A Study on Face Alignment Algorithms

Anonymous CVPR submission

Paper ID LL

Abstract

This paper studies recent development in face alignment. We summarize and critique three regression based face alignment algorithms. We also implement the supervised descent method. Our implementation gives reasonable results compared to the one reported by the author.

1. Introduction

Face alignment, also known as facial feature detection, refers to locating p facial landmarks $\mathbf{x} \in \Re^p$ (such as eye brows and corners of mouth) in a face image $\mathbf{I} \in \Re^m$ of m pixels. Recently, regression based face alignment is showing promising results. Basically, those methods try to find a series of regression matrix $\mathbf{R}^k, k = 1, 2, \cdots$ that maps the image data to the movement of landmarks $\Delta\mathbf{x}^k$ such the initial landmarks $\mathbf{x}^0$ are refined progressively by $\mathbf{x}^k = \mathbf{x}^{k-1} + \Delta\mathbf{x}^k$ to find the optimal landmarks locations $\mathbf{x}^*$. In this paper, we review three regression based methods[14][10][13] and implement one of them[14].

2. Summarize and Critique

2.1. SDM:Supervised descent method

2.1.1 Problem formulation and algorithm

Supervised descent method[14] formulates the face alignment problem as a minimization problem

$$\Delta\mathbf{x}^* = \underset{\Delta\mathbf{x}}{\operatorname{argmin}} f(\Delta\mathbf{x}) = \underset{\Delta\mathbf{x}}{\operatorname{argmin}} \|\mathbf{h}(\mathbf{I}(\mathbf{x}_0 + \Delta\mathbf{x})) - \phi_*\|_2^2 \tag{1}$$

where $\mathbf{h}(\mathbf{I}(\mathbf{x}))$ is the SIFT[9] feature extraction function that calculates feature values of image $\mathbf{I}$ at landmarks $\mathbf{x}$ and $\phi_* = \mathbf{h}(\mathbf{I}(\mathbf{x}^*))$ denotes the SIFT values at the ground truth landmarks $\mathbf{x}^*$.

One classic method to solve minimization problems is the Newton's method [3]. Yet Newton's method requires to compute the inverse of the Hessian of the function, which is not only computational expensive for large data size but also infeasible for non-differentiable functions, such as the SIFT operator here.

To overcome the limitation of Newton's method, instead of calculating the descent direction using the Hessian, supervised descent method learns such descent directions from training data. More specifically, the algorithm learns a sequence of mapping matrix $\mathbf{R}_k$ that maps the local features ϕ at landmark $\mathbf{x}$ to the motion of landmarks $\Delta\mathbf{x}$ from training dataset $\{\mathbf{I}_i\}$ with known landmarks $\{\mathbf{x}_i^*\}$

$$\underset{\mathbf{R}^k, \mathbf{b}^k}{\operatorname{argmin}} \sum_{\mathbf{I}_i} \sum_{\mathbf{x}_k^i} \|\Delta\mathbf{x}_*^{ki} - \mathbf{R}^k\phi_i^k - \mathbf{b}^k\|^2 \tag{2}$$

and updates the location of landmarks by

$$\mathbf{x}^k = \mathbf{x}^{k-1} + \mathbf{R}^{k-1}\phi^{k-1} + \mathbf{b}^{k-1} \tag{3}$$

where $\Delta\mathbf{x}_*^{ki} = \mathbf{x}_i^* - \mathbf{x}^{ki}$ is the displacement between the ground truth landmarks and the estimated landmarks at the k^{th} stage for image i.

2.1.2 Evaluation and results

The algorithm is first validated on simple analytic functions and compared to Newtons method. Then the algorithm was tested on the LFPW datasets[2][12] and evaluated by the distance between the landmarks found by the algorithm and the ground truth landmarks provided. Finally the algorithm was tested for facial feature tracking on video dataset[1][7] where the algorithm tries to detect facial landmarks in each frame.

