

Cascaded Face Alignment via Intimacy Definition Feature

Hailiang Li*, Kin-Man Lam*, Edmond M. Y. Chiu, Kangheng Wu, Zhibin Lei

*Department of Electronic and Information Engineering, The Hong Kong Polytechnic University
Hong Kong Applied Science and Technology Research Institute Company Limited
Hong Kong, China

harley.li@connect.polyu.hk, {harleyli, edmondchiu, khwu, lei}@astri.org, enkmlam@polyu.edu.hk

Abstract — In this paper, we present a fast cascaded regression for face alignment, via a novel local feature. Our proposed local lightweight feature, namely intimacy definition feature (IDF), is more discriminative than landmark shape-indexed feature, more efficient than the handcrafted scale-invariant feature transform (SIFT) feature, and more compact than the local binary feature (LBF). Experimental results show that our approach achieves state-of-the-art performance, when tested on the most challenging benchmarks. Compared with an LBF-based algorithm, our method is able to obtain about two times the speed-up and more than 20% improvement, in terms of alignment error measurement, and able to save an order of magnitude of memory requirement.

Keywords— Cascaded Face Alignment; Random Forest; Intimacy Definition Feature;

I. INTRODUCTION

Face alignment is an active research topic in computer vision. It is often used as an early, but crucial, step to other important tasks for face analysis, such as emotion and expression recognition [9], face recognition [10], and face hallucination [11]. It is also used in many other applications, such as human-machine interactions, video conferencing, gaming and animation, and has received increased attention from the computer-vision research community. Face alignment is a process to locate facial key points or landmarks (e.g. the eyebrows, eye corners, and mouth corners) from a given face image.

A vast majority of face-alignment methods assume that the face-bounding box, for a face image, is known both at the training and fitting stages. The face-bounding box is usually obtained either through a face-detection algorithm, such as the Viola-Jones [12] face detector, or from manual annotations, i.e. the ground truth. The set of coordinates of the facial landmarks is usually referred to as the face shape. Various machine-learning algorithms have been proposed to estimate the face shape. Traditional methods for face alignment usually involve these classic pioneering works: Active Shape Model (ASM) [3] and Active Appearance Model (AAM) [4]. Both ASM and AAM are statistical models; ASM represents the shape of an object, while AAM represents both texture and shape. Constrained Local Models (CLM) [24, 25, 27, 28] attempt to model shape prior, similar to AAM, by assuming that face local appearance information and global face-shape patterns lie in a linear subspace spanned by bases, learned from principal component analysis (PCA). In [26], face-shape fitting is formulated as a non-linear minimization problem, which sets a target of minimizing the error (i.e., the average distance of all corresponding landmarks which normalized by the inter-pupil distance) between the model instance and the given image, with respect to the model parameters, which control the shape and appearance variation of faces. In [26], an extension to the inverse compositional image-alignment algorithm [29] was proposed, which decouples shape from appearance, and defines a computationally efficient AAM framework. Usually, a CLM model is composed of three main parts: a point distribution model (PDM), patch experts which perform matching for local patches around landmarks of interest, and a final fitting process.

Different fitting strategies have been used in CLM models. Regularized Landmark Mean Shift (RLMS) [28] is a popular strategy, which estimates the rigid and non-rigid parameters by minimizing the misalignment error of all the landmarks, while regularized by overly complex or unlikely shapes. In [27], a local neural field (LNF) patch expert was proposed, which learns the similarity and long-distance sparsity constraints to derive relationships between pixels (neighboring and longer distance). The method achieves state-of-the-art performance, compared to the traditionally constructed CLM models.

These constructed models have limited expressive power to capture all possible complex and subtle face variations, due to variations in expression, illumination, pose, etc. Furthermore, due to the computation of the inverse of Hessian matrix and Jacobian matrix [2], it is very hard to improve the speed exponentially on constructed algorithms. Only training-based regressions can achieve the efficiency required for real applications, such as smart mobile phones.

In the past few years, a new family of face-alignment algorithms has emerged [1, 2, 5, 8, 13], which directly learns regressors from image feature descriptors to the target shape. These regression-based methods are gaining popularity, due to their excellent performance and high efficiency in the face-alignment task. Shape-indexed feature [2, 8, 13], whose index provides some clue about the hierarchical structure of the shape, is employed to enhance efficiency. In [2], the handcrafted scale-invariant feature transform (SIFT) feature is used for accurate fitting. As inspired by the works in [1, 2], an efficient and discriminative feature is a crucial element for random-forest-based cascaded face-alignment algorithms. We propose a novel and efficient feature, which can be incorporated into other face-alignment frameworks to further boost their performance.

Having given an introduction to face alignment, the remainder of this paper is organized as follows. In Section II, we will review random forest, and the random-forest-based cascaded face-alignment approach. In Section III, a feature derived from the shape-index feature, named intimacy definition feature (IDF), will be presented. Then, our proposed IDF-based cascaded random-forest face-alignment algorithm will be described and analyzed. We will also evaluate our proposed method and compare it with recent fast LBF-based methods. Section IV will discuss how to cluster the training samples into subspaces for selecting representative shapes to form initialization samples. Experiment results and parameter settings will be presented in Section V, and a conclusion is given in Section VI.

II. RANDOM FORESTS FOR LANDMARK LOCALIZATION

A landmark localization algorithm is important for face recognition and other related applications, which require extraction of local features at some specified feature points or landmarks in a face. For face alignment, a number of points, usually between 17 and 68, are selected and searched from a face image. An example of the landmarks is shown in Fig. 1, which has 68 points located around the eyes, nose, lips, and face contour. These feature points, which are useful for discriminative and generative analysis, carry the most significant information about a face. Based on these feature points, a model can then be learned from a large number of landmark-labeled face images, used for facial-shape estimation for unseen face images.

Recently, there have been two main approaches for face alignment, methods based on the Active Appearance Models (AAM) [4] that build parametric models of appearance, and regression-based models that directly model a mapping from appearance to shape [1, 2, 5, 8, 13]. In our algorithm, we adopt the regression-based approach, based on the cascaded shape-regression framework firstly proposed by Dollar et al. in [8]. Different from other methods, this approach progressively refines the initial shape estimate in several stages directly from appearance, without requiring to learn any parametric shape or appearance models. As the cascaded regression algorithms are mainly based on random forests, we give a brief

review of the main principles of random forest and cascaded-shape regression in this section.

II.1 Random Forest:

In recent years, Random Forests [14] (RFs) have emerged as very useful classifiers for a large variety of computer-vision tasks, including object detection [16], data clustering [17], image super-resolution [18, 19], etc. This method is relatively simple, and has many merits that make it particularly interesting for computer vision problems: (i) efficiency in both training and prediction, (ii) inherent unsupervised classification capability for multi-class problems, (iii) suitability for parallel processing, and (iv) good performance for high-dimensional data.

