Int J Comput Vis
DOI 10.1007/s11263-017-1012-z

Pose-Invariant Face Alignment via CNN-Based Dense 3D Model Fitting

Amin Jourabloo[1] · **Xiaoming Liu**[1]

Received: 12 June 2016 / Accepted: 6 April 2017

Abstract Pose-invariant face alignment is a very challenging problem in computer vision, which is used as a prerequisite for many facial analysis tasks, e.g., face recognition, expression recognition, and 3D face reconstruction. Recently, there have been a few attempts to tackle this problem, but still more research is needed to achieve higher accuracy. In this paper, we propose a face alignment method that aligns an image with arbitrary poses, by combining the powerful cascaded CNN regressors, 3D Morphable Model (3DMM), and mirrorability constraint. The core of our proposed method is a novel 3DMM fitting algorithm, where the camera projection matrix parameters and 3D shape parameters are estimated by a cascade of CNN-based regressors. Furthermore, we impose the mirrorability constraint during the CNN learning by employing a novel loss function inside the siamese network. The dense 3D shape enables us to design pose-invariant appearance features for effective CNN learning. Extensive experiments are conducted on the challenging large-pose face databases (AFLW and AFW), with comparison to the state of the art.

Keywords Pose-invariant face alignment · CNN · Cascaded regressor · Dense model fitting · Mirrorability constraint

Communicated by Xiaoou Tang.

✉ Xiaoming Liu
liuxm@cse.msu.edu

Amin Jourabloo
jourablo@msu.edu

[1] Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824, USA

1 Introduction

Face alignment is the process of aligning a face image and detecting a set of fiducial points, such as mouth corners, nose tip, etc. Face alignment is a key module in the pipeline of most facial analysis tasks, normally after face detection and before subsequent feature extraction and classification. As a result, improving the face alignment accuracy is helpful for numerous facial analysis tasks, e.g., face recognition (Wagner et al. 2012), face de-identification (Jourabloo et al. 2015) and 3D face reconstruction (Roth et al. 2015, 2016).

Due to its importance, face alignment has been well studied during past decades (Wang et al. 2014), with the well-known Active Shape Model (Cootes et al. 1994) and Active Appearance Model (AAM) (Matthews and Baker 2004; Liu 2009). Recently, face alignment works are very popular in top vision venues and achieve a lot of attentions. The existing approaches can be categorized into three groups: Constrained Local Model-based approaches (Cootes et al. 1994; Saragih et al. 2009), AAM-based approaches (Matthews and Baker 2004; Liu 2009, 2010) and regression-based approaches (Valstar et al. 2010; Cao et al. 2014c; Zhang et al. 2008). In spite of the fruitful prior work and ongoing progress of face alignment (e.g., the latest impressive iBUG results (Tzimiropoulos 2015) and the first 3D face alignment challenge (Jeni et al. 2016), face alignment for large-pose faces is still very challenging and there is only a few published work in this direction, as summarized in Table 1. Therefore, this is a clear research gap that needs to be addressed, which is exactly the focus of this work.

To tackle pose-invariant face alignment, our technical approach is motivated by the need to address the inherent *challenges* associated with this problem, Fig. 1. First of all, faces have different numbers of visible landmarks under pose variation, and the spatial distribution of the landmarks is

Published online: 19 April 2017

Table 1 The comparison of pose-invariant face alignment methods

Method	Dense 3D model fitting	Visibility	Database	Pose range	Training face #	Testing face #	Landmarks #	Estimation errors
RCPR (Burgos-Artizzu et al. 2013)	No	Yes	COFW	Frontal w. occlu.	1345	507	29	8.5
Wu and Ji (Wu and Ji 2015)	No	Yes	COFW, FERET	Frontal w. occlu., profile	1345; >2608	507; 465	29; 11	5.9;–
TSPM (Zhu and Ramanan 2012)	No	No	AFW	All poses	2118	468	6	11.1
CDM (Yu et al. 2013)	No	No	AFW	All poses	1300	468	6	9.1
RTSM (Hsu et al. 2015)	No	No	AFW	All poses	1100	468	6	–
TCDCN (Zhang et al. 2014b)	No	No	AFLW, AFW	[-60°, 60°]	10,000	3000; ∼ 313	5	8.0; 8.2
PIFA (Jourabloo and Liu 2015)	No	Yes	AFLW, AFW	All poses	3901	1299; 468	21.6	6.5; 8.6
Proposed method	Yes	Yes	AFLW, AFW	all poses	3901	1299; 468	34.6	4.5; 7.4