Validation on analytic function shows the algorithm converges faster than Newton's method experimentally. Results on the LFPW dataset show the supervised descent method outperforms two other recently proposed methods and the algorithm is also reported to work well on video data (though no quantitative results are provided). The method can handle images with illumination and pose changes pretty well. Yet for face images with extreme position change (such as a side face) and sever occlusion, the algorithm shows large error as compared to the ground truth.

2.2. LBF:Local binary features regression

2.2.1 Problem formulation and algorithm

SDM uses the classic SIFT method for feature extraction. LBF[10] proposes to improve the accuracy of face alignment by learning local features from training data instead. The argument is that such features are more adaptive to specific task. By learning features from a local region the method can take advantage of more discriminative features and save computational effort by learning each feature independently in a local region. The noises in the learned features are suppressed by using a global regression that maps all learned features together to give the optimal motion of landmarks,

To learn for the local features, the method takes a region around the lth landmarks and learns the local features ϕ_l in the region by solving

$$\underset{\mathbf{R}_l^k,\phi_l^k}{\operatorname{argmin}} \sum_{\mathbf{I}_i} \|\pi_l \circ \Delta\mathbf{x}_*^{ki} - \mathbf{R}_l^k \phi_l^k(\mathbf{I}_i, \mathbf{x}_i^k)\|_2^2 \qquad (4)$$

where $\pi_l\circ$ denotes the operator that gets the l th element of $\Delta\mathbf{x}_*^{ki}$. Comparing to Eq(2) in the supervised descent method, this method jointly learns a local feature ϕ_l^k and a local mapping $\mathbf{R}_l^k$ instead of using SIFT to calculate ϕ^k for all landmark locations and learning a global mapping $\mathbf{R}^k$ directly.

The author solves the learning problem in Eq(4) using the standard regression random forest[4]. During testing, each sample will traverse the trees until it reaches on leaf node for each tree. Each dimension in ϕ_l^k is 1 if the sample reaches the corresponding leaf node and 0 otherwise, resulting in a binary local feature that is highly sparse.

To suppress noise in the local feature, the algorithm discards the local mapping function $\mathbf{R}_l^k$ learns in Eq(4) and instead learns a global mapping $\mathbf{R}^k$ by solving the problem that is similar with Eq(2), except that the features used is the learned local binary features rather than SIFT features and a L2 regularization on $\mathbf{R}^k$ is needed as the learned local features have a dimension much higher than SIFT. However, due to the sparse nature of the features, the problem can be solved easily using a dual coordinate descent method[6].

2.2.2 Evaluation and results

The algorithm is evaluated on the LFPW datasets [2][12]the same as the one used in SDM and two other datasets, "Hellen"[8] and "300-W"[11] that have richer images and landmarks. The results show that the algorithm achieves comparable accuracy on the first two datasets as compared to other methods including SDM and higher accuracy on the third dataset which includes the first two as well as other challenging images. The paper shows sample results of both SDM and LBF. SDM fails on some images due to large pose variation while LBF detects most landmarks correctly. However, for images with poor illumination condition or unusual expressions (such as a widely opened mouth), both SDM and LBF fail.

Also, the speed of different algorithms is compared and the proposed method is the fastest method that runs at 300+ frames per second (FPS). The author further evaluates a faster version with smaller feature size which runs at 3000 FPS while achieving comparable accuracy as compared to other methods.

2.3. HPO:Face alignment under poses and occlusion

Both SDM and LBF treat all landmarks equally without considering occlusions and head poses explicitly. To handle occlusion, HPO [13] proposes to take the visibility of landmarks into account and learns a model to predict the probability of landmark visibility. Such model is then used to assist the face alignment by adding the visibility information to local features and shape configuration around a landmark.It also proposes to unify pose variation and occlusion by treating invisible landmarks due to pose change as a special case of occlusion.

