Non-linear Predictors for Facial Feature Tracking Across Pose and Expression

Tim Sheerman-Chase, Eng-Jon Ong and Richard Bowden
CVSSP, University of Surrey
Guildford, Surrey GU2 7XH, United Kingdom
Email: t.sheerman-chase,e.ong,r.bowden@surrey.ac.uk

***Abstract*—This paper proposes a non-linear predictor for estimating the displacement of tracked feature points on faces that exhibit significant variations across pose and expression. Existing methods such as linear predictors, ASMs or AAMs are limited to a narrow range in pose. In order to track across a large pose range, separate pose-specific models are required that are then coupled via a pose-estimator. In our approach, we neither require a set of pose-specific models nor a pose-estimator. Using just a *single* tracking model, we are able to robustly and accurately track across a wide range of expression on poses. This is achieved by gradient boosting of regression trees for predicting the displacement vectors of tracked points. Additionally, we propose a novel algorithm for simultaneously configuring this hierarchical set of trackers for optimal tracking results. Experiments were carried out on sequences of naturalistic conversation and sequences with large pose and expression changes. The results show that the proposed method is superior to state of the art methods, in being able to robustly track a set of facial points whilst gracefully recovering from tracking failures.**

I. Introduction

In this paper, we propose a person-specific and *data-driven* approach to achieve accurate and real-time tracking of independent facial points across a wide range of pose and expression. This was achieved by learning different subsets of appearance and shape-based features that are then selected on-the-fly during the tracking process. Crucially, this allows our system to track facial feature points across a wide range of pose and expressions *without* the explicit need for pose estimation. This was achieved without any further constraints based on the dynamics of the tracked features.

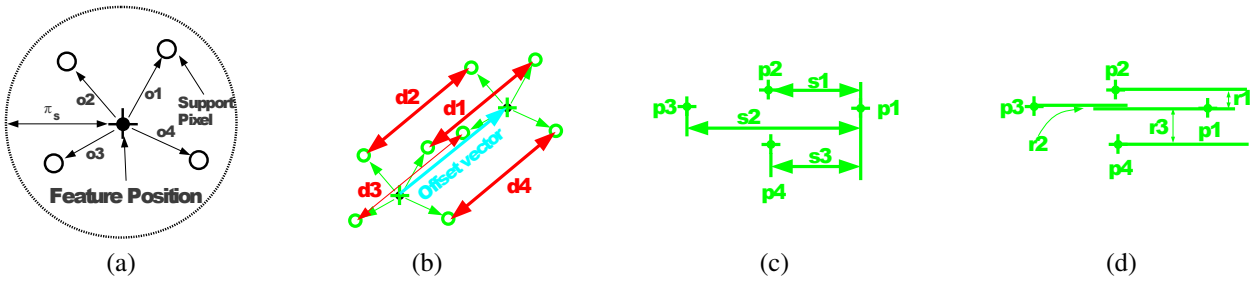
There exists a number of different methods for facial feature tracking. A widely used method for facial feature tracking is the Active Appearance Model (AAM), originally proposed by Cootes et al [1]. AAMs use an iteratively updated generated model, with both shape and appearance, to fit an observed face. Regression is used to determine the change in model parameters, based on differences between the target image and generative model [2]. This was later extended to various model fitting approaches, such as Boosting methods [3], [4], [5] and neural networks [6], [7]. However, one main disadvantage of the above AAM methods is the requirement for a large training database of different facial shapes and appearances. AAMs have previously been used for tracking lip shapes by Matthews et al[8]. A related method which uses only shape variation is Active Shape Model (ASM) [9], which have also been used for facial feature tracking. Pose invariance can be obtained by using a mixture pose-specific ASMs as proposed in [10]. However, a mechanism for switching selecting the relevant pose-specific AAMs or ASMs is then required. Pose invariance can also be achieved by coupling AAMs with an underlying 3D model, for example by Xiao et al [11], followed by Dornaika et al [12] and more recently by Sung et al [13].

Another recent method for tracking facial features are Linear Predictors (LP) [14]. A linear predictor provides a linear mapping from "support pixels" (i.e a sparse template) differences to the displacement vector of a tracked facial feature. Linear regression is only effective when pose changes are limited. As pose variations become significant, it is not possible to find the above subset of "linear" support pixels. This in turn results in catastrophic tracking failure when large pose variations is present in the target subject. Offset predictions have also been used for facial feature detection [15] by vote accumulation. Regression based detection was also performed on faces across a range of poses by Dantone et al. [16], using conditional regression trees.

The main contribution of this paper is a novel method that attempts to overcome these fundamental limitations by utilising a non-linear mapping based on gradient boosting (Section II). Here, we also propose a novel hierarchical configuration of gradient boosting to provide accurate *and* robust tracking. Furthermore, a novel algorithm was proposed in Section III for automatically learning the optimal configuration of gradient boosting regression trees. Crucially, the resulting learnt gradient boosting tracker is accurately track facial features across large changes in pose and expression *without* the need for explicit 3D models, pose-specific models or pose estimators. This is detailed in our experimental results in Section IV. Finally, we conclude in Section V.

II. Facial Feature Tracking using Non-linear Predictors

The tracking approach proposed in this paper attempts to tackle the problem of feature tracking as a regression problem. More specifically, tracking involves predicting the *displacement offset* vector (x, y) coordinates of a feature point from its correct location. In order to achieve this, we firstly



(a) (b) (c) (d)

Fig. 1: Illustrated here are the 3 support pixels in (a) along with the support offsets $(o1, o2, o3)$ within the support region (enclosing circle). (b) shows how support pixel differences that form the intensity-based features are obtained given an offseted position.(c) and (d) shows the different components for the shape-vector.

make available two types of features: intensity and shape-based features (Section II-A). Following this, Section II-B details how gradient boosting regression trees (GBRT) are used for predicting the offset vector for a tracked feature point. We then increase both the tracking accuracy and robustness by combining regressors in a hierarchical manner as described in Section II-C.