Fig. 1 The proposed method estimates landmarks for large-pose faces by fitting a dense 3D shape. From *left* to *right*: initial landmarks, fitted 3D dense shape, estimated landmarks with visibility. The *green/red/yellow dots* in the *right column* show the visible/invisible/cheek landmarks, respectively (Color figure online)

highly pose dependent. This presents challenges for existing face alignment approaches since most of them are based on 2D shape models, which inherently have difficulty in modeling the 3D out-of-plane deformation and handling large-pose face alignment. In contrast, given the fact that a face image is a projection of a 3D face, we propose to use a dense 3D Morphable Model (3DMM) to reconstruct the 3D shape of face and the projection matrix as the *latent representation* of a 2D face shape. Therefore, face alignment amounts to estimating this representation, i.e., performing the 3DMM fitting to a face image with *arbitrary* poses.

Second, our approach to large-pose face alignment is through 3DMM fitting. However, the classic analysis-by-synthesis-based optimization approach for 3DMM fitting is not only inefficient (e.g., 40 seconds per image in Amberg et al. (2008)), but also based on the assumption that the 2D landmarks are provided either manually or with a separate face alignment routine (Qu et al. 2015; Jeni et al. 2015), which conflicts with the goal of our work. This motivates us to employ the powerful cascaded regressor approach to learn the mapping from a 2D face image to its representation. Since the representation is composed of 3D parameters, the mapping is likely to be more complicated than the cascaded regressor in 2D face alignment (Cao et al. 2014c). Therefore, we propose to use Convolutional Neural Networks (CNN) as the regressor in the cascaded framework, to learn the mapping. While most prior work on CNN-based face alignment estimate no more than six 2D landmarks per image (Zhang et al. 2014b; Sun et al. 2013), our cascaded CNN can produce a substantially larger number (34) of 2D and 3D landmarks. Further, using landmark marching (Zhu et al. 2015b), our algorithm can adaptively adjust the 3D landmarks during the fitting, so that the local appearances around cheek landmarks contribute to the fitting process.

Third, conventional 2D face alignment approaches are often driven by the local appearance feature patch around each estimated 2D landmark. Even at the ground truth landmark, such as the outer eye corner, it is hard to assume that the local patches from faces at various poses cover the same area of facial skin *anatomically*, which poses additional challenge for the learning algorithm to associate a unified and distinctive pattern with the ground truth landmark. Fortunately, in our work, we can leverage the dense 3D face model as an oracle to build dense feature *correspondence* across various poses and expressions. Therefore, we propose two novel pose-invariant local features, as the input layer for CNN learning. We also utilize person-specific surface normals to estimate the visibility of each landmark by inspecting whether its surface normal has a positive z coordinate, and the estimated visibilities are dynamically incorporated into the CNN regressor learning such that only the extracted features from visible landmarks contribute to the learning.

Fourth, the CNN regressor deals with a very challenging learning task given the diverse facial appearance across all poses. To facilitate the learning task under large variations of pose and expression, we develop two new constraints to learn the CNN regressors. One is that, there is inherent ambiguity in representing a 2D face shape as the combination of the 3D shape and projection matrix. Therefore, in addition to regressing toward such a non-unique latent representation, we also propose to constrain the CNN regressor in its ability to directly estimate 2D face shapes. The other is that, a horizontally mirrored version of a face image is still a valid face and their alignment results should be the flip version of each other. In this work, we propose a CNN architecture with a new loss function that explicitly enforces these constraints. The new loss function minimizes the difference of face alignment results of a face image and its mirror, in a siamese network architecture (Bromley et al. 1993). Although this mirrorability constraint was an alignment accuracy measure used in post-processing (Yang and Patras 2015), we integrate it directly in CNN learning.

These algorithm designs collectively lead to the proposed pose-invariant face alignment algorithm. We conduct extensive experiments to demonstrate the capability of proposed method in aligning faces across poses on two challenging datasets, AFLW (Köstinger et al. 2011) and AFW (Zhu and Ramanan 2012), with comparison to the state of the art.

We summarize the main contributions of this work as:

- Pose-invariant face alignment by fitting a dense 3DMM, and integrating estimation of 3D shape and 2D facial landmarks from a single face image.
- The cascaded CNN-based 3D face model fitting algorithm that is applicable to all poses, with integrated landmark marching and contribution from local appearances around cheek landmarks during the fitting process.
- Dense 3D face-enabled pose-invariant local features and utilizing person-specific surface normals to estimate the visibility of landmarks.
- A novel CNN architecture with mirrorability constraint that minimizes the difference of face alignment results of a face image and its mirror.