A random forest is an ensemble of T binary decision trees $T^t(x): V \rightarrow R^K$, where t is the index of the trees, $V \in R^M$ is the M -d feature space, and $R^K = [0, 1]^K$ represents the space of class probability distributions over the label space $Y = \{1, \dots, K\}$, as shown in Fig. 1.

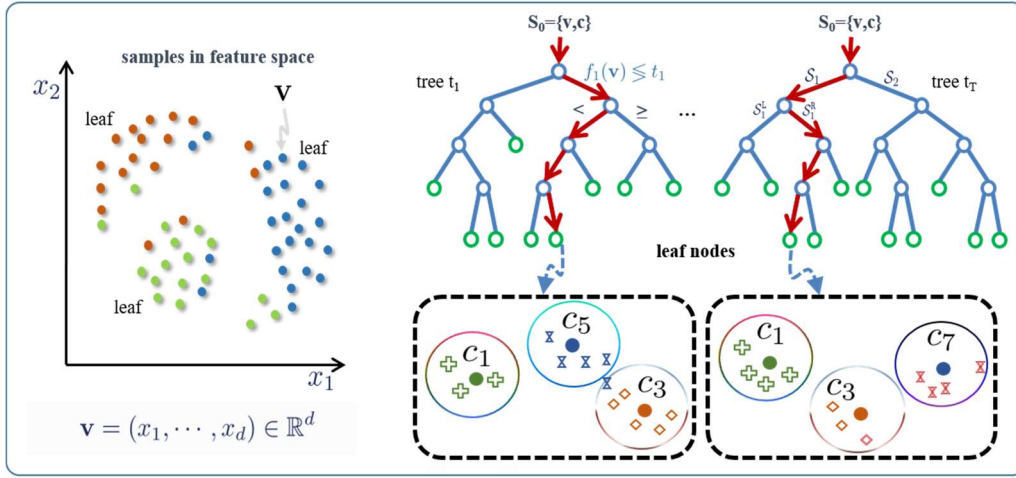


Fig. 1: Random forest for clustering data.

In the testing/prediction stage, each decision tree returns a class probability $p_t(y|\mathbf{v})$ for a given test sample $\mathbf{v} \in R^M$, and the final class label y^* is then obtained via averaging:

$$y^* = \arg \max_y \frac{1}{T} \sum_{t=1}^T p_t(y|\mathbf{v}). \quad (1)$$

A splitting function $s(\mathbf{v}; \Theta)$ is typically parameterized by two values: (i) a feature dimension $\Theta_1 \in \{1, \dots, M\}$, and (ii) a threshold $\Theta_2 \in R$. The splitting function is defined as follows:

$$s(\mathbf{v}; \Theta) = \begin{cases} 0, & \text{if } v(\Theta_1) < \Theta_2, \\ 1, & \text{otherwise,} \end{cases} \quad (2)$$

where the outcome defines to which child node the sample \mathbf{v} is routed, 0 and 1 are two labels belonging to the left and right child nodes, respectively. Each node chooses the best splitting function Θ^* out of a randomly sampled set $\{\Theta^i\}$ by optimizing the following function:

$$I = \frac{|L|}{|L|+|R|} H(L) + \frac{|R|}{|L|+|R|} H(R), \quad (3)$$

where L and R are the sets of samples that are routed to the left and the right child nodes, and $|S|$ represents the number of samples in the set S . During the training of a random forest (RF), the decision trees are provided with a random subset of the training data (i.e. bagging), and are trained independently of each other. Training a single decision tree involves recursively splitting each node, such that the training data in the newly created child nodes is clustered according to class labels. Each tree is grown until a stopping

criterion is reached (e.g. number of samples in a node is less than a threshold or the tree depth reaches a maximum value) and the class probability distributions are estimated in the leaf nodes. $H(S)$ is the local score for a set of samples (S is either L or R), which normally is calculated using entropy as in (6), but it can be replaced by the variance [1] or Gini index [14].

$$H(S) = - \sum_{k=1}^K [p(k|S) * \log(p(k|S))] \quad (4)$$

where K is the number of classes, and $p(k|S)$ is the probability for class k , which is estimated from the set S .

II.2 Cascaded shape regression:

Many face alignment methods work under a cascaded framework, where an ensemble of N regressors operates in a stage-by-stage manner, which are referred to as stage regressors. This approach was first explored in [8]. At the testing stage, the input to a regressor R_t at stage t is a tuple (I, S^{t-1}) , where I is an image and S^{t-1} is the shape estimate from the previous stage (the initial shape S^0 is typically the mean shape of the training set). The regressor extracts features with respect to the current shape estimate, and regresses a vector of shape increment as follows:

$$S_t = S_{t-1} + R_t(\phi_t(I, S_{t-1})), \quad (5)$$

where $\phi_t(I, S_{t-1})$ is referred to as the shape-indexed features, i.e. they depend on the current shape estimate. The cascade progressively infers the shape in a coarse-to-fine manner – the early regressors handle large variations in shape, while the later ones ensure small refinements. After each stage, the shape estimate resembles the true shape closer and closer.

In our algorithm, the feature mapping function $\phi_t(I, S_{t-1})$ generates the local IDF values derived from the shape-indexed feature. There is an assumption, proved by intensive experimental results, that the shape increments have close correlation with the local features of the landmarks, which define the face shape. Thus, given the features and the target shape increments $\{\Delta S_t = S - S_{t-1}\}$, we can learn a linear projection matrix R_t . Most cascaded regression models [1, 2, 5, 8, 13] have a similar workflow, as shown in Fig. 4.

III. INTIMACY DEFINITION FEATURE BASED CASCADED REGRESSION MODEL

In this section, we will first introduce a new feature, which is efficient for local pattern representation and matching, based on measuring the degree of intimacy (DoI) between two features.

III.1 Efficient Metric on Intimacy Definition Feature:

To explain the features, we replace a feature with a member in a family, and we measure the DoI between two family members. The relationships between the family members are represented by using a binary family tree, as shown in Fig. 2. The DoI between two family members can be computed by their respective intimacy definition feature (IDF) values. In Fig. 2, the DoI between David and Daniel should be stronger than that between David and Denis. This is because David and Daniel have the same father, while David and Denis do not have the same father but they share the same grandfather only. The way to let the computer learn the DoI value, between any two members in the same generation or level in the hierarchical family tree, is to digitize the DoI values. This means to set values to all the nodes in the family tree and define a distance metric between any two of the members.

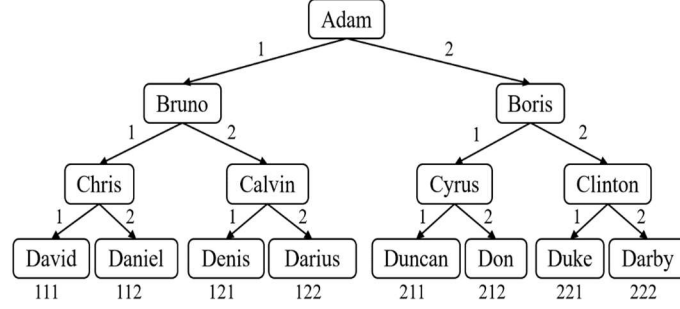


Fig. 2: A family tree shows the degree of intimacy between two persons in the 4th generation.