Firstly, a number of preliminary definitions are provided. In this paper, the input image is defined as I with height H and width W. The intensity of the pixels in the image at location (x, y) is defined as $I(x, y)$ where $I : \mathbb{R}^2 \rightarrow [0, 255]$. We define the number of points being tracked as N_F and for the rest of the paper, $f \in [1, N_F]$ will be used to denote a single feature index.

A. Input Features

The proposed tracker will utilise two types of features for tracking feature points: intensity-based and shape-based. These will then be fused into a high-dimensional vector that will then undergo feature selection in Section II-B before being used for feature offset prediction. We will start by defining the intensity-based features.

The intensity-based features are similar to support pixel features that are employed by linear predictor trackers[14]. Each tracked feature point is associated with a set of support pixels representing a sparse template of the feature appearance. Formally, each support pixel (s) is defined as the pair $s = (p, o)$ where $p \in [0, 255]$ represents the template pixel intensity and $o \in \mathbb{R}^2$ is the displacement vector of the support pixel from its associated feature. The offset vector o is randomly sampled from a *support region* which is an enclosing circle of radius σ_S (i.e. $||o|| \leq \sigma_S$). The set of support pixels for a f^{th} feature is defined as $\mathbf{S}^f = \{s_i^f\}_i^{D_S}$, where $s_i^f = (p_i^f, o_i^f)$ is the i^{th} support pixel defined above and D_S is the number of support pixels. An illustration of the support pixels and its offsets can be seen in Fig. 1(a).

For the purpose of tracking, the support pixel intensities are used to obtain sparse template differences that are then used by gradient boosting regression trees as input features for offset vector predictions. To achieve this, we first need to obtain corresponding support pixel intensities at a particular pixel location. Specifically, let $\mathbf{p} = (p_x, p_y)$ be an initial location for the f^{th} tracked point in input image I. We next define the function δP for extracting a vector of differences between current support pixel intensities at $\mathbf{p}$ to the template support pixel intensities for the tracked point f:

$$\delta P(\mathbf{p}, I; \mathbf{S}^f) = (I(x' - o_{i,x}^f, y' - o_{i,y}^f) - p_i^f)_{i=1}^{D_S} \quad (1)$$

The function δP returns a D_S-dimensional vector which we denote as the *intensity-based feature vector* (Fig. 1b).

Features that indirectly contain shape information are also provided. Features encoding shape information were included due to their potential for giving information that will increase the robustness of a point tracker (e.g. detecting tracking failure of a tracked point). In this paper, we use a simple feature based on the differences in positions between two tracked features. This allows the regression tree tracker in Section II-B to select only specific pairs of tracked points that are advantageous to giving more robustness whilst explicitly ignoring others. This is in contrast to popular shape information (e.g. PDMs[9] and AAMs[1]) that is forced to utilise the locations of all tracked points. Each tracked point is associated with $N_F - 1$ features. In order to define the shape-based features for the f^{th} tracked point, firstly let $\mathbf{P} = \{(p_x^f, p_y^f)\}_{f=1}^{N_F}$ be the positions of all the tracked points. We then define the function δS for generating the shape-based feature vector for tracked point f as follows:

$$\delta S(\mathbf{P}, f) = (p_x^j - p_x^f, p_y^j - p_y^f)_{j \neq f} \quad (2)$$

The function δS returns a $2(N_F - 1)$-dimensional vector containing the differences between the position of the f^{th} tracked point and other tracked points respectively. This vector will be denoted as the *shape-based feature vector* (Fig. 1c,d).

In order to generate the input feature vector for gradient boosting regression trees, we fuse all both the above intensity-based (Eq. 1) and shape-based features (Eq. 2) into a single high dimensional vector using the following function F:

$$F(\mathbf{P}, I; \mathbf{S}^f, f) = (\delta P(I; \mathbf{S}^f), \delta S(\mathbf{P}, f)) \quad (3)$$

where $\mathbf{P}$ is the set of tracked point locations, I the image, $\mathbf{S}^f$ then support pixel offsets for feature f. Let $A = F(\mathbf{P}, I; \mathbf{S}^f, f)$, we denote the vector A as an *input feature vector* with its dimensionality denoted as $D_A = D_S + 2(N_F - 1)$. In order to simplify the equations in the next few sections, we

note that the support pixel offsets for each feature ($\mathbf{S}^f$) once generated, are fixed. As a result, when generating the input feature vector for feature f, we will instead denote Eq. 3 as:

$$F^f(\mathbf{P}, I) = F(\mathbf{P}, I; \mathbf{S}^f, f) \tag{4}$$

B. Gradient Boosting Regression Trees Feature Offset Predictors

In this section, we will describe how a gradient boosting regression trees are used to predict the displacement offset vectors of a single tracked feature point using the feature vector defined in Eq. 3. The regressor consists of a collection of regression trees, where each tree provides a single (but potentially unreliable) prediction for a displacement vector coordinate. The set of individual tree predictions will then be combined together to yield a more accurate and reliable displacement prediction. For our purpose, each tree will predict only a *single* coordinate for the displacement vector (e.g. x or y coordinate only). As a result, two sets of regression trees are required to predict the 2D displacement vector for a tracked point: one for the *x*-coordinate and one for the *y*-coordinate.