The rest of this paper is organized as follow. In the next section prior works are summarized. Section 3 describes pose-invariant face alignment algorithm with two different CNN architectures and two types of pose-invariant local features. Experimental results are reported in Section 4. Conclusions and future directions are in Section 5.

The preliminary version of this work appears in Jourabloo and Liu (2016). We extend it in a number of ways: (i) proposed a new CNN architecture and new loss function for integrating mirrorability constraint in the training process of CNN; (ii) added synthetic face images to the training set; and (iii) compared implementation of proposed method with two different CNN toolboxes.

2 Prior Work

We review prior work in four areas related to the proposed method: pose-invariant face alignment, 3D face model fitting to a single image, face alignment via deep learning, and sharing information in face alignment and deep learning.

Pose-invariant face alignment The methods of Yu et al. (2013); Zhu and Ramanan (2012); Hsu et al. (2015) combines face detection, pose estimation and face alignment. By using a 3D shape model with optimized mixture of parts, (Yu et al. 2013) is applicable to faces with a large range of poses. In Wu and Ji (2015), a face alignment method based on cascade regressors is proposed to handle invisible landmarks. Each stage is composed of two regressors for estimating the probability of landmark visibility and the location of landmarks. This method is applied to profile-view faces of FERET database (Phillips et al. 2000). However, as a 2D landmark-based approach, it cannot estimate 3D face poses. Occlusion-invariant face alignment, such as RCPR (Burgos-Artizzu et al. 2013), may also be applied to handle large poses since non-frontal faces are one type of occlusions. Tulyakov and Sebe (2015) is a very recent work that estimates 3D landmark via regressors. However, it only tests on synthesized face images up to $\sim 50^\circ$ yaw. The most relevant prior work is Jourabloo and Liu (2015), which aligns faces of arbitrary poses with the assistant of a sparse 3D Point Distribution Model (PDM). The model parameter and projection matrix are estimated by the cascade of linear or non-linear regressors. We extend (Jourabloo and Liu 2015)

Table 2 The comparison of most recent 3D face model fitting methods

Method	Integrated 2D landmark	No of 2D landmarks	Testing database	Pose range	3D bases	Method
(Qu et al. 2015)	No	68	Basel	[-30°, 30°]	Basel bases	Adaptive contour fitting
(Jeni et al. 2015)	No	77 to 1024	BU-4DFE; BP-4DS; videos	[-60°, 60°]	Bases from BU-4DFE & BP-4DS	Cascaded regressor; EM
(Zhu et al. 2015c)	Yes	–	FRGC	Frontal	Basel bases	Cascaded regressor
Proposed method	Yes	–	AFW; AFLW	All poses	Basel bases	3D cascaded regressor

in a number of aspects, including *fitting a dense 3D morphable model, employing the powerful CNN as the regressor, using 3D-enabled features, estimating cheek landmarks and utilizing the mirror CNN architecture*. Table 1 compares the pose-invariant face alignment methods.

3D face model fitting Table 2 shows the most recent 3D face model fitting methods to a single image. Almost all prior works assume that the 2D landmarks of the input face image is either manually labeled or estimated via a face alignment method. In Jeni et al. (2015), a dense 3D face alignment from videos is proposed. At first, a dense set of 2D landmarks are estimated by using the cascaded regressor. Then, an EM-based algorithm is utilized to estimate the 3D shape and 3D pose of the face from estimated 2D landmarks. The authors in Qu et al. (2015) aim to make sure that the locations of 2D contour landmarks are consistent with the 3D face shape. In Zhu et al. (2015c), a 3D face model fitting method based on the similarity of frontal view face images is proposed. In contrast, our proposed method is the first approach to *integrate 2D landmark estimation as part of the 3D face model fitting for large poses*. Furthermore, all prior 3D face model fitting works process faces with up to 60° yaw while our method can handle all view angles.