As we can see in the family tree in Fig. 2, two persons, who share more recent parents, should be more intimate than those who share relatively distant parents, as described in the previous example. However, how can a computer know this intimacy, based on this logic comparison operation? In this paper, we propose a simple, yet efficient, method to compare the DoI values between two members in the same generation. We firstly set two persons in the same generation with values of very small difference, for example, we set 1 and 2 as the respective *path values* to the two offspring nodes (e.g. David is the younger brother so his *path_value* is 1, while Daniel is the older brother so his *path_value* is 2) in the full binary family tree. Then, we set a relatively larger value, e.g. 10, to the *generation value* k for each generation level. Each node (except the root node) can then be encoded by summing up all the corresponding level weights along the path from the root to the node of a member concerned, where a level weight of a node is computed by multiplying the value of the node and its corresponding *generation value* k . We name this as the intimacy definition feature (IDF) value of the node or family member, which can be calculated as follows:

$$IDF = \sum_{l=1}^L path_value_l * k^l, \quad (6)$$

where L is the total number of levels in the family tree. Therefore, the IDF value of David can be encoded as: 111 ($1*10^2+1*10^1+1*10^0$), and Daniel with IDF value: 112 ($1*10^2+1*10^1+2*10^0$). We can also encode Denis as IDF value: 121 ($1*10^2+2*10^1+1*10^0$). The intimacy distance between David and Daniel is $1(1=abs(111-112))$, and the distance between David and Denis is 10 ($10=abs(111-121)$). The distances show that the intimacy between David and Daniel should be greater than that between David and Denis. Based on the proposed IDF, we can compute the DoI value between the IDF values of two family members belonging to the same generation. The family members can be replaced by visual features, which are encoded by IDF values. Consequently, the similarity between two features can be computed by measuring their DoI.

In our study, we found that this simple, yet efficient, feature can be computed by traveling a random forest, which can achieve promising performance, in terms of both accuracy and speed, as shown in Section IV. When using the encoded feature value for linear regression on the leaf nodes for prediction, for more reliable and better performance, the feature is normalized as follows:

$$normalized_IDF = \frac{(IDF-IDF_{min})}{(IDF_{max}-IDF_{min})} \quad (7)$$

where IDF_{min} and IDF_{max} are the minimum and maximum IDF values, respectively, in the same level under consideration. Using our example, the range of the IDF values in the binary tree is [100, 222], i.e., $IDF_{min}=100$ and $IDF_{max}=222$. Therefore, based on equation (7), the normalized IDF value for David (111) can be calculated as: $(111-100)/(222-100) = 0.090164$.

III.2 Derive IDF feature from shape-indexed feature:

For each stage, the whole feature vector Φ_t is a concatenating vector from a set of independent local-feature mapping functions: $\phi_t(I, S_{t-1})$, which is the local IDF values, derived from the shape-indexed feature in our algorithm. A shape-indexed feature is the value of two pixels' intensity difference. For every landmark point, those two pixels used to compute the shape-indexed value are chosen with two randomness in the random forest splitting rule, which means that they are randomly sampled from a number of candidate pixels (e.g. 500) and the threshold is also randomly selected. The positions of the pixel pair and the threshold to be used are decided, based on maximizing the information gain obtained when splitting all the samples in a node into its left and right nodes.

Same as LBF [1], we discard such learned local shape-indexed features since it is not sufficiently discriminative, or does not encode the path of a sample along a tree explicitly. Instead, we encode the path of a sample along a tree ends to a leaf node, using our proposed IDF values. All the IDF features are concatenated to form a global feature mapping function Φ_t for learning a global linear projection, i.e. the regressor R_t , in the next step.

All the pixel pairs are sampled from the neighborhood, centered at each landmark point. Different faces should normally be aligned with different rectangles, but the same neighborhood size can be used for the landmark points of different faces. The idea of our shape-indexed feature is described in Fig. 3.

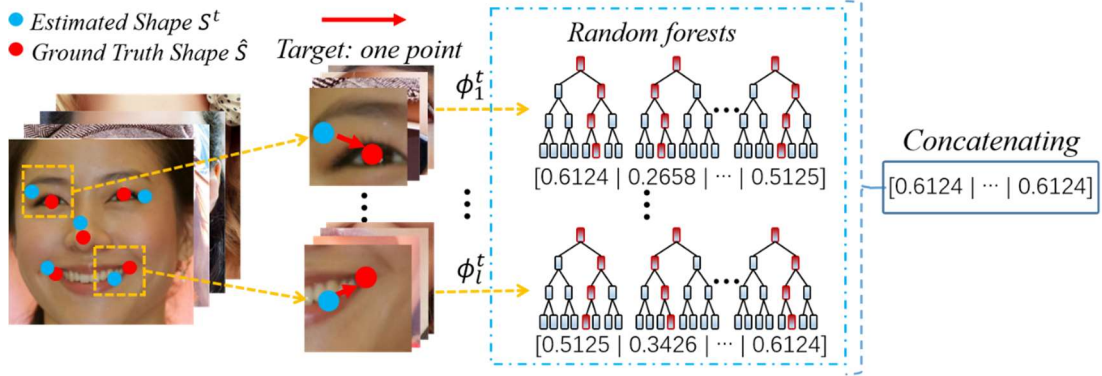


Fig. 3: The process of IDF-based feature vector extraction

In the prediction stage, the neighborhood size around the landmark points can be reduced when moving from one cascade to another cascade, so the cascaded shape regression operates from coarse to fine progressively.

III.3 Cascaded shape regression based on IDF feature:

A similar work to our proposed algorithm is the LBF-based method in [1], which improves the supervised descent method (SDM) [2] used in linear regression. Random forests were used for training, so as to minimize the alignment error for the respective landmarks with LBF, rather than the shape-indexed feature in the leaf nodes. LBF is a type of local feature, which is coded as a binary array, by placing the value '1' for leaf nodes, where samples fall into them eventually while traversing the tree, and the value '0' otherwise. Each landmark is coded individually, and the local features are concatenated to form a global feature vector, which is then learned by using ridge regression (i.e., linear regression with L^2 regularization). Rather than using LBF, our proposed IDF is employed in the cascaded alignment framework, as depicted in Fig. 4. To further improve the performance, we refine the shape initialization by using the k -means clustering algorithm. The success of this method is due to its feature-learning step, where features are explicitly learned for the given specific task. Due to the sparse nature of the feature vector, the testing phase can be reduced to traversing the forest, and performing simple table look-ups

and additions. The method can achieve an impressive speed of 3,000 fps, the fastest.

