quantitative and Qualitative, are presented that confirms that the regression trees with Online Multiple Instance Learning (MIL) results in high quality predictions with more efficiency than the previous method.

TECHNICAL PRELIMINARIES

2.1 Facial Landmark Features

The problem of real-time facial feature detection concerns the localizing the facial feature points or landmarks deterministically. Landmarks examples are points that are positioned around jaw, nose, eyes or the lips. These regions carries the most significant semantic.

Hence it deals with detecting distinctive features automatically in human faces. The landmark features detected are used in several applications that are demonstrated in this paper.

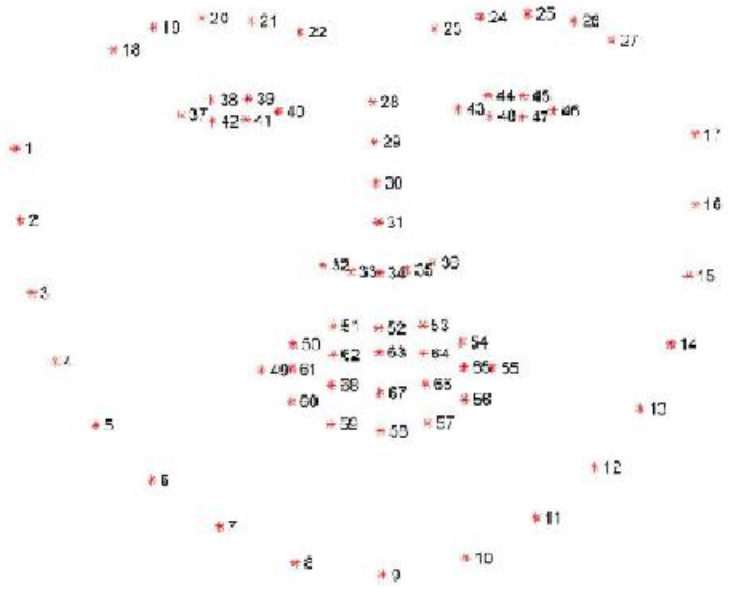


Fig 2.1. Visualizing the 68-facial coordinate points in the iBUG dataset

Therefore, prior to detection of facial features, the first work is to find out where the face is located in the frame of the image of video stream. Typically, this is

solved with an annotated rectangle bounding box which contains the face to be detected. Hence, the facial keypoint detection pipeline starts with detecting the face, which accepts pixel intensities matrices and returns the faces detected in the bounding boxes as the output.

Facial Landmarks detection is the subset of shape prediction problem. In the input image, the ROI which specifies the object detected, the shape predictor attempts in localizing the interested keypoint along the object's shape. Hence, using shape prediction methods, the goal is to detect the important facial landmark structures on the person's face.

Therefore, the Facial Landmarks detection is a step-by-step process:

1 Step#1: Localizing the face in the frame.

2 Step#2: Detecting the required facial structures on face ROI.

This method is as follows:

1 A trained set of facial landmarks labelled on an image. The images as labeled with specific (x, y) coordinates of locations that surrounds each of the facial structures.

2 Priors, the probability on nearest distance calculated between the pairs of each input pixels.

End results obtained is facial landmark detector which can be used to detect the facial keypoints in images or real-time along with high-quality and reliable predictions.

2.2 Decision Trees

The approach is a powerful method for shape prediction in machine learning. Decision Trees are the foundation for the ensemble methods such as gradient boosting, bagging and random forests. The input is a feature vector S to the decision tree, which is placed on the root and then yields down to a leaf node. Question is represented by the tree's internal node and consequently branch is based on the answer. However, this is not a binary tree since the question is not typically mere yes/no. The decision tree structure is depicted below in [Fig. 2.2].

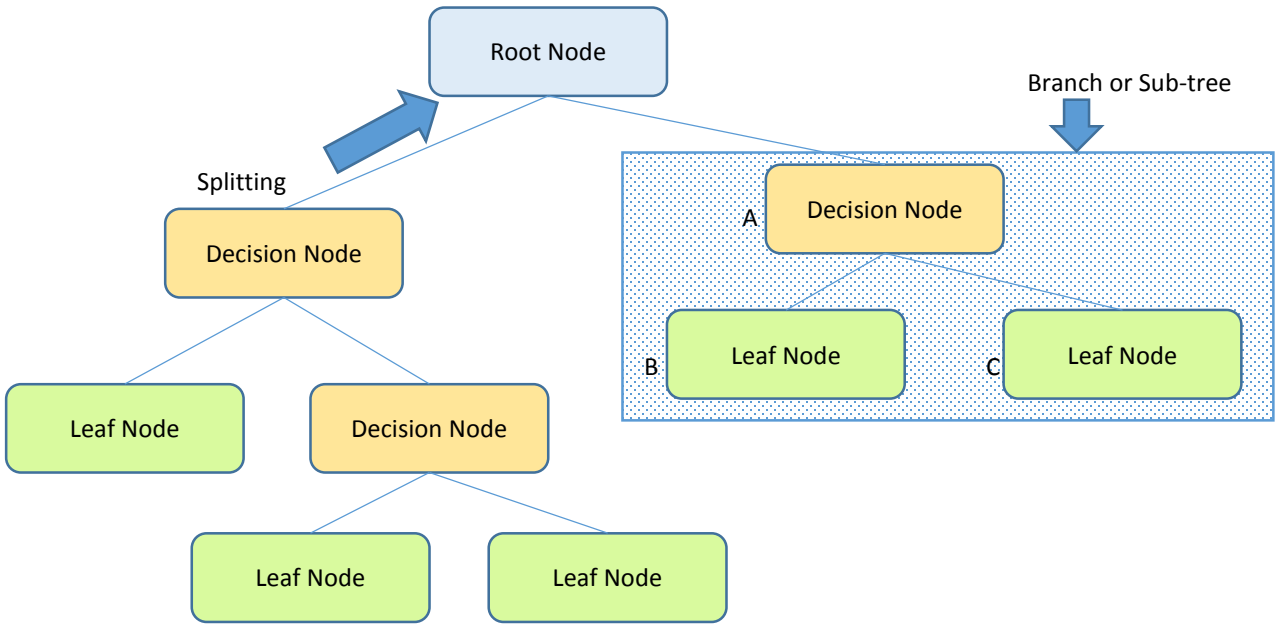


Fig. 2.2: Decision Trees Terminology

NOTE:

1 **Root Node**: Represents sample that further divides into homogenous sets.
2 **Decision Node**: Sub-node splits to further sub-nodes.
3 **Leaf Node**: Terminal nodes that do not split further.
4 **Pruning**: Opposite process of splitting. Removing sub-nodes of a decision node.