First, the algorithm learns a function from training dataset as well as synthetic data that gives a prior probability of possible occlusion patterns. For example, occlusion usually happens in a block manner (Fig 1 a) where an entire block of the face is missing while it is very unlikely to have a face image where every other pixel is occluded (like a checkboard in Fig 1 b). Fig 1 shows example synthetic data that models possible occlusion patterns and another less likely block patterns. The probability function $Loss(\mathbf{c})$, where $\mathbf{c}$ represents the probability of each possible occlusion patterns to occur in an image of m pixels and it is a vector of length 2^m. After learning $Loss(\mathbf{c})$, the probability of landmark visibility is learned by optimizing

$$\begin{aligned} \underset{\Delta\mathbf{p}^k, T^k}{\operatorname{argmin}} \quad & \|\Delta\mathbf{p}^k - T^k\phi(\mathbf{I}, \mathbf{x}^{k-1})\|_2^2 + \lambda\mathbb{E}_{\mathbf{p}^k}[Loss(\mathbf{c})] \\ \text{s.t.} \quad & \mathbf{p}^k = \mathbf{p}^{k-1} + \Delta\mathbf{p}^k \end{aligned} \qquad (5)$$

The algorithm solves the problem in Eq(5) iteratively by optimizing the objective function with respect to $\Delta\mathbf{p}$ and T. Solving for T is a simple least square problem while solving for $\Delta\mathbf{p}$ is more complicated and gradient descent method is used.

Given the predicted landmark visibility probabilities, the algorithm updates the landmark locations by learning the descent direction $\mathbf{R}^k$ that maps local image features to movement of landmarks. For occluded landmarks, the region around them is not facial part thus the local appearance around them should be less useful in predicting landmark locations. Therefore, the local features around landmark $\mathbf{x}^k$

(a) Natural Occlusion (b) Checkboard-like Occlusion

Figure 1: Example synthetic data shows different occlusion patterns (gray blocks). Small numbers indicate facial landmarks.

is weighed by the corresponding visibility probability $\mathbf{p}^k$. Mathematically, this modifies Eq(2) to

$$\underset{\mathbf{R}^k,\mathbf{b}^k}{\operatorname{argmin}} \sum_{\mathbf{I}_i} \sum_{\mathbf{x}_k^i} \|\Delta \mathbf{x}_*^{ki} - \mathbf{R}^k[\mathbf{p}^k \star \phi_i^k]\|^2 \qquad (6)$$

where the $\star$ operator denotes the product between the visibility probability and the feature vector around each of the landmark.

2.3.1 Evaluation and Results

The algorithm was evaluated on three datasets, the LFPW dataset[2][12] which is used by both SDM and LBF, the Helen dataset[8] used by LBF and a new dataset that has more occluded faces and larger pose variations[5]. The proposed algorithm achieves comparable accuracy in terms of the distance between ground truth and detected landmarks on the first two datasets while outperforming SDM and LBF on the third dataset. In general, the algorithm is able to detect occluded landmarks and handle pose variance better than SDM . However, the algorithm still fails for extreme head poses. Besides, example images of occlusion detection shown in the paper all have a nearly fronted pose, which should reduce the complexity of occlusion detection. These observations suggest that the power of the occlusion modeling and the idea of viewing side faces as a case of occlusion is still limited.

In terms of computational efficiency, the algorithm is slower than the previous two methods. The proposed algorithm runs at 2 FPS while the SDM and LBF runs at 160 and 460 FPS respectively on the LFPW data base.

2.4. Discussion and Comparison

The three algorithms all aim at solving face alignment problem using regression method and achieve promising results. Table 1 summarizes the quantitative results (mean absolute error/FPS) obtained by the 3 algorithms on different datasets, "NA" indicates no result reported.

	LFPW	Helen	300-W	COFW
SDM	3.49/160	5.85/21	7.52/70	6.69/NA
LBF	3.35/460	5.41/200	6.32/320	NA
HPO	3.93/2	5.49/2	NA	5.18/2

Table 1: Summary of quantitative results of different algorithms

SPM proposes a general frame work to solve face alignment as an optimization problem without calculating the inverse of Hessian. The method is general to handle different feature extraction functions as it does not impose any specific requirement such as differentiability. Also, the learning process requires solving a simple least square problem only and appears to have a very fast convergence rate (4 or 5 steps) experimentally.