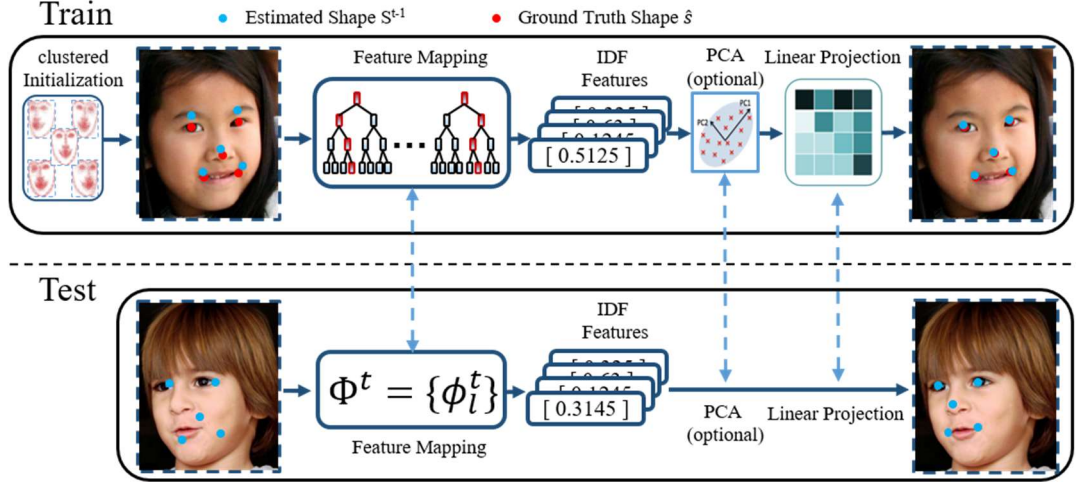


Fig. 4: An overview of the workflow for IDF-based cascaded regression face alignment

However, LBF has a high dimensionality. Assume that the number of landmarks (or forests) for a face is l , the number of trees for a forest is t , and the depth of a tree is d . The dimensionality of LBF will then be $l \cdot t \cdot 2^{(d-1)}$. For a normal setting of $l = 68$, $t = 10$, and $d = 7$, the feature dimension is $68 \times 10 \times 2^{(7-1)} = 43,520$, which is relatively high. Usually, with more and deeper trees, the alignment errors will become smaller. However, the high dimensionality of LBF restricts it from using deeper trees. Although the feature is sparse, its high dimensionality imposes a high burden on the computation of linear regression and the storage requirement. An intuitive way to solve the problem is to employ Principal Component Analysis (PCA) to reduce the dimensionality. However, LBF is a binary, sparse feature, and carries labelling information, which makes PCA not acceptable for the feature. To avoid the computational complexity, the LBF-based approach has to limit the tree depth to 5, which means that there are, at most, 16 leaf nodes in each tree. Consequently, this heavily restricts its capability on classification and prediction.

Compared to the pixel shape-indexed feature [13], LBF is more discriminative because it explicitly encodes the full path, from the root to the leaf node of each sample. Although LBF is discriminative, it is hard to greatly improve its performance because of its high dimensionality when using deeper trees. To improve the performance, an intuitive way is to replace LBF with another more compact and efficient *index feature*, which encodes the path of a sample along a tree. However, the performance is very poor, because index values are similar to labels, which inclines more to results with over-fitting. A simple analysis in Fig. 2 can help describe the problem of using an *index feature*. Suppose that we set the indices for David, Daniel, and Denis at 1, 2, and 3, respectively, as shown in Fig. 2. With these values, we can find that the DoI value between David and Daniel is the same as that between Daniel and Denis. However, from Fig. 2, we know that the intimacy between David and Daniel should be higher than between Daniel and Denis.

Our algorithm is based on extracting the IDF value at each facial landmark, using the full binary family tree. With the IDF values, leaf nodes can be compared based on their DoI. The main contribution of this paper is that the efficient IDF feature is proposed to replace LPF. This can greatly reduce the feature dimensionality, while a promising performance can still be achieved. More importantly, our algorithm runs much faster and requires less memory than that using LPF. For example, for $l = 68$, $t = 10$, and $d = 7$, the feature dimensionality of IDF is $68 \times 10 \times 1 = 690$, rather than 43,520 for LBF. In other

words, the dimensionality is reduced by 64 times.

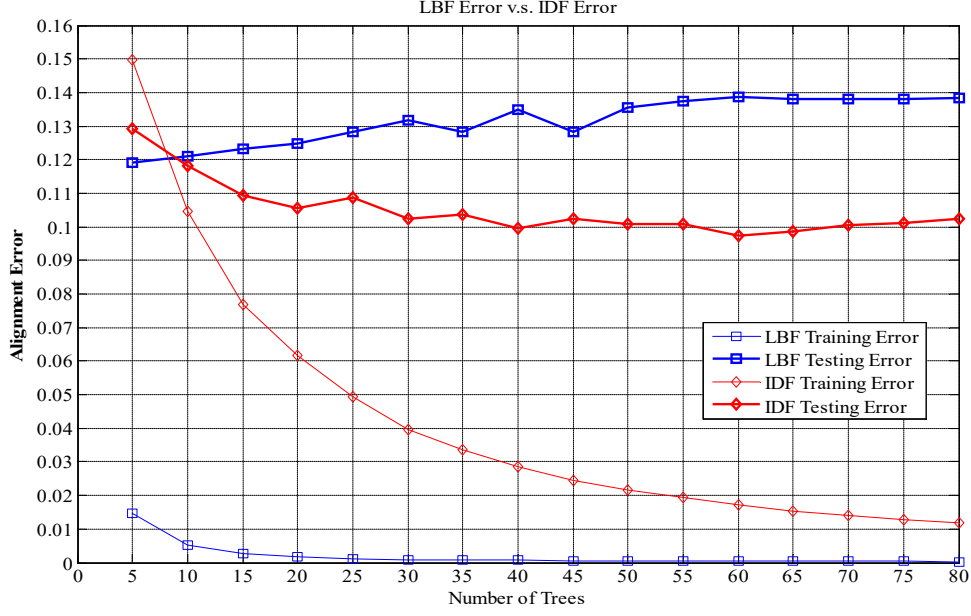


Fig. 5: A comparison of the alignment errors of the IDF and LBF algorithms on the LFPW dataset [20], with tree depth = 7, training samples: 1000, testing samples: 300)

To demonstrate the effectiveness of IDF for face alignment, we set tree depth, maximum number of stages, and number of landmarks at 7, 7, and 68, respectively, and measure the respective alignment errors using LBF and IDF. Fig. 5 shows the alignment errors in the training and testing stages, based on the LFPW dataset [20], with different numbers of trees. From the results, we can see that our proposed IDF algorithm is able to achieve, on average, an error of around 0.10, when the number of trees is more than 10, while the minimum error achieved by the LBF-based algorithm is 0.12. Therefore, our algorithm can achieve an improvement of about 20%, in terms of alignment error, when compared to the LFP-based algorithm.