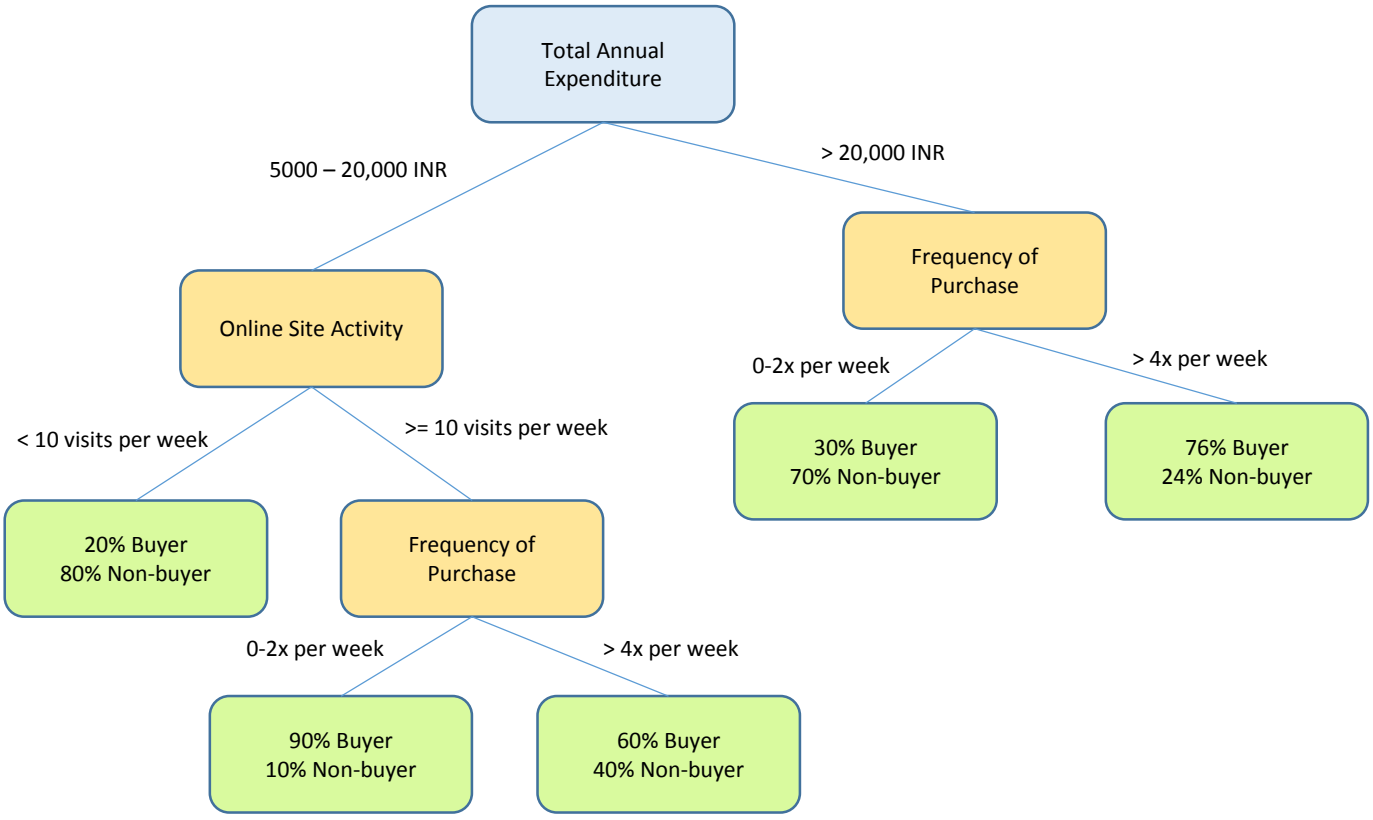


Fig. 2.3: Decision Tree Example in e-commerce setting

Decision tree training requires the important decision to be taken on features to split on & what thresholds it can be compared with.

Data is represented as:

$$(x,Y) = (x_1,x_2,x_3 \ldots x_K,Y) \quad (1)$$

The classification or generalized variable is Y which is dependent or target variable. The input variables x_1,x_2,x_3 etc, are the input variables used for the task.

In data mining, decision trees are two important types:

Classification Tree – Data analysis method where the class to which the data belongs is the predicted outcome.

Regression Tree – Data analysis method where the anticipated outcome is a real number.

CART Algorithm

Decision Trees are also referred to as "Classification & Regression Trees" that are used for classification that has similarities and dissimilarities, like procedure to determine where to split.

Metrics

Decision trees uses algorithms to decide on splitting a node into two or more sub nodes. The creation of resultant sub-nodes have more purity or homogeneity with respect to the target variable. The algorithm selection is based on target variable type and works on top-down approach. The best split are measured by different algorithms using different metrics. There are three metrics when put to each and every resulting subset, the resulting values are averaged or summed to provide the quality measurement of split.

Gini Impurity

Gini Index is the cost function that is used for split evaluation in the dataset. It measures incorrectly labeled element that is chosen randomly from the set that was randomly labeled according to distribution of the labels in given subset.

Defined as adding the probability P_i with label i of an item selected, times the Probability 1 - P_i of mistake in categorizing the item. It becomes 0 or minimum, if all cases in node falls unanimously into one category of target.

$$I_x(f) = \Sigma P_i (1- P_i) = \Sigma (P_i- P_i^2) = \Sigma P_i- \Sigma P_i^2 = 1 - \Sigma P_i^2 = \Sigma P_i P_K (i \neq k) \qquad (2)$$

Information Gain

Information Gain decides on which feature for split on at step in tree building. Hence, only the split is chosen that gives the pure daughter nodes. Nodes that are impure require less information for description and nodes that are more impure require of more information. Information Theory states that Entropy measures the disorganization degree in system. Entropy is equal to zero when the sample is pure or homogeneous and the entropy is one when the sample is equally or similarly divided.

$$\text{Entropy} = -a \log_2 a - b \log_2 b \qquad (3)$$

a is the probability of success and b is the probability of failure in that node. Entropy choses the split that has lowest entropy compared to the parent node and the other splits. The best case is lesser entropy.

Variance Reduction

The above algorithms were discussed for categorical target variable. For target variables that are continuous in regression problems, the Variance Reduction algorithm is used. To choose the desired split, variance standard formula is used

in algorithm. The spilt that results to lower variance is chosen as the criteria for population split.

$$\text{Variance} = \Sigma (Y - \bar{Y})^2 / n \quad (4)$$

$\bar{Y}$ is mean and Y is actual.

Next, overfitting is one of the key problems while decision tree modelling. Training set will give 100% accurate results if no limit is set on decision tree since it will make one leaf node for every observation. Thus, there are two ways in preventing overfitting:

1 Setting Tree Size Constraints

2 Tree Pruning

This will be further discussed in Method 3.1.

2.3 Regression Trees and Ensemble

In Ensemble Methods, a single estimator's generalizability or robustness is improved by combining the predictions of parent estimators that is based on a learning algorithm. Hence, in **Boosting Methods**, the base estimators are sequentially built and the bias of the combined estimator is reduced. Thus powerful ensemble is produced after combining weak estimators. The examples are AdaBoost, Random Forest, Gradient Tree Boosting.

The Tree ensemble methods are invariant to the scaling of inputs, so normalization of features is not needed.

Model: Assuming there are K trees

$$\hat{y}_i = \sum^{K}_{k=1} f_k(x_i),\ f_k \in \mathcal{F} \quad (5)$$

$\mathcal{F}$ – Function space containing all the Regression Trees

Parameters:

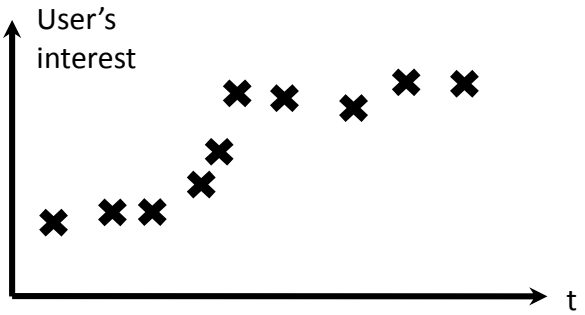
- Includes the tree structure and the leaf score.
- Functions are parameters are used:

  $$\Theta = \{f_1, f_2 \ldots f_K\} \quad (6)$$

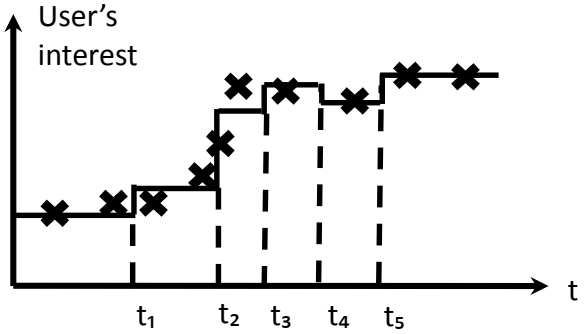
- Instead of learning weights in $\mathbf{R}^d$, we learn trees or functions.
- The objectives viz. loss and regularization are defined and optimized.