Face alignment via deep learning With the continuous success of deep learning in vision, researchers start to apply deep learning to face alignment. Sun et al. (2013) proposed a three-stage face alignment algorithm with CNN. At the first stage, three CNNs are applied to different face parts to estimate positions of different landmarks, whose averages are regarded as the first stage results. At the next two stages, by using local patches with different sizes around each landmark, the landmark positions are refined. Similar face alignment algorithms based on multi-stage CNNs are further developed by Zhou et al. (2013) and CFAN (Zhang et al. 2014a). In Zhang et al. (2014a), a face alignment method based on cascade of stacked auto-encoder (SAE) networks can progressively refine locations of 2D landmarks at each stage. TCDCN (Zhang et al. 2014b) uses one-stage CNN to estimates positions of five landmarks given a face image. The commonality among most of these prior works is that they only estimate 2D landmarks and the number of landmarks is limited to 6. In comparison, our proposed method *employs CNN to estimate 3D landmarks, as the byproduct of the 3D surface reconstruction*. As a result, the number of estimated landmarks is bounded by the number of 3D vertexes, although the evaluation is conducted for 34 landmarks.

Sharing information in face alignment and deep learning Utilizing different side information in face alignment can improve the alignment accuracy. TCDCN (Zhang et al. 2014b) jointly estimates auxiliary attributes (e.g., gender, expression) with landmark locations to improve alignment accuracy. In Yang and Patras (2015), the mirrorability constraint, i.e., the alignment difference between a face and its mirrored counterpart, is used as a measure for evaluating the alignment results without the ground truth, and for choosing a better initialization. Consensus of occlusion-specific regressors (Yu et al. 2014) in a Bayesian model is used to share information among different regressors. In Zhu et al. (2015a) multiple initializations are used for each face image and a clustering method combines, the estimated face shapes. For deep learning methods, sharing information is performed either by transferring the learned weights from a source domain to the target domain (Yosinski et al. 2014), or by using the siamese networks (Zagoruyko and Komodakis 2015; Bell and Bala 2015) to share the weights among branches of the network and make a final decision with combined responses of all branches. *Compared to the prior work, the proposed method integrates the mirrorability constraint in a siamese network, which explicitly contributes to network learning, rather than as a post-processing metric* (Yang and Patras 2015). *The siamese network allows us to share information between face images and their mirrored ones. Feeding all local patches into CNN shares information among patches, rather than one local patch for one CNN* (Zhang et al. 2014b; Sun et al. 2013).

3 Unconstrained 3D Face Alignment

The core of our proposed 3D face alignment method is the ability to fit a dense 3D Morphable Model to a 2D face image with arbitrary poses. The unknown parameters of fitting, the

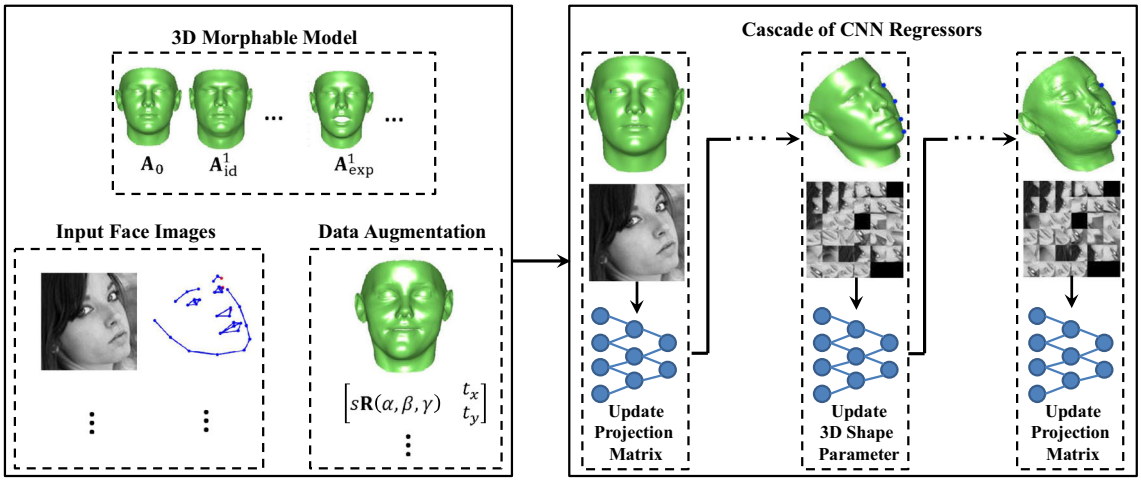


Fig. 2 The overall process of the proposed method

3D shape parameters and the projection matrix parameters, are sequentially estimated through a cascade of CNN-based regressors. By employing the dense 3D shape model, we enjoy the benefits of being able to estimate 3D shape of face, locate the cheek landmarks, use person-specific 3D surface normals, and extract pose-invariant local feature representation, which are less likely to achieve with a simple PDM (Jourabloo and Liu 2015). Figure 2 shows the overall process of the proposed method.