	Number of Trees									
stage	5	10	20	30	40	50	60	70	80	Avg.
1	0.1765	0.1714	0.1630	0.1583	0.1583	0.1583	0.1533	0.1495	0.1485	0.1597
2	0.1411	0.1341	0.1300	0.1315	0.1315	0.1315	0.1397	0.1387	0.1390	0.1352
3	0.1276	0.1251	0.1252	0.1292	0.1292	0.1292	0.1390	0.1382	0.1386	0.1312
4	0.1232	0.1226	0.1240	0.1287	0.1287	0.1287	0.1389	0.1381	0.1385	0.1301
5	0.1209	0.1217	0.1235	0.1285	0.1285	0.1285	0.1388	0.1380	0.1384	0.1296
6	0.1198	0.1212	0.1234	0.1284	0.1284	0.1284	0.1388	0.1380	0.1384	0.1294
7	0.1193	0.1209	0.1233	0.1283	0.1283	0.1283	0.1388	0.1380	0.1384	0.1293

Table-1: Alignment errors at different stages, with different number of trees, based on the LBF algorithm

	Number of Trees									
stage	5	10	20	30	40	50	60	70	80	Avg.
1	0.1924	0.1915	0.1873	0.1937	0.1886	0.1914	0.1826	0.1810	0.1856	0.1882
2	0.1636	0.1583	0.1472	0.1462	0.1412	0.1360	0.1312	0.1318	0.1326	0.1431
3	0.1540	0.1412	0.1294	0.1283	0.1206	0.66	0.1112	0.1129	0.1136	0.1254
4	0.1445	0.1309	0.1188	0.1192	0.1119	0.1091	0.1041	0.1059	0.1073	0.1168
5	0.1380	0.1249	0.1136	0.1142	0.1076	0.1057	0.1010	0.1032	0.1049	0.1126
6	0.1334	0.1200	0.1114	0.1104	0.1051	0.1039	0.0990	0.1015	0.1034	0.1098
7	0.1291	0.1180	0.1093	0.1089	0.1036	0.1024	0.0974	0.1005	0.1025	0.1080

Table-2: Alignment errors at different stages, with different number of trees, based on the IDF algorithm

Another factor we should consider is the number of trees required to achieve a specific alignment error. From Fig. 5, we can see that using about 10 trees for our algorithm can achieve even smaller errors than that of LBF using more than 80 trees. As shown in Table 1 and Table 2, although LBF performs better in the first 3 stages, IDF can always achieve better performance at later stages, since its alignment error converges steeper than LBF. In other words, IDF converges faster in the coarse-to-fine search of the true landmarks, with a higher discriminative power.

Fig. 6(a) shows the alignment errors of the LBP and IDF methods, with different numbers of stages used. We can see that the curve for IDF is much steeper than that for LBF, which means that the IDF feature converges to lower errors faster, and is more discriminative than LBF at later stages. An explanation for this is that the IDF value is represented as floating point numbers, and has a stronger representation than LBF represented by binary numbers. Fig. 6(b) shows the alignment errors of IDF with a larger number of stages used. To make a balance between computational complexity and fitting accuracy, using 7 stages is a compromise. Therefore, in the rest of this paper, our algorithm uses 7 stages of cascade.

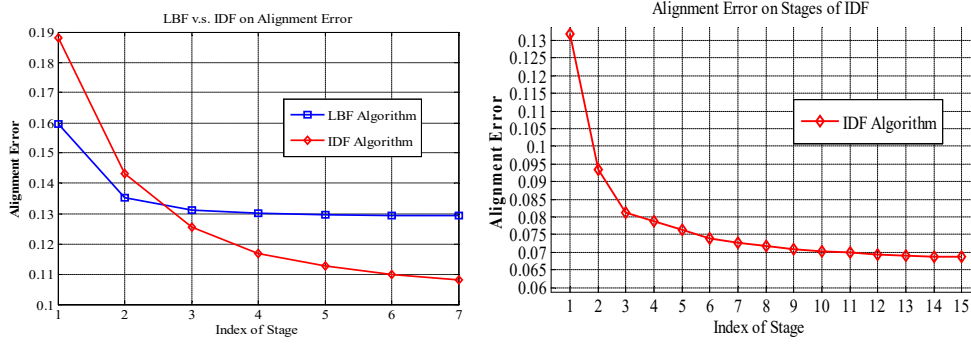


Fig. 6: Alignment errors with different number of stages of cascade: (a) based on LBF and IDF, up to 7 stages, and (b) based on IDF only, up to 15 stages (tree depth = 7, LFPW dataset [20]).

Having analyzed the LBF algorithm, we found that, in the prediction stage, feature extraction and linear regression take up about 30% and 70% of the total computation, respectively. Since our proposed IDF is derived from the shape-indexed feature, which takes up less computation in the extraction. Furthermore, IDF has its dimensionality an order of magnitude lower than that of LBF, so the computational complexity for linear regression (the *LibLinear* package is used for IDF and LBF) is greatly reduced. As shown in Fig. 7, the number of frames processed per second, based on IDF, is about 2 times faster than LBF, with the same setting.

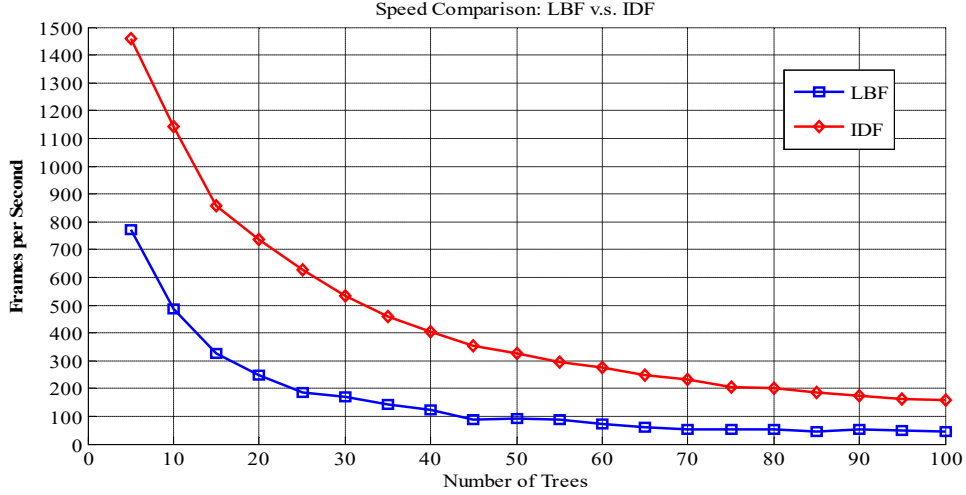


Fig. 7: The speeds in terms of number of frames per second for the IDF and LBF algorithms (tree depth = 7, Helen dataset [21]).

When the tree depth increases, the feature dimensionality of LBF increases exponentially, while the IDF algorithm increases linearly. In addition to computational complexity, memory requirement is also an important issue for real applications. Because of the high dimensionality of LBF, more weights for linear regression need to be saved for the regression model. That means the LBF-based method needs much more memory in the prediction step, as all the progression models for the cascaded stages are to be kept in memory, as shown in Fig. 8.