  Single variable regression tree has objectives:

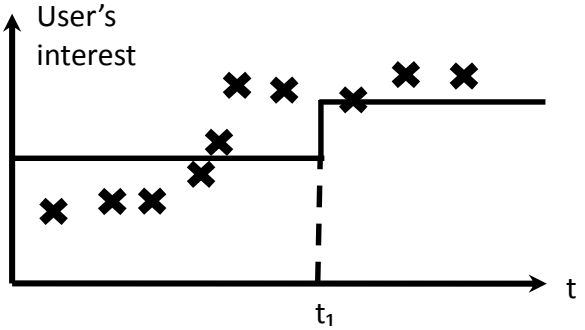
  - Training Loss: The function fitting on the points.
  - Regularization: Defining the function complexity i.e. the number of splitting points, segment height.



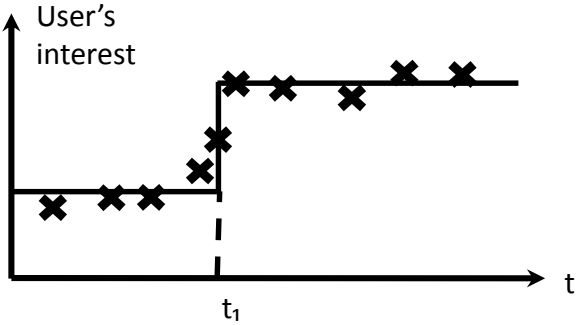
Step function on k



Too many splits Ω(f) is high



Wrong split point L (f) is high



Good balance of Ω (f) and L(f)

XX

Model: Assuming there are K trees

$$\hat{y}_i = \sum^{K}_{k=1} f_k (x_i), f_k \in f \quad (7)$$

Objective:

$$Obj = \sum_{i^N=1} l (y_i, \hat{y}_i) + \sum^{K}_{k=1} \Omega (f_k) \quad (8)$$

$$Training\ Loss = \sum_{i^N=1} l (y_i, \hat{y}_i) \quad (9)$$

$$Complexity\ of\ Trees = \sum^{K}_{k=1} \Omega (f_k) \quad (10)$$

There are possible ways to define Ω by number of nodes in tree or depth, L2 norm of leaf weights

The prediction score is defined by the Regression Tree Ensemble that depends on the objective function definition.

Using Square Loss

$$l (y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2 \quad (11)$$

results in Gradient Boosting Machine.

2.4 Boosting

Boosting is an ensemble method that generally creates strong classifier from many weak classifiers. Thus, Boosting uses an weak classifiers ensemble that "boost" the classifiers to produce a strong classifier. Weak classifiers considers "better than random guess" i.e., they are really simplistic models.
However, when their outputs are combined, they have a better result

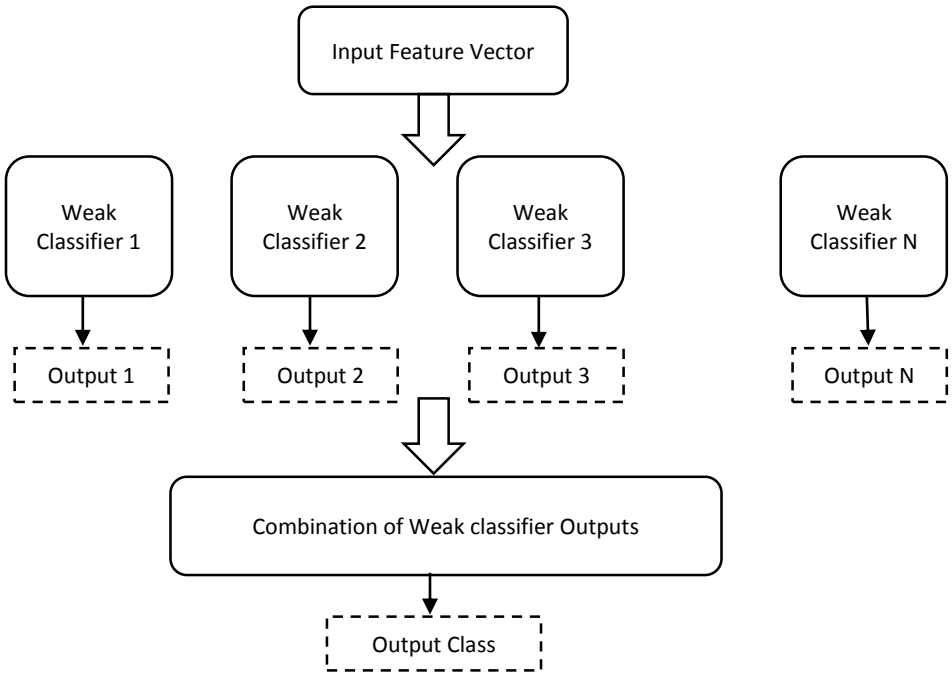


Fig. 2.4: Representation of ensemble/boosting classification methods

From the pictorial representation, the model is built from training data, then gradually creates a second model as it attempts to rectify the errors from thereon. Models is kept on adding with maximum number of of models added till training set is precisely predicted. **AdaBoost** is the most successful boosting algorithm developed for binary classification.

Gradient Boosting

Gradient Boosting works with three elements:

1 **Loss Function**: In this project, regression uses squared error. It is differentiable.

2 **Weak Learner**: The weak learners in Gradient Boosting are decision trees used. Here, specifically regression tree are used whose subsequent model outputs when added "corrects" the predictions' residuals. The tree is built in "greedy manner", with the split points best chosen based on loss minimization or purity scores.

3 **Additive Model**: Gradient descent minimizes parameters' set, such as coefficients in regression function or the weights in neural network. After error or loss calculation, error is minimized by updating the weights.

In weak learner sub models or decision trees, after loss calculation, to perform gradient descent. A tree is added to the model that follows the gradient and hence results in loss reduction. Tree parameterizing is done, then tree parameters are modified and moved in such a direction so that it results in less residual loss. The new tree's output is then added to output of existing sequence of trees in the effort to correct or improve the model's final output.

2.5 Online Multiple Instance Learning

Multiple Instance Learning (MIL) is a method for targeting problems having incomplete base of knowledge of training examples' labels. Every training sample in supervised learning is assigned with real valued label. However, in MIL to each bag of instances, training labels are given.
For example a binary case, if any one instance in the bag is positive then the bag is labeled as positive and if all the instances in the bag are negative then the bag is labeled negative.The goal of the MIL method is classification of unseen instances or bags that is based on the labeled bags as training data.
The MIL Tracker considers the neighborhood around the current location of the object in order to generate many potential positive samples. There may be apprehensions that it is a bad idea because the object may not be centered in most of the positive samples. This is where the MIL differs. In MIL, the concept of positive and negative "bags" exists instead of positive and negative examples. The images collection are not all positive examples in the positive bag. Instead, only one image in positive bag need to be positive sample.

In the example presented above, a patch that is centered on the current location of object is contained along with the patches in small neighborhood around the object. Hence, when the samples from neighborhood of current location are put in bag, even if the tracked object's current location is inaccurate, a good chance is there that the bag contains at-least one image in which the target object is nicely centered. The MIL does a reasonable job under partial occlusion also.

The appearance model is updated by MIL with image patches set, although not sure of the image patch that contains the fully centered target object. This algorithm works at real-time speeds.