A regression tree is similar to decision trees, with the difference of predicting real valued outputs as opposed to class labels. Each regression tree is a binary tree of nodes that consists of a set of internal decision-making nodes and a set of leaf nodes providing output values. We define the regression tree for the track point f as T^f. Associated with T^f is the set of terminal leaf-node values $L = \{l_i : l_i \in \mathbb{R}\}_{i=1}^{|T^f|}$, where $|T^f|$ is the number of leaf nodes in T^f. In order to define the internal nodes, suppose that we are given the feature vector A^f (Eq. 3). Then, an internal node B in T^f is a binary node that is then associated with a feature dimension i and decision function of the form:

$$B(A^f) = \begin{cases} B^+ & \text{if } \{A_i^f \leq c\} \\ B^- & \text{if } \{A_i^f > c\} \end{cases} \tag{5}$$

where $c \in \mathbb{R}$ is the feature threshold obtained during the tree learning process described in Section III, B^+ and B^- are the children tree nodes of B and can either be another internal node or a leaf node. Each leaf node is assigned a particular prediction real value which is also obtained during the tree learning process. Thus, it is possible to assign a predicted value $\mu \in \mathbb{R}$ from the tree T^f given the feature vector A^f by recursively applying Eq. 5, starting from the root node of the tree and ending with a leaf node (whose associated value is μ). To make the above statement more concise, we will use $g(A, T)$ to denote the function that assigns a leaf-node value μ to the feature vector A given a regression tree T.

However, a single regression tree is often inadequate for providing good predictions. This is due to quantisation errors that exists due to the prediction mechanism used by regression trees. To address this issue, a set of different regression trees can be combined together in a weighted manner into gradient boosted regression trees set to yield a smoother and more accurate prediction. Specifically, suppose have a set of N_T number of weighted regression trees, $\mathbf{F} = \{(\gamma_i, T_i)\}_{i=1}^{N_T}$, and input feature vector A, a combined prediction can be obtained as:

$$G(A; \mathbf{T}) = \sum_{t=1}^{N_T} \gamma_t g(A, T_t^f) \tag{6}$$

We will now describe how to obtain the predicted positions for all the tracked points using Eq. 6 given an input image I, a set of initial locations $\mathbf{P} = \{(p_x^f, p_y^f)_{i=1}^{N_F}\}$ and the set of regressor pairs (one regressor for each offset axis), ($\mathbf{F} = \{(\mathbf{T}^{f,x}, \mathbf{T}^{f,y})\}_{f=1}^{N_F}$ via the following function:

$$K(\mathbf{P}, I; \mathbf{F}) = (G(F^f(\mathbf{P}, I); \mathbf{T}^{f,x})), G(F^f(\mathbf{P}, I); \mathbf{T}^{f,y}))) \tag{7}$$

In order to acquire a more stable set of tracking, we typically recursively iterate function K a fixed number of times (r), denoted as K^r:

$$K^1(\mathbf{P}, I; \mathbf{F}) = K(\mathbf{P}, I; \mathbf{F}) \tag{8}$$

$$K^i(\mathbf{P}, I; \mathbf{F}) = K^{i-1}(K(\mathbf{P}, I; \mathbf{F}) \tag{9}$$

C. Hierarchical Regression Tracker

Previous work on visual tracking of points have found that there is a trade-off between the ability for a tracker to be robust and accurate [14]. In particular, a robust tracker can often tolerate large amounts of movements in a point, but lack the ability to provide accurate predictions. Conversely, an accurate tracker can often locate to a good degree of accuracy the location of a track point, but is unable to cope with large displacements in the point between subsequent frames. To improve on this, we propose to use a *hierarchical regression tracker* that sequentially applies a series of regression trackers to yield a more robust and accurate prediction. Learning the individual regression trackers in the hierarchy and determining the series of trackers to string together is done automatically and described in Sections III-B and III-C respectively.

In order to define a hierarchical regression tracker, suppose we have a series of N_H number of regression tracker pairs: $\mathbf{H} = (\mathbf{F}_i)_{i=1}^{N_H})$, where $\mathbf{F}_i$ is a pair of regressors described in the previous section. Given an input feature vector $A \in \mathbb{R}^F$, a hierarchical prediction can be obtained by a recursive application of the iterated composition of Eq. 6, which we denote as:

$$h(\mathbf{P}, I; \mathbf{H}) = H(\mathbf{P}, N_H + 1; \mathbf{H}) \tag{10}$$

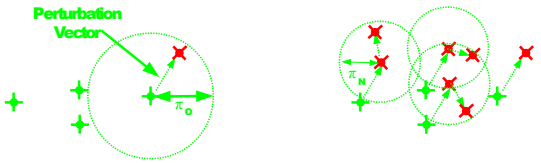
where the recursive function H is defined as:

$$H(\mathbf{P}, 1; \mathbf{H}) = K^r(\mathbf{P}, I; \mathbf{F}_{N_H}) \tag{11}$$

$$H(\mathbf{P}, i; \mathbf{H}) = H(K^r(\mathbf{P}, I; \mathbf{F}_{N_H - i}), i - 1; \mathbf{H}) \tag{12}$$

D. Computational Complexity of Hierarchical Regression Tracker

The proposed hierarchical tracker is highly efficient, as it is based on regression trees. Specifically, in order to perform a prediction using a hierarchical gradient boosting regression tree tracker of L levels, with each regressor containing a set of N_T trees of depth D, only LN_TD number of if-statement comparisons need to be performed along with an



(a) (b)

Fig. 2: (a) shows the standard deviation of the region (enclosing circle) used to generate random offset vectors. (b) Using the generated offset vector, all other tracked points are similarly displaced, before being further displaced by the shape noise Gaussians (small enclosing circles around tracked points).

average of N_T predictions. As an example, for our experiments $N_T = 100$, $D = 5$, $L = 2$, results in only 2000 if-statement evaluations required for a prediction. As a result, we have found that using this method, the speed of the tracking system is limited to how fast image frames can be extracted from video sequences.