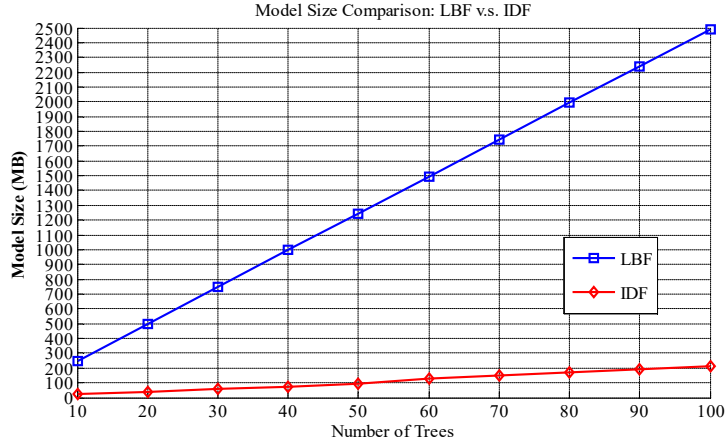


Fig. 8: Memory requirements (MB) of the IDF and LBF with different numbers of trees (tree depth: 7, Helen dataset [21])

IV. TRAINING WITH REPRESENTATIVE SHAPES FROM CLUSTERED SUBSPACES

A limitation of the cascaded regression model is that the approach is sensitive to the initialized shape, which means that using a common mean shape, it may not be possible to obtain good performances on some images with unseen face profiles. In [5], a conditional regression forest was proposed for face alignment, in which annotated samples are used to train a classifier to detect the face pose with discriminative features inside and outside the face-bounding boxes. Based on the annotated face poses, a number of cascade regression forest models are trained, instead of a single model only. In the testing stage, once the face pose has been detected using the pose detector, the probability of the head pose is estimated from the testing image, and the corresponding trees are selected for later face alignment.

In [6], a pose detector is employed for estimating initial shapes, based on the k nearest neighbors selected from training samples, which uses two efficient and effective features, namely the Histogram of

Oriented Gradients (HOG) [22] and Local Binary Patterns (LBP) [23], for searching example face images with a similar pose and texture appearance to the query face, respectively. The local appearance of feature points can be accurately approximated with locality constraints. Therefore, with the searched training faces, which have similar poses and textures to a test face, more accurate initial shape model can be constructed in the prediction stage.



Fig. 9: Results from clustering with 7 groups of faces with different poses.

Different from the two above-mentioned methods [5, 6], our algorithm does not use any pose detector that considers similar texture. The training samples are selected, based on the similar shapes spanned in face-shape subspaces. As using random initial shapes in the training phase can improve the generalization capability of the alignment method, this means that the trajectory of face alignment, during the regression stages in the prediction process, can be learnt from training samples. Intuitively, for a face with a large pose, the shape trajectory of a left-pose face cannot be learnt from a right-pose face. Therefore, training samples selected for the initial model should have similar poses, which can help to learn the pose information implicitly. In [5], a face dataset with different poses and with 10 landmark points was created. However, we consider 68 landmark points in face images, and we will evaluate our algorithm using some standard public datasets, such as the LFPW dataset [20] and Helen dataset [21]. The dataset can be labelled manually, as it was in [5], so that the learning will be more precise. In this paper, we have devised a more efficient scheme for this training process. This is important because the tedium of labeling faces and their poses is challenging and the work may be imprecise. For example, it is difficult to decide on a face with a pose (e.g. at 45-degree angle) from two clusters (e.g. 30-degree angle and 60-degree angle). It is often an ambiguous task for human eyes. In [6], although k nearest neighbors similar to the test face are searched with locality constraints, a relatively narrow subspace may be produced, based on the k training samples. This may spoil the generalization capability of the learned model.

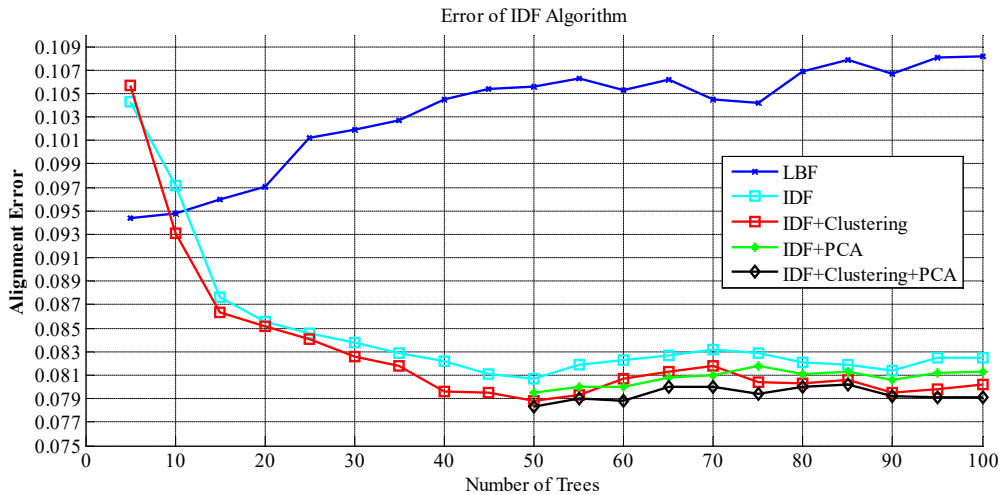


Fig. 10: Alignment errors of the IDF algorithm with clustering and/or PCA (tree depth: 7, stages: 5, Helen dataset [21]).

We use the k -means algorithm to group the training samples into a number of clusters, as shown in Fig. 9. Then, for each test face image, instead of learning the initial shape using the whole training dataset, we select example shapes only from the cluster with a similar pose to the test face. Therefore, the model

is learned with the pose information, and a smaller number of selected examples are necessary to represent the test face well. Experimental results in Fig. 10 show that the “IDF+Clustering” training scheme further improves the alignment error, when compared to the non-clustering scheme.

	Number of Trees									
	10	20	30	40	50	60	70	80	90	100
IDF	680	1360	2040	2720	3400	4080	4760	5440	6120	6800
PCA*	N/A	N/A	N/A	N/A	768	806	837	852	879	888

Table-3: Feature dimensions of IDF and IDF after using PCA (PCA* means: IDF+PCA, keeping 95% of variance at Stage 1).

The higher the feature dimension, the more the number of linear-regression weights for the regression model, which requires more computations and more memory space for the prediction step. All the weights of the models for the cascaded stages need to be kept in memory. Another advantage of using IDF, compared to LBF, is that IDF can apply PCA to reduce its feature dimensionality, because of its float property on feature values. From Table 3, we can see that, when the dimension becomes higher, retaining eigenvectors with 95% of variance can reduce the feature dimension by 80% to 90%, while comparable or even better performance can be achieved. Balance the overhead cost of PCA computation and the relax on linear regression after dimension reduction, theoretically a more optimal and faster solution can be found when the feature of IDF growing up. But it is hard to implement PCA with LBF feature’s binary values, therefore, IDF with a higher dimensionality can be adopted so as to achieve both efficiency and accuracy, which is hard with LBF feature. Fig. 4 shows the whole process of our proposed algorithm, and the training and fitting stages of our proposed algorithm are described in in Algorithm 1 and Algorithm 2 respectively.