3.1 3D Morphable Model

To represent a dense 3D shape of an individual's face, we use 3D Morphable Model (3DMM),

$$\mathbf{S} = \mathbf{S}_0 + \sum_{i=1}^{N_{id}} p_{id}^i \mathbf{S}_{id}^i + \sum_{i=1}^{N_{exp}} p_{exp}^i \mathbf{S}_{exp}^i, \tag{1}$$

where $\mathbf{S}$ is the 3D shape matrix, $\mathbf{S}_0$ is the mean shape, $\mathbf{S}_{id}^i$ is the ith identity basis, $\mathbf{S}_{exp}^i$ is the ith expression basis, p_{id}^i is the ith identity coefficient, and p_{exp}^i is the ith expression coefficient. The collection of both coefficients is denoted as the shape parameter of a 3D face, $\mathbf{p} = (\mathbf{p}_{id}^\intercal, \mathbf{p}_{exp}^\intercal)^\intercal$. We use the Basel 3D face model as the identity bases (Paysan et al. 2009) and the face wearhouse as the expression bases (Cao et al. 2014b). The 3D shape $\mathbf{S}$, along with $\mathbf{S}_0$, $\mathbf{S}_{id}^i$, and $\mathbf{S}_{exp}^i$, is a $3 \times Q$ matrix which contains x, y and z coordinates of Q vertexes on the 3D face surface,

$$\mathbf{S} = \begin{pmatrix} x_1 & x_2 & \cdots & x_Q \\ y_1 & y_2 & \cdots & y_Q \\ z_1 & z_2 & \cdots & z_Q \end{pmatrix}. \tag{2}$$

Any 3D face model will be projected onto a 2D image where the face shape may be represented as a sparse set of N landmarks, on the facial fiducial points. We denote x and y coordinates of these 2D landmarks as a matrix $\mathbf{U}$,

$$\mathbf{U} = \begin{pmatrix} u_1 & u_2 & \cdots & u_N \\ v_1 & v_2 & \cdots & v_N \end{pmatrix}. \tag{3}$$

The relationship between the 3D shape $\mathbf{S}$ and 2D landmarks $\mathbf{U}$ can be described by using the weak perspective projection, i.e.,

$$\mathbf{U} = s\mathbf{R}\mathbf{S}(:, \mathbf{d}) + \mathbf{t}, \tag{4}$$

where s is a scale parameter, $\mathbf{R}$ is the first two rows of a 3×3 rotation matrix controlled by three rotation angles α, β, and γ (pitch, yaw, roll), $\mathbf{t}$ is a translation parameter composed of t_x and t_y, $\mathbf{d}$ is a N-dim index vector indicating the indexes of semantically meaningful 3D vertexes that correspond to 2D landmarks. We form a projection vector $\mathbf{m} = (s, \alpha, \beta, \gamma, t_x, t_y)^\intercal$ which collects all parameters to this projection. We assume the weak perspective projection model with six degrees of freedom, which is a typical model used in many prior face-related work (Xiao et al. 2004; Jeni et al. 2015).

At this point, we can represent any 2D face shape as the projection of a 3D face shape. In other words, the projection parameter $\mathbf{m}$ and shape parameter $\mathbf{p}$ can uniquely represent a 2D face shape. Therefore, the face alignment problem amounts to estimating $\mathbf{m}$ and $\mathbf{p}$, given a face image. Estimating $\mathbf{m}$ and $\mathbf{p}$ instead of estimating $\mathbf{U}$ is motivated by a few factors. First, without the 3D modeling, it is non-trivial to model the out-of-plane rotation, which has a varying number of landmarks depending on the rotation angle. Second, as pointed out by Xiao et al. (2004), by only using $\frac{1}{6}$ of the number of the shape bases, 3DMM can have an equivalent representation power as its 2D counterpart. Hence, using

3D model leads to a more compact representation of shape parameters.

Cheek landmarks correspondence The projection relationship in Eq. 4 is correct for frontal-view faces, given a constant index vector **d**. However, as soon as a face turns to the non-frontal view, the original 3D landmarks on the cheek become invisible on the 2D image. Yet most 2D face alignment algorithms still detect 2D landmarks on the contour of the cheek, termed "cheek landmarks". Therefore, in order to still maintain the 3D-to-2D correspondences as Eq. 4, it is desirable to estimate the 3D vertexes that match with these cheek landmarks. A few prior works have proposed various approaches to handle this (Qu et al. 2015; Zhu et al. 2015b; Cao et al. 2014a). In this paper, we leverage the landmark marching method proposed in Zhu et al. (2015b).