III. Learning the Hierarchical Tracker

In this section, we will describe the method used for learning the non-linear predictor regressor described in Section II. To this end, we will firstly describe how the training dataset can be synthesised using a small number of landmarked example images (Section III-A). Using the synthesised training data, the Gradient-Boosting method is used to learn a set of regression trees (Section III-B).

A. Synthesis of Training Data

The training dataset for learning the non-linear predictor regressors takes the form of a set of N_T number of feature vector-label pairs: $\{(A_i, y_i)\}_{i=1}^{N_T}$, where $y_i \in \mathbb{R}^2$ is the displacement vector to predict using the input feature vector $A_i \in \mathbb{R}^F$. In order to generate this training dataset, suppose we are originally given N_I number of training images with the location of the tracked points landmarked on each image. A number (N_R) of in-plane rotations are applied to each training image and its landmark positions. This is to account for unseen in-plane rotational variations that may occur during the tracking process. Specifically, each image will be artificially rotated in-plane N_R number of times with the rotation parameter sampled from the Gaussian distribution ($\mathbb{N}(0, \sigma_R)$), where the variance of the rotation distribution (σ_R) is determined automatically in Section III-C. This results in a new set of $N_I' = N_I N_R$ number of landmarked images. We shall denote the set of landmarked points of tracked point f in image i as: $\{(l_{i,x}^f, l_{i,y}^f)\}$, where $i \in [1, N_I']$.

A training pair is generated for tracked point f in training image i as follows: We initially generate an offset vector ($\mathbf{o}^f = (o_x^f, o_y^f)$) from its landmark position yielding the displaced location of $\mathbf{l'}^f = (l_{i,x}^f + o_x^f, l_{i,y}^f + o_y^f)$. The displacement vector $\mathbf{o}^f$ is sampled from the 2D radial Gaussian distribution $\mathbb{N}(0, \sigma_O)$, where the variance of the distribution σ_O is again determined automatically in Section III-C. The setting of σ_O has important implications on what "type" of tracker will be learnt. A large value of σ_O will produce a tracker that capable of dealing with large perturbations in a tracked point, but at the cost of accuracy loss, whilst a small σ_O results in the converse.

In order to generate the shape vector, a new set of perturbed landmark points $\mathbf{P} = \{(p_x^j, p_y^j\}_{j=1}^{N_F}$ for all features is generated:

$$p_*^j = \begin{cases} o_*^j + l_{i,*}^j & \text{if } j = f \\ o_*^j + l_{i,*}^j + \epsilon & \text{if } j \neq f \end{cases} \tag{13}$$

where $* \in \{x, y\}$ is a placeholder denoting the coordinate components, since the equation is the same for both x and y coordinates. Additionally, ϵ is the shape-noise factor that is sampled from the Gaussian distribution $\mathbb{N}(0, \sigma_N)$. The addition of gaussian noise is necessary to prevent over-reliance on shape constraints and to force the regressor to consider pixel intensity information. Using Eq. 4, a feature vector A^f is obtained at location $\mathbf{l'}^f$ in image i given the above perturbed shape configuration: $A^f = F^f(\mathbf{P}, I_i)$, where I_i denotes the i^{th} image. The training example generated for this tracked point is the pair: $(A^f, \mathbf{o}^f)$. This is repeated for each training image N_A number of times, yielding a dataset of $N_T = N_A N_I$ number of training examples for each tracked point.

B. Learning the Tracker: Gradient Boosting Regression Trees

Given a training dataset described in the previous section, for a feature f, we use the gradient boosting regression trees method [17] for learning two separate regressors, one for predicting the x and another for the y component of the displacement vector from an input feature vector. To this end, suppose that the training set of N_T examples given is: $\mathbf{A}^f = \{(A_i^f, \mathbf{o}_i^f)\}_{i=1}^{N_T}$. Using $\mathbf{A}^f$, we form two new datasets, with the x and y components of the target offset vector isolated: $\mathbf{A}^{f,x} = \{A_i^f, o_{x,i}^f\}$ and $\mathbf{A}^{f,y} = \{A_i^f, o_{y,i}^f\}$, where $o_{x,i}^f$ and $o_{y,i}^f$ are the x and y components of the offset vector $\mathbf{o}_i^f$ respectively.

The aim of gradient boosting regression trees is to learn a set of regression trees (Eq. 6) that minimises the squared error between the predicted and groundtruth target values, given one of the above target datasets. This is achieved by sequentially training regression trees for predicting the residuals between the prediction from the previous set of regression trees and its respective groundtruth values and is detailed in Algo. 1. The training of a single regression tree given a dataset uses the standard approach of building the tree top-down, where the dimension index and threshold value of each non-leaf node is chosen such that the squared error between the tree prediction and target values are minimised.

C. Automated Training Parameters Selection

One issue that remains is the need to train a number of different trackers and sequentially order them to form a hier-

Algorithm 1 Gradient Boosting of regression $\mathbf{F}$ with M trees using dataset $\mathbf{X} = \{A_i, o_i\}_{i=1}^{N_T}$

Fit initial tree $\mathbf{T}_1$ minimising: $\sum_{(A,o)\in\mathbf{X}}(o - g(A, T_1))^2$
$\mathbf{F} = \{(1, \mathbf{T}_1)\}$
for $m \in [2, M]$ **do**
  Obtain "residual-dataset": $\mathbf{X}' = \{A_i, o_i - G(A, \mathbf{F})\}$
  Fit tree $\mathbf{T}_m$ minimising: $\sum_{(A,r)\in\mathbf{X}'}(r - g(A, T_m))^2$
  Compute γ_m using line-search optimising:

$$\gamma_m = \arg\min_{\gamma} \sum_{(A,o)\in\mathbf{X}} (o - G(A, \mathbf{F} \cup (\gamma, \mathbf{T}_m)))^2$$

  Update model: $\mathbf{F} = \mathbf{F} \cup (\gamma_m, \mathbf{T}_m)$
end for
return Gradient boosted regression tree set $\mathbf{F} = \{(\gamma_i, \mathbf{T}_i)\}_{i=1}^{M}$.

archical tracker that can cope with large perturbations whilst still being accurate. Crucially, the properties of the individual regression based trackers are dependent on the training set parameters (e.g. a training set with large perturbations will result in a robust but inaccurate tracker). To achieve this, we propose a genetic-algorithm inspired method that can automatically determine the optimal parameters for generating training data sets that will be used to learn a hierarchical tracker that is both robust and accurate.