Algorithm 1: IDF Training Stage:

Input: training data (I_i, S_i, \bar{S}_i) for $i=1, \dots, N$, where I_i represents a face image, S_i is the corresponding shape, and N is the number of training samples.

Output: regressors: $R = (R_1, \dots, R_T)$, T : stage count.

1: Using k -means to cluster shapes in S into M clusters $C = (C_1, \dots, C_M)$, randomly sample initial shapes for each target shape from its belonging cluster $\bar{S}_i \in C_i$ as source shapes;

2: for $t=1$ to T **do**

3: for all $i \in (1 \dots N)$ **do**

4: $\Delta S_t^i = S_t^i - \bar{S}_t^i$ \Rightarrow Calculate shape increment: ΔS_t^i

5: $f_t^i = \phi_t(I^i, S_{t-1}^i)$ \Rightarrow IDF features derived from Shape-indexed features

6: end for

7: $R_t = \arg \min_R \sum_i |R(f_t^i) - \Delta S_t^i|$

8: for all $i \in (1 \dots N)$ **do**

9: $\bar{S}_t^i = \bar{S}_t^i + R(f_t^i)$ \Rightarrow update current shape

10: end for

11: end for

Algorithm 2: IDF Fitting Stage:

Input: testing image I , initial (mean) shape S^0 , trained regressors: $R = (R_1, \dots, R_T)$, T : stage count.

Output: Estimated pose S^T

1: for $t=1$ to T **do**

2: $f_t = \phi_t(I, S_{t-1})$ \Rightarrow IDF features derived from Shape-indexed features

3: $\Delta S = R_t(\phi_t(I, S_{t-1}))$ \Rightarrow apply linear regressor R_t

4: $S_t = S_{t-1} + \Delta S$ \Rightarrow update pose

5: end for

V. EXPERIMENTAL RESULTS AND PARAMETER SETTING

Using the LBF or IDF feature on random forest can avoid the problem of having insufficient numbers of training samples in the leaf nodes, because these features are not extracted from the samples, but computed from the full binary tree. We have analyzed the encoding process of IDF, and found that the IDF value of each node in the full binary tree is affected by two parameters: the *difference value* d between two brother nodes and the *magnitude value* k for each generation level. However, since the final encoded values of all the nodes are relative values, one of these two parameters can be fixed and the other one used for fine-tuning. In our experiments, we fix the value of d to 1, and plot the alignment error curves for different values of k .

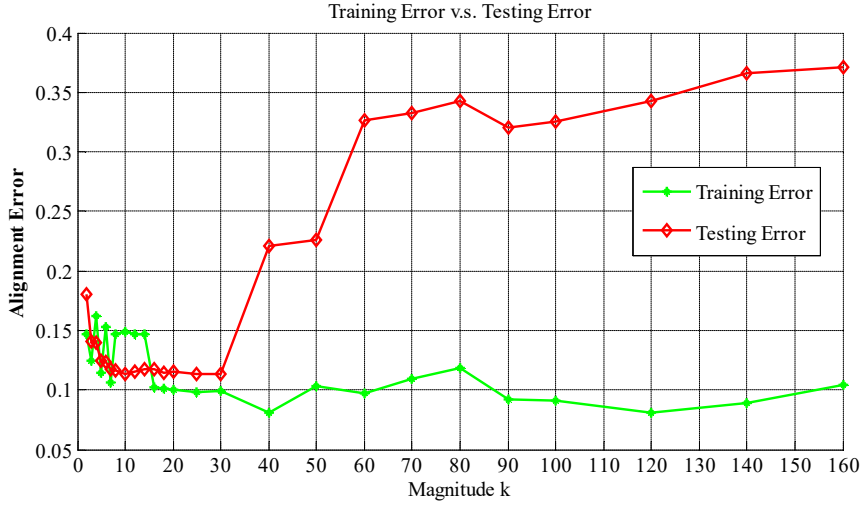


Fig. 11: Alignment errors of various magnitude values of k .

As shown in Fig. 11, the alignment errors become the lowest, when the *magnitude value* k is in the range from 10 to 30 (for the tree depth set at 7). This means when the *magnitude value* k is within this range, the encoded values keep discriminative capability. Therefore, for our proposed IDF feature, the optimal setting is as follows: tree depth: 7, maximum number of stages: 7, number of trees in a forest: 11, number of initialization faces: 50, number of shape clusters: 7, and *magnitude value* k : 10. The trained model, based on our proposed IDF feature and framework, is capable of achieving a comparable alignment quality to state-of-the-art methods [1, 2, 13, 15]. Meanwhile, our algorithm can run at a speed of 1,000 frames per second (FPS) on a desktop computer (Intel Core i7 4790 CPU @3.6GHz, 16GB RAM) with C++ code after thread parallelization on 8-core CPUs. Fig. 12 illustrates some results of the IDF method, and shows that IDF can locate landmarks accurately on faces with different poses and expressions, with occlusion, as well as faces with accessories (glasses). Our proposed method achieves

promising performance, compared to the state-of-the-art algorithms. Although [1] states that it can achieve the speed of 3,000 FPS, its experimental result is generated from trimmed version on most parameter settings.