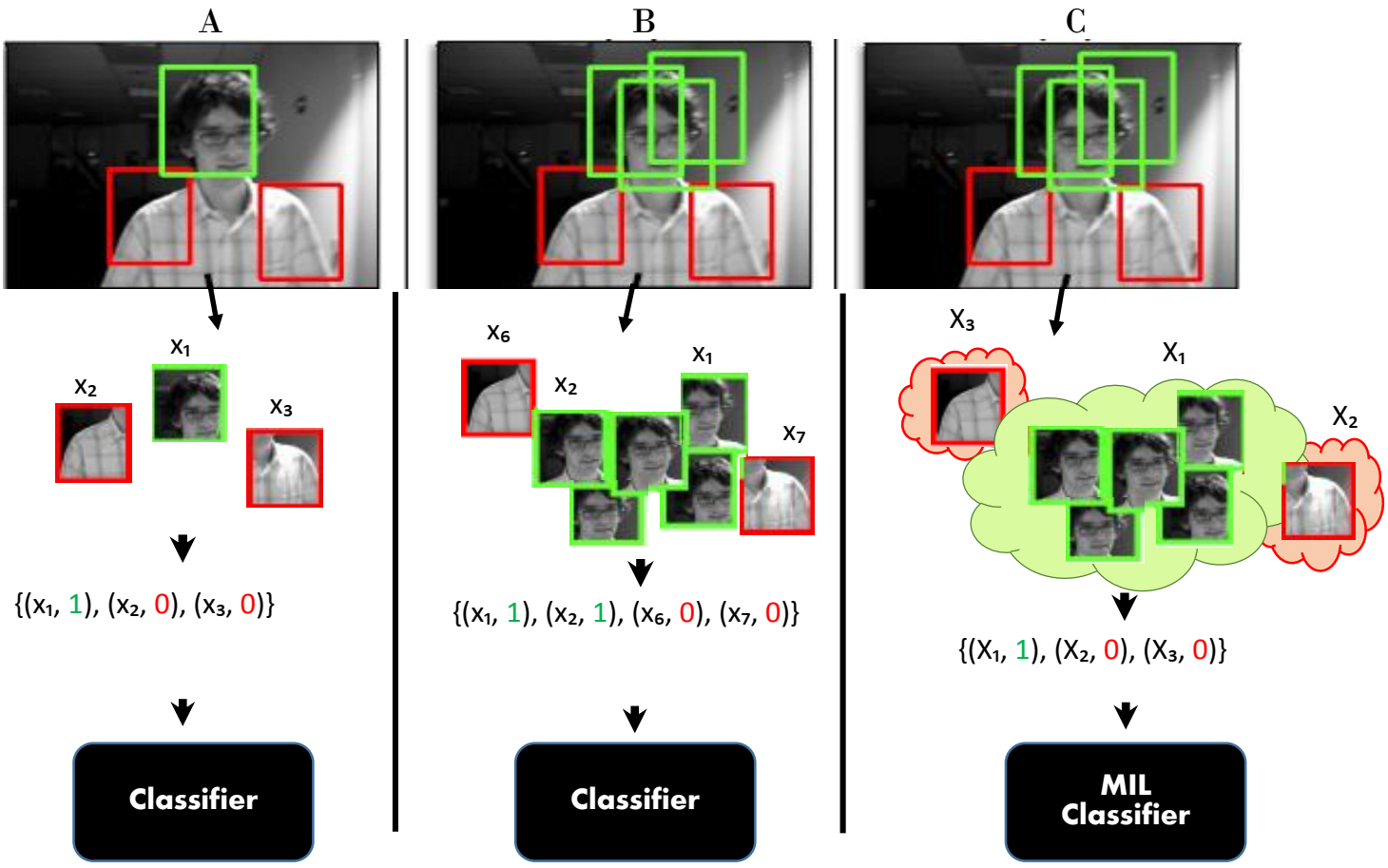


Fig 2.5 illustrates the three different strategies built for a standard classifier and MIL Classifier.

(A) Discriminative classifier updated using only a single positive image. Notice that positive image patch is not perfectly capturing the target object.

(B) Updating discriminative classifier using several positive images. This confuses the classifier resulting in poor performance.

(C) Updating MIL classifier using a positive bag consisting of several image patches.

This will be further discussed in Section 3.2.

METHOD

3.1 Facial Landmark Detection and Prediction with regressors

In this method, the primitive regressor used is a decision tree. Thus, the Cascaded Shape regression framework is called as ensemble of the Regression Trees. The aim of building decision tree is to minimize the sum of square error in training, which is same goal as in testing. There are three rules that governs the decision tree:

1 The method maximizes the **reduction in variance** in child nodes since the split in each of the internal node of a decision tree will be chosen from the random pool of candidate splits.

2 Every leaf node consists of mean of all the training samples on the leaf multiplied with the regularization parameter ν=0.1 multiplied with each leaf node.

3 When the splits are chosen at the internal nodes, a group of features are selected at random with the exponential prior distribution, which are biased towards the closer pixel-pairs. From this pool onwards, the features are selected to maximize the reduction in variance.

3.1.2 Method

The method represents the algorithm that precisely estimates the facial landmarks position.

3.1.2.1 Cascade of Regressors

Some notations are introduced.
In the image I, $x_i \in \mathbb{R}^2$, are the 2-D coordinates.

$S = (x_1^T, x_2^T \ldots x^T)^T \in \mathbb{R}^{2p}$ denotes coordinates of the p facial landmarks in I.
$\hat{S}$ = current estimate of S

Each regressor r_i (.,.) in cascade predicts update vector from the image and $\hat{S}$ is added to current shape estimate to improve the estimate. The necessary point of the cascade is that regressor r_i predicts based on features.

$$\hat{S}^{(t+1)} = \hat{S}^{(t)} + r_t(I, \hat{S}^{(t)}) \tag{12}$$

The mean shape of the training data is selected as initial shape which is always centered and scaled according to the bounding box output of face detector.

3.1.2.2 Cascade Regressor Learning

To learn r_o in first cascade, we create training data triplets from face image, an initial shape estimate and target update step, i.e.

$$(I_{\pi_i}, \hat{S}_i^{(0)}, \Delta S_i^{(0)})$$

$$\pi_i \in \{1, \ldots, n\} \tag{13}$$

$$\hat{\mathbf{S}}_i^{(0)} \in \{\mathbf{S}_1, \ldots, \mathbf{S}_n\} \backslash \mathbf{S}_{\pi_t} \tag{14}$$

$$\Delta \mathbf{S}_i^{(0)} = \mathbf{S}_{\pi_t} - \hat{\mathbf{S}}_i^{(0)} \tag{15}$$

From this data, we learn the r_o, implemented by gradient tree boosting with the sum of square error loss.

By setting t=0, the triplet set having training data is again updated which is to provide input for the next regressor r_1 in the cascade.

$$\hat{\mathbf{S}}_i^{(t+1)} = \hat{\mathbf{S}}_i^{(t)} + r_t(I_{\pi_t}, \hat{\mathbf{S}}_i^{(t)}) \tag{16}$$

$$\Delta \mathbf{S}_i^{(t+1)} = \mathbf{S}_{\pi_t} - \hat{\mathbf{S}}_i^{(t+1)} \tag{17}$$

The process keeps on iterating until T cascade of regressors are learnt which when combined gives good accuracy.

Algorithm 1 Learning rt in the cascade

1. Initialize

$$f_0(I, \hat{\mathbf{S}}^{(t)}) = \arg\min_{\gamma \in \mathbb{R}^{2p}} \sum_{i=1}^{N} \|\Delta \mathbf{S}_i^{(t)} - \gamma\|^2$$

2. For k = 1, …, K
   a) Set for i = 1,…,N

$$r_{ik} = \Delta \mathbf{S}_i^{(t)} - f_{k-1}(I_{\pi_t}, \hat{\mathbf{S}}_i^{(t)})$$

   a) Fit the tree to the regression target giving back a weak regression function $g_k(I, \hat{\mathbf{S}}^{(t)})$.
   b) Update

$$f_k(I, \hat{\mathbf{S}}^{(t)}) = f_{k-1}(I, \hat{\mathbf{S}}^{(t)}) + \nu\, g_k(I, \hat{\mathbf{S}}^{(t)})$$

   c) Output

$$r_t(I, \hat{\mathbf{S}}^{(t)}) = f_K(I, \hat{\mathbf{S}}^{(t)})$$

3.1.2.3 Selecting splits

To train each regression tree, at every split node, there are two pixel coordinates whose difference is a feature and this feature will be compared against the threshold that is the difference between two pixels' intensities. In the test, the pixels at positions u and v in mean shapes coordinate system. The points in a face image of arbitrary shape, having the same position relative to mean shape's u and v. For achieving this, the image is warped to the mean shape that is based on current shape estimation before feature extraction.