Firstly, we require a definition of a *gene* to represent a configuration of the hierarchical predictor. To this end, we start by noting that for each regression based tracker, we aim to optimise the following parameters: support region size (σ_S); landmark offset variance (σ_O); shape noise variance (σ_N) and the shape flag ($\delta_S \in \{0, 1\}$). The shape flag will determine if the shape features in the input feature vector will be used. Thus each regression based tracker is represented by a 4-dimensional vector, which we define as the *tracker gene*: $\mathbf{g} = (\sigma_S, \sigma_O, \sigma_N, \delta_S)$. In order to represent a hierarchical predictor of N_H regression based trackers, a $4N_H$-dimensional gene vector is made, which we denote as *hierarchical gene*. This vector is the result of concatenating N_H separate gene vectors, one for each regression based predictor in the hierarchy. The aim of the proposed algorithm is to find the configuration of a hierarchical gene vector that produces an optimal hierarchical tracker.

A hierarchical gene $\mathbf{g}$ can be translated into a hierarchical predictor by firstly generating a set of support pixel offsets, followed by the appropriate training dataset for each tracker in the hierarchy. The algorithm for carrying this out is as given in Algorithm 2.

Next, we require a method for evaluating the performance of the gene $\mathbf{g}$. Using Algorithm 2, a hierarchical predictor consisting of a set of regressor trackers is learnt: $\{\mathbf{F}_i\}_{i=1}^{N_H}$. The training video where the landmarked training images originated from is then used. With the hierarchical predictor, the F points of interest are tracked at every frame, where the initial locations for points at frame t set to the tracking results from

Algorithm 2 Learn Regressor $\mathbf{F}$ from hierarchical gene $\mathbf{g}$

for $i \in [1, N_H]$ **do**
  Get current gene: $(\sigma_S, \sigma_O, \sigma_N, \delta_S) = \mathbf{g}[4(i-1) + 1\ 4i]$
  Generate support pixel within support region of σ_S for each feature.
  Synthesise training set $\mathbf{X} = (A^f, \mathbf{o}^f)$ with parameters σ_O and σ_N as described in Section III-A.
  Learn regressor pair $(\mathbf{F}_i^{f,x}, \mathbf{F}_i^{f,y})$ using $\mathbf{X}$ as described in Section III-B.
end for
return Set of learnt regressor pairs: $\{(\mathbf{F}_i^{f,x}, \mathbf{F}_i^{f,y})_{i=1}^{N_H}\}$

frame $t-1$. When a landmarked frame is reached, the average Euclidean distance between the landmark positions ($\{(l_x^f, l_y^f)\}$) and corresponding tracked positions ($\{(p_x^f, p_y^f)\}$) is obtained: $\epsilon = \sum_{i=1}^{N_F} \sqrt{(p_x^f - l_x^f)^2 + (p_y^f - l_y^f)^2}$. This distance ϵ is then applied into the following equation to penalise tracked points that have catastrophic failures:

$$C(\epsilon) = 1 - e^{-s\epsilon} \tag{14}$$

where the scaling factor $s \in \mathbb{R}$ is proportional to the maximum allowable pixel error before we conclude that a catastrophic failure has occurred. The position of the tracker is the re-initialised to the landmark positions of the respective frame. This is then repeated until all the landmark frames have been reached and the overall average error calculated. This error will serve as an inverse fitness of the gene (i.e. we aim to find a gene that minimises the overall average error).

One important element in our proposed algorithm is the ability to mutate genes. The mutation mechanism will allow a larger exploration of the parameter space modelled by the genes to happen. As a result, configurations that are more optimal can be discovered. In order to achieve this, we firstly introduce the function m for mutating a tracker gene, $\mathbf{g}$: $m(\mathbf{g}) = (g_1 + \epsilon_S, g_2 + \epsilon_O, g_3 + \epsilon_N, \neg^{0.5} g_4)$. The function m essentially adds a small amount of noise to the different components of the tracker gene. Here, $\neg^{0.5}$ denotes the operation of randomly flipping the shape vector flag. The noise factors ϵ_1, ϵ_2, ϵ_3 are all derived from uniform distributions($\mathbb{U}$): $\epsilon_i = \mathbb{U}(\alpha_i, \beta_i), i \in \{1, 2, 3\}$. The parameters for these uniform distributions (i.e. $(\alpha_i, beta_i)$) are determined heuristically and will be detailed in Section IV. When given a hierarchical gene $\mathbf{g}'$ of $4N_H$ dimensions, we iteratively apply m to each sections of $\mathbf{g}'$, and denote this with the function M:

$$M(\mathbf{g}') = (m(g(1:4)), ..., m(g(4(N_H - 1) : 4N_H)) \tag{15}$$

Using the above we propose a novel algorithm for obtaining the optimal hierarchical predictor in Algorithm 3.