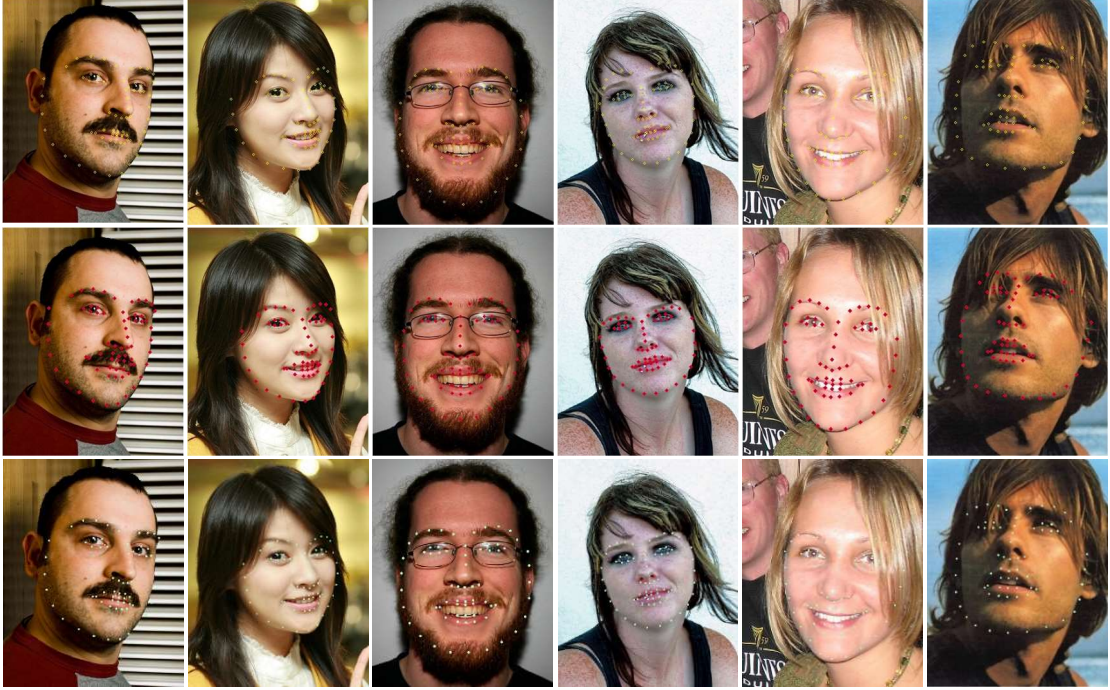


Fig. 12: Fitting results, with 68 landmarks, based on different methods and the Helen dataset [21]: Row 1: [15]; Row 2: [27], and Row 3: IDF.

For the linear regression setting, the *LibLinear* package [7] is used for both LBF and IDF, and the linear regression type is set at L2R_L2LOSS_SVR, i.e., L^2 -regularized L^2 -loss support vector regression (primal), in which Newton method with trust region step control is employed to achieve faster convergence [30].

VI. CONCLUSIONS

Real-time face alignment is still a challenging task. Although many researchers have put efforts into this research area and many algorithms have already been proposed, a highly robust and efficient algorithm is still on the way. In this paper, we have proposed a fast cascaded random-forest regression face-alignment method, based on a novel, simple, but effective, feature, which can achieve state-of-the-art performances, in terms of alignment accuracy and computational efficiency.

We have also addressed the fact that cascaded regression approaches are sensitive to shape initialization. Rather than using a number of blind initializations at the training stage, we propose to initialize shapes, using similar shape samples spanned in different subspaces. With this training strategy, the cascaded regression approach is capable of learning more accurate alignment trajectory and further improving the generalization capability of the trained forests, which result in higher accuracy in the prediction stage.

IDF is a very efficient feature for face alignment, since it is derived from the efficient pixel-based shape-indexed feature. However, limited by the capacity of pixel-based features, it is more susceptible to image noise, compared to the manually crafted SIFT-based methods. Furthermore, due to the limitation of the random-forest-based cascade regression framework, further work is necessary to tackle these problems for faces with large pose and occlusion.

References

- [1] Shaoqing Ren, Xudong Cao, Yichen Wei, Jian Sun, Face Alignment at 3000 FPS via Regressing Local Binary Features, CVPR, 2014
- [2] Xiong, X., De la Torre, F, Supervised descent method and its applications to face alignment. CVPR-2013.
- [3] T. F. Cootes and C. J. Taylor. Active shape models — "smart snakes". In BMVC, pp. 266–275, 1992.
- [4] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In ECCV, vol. 1407, pp. 484–498, 1998. 1,
- [5] M. Dantone, J. Gall, G. Fanelli, and L. Van Gool, Real-time Facial Feature Detection using Conditional Regression Forests, CVPR, 2012
- [6] Huiling Zhou, Kin Man Lam, and Xiangjian He, Shape-Appearance-Correlated Active Appearance Model, Pattern Recognition, vol. 56, pp. 88-99, August 2016.
- [7] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. Journal of Machine Learning Research(JMLR), 9:1871–1874, 2008
- [8] P. Dollár, P. Welinder, and P. Perona. Cascaded pose regression. In CVPR, pp. 1078–1085, 2010.
- [9] S. W. Chew, P. Lucey, S. Lucey, J. M. Saragih, J. F. Cohn, and S. Sridharan. Person-independent facial expression detection using constrained local models. In FGR, pp. 915–920, 2011.
- [10] H. Gao, H. K. Ekenel, and R. Stiefelhagen. Pose normalization for local appearance-based face recognition. ICB-2009.
- [11] N. Wang, D. Tao, X. Gao, X. Li, and J. Li. A comprehensive survey to face hallucination. Journal of Computer Vision, 106(1):9–30, 2014.
- [12] P. Viola, and M. Jones, Robust Real-Time Face Detection, IJCV- 2004.
- [13] Xudong Cao et al., Face Alignment by Explicit Shape Regression, IJCV-2014
- [14] Leo Breiman. Random Forests. ML, 45(1):5–32, 2001.
- [15] V. Kazemi and J. Sullivan, One millisecond face alignment with an ensemble of regression trees, CVPR-2014
- [16] Samuel Schuster, Paul Wohlhart, Christian Leistner, Amir Saffari, Peter M. Roth, Horst Bischof, Alternating Decision Forests, CVPR-2013
- [17] F. Moosmann, B. Triggs, and F. Jurie. Fast discriminative visual codebooks using randomized clustering forests. In NIPS, 2006.
- [18] Samuel Schuster, Christian Leistner, and Horst Bischof, Fast and Accurate Image Upscaling with Super-Resolution Forests, CVPR-2015
- [19] Jordi Salvador, Eduardo Pérez-Pellitero, Naive Bayes Super-Resolution Forest, ICCV-2015
- [20] P. Belhumeur, D. Jacobs, D. Kriegman and N. Kumar, Localizing parts of faces using a consensus of exemplars, CVPR-2011.
- [21] V. Le, J. Brandt, Z. Lin, L. Boudev and T. S. Huang, Interactive Facial Feature Localization, ECCV-2012.
- [22] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, CVPR-2005, 886–893.
- [23] T. Ahonen, A. Hadid, M. Pietikainen, Face description with local binary patterns: application to face recognition, TPAMI 28 (12) (2006) 2037–2041.
- [24] D. Cristinacce and T. F. Cootes. Feature detection and tracking with constrained local models. In Proceedings of the British Machine Vision Conference, pages 95.1–95.10. BMVA Press, 2006.
- [25] D. Cristinacce and T. Cootes, Automatic feature localization with constrained local models, Journal of Pattern Recognition, vol. 41, no. 10, pp. 3054–3067, 2008.
- [26] I. Matthews and S. Baker, Active appearance models revisited, IJCV, vol. 60, no. 2, pp. 135–164, 2004.
- [27] Baltrusaitis T, Robinson P, Morency L P. Constrained local neural fields for robust facial landmark detection in the wild, Proceedings of the IEEE International Conference on Computer Vision Workshops. 2013: 354-361
- [28] J. Saragih, S. Lucey, and J. Cohn. Deformable Model Fitting by Regularized Landmark Mean-Shift. IJCV, 2011.
- [29] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. IJCV, 56(3):221–255, 2004. 7.
- [30] C.-J. Lin, R. C. Weng, and S. S. Keerthi. Trust region Newton method for large-scale logistic regression. JMLR-2008