However, the underlying assumption for $\mathbf{R}_k$ to be the optimal descent directions is that the entire dataset represents a set of similar functions with similar descent directions. Under this assumption, it is possible to average over the training dataset and learn the descent directions that are optimal for both training and testing data. If this assumption does not hold, the algorithm can fail easily. Consider a simple case where we sample two points of a function that have opposite descent direction as training samples, averaging over them will learn a 0 direction that leads the optimization no where. This should also be the reason that the algorithm fails on faces with extreme poses. Therefore, it is very important to preprocess the dataset to make sure the dataset is homogeneous. Details of such preprocessing will be described in the experimental section.

LBF extends the SDM by learning local binary features, which contributes to both computational efficiency and accuracy of the algorithm. On the other hand, learning local features also adds on model complexity and requires more parameter tuning, such as the size of the local region for the local feature learning process, which varies at different training stages. The author proposes to perform cross-validation to determine the optimal model parameters yet it is not clear in the paper how stable the parameters are across different datasets and how robust the method is to different parameters.

HPO aims at resolving variation of head poses and occlusions, which is a major challenge in face alignment and proposes to view the two problems in a unified way. The results show the method's efficacy in detecting occluded points and various poses yet the power still seems limited for some extreme cases.

3. Implementation of SDM

3.1. Face Detection with OpenCV

SDM algorithm requires predefined face location. The face is detected with OpenCV's Haar featured-based cascade classifier. The face detection rate is 82.3% on LFPW dataset. We discarded those images that cannot be detected and ended up with 434 training images and 120 test images. The output from OpenCV's face detector is a rectangle which contains the region of the face. The face rectangle will later be used as a reference to normalize the dataset and initialize the algorithm.

3.2. Calculate Mean Face

The face detection information gathered from the previous step is used to center and normalize the face images as well as the labeled landmarks. We normalized the face rectangle to 200 pixels by 200 pixels and take 100 pixels of padding from each side, since some of the labeled landmarks is outside the face rectangle. We then used the normalized landmarks position to generate the initial mean face by calculating the mean position value of each landmark from all images in the training dataset. This initial mean face will be used in the initialization step of both learning and predicting stage.

3.3. Learning Stage

3.3.1 Aligned Landmarks with the Face Position

The first step of SDM is generating the initial estimation of facial landmarks based on the mean face shape we calculated in previous step and the variance of scale and translation of training face images. In our implementation, we first resized the image based on the face size detected by OpenCVs face detector; therefore initial face shape will be the same across different images.

3.3.2 Extract HoG features from Landmarks

The next step is extracting the HoG features from patches around the landmarks with size 32 pixels by 32 pixels. The HoG features is equivalent to the SIFT features of fixed scale and predefined local patch as described in the original paper. The HoG features Given an image $\mathbf{d} \in \Re^{m\times 1}$ of m pixels and $\mathbf{x} \in \Re^{p\times 1}$ indexes p landmarks in the image, we generated a feature vector $\phi(\mathbf{d}^i(x)) \in \Re^{128\times 1}$ per landmark from the patch. The generated features for each landmarks of the image is than concatenate into a vector $\phi(\mathbf{d}^i(\mathbf{x})) \in \Re^{128p\times 1}$, where p is the number of landmarks. For the purpose of matrix calculation in MATLAB, we concatenate all feature vectors in the training dataset into a matrix $\Phi \in \Re^{n\times 128p}$, where n is the number of images in training dataset and each row of the matrix is the transpose of corresponding feature vector of that image.