IV. Experiments

In this section, we will detail the experiments carried out to test the ability of our proposed method to cope with simultaneous and large change in pose and expression. To this end, two sets of experiments were carried out: tracking

Algorithm 3 Genetic Algorithm for Optimising Hierarchical Regression based Predictor

Randomly initialise population of N_G number of hierarchical genes: $\mathbf{G} = (\mathbf{g}_i)_{i=1}^{N_G}$.
$\epsilon^P = 0$
while 1 **do**
  Evaluate errors for each gene: $(C(\epsilon_i))_{i=1}^{N_G}$
  Extract gene with the smallest error: $\mathbf{g}_j$ where $j = \arg\min_{i \in [1,N_G]}(C(\epsilon_i))_{i=1}^{N_G}$
  $\mathbf{G}^{new} = []$
  for $i \in [1, N_G]$ **do**
    Random mutation (Eq. 15) on $\mathbf{g}^{new} = M(\mathbf{g}_j)$
    $\mathbf{G}^{new}[i] = \mathbf{g}^{new}$
  end for
  if $|\epsilon^P - \min((C(\epsilon_i))_{i=1}^{N_G})| < \theta$ **then**
    break
  end if
  $\epsilon^P \leftarrow \min((C(\epsilon_i))_{i=1}^{N_G})$
end while
return Hierarchical predictor from gene $\mathbf{g}_j$(Algo. 2).

on long sequences of naturalistic conversation(Section IV-A); and sequences containing continuous and large changes in pose and expressions (Section IV-B). We also provide a comparison against the tracking performance of the linear predictor trackers from [14]. For all the experiments described in this section, we have chosen to track a set of 48 facial feature points as shown in Figure 3 and 4. These points were chosen as they are commonly used in facial feature tracking work.

A. Tracking in Naturalistic Conversation

We now describe a set of experiments for evaluating the tracking performance of the proposed method over a long sequences involving a set of subjects involved in unconstrained conversation. To this end, we have chosen to use the TwoTalk dataset [18], whereby subjects exhibited considerable expression changes along with pose variations. The TwoTalk dataset consists of four pairs of participants recorded in informal conversation for 12 minutes. The only instructions given to the participants were to remain seated and to communicate until asked to stop. The conversation was recorded by two progressive scan digital video cameras with a frame resolution of 720 by 576 pixels and 25 Hz frame rate. The dataset contains a range of spontaneous emotions, lip movements, head pose changes and occasional hand gestures that occasionally occlude the face. The colour images are converted to greyscale using the ITU-R 601-2 luma transform. The result is a test dataset of 8 video sequences, each containing 18000 frames. For each video sequence, the first half was retained for training purposes whilst the later half was used for testing. In total, there were 72000 test frames.

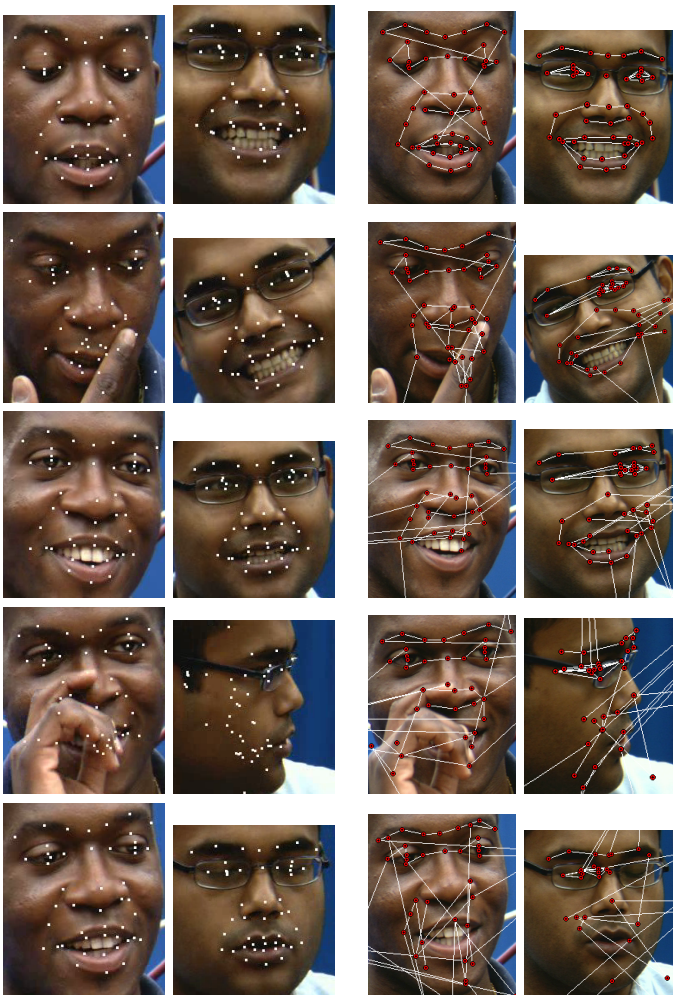
For the experiments in this dataset, 19 images were extracted from the training clip for each subject. 48 facial points were then manually annotated on the images. Subsequently, the proposed method described in Section II was used to train 48 separate trackers, one for each facial point of interest. A 2 level hierarchy was found to be sufficient for good tracking performance. Additionally, the error measure scale (s) used in Eq. 14 was set to 0.2 to allow a reasonable tradeoff between penalising the tracker for accuracy vs robustness in the learning process. Additionally, each regression regressor contained 100 trees, each with the depth of 5. In order to learn the hierarchical configuration of trackers, a population of 8 genes was sufficient to allow for an optimal configuration to be found.