Let k_u be the facial landmark index in mean shape which is closest to u and the offset from u as

$$\delta \mathbf{x}_\mathbf{u} = \mathbf{u} - \bar{\mathbf{x}}_{k_\mathbf{u}} \tag{18}$$

u' is the position in the image I that is relatively similar to **u** in mean shape with scale and rotation matrix is given as

$$\mathbf{u}' = \mathbf{x}_{i,k_\mathbf{u}} + \frac{1}{s_i} R_i^T \delta \mathbf{x}_\mathbf{u} \tag{19}$$

Sum of squares minimization is possible by scale and rotation:

$$\sum_{j=1}^{p} \|\bar{\mathbf{x}}_j - (s_i R_i \mathbf{x}_{i,j} + \mathbf{t}_i)\|^2 \tag{20}$$

v' is similarly defined as **u'**.

Hence, each split is a tuple of three parameters $\theta = (u,v,\tau)$ and

$$h(I_{\pi_i}, \hat{\mathbf{S}}_i^{(t)}, \boldsymbol{\theta}) = \begin{cases} 1 & I_{\pi_i}(\mathbf{u}') - I_{\pi_i}(\mathbf{v}') > \tau \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

A pool of candidate splits $\theta_{pool} = \{(u_1, v_1, \tau), (u_2, v_2, \tau), \ldots, (u_i, v_i, \tau)\}$ is selected to determine the split at node, the split θ^* that maximizes the variance reduction in the child node is selected. The difference in variance between a node and its children is selected. The leaf value stored is the mean of all training samples within the leaf, and hence the maximization of variance reduction is equivalent to sum of square errors minimization.

To reduce the computation time, the average of samples falling into right child is calculated from the average in left child and the overall average.

$$\mu_{\boldsymbol{\theta},r} = \frac{|\mathcal{Q}|\mu - |\mathcal{Q}_{\boldsymbol{\theta},l}|\mu_{\boldsymbol{\theta},l}}{\mathcal{Q}_{\boldsymbol{\theta},r}} \tag{22}$$

3.1.2.4 Exponential prior distribution of Feature selection

The feature pool θ_{pool} is randomly selected. However, as pixel pairs that are closer together have more discriminative features as compared to those which are away, the exponential prior is used over the pixels closeness used in a split to encourage closer pixel pairs to be chosen.

$$P(\mathbf{u}, \mathbf{v}) \propto e^{-\lambda \|\mathbf{u}-\mathbf{v}\|} \tag{23}$$

The framework presented is faster in error reduction of partial or uncertain labelled data.

3.2 Improved Method of Facial Landmark Detection with decision trees and Online Multiple Instance Learning

In the existing method if Facial Landmark Detection using regressor, slight inaccuracies like appearance changes and distance results in causing drift at real-time speeds. Hence, it is usually challenging to provide robust and stable facial localization tracking on adaptive appearance models. Hence, an Online MIL is applied on the existing method to model adaptively the appearance for object tracking, and the tracker requires no human input and bootstraps itself. The MIL framework based on boosting, updates the appearance model with a set of the image patches, even though it is not known that which image patch precisely captures object of interest. In this framework, training examples comes in "bags" which contains positive and negative instances that share a common label; a positive bag contains at least one positive instance and a negative bag contains no positive instances at all. As this online algorithm does not make any provision for feature selection, we implement this by wrapping weak learners inside Facial Landmark Features. So the proposed algorithm requires knowledge of whether the key facial structures are present in the visual field for training. The Online MIL using boosting –based approaches lends themselves more naturally to modification for online learning because of iterative nature of boosting algorithms. Here the multiple instance problem is handled by the ensemble algorithm unlike the previous methods of MIL. The key facial landmark features are extracted from the whole image; thus the object of interest is represented by a group of the instances embedded in a positive bag, which also includes interest points from the background.

Generally, the tracking system consists of 3 essential components: an image representation, the appearance model and the motion model. In this

framework, the training data has the form $\{(X_1,y_1), \ldots , (X_n,y_n)\}$ where a bag $X_i = \{x_{i1}, \ldots, x_{im}\}$ and y_i is a bag label. Bag Labels defined as:

$$y_i = \max_j (y_{ij}) \tag{24}$$

Y_{ij} are the instance labels which are assumed to exist but unknown during training. A bag is considered to be positive if it contains at least one positive instance. The MIL framework uses the gradient boosting framework t train a boosting classifier that maximizes the log likelihood of bags:

$$\log \mathcal{L} = \sum_i \Big(\log p(y_i|X_i) \Big) \tag{25}$$

The likelihood is defined over the bags and not instances since the instance labels are unknown during the training time. However, the goal is to train an instance classifier that does estimation of $p(y|x)$. And therefore the expression $p(y_i|X_i)$ need to be expressed the probability of its instances.

$$p(y_i|X_i) = 1 - \prod_j \Big(1 - p(y_i|x_{ij})\Big) \tag{26}$$

The equation above means that if at least one instance in a bag has a high probability, the bag probability will be high as well.

Boosting combines many weak classifiers $\mathbf{h}(x)$ into an additive strong classifier:

$$\mathbf{H}(x) = \sum_{k=1}^{K} \alpha_k \mathbf{h}_k(x) \tag{27}$$

Where α_k are the scalar weights. After each of the weak classifier is trained, the training examples are again re-weighted such that the examples that were previously misclassified receives more weight. It chooses one feature that has the most discriminative power than the whole weighted training set.

So, for each incoming example x, each h_i is sequentially updated and the weight of the example x is being adjusted after each update. The weak classifiers are sequentially chosen to optimize:

$$(\mathbf{h}_k, \alpha_k) = \underset{\mathbf{h}\in\mathcal{H},\alpha}{\operatorname{argmax}} J(\mathbf{H}_{k-1} + \alpha \mathbf{h}) \tag{28}$$

Where $\mathbf{H}_{k-1}$ is the strong classifier made up of first (k-1) weak classifiers, and h is the set of all possible weak classifiers.

Given image patch x and binary label y=1 which indicates presence of the tracked object, the instance probability p(y = 1 | X) is represented as:

$$p(y|x) = \sigma(\mathbf{H}(x)) = \frac{1}{1 + e^{-\mathbf{H}(x)}}. \tag{29}$$

To update the classifier, first is to update all of the weak classifiers in parallel. The bag label y_i is passed for all instances x_{ij} to the weak training procedure. Then K weak classifiers **h** are chosen from candidate pool sequentially using the criteria:

$$\mathbf{h}_k = \underset{h \in \{h_1, \ldots, h_M\}}{\operatorname{argmax}} \log \mathcal{L}(\mathbf{H}_{k-1} + h) \tag{30}$$