$$\Phi = \begin{bmatrix} \phi(\mathbf{d}^1(\mathbf{x}))^T \\ \phi(\mathbf{d}^2(\mathbf{x}))^T \\ \phi(\mathbf{d}^3(\mathbf{x}))^T \\ \vdots \\ \phi(\mathbf{d}^{n-1}(\mathbf{x}))^T \\ \phi(\mathbf{d}^n(\mathbf{x}))^T \end{bmatrix} \tag{7}$$

3.3.3 Perform SDM Regression

The learning for SDM follows the Eq(2) and Eq(3). The landmarks are predicted by a generic linear combination. The SDM will learn a sequence of generic descent direction R_k and bias term b_k. To learn the generic descent direction R_k and bias term b_k, we minimized Eq(2), which is a well-known linear least squares problem which can be solved in closed form. To reformulate the equation in a matrix form, we got

$$\mathbf{D} = \Phi\mathbf{A} \tag{8}$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{b}^k \\ \mathbf{R}^k \end{bmatrix} \tag{9}$$

$$\mathbf{D} = \begin{bmatrix} \Delta\mathbf{x}^1 \\ \Delta\mathbf{x}^2 \\ \Delta\mathbf{x}^3 \\ \vdots \\ \Delta\mathbf{x}^{n-1} \\ \Delta\mathbf{x}^n \end{bmatrix}, \Phi = \begin{bmatrix} 1 & \phi^1 \\ 1 & \phi^2 \\ 1 & \phi^3 \\ \vdots & \vdots \\ 1 & \phi^{n-1} \\ 1 & \phi^n \end{bmatrix} \tag{10}$$

where $D \in \Re^{n\times 2p}$, $\Phi \in \Re^{n\times(128p+1)}$and $A \in \Re^{(128p+1)\times 2p}$. The dimension of D is $n \times 2p$ is because we concatenate the x and y coordinate of the landmarks into a row vectors, thus the length becomes two times the number of landmarks. The close form solution for Eq(8) is

$$\mathbf{A} = (\Phi^T\Phi)^{-1}\Phi^T\mathbf{D} \tag{11}$$

One important implementation detail to notice is that to perform the learning, the landmarks position need to be normalized to a region of an unit square, i.e.$\Delta\hat{\mathbf{x}}^k = \Delta\mathbf{x}^k/s$,where s is calculated as the distance between the minimal and maximal coordinates among all landmarks $\mathbf{x}^k$.

3.3.4 Recalculate the updated landmarks

After we learned the new $\mathbf{R}^k$ and $\mathbf{b}^k$, we can use these parameters to predict the new landmarks position by computing Eq(3). Since we normalized the landmark position to a region of an unit square, we need to remapping the $\Delta\mathbf{x}^k$ from Eq (1). back to its original dimension.

3.3.5 Iterate the Learning Process

The new landmark position we got from 3.3.4 are now the new initial land marks, and we can repeat the learning process from 3.3.2 to 3.3.4 several times, and store the $\mathbf{R}^k$ and $\mathbf{b}^k$ value in each iteration. These values will be used in the predicting stage later. The learning error decreases monotonically, and the algorithm converges in less than five iterations.

3.4. Predicting Stage

The steps in predicting stage is very similar to the learning stage. Instead of learning the $\mathbf{R}$ and $\mathbf{b}$ parameter in each interation, we used the parameter $\mathbf{R}$ and $\mathbf{b}$ learned from training data. The mean face we got from training data was used to generate the initial landmark positions in the testing stage. Eq(2) was used to compute the predicted landmark position. Notice that the landmark positions also need to be normalized to a region of an unit square when calculating the $\Delta\mathbf{x}$.