In order to evaluate the robustness and accuracy performance of the tracker, we firstly note that it will be prohibitively time consuming to manually annotate 72000 frames with 48 facial feature points. As a result, we have instead chosen to annotate a subset of frames randomly distributed over the test sequence. The entire sequence is then tracked and at the annotated frames, the tracking pixel error (Euclidean distance) is measured and used for reinitialisation of the tracker positions. For this paper, a total of 60 test frames were landmarked over the test sequence of each subject. A catastrophic failure of tracking will result in a large error for the respective point when evaluate at the next annotated frame. We quantify the amount of catastrophic failures by measuring the percentage of points with tracking errors greater than 10 pixels.

The percentage of points over the test sequences within 10 pixels of the landmarked positions was found to be 96.3% compared to 89.2% using the LP method. This shows that the proposed method is significantly more robust than the linear predictor method. The pixel errors for the proposed method had a median of 2.88 compared to 3.19 of the LPs. This shows the proposed method is more accurate at the tracking. We find that the majority of tracking failures in the LP method occurs when there is a large change in pose or expression. In comparison, the our proposed method is able to overcome many of these limitations and continue to maintain track across these challenging conditions. We also find that the accuracy of the points when tracking failure did not occur is comparable with the state of the art LPs. The tracking results are shown in in Fig. 3 where the positions of the tracked points using our method and the LP method are shown at various frames in the test sequences. Additional results can be seen in the supplementary material videos.

B. Tracking across Significant Pose Changes

In addition to the TwoTalk dataset, we have carried out experiments that were specifically tailored to test how the proposed method copes with simultaneous occurrence of significant pose and expression changes. To this end, two sequences (train and test) were captured on 4 subjects using a hand-held commercial camcorder with a frame rate of 30 fps at a resolution of 920x540. The task of the subject was to show various facial expressions whilst changing the orientation of their head. Additionally, the lack of a tripod introduces further complexity of random translations due to an unsteady



(a) Proposed Method (b) Linear Predictor Method

Fig. 3: Tracking results from the TwoTalk dataset for the proposed method compared against the linear prediction method. Each column of (a) and (b) is the results for a subject. The top represents an earlier part of the sequence. Shown is how the proposed trackers recovered after occlusions and pose changes whilst the LP method results in catastrophic tracking failure.

hand holding the camera. Each sequence was approximately 15 seconds in length. The resulting dataset consisted of 4 test sequences with approximately 8000 number of frames in total.

In order to train the proposed hierarchical regressor based trackers, 13 example frames from the training sequence of each subject was extracted and manually annotated. These images were selected to cover the pose and expression variations present in the training sequence. As before, using these annotated training frames, 48 separate hierarchical trackers were learnt using the method proposed in Section II. For each tracker, a 2-level hierarchical tracker was sufficient to give good tracking results. Each tracker in the hierarchy consisted of 100 regression trees that are 5 levels deep. Again, the resulting trackers are highly efficient, and the computational runtime limited to how fast the system is able to extract image frames from the video sequence. Additionally, we have also trained a state-of-the-art LP tracker system for comparison using the same training set.

The tracking results can be seen in Figure 4 and supplementary video. It was found that for all the test sequences, the proposed method did not suffer from any catastrophic failures with good tracking accuracy and thus required no manual reinitialisation during the course of tracking all the test sequences. This demonstrates the ability for the hierarchical regressors to cope with highly non-linear mappings between the feature space and predicted offset vectors. In contrast, the LP tracking method was not able to cope with any of the pose changes that was present, resulting in failure very early on in the test sequence. Even with manual reinitialisation, many points lost track the moment the face of the subject underwent significant changes in pose. In particular, for two subjects, it was not possible to maintain track of all the points at any one instance (i.e. there was always some tracked point that suffered from unrecoverable tracking error). Examples of such failures can be seen in Fig. 4b.

V. Conclusions

In this paper a non-linear predictor for estimating the displacement of tracked feature points on faces was proposed. We have shown that our method is able to robustly and accurately track facial feature points across significant changes in pose and expression. Crucially, this was achieved with neither pose-specific models, 3D-models nor pose-estimators. Using just a *single* tracking model, we are able to robustly and accurately track across a wide range of expression on poses. This was achieved by learning a hierarchical set of gradient boosted regression trees for predicting the displacement vectors of tracked points. Additionally, we have proposed a novel algorithm that simultaneously configures a hierarchical set of trackers for optimal tracking results. Experiments carried out on sequences of naturalistic conversation and sequences with large pose and expression changes show that the proposed method is superior to state of the art methods, in being able to robustly track a set of facial points whilst gracefully recovering from tracking failures.

Acknowledgment

This work was supported by funding from the UK Home Office and the EPSRC project EP/I011811/1.