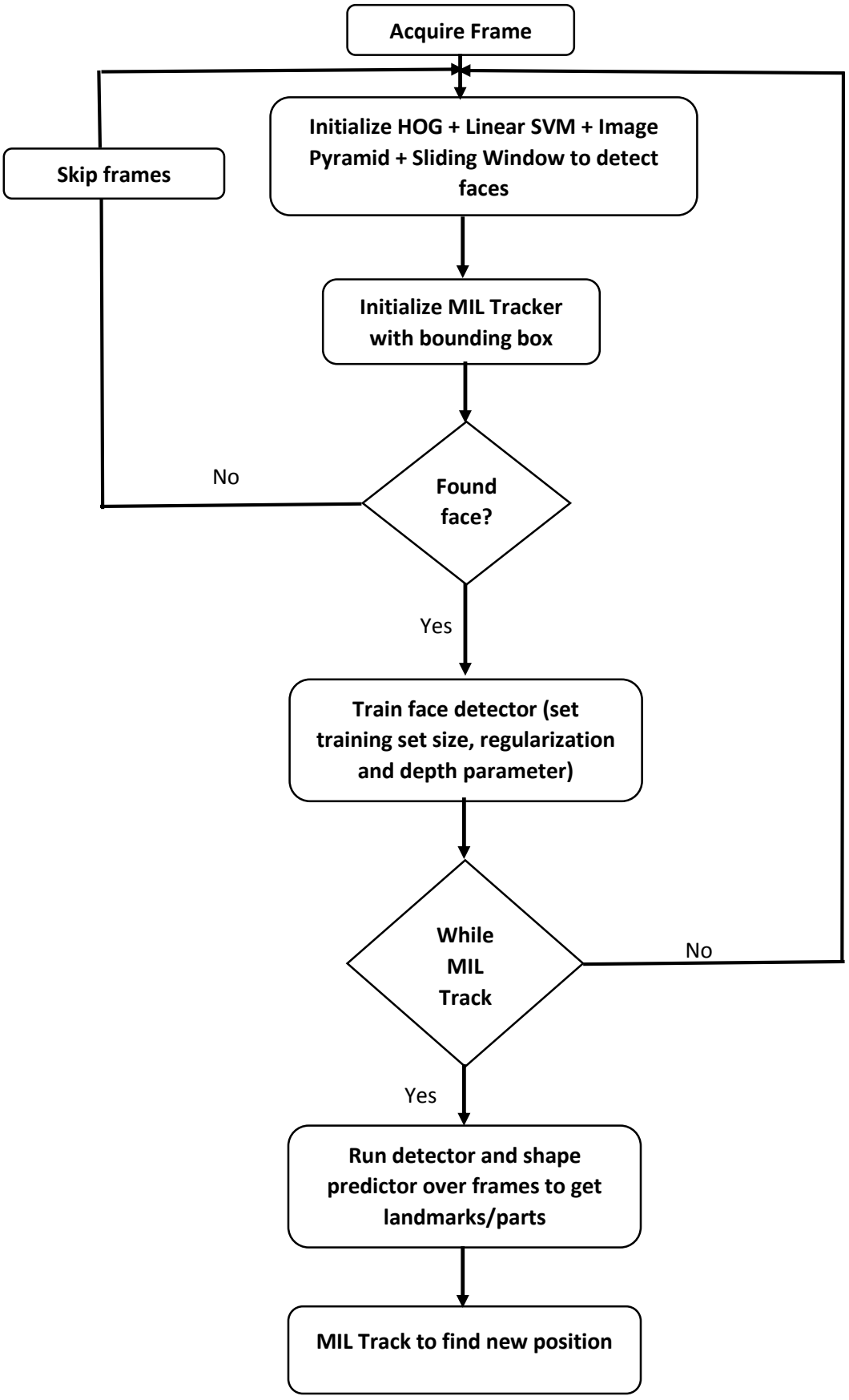
Algorithm 2: Online MIL

Input: Dataset $\{X_i, y_i\}_{i=1}^{N}$, where $X_i = \{x_{i1}, x_{i2}, \ldots\}, y_i \in \{0, 1\}$

Update all M weak classifiers in the pool with data $\{x_{ij}, y_i\}$
Initialize $H_{ij} = 0$ for all i, j
for $k = 1$ to K **do**
  for $m = 1$ to M **do**
    $p_{ij}^m = \sigma\left(H_{ij} + h_m(x_{ij})\right)$
    $p_i^m = 1 - \prod_j \left(1 - p_{ij}^m\right)$
    $\mathcal{L}^m = \sum_i \left(y_i \log(p_i^m) + (1 - y_i) \log(1 - p_i^m)\right)$
  end for
  $m^* = \operatorname{argmax}_m \mathcal{L}^m$
  $\mathbf{h}_k(x) \leftarrow h_{m^*}(x)$
  $H_{ij} = H_{ij} + \mathbf{h}_k(x)$
end for

Output: Classifier $\mathbf{H}(x) = \sum_k \mathbf{h}_k(x)$, where $p(y|x) = \sigma(\mathbf{H}(x))$

Fig. 3.1 illustrates the proposed algorithm on the HELEN and LFPW dataset.



APPLICATIONS

4.1 Facial Expression Predictions Evaluation

The project development compares the performance of the proposed method with the Support Vector Machine Classifier to train the machine learning algorithm. The results of proposed method outperforms with less computation.

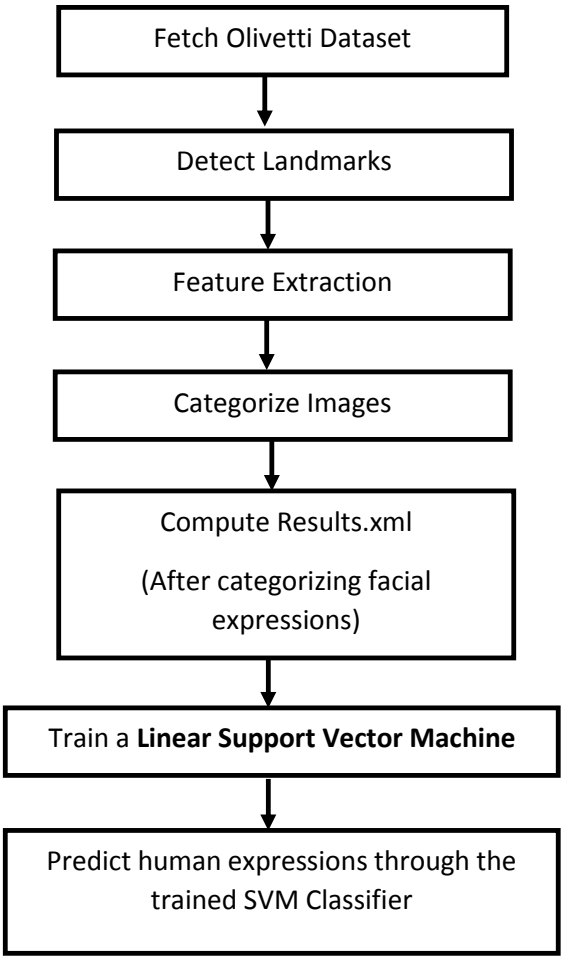


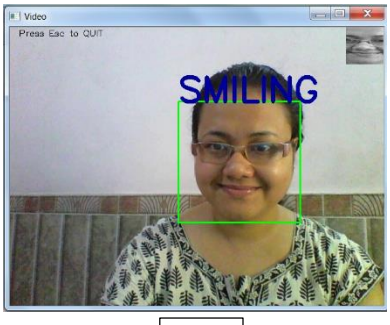
Fig. 4.1 illustrates the project development on Support Vector Machine Classifier

The Olivetti dataset contains set 400 face images of 40 different subjects. These images are quantized upto 256 gray-levels and are stored as unsigned 8 bit integers; which are converted to values of type floating point in the interval [0, 1] which is easier to work with algorithms.

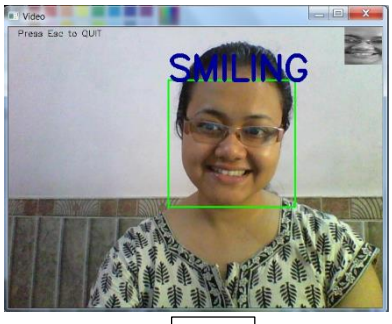
The feature landmark detections are transformed into features to feed the classifier. The facial landmarks extracted describe the position of all the "moving parts" of the depicted face, to express emotions. The features extracted are divided into categories. The face expressions in the dataset are classified in the results.xml file which contains the index of the particular image and the sort of expression for that image.

The linear Support Vector Machine Classifier is trained on this dataset which is used for further classifying the expression classes.

Fig. 4.2 (a), (b), (c) illustrates frames screenshots via a webcam feed on how the classifier after training is applied on the frame of video stream with two emotion categories.



(a)



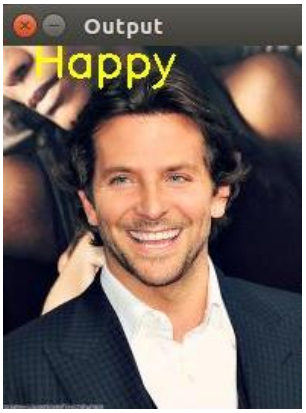
(b)

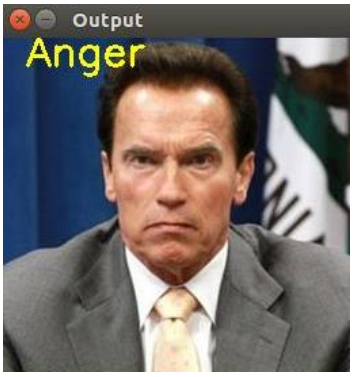
(c)

Running the same slice of data on **Decision Tree** and **Online MIL** with six emotion categories.