Algorithm 1 SDM Learning Algorithm

function REGRESS($imgs, marks, ans$)
  $marks \leftarrow normalized(marks)$
  $\Phi \leftarrow Features(imgs, tmpMarks)$
  $\Delta\mathbf{x} \leftarrow ans - marks$
  $\Phi \leftarrow AddBias(\Phi)$
  $R \leftarrow (\Phi^T\Phi)^{-1}\Phi^T\Delta x$
  $\Delta\mathbf{x} \leftarrow \Phi R$
  $marks \leftarrow marks + Remap(\Delta\mathbf{x}, faceRects)$
  return $R, marks$

function SDM TRAIN($imgs, marks, ans$)
  for k = 1 : 5 **do**
    if First Iteration **then**
      $faceRects \leftarrow FaceDetect(imgs)$
      $marks \leftarrow MeanFace(faceRects)$
    $[R[k], marks] \leftarrow Regress(imgs, marks, ans)$
  return R

4. Evaluation and Results

4.1. Dataset

In our experiment, we used the LFPW dataset the same as the one in the paper. LFPW consists of 1,432 faces from images download from the web and each image is labeled with 35 fiducial landmarks by three MTurk workers. The 35 fiducial landmarks mark eyebrows (4 landmarks each), eyes (5 landmarks each), nose (4 landmarks), mouse (5 landmarks) and chin (1 landmark). We discarded the 6 landmarks which labeled ears since ears are often covered by hair in our dataset; therefore, 29 landmarks were used. Fig 2 shows the 29 labeled landmarks on an example image. Due to the copyright issue, the dataset only contains URL to the labeled images. Most of the URLs are no longer valid. We downloaded 527 of the 1132 training images and 146 of the 300 test images.

Algorithm 2 SDM Predicting Algorithm

function REGRESS($imgs, R, marks$)
  $marks \leftarrow normalized(marks)$
  $\Phi \leftarrow Features(imgs, tmpMarks)$
  $\Phi \leftarrow AddBias(\Phi)$
  $\Delta\mathbf{x} \leftarrow \Phi R$
  $marks \leftarrow marks + Remap(\Delta\mathbf{x}, faceRects)$
  return $marks$

function SDM PREDICT($imge, R$)
  for cascade = 1 : 5 **do**
    if First Iteration **then**
      $faceRects \leftarrow FaceDetect(imgs)$
      $marks \leftarrow MeanFace(faceRects)$
    $marks \leftarrow Regress(imgs, R[k], marks)$
  return $marks$

Figure 2: 29 Labeled Landmarks

4.2. Results

The author reported a mean error ($\times 10^{-2}$) of 3.47. Our implementation shows a mean error($\times 10^{-2}$) of 4.95. Considering that our dataset contains only one half of the images the original authors used, the difference between our result and the original authors' is acceptable. The smaller dataset might make our learning model less representative and more vulnerable to the face variance, such as the aspect ratio of the face, the distance between different organs, and the orientation of the face, hence increase the alignment error. Fig 5a shows an example of initial landmarks with mean face normalized using the face detector and Fig 5b shows the predicted landmarks using the SDM algorithm. And the second row of Fig 3 demonstrates that the SDM algorithm's prediction is robust under different light condi-

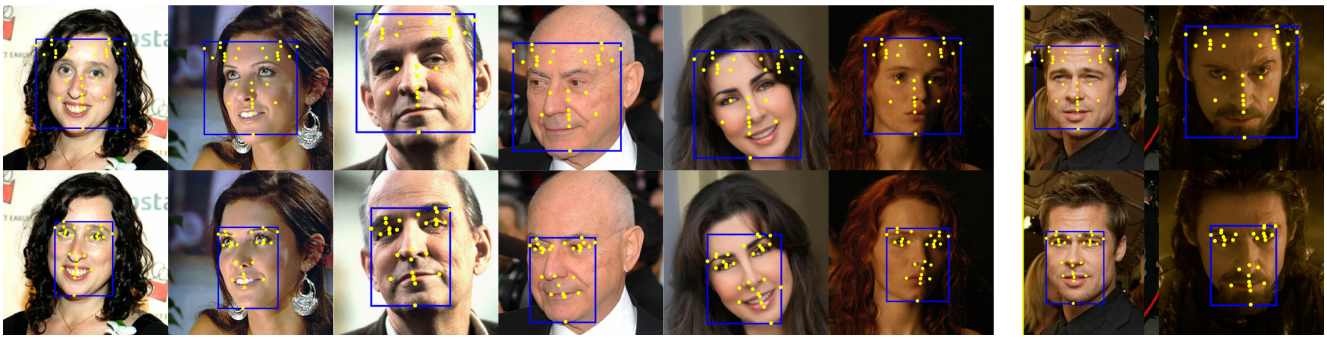
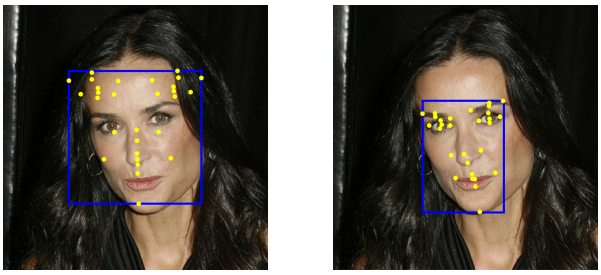