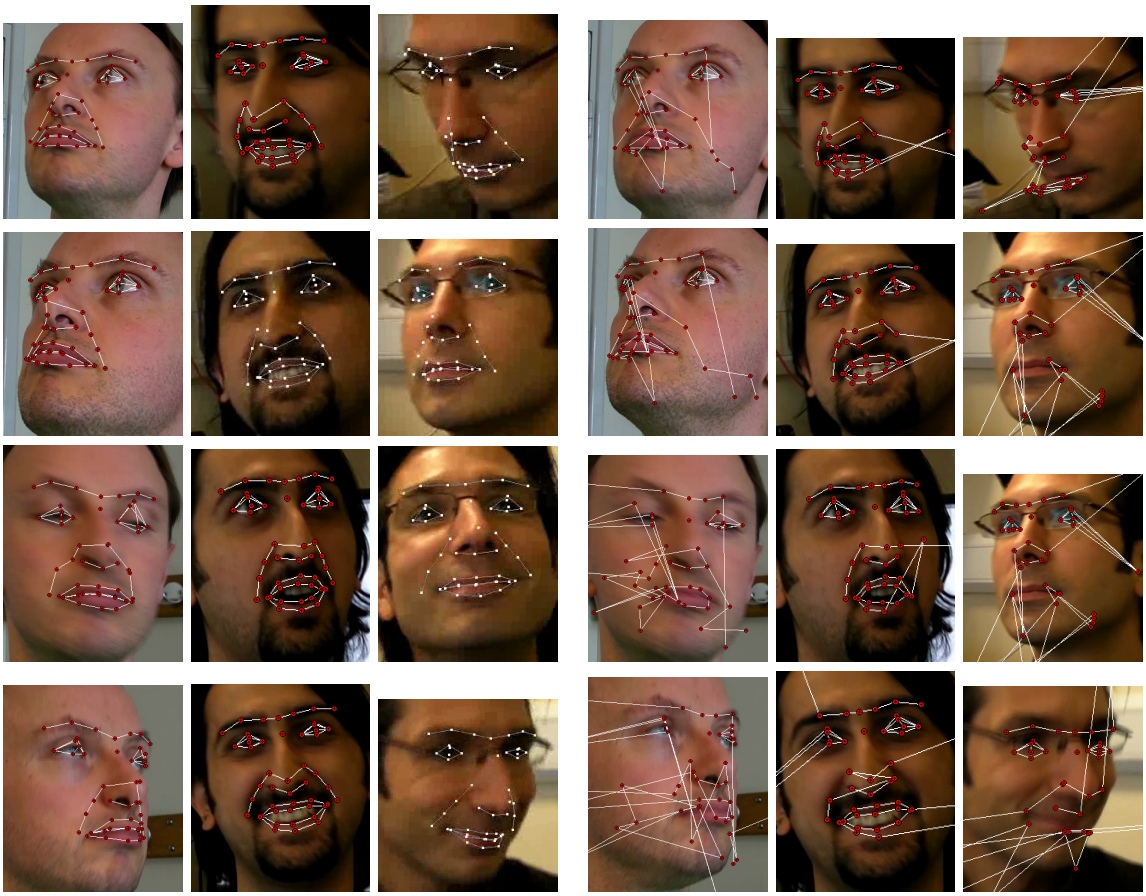
References

[1] T. Cootes, G. Edwards, and C. Taylor, "Active appearance models," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 681–685, 2001.

[2] T. C. Patrick Sauer and C. Taylor, "Accurate regression procedures for active appearance models," in *Proc. BMVC*, 2011, pp. 30.1–30.11, http://dx.doi.org/10.5244/C.25.30.

[3] L. Zhang, H. Ai, S. Xin, C. Huang, S. Tsukiji, and S. Lao, "Robust face alignment based on local texture classifiers," in *IEEE International Conference on Image Processing*, vol. 2, September 2005, pp. II–354–7.

[4] D. Cristinacce and T. Cootes, "Feature detection and tracking with constrained local models," in *Proc. of British Machine Vision Conference*, 2006, pp. 929–938.



(a) Proposed Method (b) Linear Predictor Method

Fig. 4: The tracking results on sample frames for 3 different subjects used in the experiments of Section IV-B. Shown here is how the proposed method copes with large pose changes, whilst the LP method results in failure of tracking for multiple points.

[5] L. Ding and A. Martinez, "Features versus context: An approach for precise and detailed detection and delineation of faces and facial featurse," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2010.

[6] F. Sukno, S. Ordas, C. Butakoff, S. Cruz, and A. Frangi, "Active shape models with invariant optimal features: Application to facial analysis," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 29, no. 7, pp. 1105–1117, 2007.

[7] I. Castelli, M. Maggini, S. Melacci, and L. Sarti, "Auto associative neural network based active shape models," in *Proc. of 8th IEEE International Conference on Face and Gesture*, 2008, pp. 1–6.

[8] I. Matthews, T. Cootes, and J. Bangham, "Extraction of visual features for lipreading," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, pp. 198–213, 2002.

[9] T. Cootes, C. Taylor, D. Cooper, and J. Graham, "Active shape models - their training and application," *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 38–59, January 1995.

[10] K. Atul, Y. Huang, and D. Metaxas, "Tracking facial features using mixture of point distribution models," in *Proc. of Indian Conference on Computer Vision Graphics and Image Processing*, India, December 2006, pp. 492–503.

[11] J. Xiao, S. Baker, I. Matthews, and T. Kanade, "Real-time combined 2d+3d active appearance models," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, June 2004, pp. 535–542.

[12] F. Dornaika and J. Ahlberg, "Fitting 3d face models for tracking and active appearance model training," *Image and Vision Computing*, vol. 24, no. 9, pp. 1010 – 1024, 2006.

[13] J. Sung, T. Kanade, and D. Kim, "Pose robust face tracking by combining active appearance models and cylinder head models," *International Journal on Computer Vision*, vol. 80, no. 2, pp. 260–274, 2008.

[14] E. Ong, Y. Lan, B. Theobald, R. Harvey, and R. Bowden, "Robust facial feature tracking using selected multi-resolution linear predictors," in *Proc. of the 12th International Conference on Computer Vision*, 2009.

[15] T. F. Cootes, M. C. Ionita, C. Lindner, and P. Sauer, "Robust and accurate shape model fitting using random forest regression voting," in *Proceedings of the 12th European conference on Computer Vision - Volume Part VII*, ser. ECCV'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 278–291.

[16] M. Dantone, J. Gall, G. Fanelli, and L. V. Gool, "Real-time facial feature detection using conditional regression forests," in *CVPR*, 2012.

[17] J. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[18] T. Sheerman-Chase, E.-J. Ong, and R. Bowden, "Cultural factors in the regression of non-verbal communication perception," in *Proc. of the Workshop on Human Interaction in Computer Vision*, Barcelona, Nov 2011.