(a)

(b)





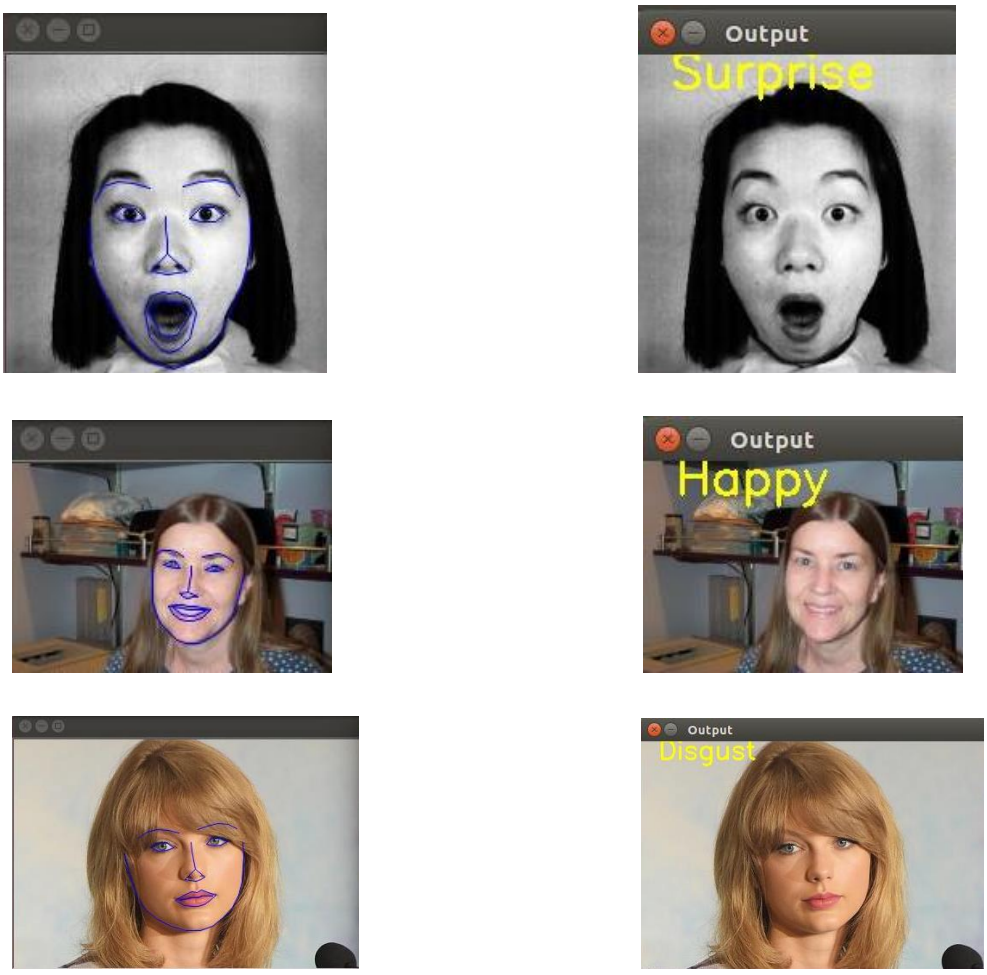


Fig. 4.3 (a) is the prediction using trained model, (b) Predicted Emotion using the trained data.

4.1.1 Third Party Libraries

Keeping the dependencies to minimum, I extensively used the real-time video and HELEN dataset, on which an ensemble of randomized regression trees is used for detecting 68 landmarks on the face from a single image in a millisecond. All the numerical computing for Python was done on Python Library (numpy).

```
 Please Wait . . . . .


Scores:  [ 0.8         0.8         0.71666667  0.68333333  0.77966102]
Mean score: 0.756 (+/-0.024)
Accuracy on training set:
1.0
Accuracy on testing set:
0.86
Classification Report:
             precision    recall  f1-score   support

          0       0.81      0.81      0.81        36
          1       0.89      0.89      0.89        64

avg / total       0.86      0.86      0.86       100

Confusion Matrix:
[[29  7]
 [ 7 57]]
```

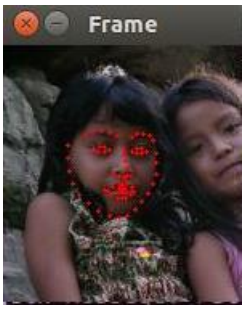
Fig. 4.4 shows the classifier computation with the scores

4.2 Real time Facial Landmark Detection Evaluation

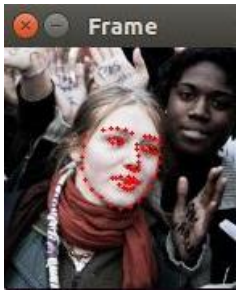
This demonstration expands the implementation of facial landmarks on Decision trees and the proposed method to work in real-time video streams, hence paving the way for more real-world applications. Here, the results demonstrate that the proposed method are fully capable of detecting facial landmarks in video steam in real-time using a system with modest CPU. The algorithm operates reliably and also accurately under a broad range of variations in appearance, including pose, expression, occlusion and lightning. Hence such a system can be applied for more computer vision applications like the blink detection. The method works with multiple faces provided that the face in the image/video stream is detected. The dataset used for implementation and evaluation purpose is the HELEN Dataset.

(a)

(b)



(c)

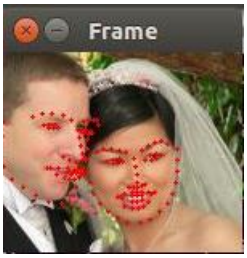


(d)

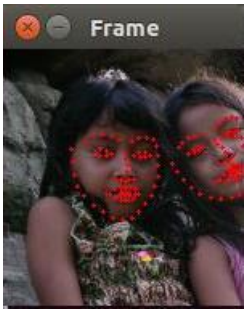
(e)

Fig. 4.5 (a), (b), (c), (d) and (e) resultant frames from HELEN dataset after applying Method 3.1.

(a)



(b)



(c)

(d)

(e)

Fig. 4.6 (a), (b), (c), (d) and (e) improved resultant frames from HELEN dataset after applying Method 3.2.

EXPERIMENTS and DATASET

The proposed system is tested on challenging datasets and video sequences. The goal of this work is to demonstrate that using Regression trees with Online MIL results in more stable and robust tracker. The framework is implemented for both single and multi-target tracking. For performance evaluation, quantitative and qualitative results the error rate and performance comparisons are presented on dataset. This implementation currently runs at 25 frames per second on Ubuntu VirtualBox virtual machine and on native development Ubuntu environment.

5.1 Parameters

The parameter settings that is used to conduct the experiment are mentioned. The number of the strong regressors r_t, in cascade is T = 10 and each of the rt consists of K = 500 weak regressors g_k. The depth of trees that is to be used to represent g_k is F = 5. At each of the level of cascade P = 400 pixel location are being sampled from image. To train weak regressors, we by random sample pair of the P pixel locations according to the prior and choosing a random threshold for creating a potential split. The suitable split is found by repeating the aforementioned process S = 20 times and hence choosing the one that best optimizes our objective.

5.2 Performance

The runtime complexity of the regression tree on a single image is O(TKF). The training time complexity depends on the training data as O(NDTKFS) where N is the training data and D is the target's dimension. On HELEN dataset, it takes about one millisecond per image.

5.3 Feature Selection

Equation (22) describes the parameter λ, which is the maximum effective distance between two pixels in the feature set and it is set to 0.1 in the experiments. Cross validating this parameter on our dataset significantly improves the process. Table 1 gives the comparison and effects of priors on feature selection for the final error.

Table 1.