Figure 3: This top row shows the initial guess of the face landmarks. The bottom shows the predicted landmarks from SDM algorithm after 5 iterations.

Figure 4: Example results on the YouTube Celebrities dataset.



(a) Initial mean face (b) Predicted landmarks

Figure 5: a) The initial mean face normalized using face detector. b) The predicted landmarks after five iterations

tion, and a large variation in poses. Fig 4 shows example of tracking result on YouTube Celebrities dataset. The result video can be viewed at https://www.youtube.com/watch?v=JCIR_BmhGfY. The processing time for the SDM algorithm is around 0.055 seconds per frame.

5. Conclusion

In this paper, we summarize and compare three regression based face alignment algorithm. All three method reports promising results. The first algorithm proposes a general framework to learn regressor and the remaining two algorithms extend the first one in terms of accuracy and robustness to occlusion. Our implementation of the first algorithm reproduces its experiments on both image and video data and achieves comparable result to the one reported.

References

[1] M. S. Bartlett, G. C. Littlewort, M. G. Frank, C. Lainscsek, I. R. Fasel, and J. R. Movellan. Automatic recognition of facial actions in spontaneous expressions. *Journal of multimedia*, 1(6):22–35, 2006. 1

[2] P. N. Belhumeur, D. W. Jacobs, D. J. Kriegman, and N. Kumar. Localizing parts of faces using a consensus of exemplars. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(12):2930–2940, 2013. 1, 2, 3

[3] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84, 1998. 1

[4] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. 2

[5] X. P. Burgos-Artizzu, P. Perona, and P. Dollár. Robust face landmark estimation under occlusion. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1513–1520. IEEE, 2013. 3

[6] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008. 2

[7] M. Kim, S. Kumar, V. Pavlovic, and H. Rowley. Face tracking and recognition with visual constraints in real-world videos. In *Computer Vision and Pattern*

Recognition, 2008. CVPR 2008. IEEE Conference on, pages 1–8. IEEE, 2008. 1

[8] V. Le, J. Brandt, Z. Lin, L. Bourdev, and T. S. Huang. Interactive facial feature localization. In *Computer Vision–ECCV 2012*, pages 679–692. Springer, 2012. 2, 3

[9] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999. 1

[10] S. Ren, X. Cao, Y. Wei, and J. Sun. Face alignment at 3000 fps via regressing local binary features. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1685–1692. IEEE, 2014. 1, 2

[11] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic. A semi-automatic methodology for facial landmark annotation. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on*, pages 896–903. IEEE, 2013. 2

[12] J. Saragih. Principal regression analysis. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2881–2888. IEEE, 2011. 1, 2, 3

[13] Y. Wu and Q. Ji. Robust facial landmark detection under significant head poses and occlusion. In *Proc. Int. Conf. Comput. Vision. IEEE*, volume 1, 2015. 1, 2

[14] X. Xiong and F. De la Torre. Supervised descent method and its applications to face alignment. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 532–539. IEEE, 2013. 1