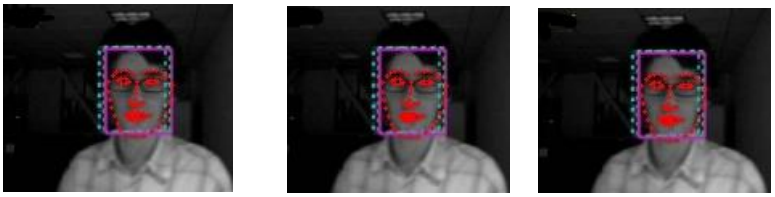
Parameter	Exponential	Uniform
Error	0.027	0.053

5.4 Video Sequences

The experiments are performed on own video sequence taken. All video frames were converted to gray scale and resized to 320x240 pixels. The quantitative results are summarized in Table 2 and Fig. 5.1.

Frame 0 Frame 28 Frame 40 Frame 56

Fig. 5.1 shows the Screenshots of final Results on the (A) real-time video highlighting in-plane rotation and object occlusion.



Frame 1 Frame 2 Frame 3

Fig. 5.1 (B) Screenshots of David Indoor which undergoes illumination change shows Landmarks with Online MIL Tracker

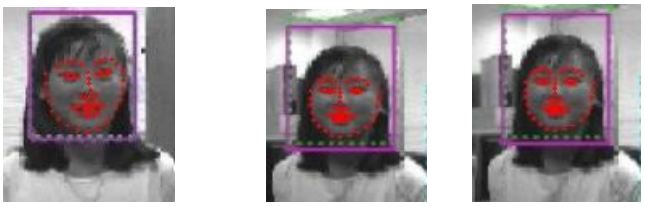


Fig. 5.1 (C) Screenshots of Girl which undergoes out-of-plane rotation shows Landmarks with Online MIL Tracker

Fig. 5.1 (D) Screenshots of result of some images showing 194 landmarks from HELEN Dataset.

The Online MIL applied ensemble of regression trees significantly improves the results over ensemble of regression trees only.

Table 2: Sequence Properties for evaluation

Video Clip	Frames	Size of Image	Moving Object	Scale and Illumination Change	Partial Occlusion
Our Video	400	320x240	yes	yes	yes
David Indoor	120	320x240	yes	yes	yes
Girl	210	150x150	yes	yes	no
HELEN	Image	150x150	Image	Image	yes

5.4 Comparison

Table 3: A summary of the results of different algorithms on the real-time video feed. The error is average normalized distance of each of the landmark to the ground truth position. The methods were initialized with the mean shape.

Table 3

	ERT	ERT and Online MIL
Error	.049	.032

The average error rate on number of images normalized by interocular distance is calcutated for our approach.

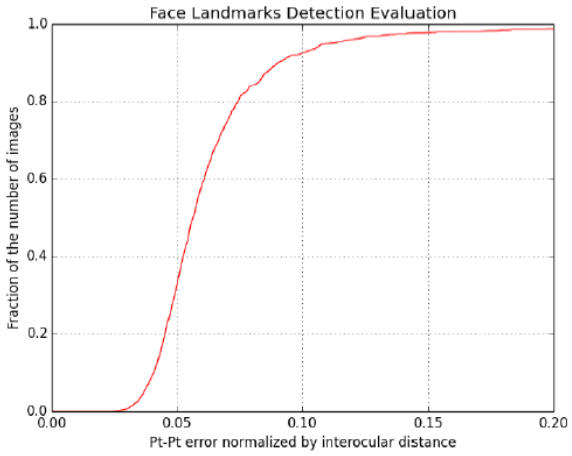


Fig 5.2 Evaluation Plot between Normalized Error with No of Frames

COMPARED ALGORITHMS

EF – In this approach, features are selected randomly on randomized ferns.

EF+CB – In this approach, feature selection is based on correlation that projects target's output on a random direction.

Table 4: Comparison Table on pixel location errors with radius = 4.

Video Clip	EF	EF+CB	ERT	ERT with Online MIL
Our Video	0.051	0.048	0.040	0.029
David Indoor	0.060	0.052	0.046	0.032
Girl	0.070	0.061	0.054	0.043
HELEN	0.043	0.039	0.029	0.015

CONCLUSION

In this work, the regression tree ensemble along with Online Multiple Instance Learning (or MIL) for faster and deterministic implementation with reduced error on certain, uncertain and partial labels of the shape parameters. The comparison between algorithms was laid out to summarize the fast performance of the proposed method under various conditions. Online MIL takes multiple image patches instead of one single positive image and hence it possess the flexibility of choosing the label instances. Unlike other algorithms, this method retains the useful information after feature extraction starting from the first frame itself. With this implementation, the algorithm attains real time speed.

This algorithm can be extended to work towards more robust performance for severe occlusions by implementing a part based model to handle at real time speeds. It can be based on the shape parameters correlation for effective training with partial labels.

BIBLIOGRAPHY

1. Nannan Wang et al. `Facial Feature Point Detection: A Comprehensive Survey'. In: CoRR abs/1410.1037 (2014). url: http://arxiv.org/abs/1410.1037.

2. Davis E. King. `Max-Margin Object Detection'. In: CoRR abs/1502.00046 (2015). url: http://arxiv.org/abs/1502.00046.

3. P. Doll_ar, P. Welinder and P. Perona. `Cascaded Pose Regression'. In: CVPR. 2010.

4. P. N. Belhumeur et al. `Localizing Parts of Faces Using a Consensus of Exemplars'. In: Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition. CVPR '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 545{552. isbn: 978-1-4577-0394-2. doi: 10.1109/CVPR.2011.5995602. url: http://dx.doi.org/10.1109/CVPR.2011.5995602.

5. Yoav Freund and Robert E Schapire. `A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting'. In: Journal of Computer and System Sciences 55.1 (1997), pp. 119 {139. issn: 0022-0000. doi: http://dx.doi.org/10.1006/jcss.1997.1504. url: http://www.sciencedirect.com/science/article/pii/S002200009791504X.

6. S. Avidan, "Ensemble Tracking," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no. 2, pp. 261-271, Feb. 2007.

7. J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. The Annals of Statistics, 28(2):337–407, 2000.

8. P. Viola, J. C. Platt, and C. Zhang. Multiple instance boosting for object detection. In NIPS, pages 1417–1426, 2005.

9. Timothy F. Cootes, Gareth J. Edwards and Christopher J. Taylor. `Active Appearance Models'. In: IEEE Trans. Pattern Anal. Mach. Intell. 23.6 (June 2001), pp. 681{685. X. Liu and T. Yu. Gradient feature selection for online boosting. In ICCV, pages 1–8, 2007.

10. S. Avidan. Ensemble tracking. In CVPR, volume 2, pages 494–501, 2005.

11. J. H. Friedman. Greedy function approximation: A gradient boosting machine. The Annals of Statistics, 29(5):1189–1232, 2001.

12. N. C. Oza. Online Ensemble Learning. Ph.D. Thesis, University of California, Berkeley, 2001.

13. X. Xiong and F. De la Torre, \Supervised descent methods and its applications to face alignment," in Proc. CVPR, 2013. 2, 17, 18, 20, 31, 36.

14. Yang Hua, Karteek Alahari, Cordelia Schmid."Occlusion and Motion Reasoning for Long-term Tracking", ECCV 2014 - European Conference on Computer Vision, Sep 2014, Zurich,Switzerland. Springer, 2014.

15. T. Drutarovsky and A. Fogelton, \Eye blink detection using variance of motion vectors," in Computer Vision - ECCV Workshops, 2014. 2, 7, 8, 20, 21, 22, 23, 24.

16. Vuong Le et al. `Interactive Facial Feature Localization'. In: Proceedings of the 12th European Conference on Computer Vision - Volume Part III. ECCV'12. Florence, Italy: Springer-Verlag, 2012, pp. 679{692. isbn: 978-3-642-33711-6. doi: 10.1007/978-3-642-33712-3_49. url: http://dx.doi.org/10.1007/978-3-642-33712-3_49.

17. Wikipedia. Computer Vision. url: http://en.wikipedia.org/wiki/Computer_Vision (visited on 2015).

18. Dong Chen et al. `Joint Cascade Face Detection and Alignment'. In: ECCV (6)'14. 2014, pp. 109{122}.

19. E. Antonakos et al. `Feature-Based Lucas-Kanade and Active Appearance Models'. In: IEEE Transactions on Image Processing 24.9 (2015), pp. 2617{2632.