Specifically, we define a set of *paths* each storing the indexes of vertexes that are not only the most closest ones to the original 3D cheek landmarks, but also on the contour of the 3D face as it turns. Given a non-frontal 3D face **S**, by ignoring the roll rotation γ, we rotate **S** by using the α and β angles (pitch and yaw), and search for a vertex in each predefined path that has the maximum (minimum) x coordinate, i.e., the boundary vertex on the right (left) cheek. These resulting vertexes will be the new 3D landmarks that correspond to the 2D cheek landmarks. We will then update relevant elements of **d** to make sure these vertexes are selected in the projection of Eq. 4. This landmark marching process is summarized in Algorithm 1 as a function $\mathbf{d} \leftarrow g(\mathbf{S}, \mathbf{m})$. Note that when the face is approximately of profile view ($|\beta| > 70°$), we do not apply landmark marching since the marched landmarks would overlap with the existing 2D landmarks on the middle of nose and mouth. Figure 3 shows the defined set of pathes on the 3D shape of face and one example of applying Algorithm 1 for updating vector **d**.

Algorithm 1: Landmark marching $g(\mathbf{S}, \mathbf{m})$.

Data: Estimated 3D face **S** and projection matrix parameter **m**
Result: Index vector **d**
/* Rotate **S** by the estimated α, β */
$\hat{\mathbf{S}} = \mathbf{R}(\alpha, \beta, 0)\mathbf{S}$
if $0° < \beta < 70°$ **then**
  foreach $i = 1, \cdots, 4$ **do**
    $V_{\text{cheek}}(i) = \arg\max_{id}(\hat{\mathbf{S}}(1, \text{Path}_{\text{cheek}}(i)))$
if $-70° < \beta < 0°$ **then**
  foreach $i = 5, \cdots, 8$ **do**
    $V_{\text{cheek}}(i) = \arg\min_{id}(\hat{\mathbf{S}}(1, \text{Path}_{\text{cheek}}(i)))$
Update 8 elements of **d** with V_{cheek}.

3.2 Data Augmentation

Given that the projection matrix parameter **m** and shape parameter **p** are the representation of a face shape, we should have a collection of face images with ground truth **m** and **p** so that the learning algorithm can be applied. However, while **U** can be manually labeled on a face image, **m** and **p** are normally unavailable unless a 3D scan is captured along with a face image. For most existing face alignment databases, such as the AFLW database (Köstinger et al. 2011), only 2D landmark locations and sometimes the visibilities of landmarks are manually labeled, with no associated 3D information such as **m** and **p**. In order to make the learning possible, we propose a data augmentation process for 2D face images, with the goal of estimating their **m** and **p** representation.

Specifically, given the labeled visible 2D landmarks **U** and the landmark visibilities **V**, we estimate **m** and **p** by minimizing the following objective function:

$$J(\mathbf{m}, \mathbf{p}) = ||(s\mathbf{RS}(:, g(\mathbf{S}, \mathbf{m})) + \mathbf{t} - \mathbf{U}) \odot \mathbf{V}||_F^2 , \quad (5)$$

which is the difference between the projection of 3D landmarks and the 2D labeled landmarks. Note that although the landmark marching $g(:, :)$ makes cheek landmarks "visible" for non-profile views, the visibility **V** is still necessary to avoid invisible landmarks, such as outer eye corners and half of the face at the profile view, being part of the optimization.

3.2.1 Optimization

For convenient optimization of Eq. 5, we redefine all projection parameters as a projection matrix, i.e.,

$$\mathbf{M} = \begin{bmatrix} s\mathbf{R} & \begin{matrix} t_x \\ t_y \end{matrix} \end{bmatrix} \in \mathbb{R}^{2\times 4}. \quad (6)$$

Also, we denote $\mathbf{d} = g(\mathbf{S}, \mathbf{m})$ in Eq. 5 by assuming it is a constant given the currently estimated **m** and **p**. We then rewrite Eq. 5 as,

$$J(\mathbf{M}, \mathbf{p}) = \left|\left|\left(\mathbf{M}\begin{bmatrix} \mathbf{S}(:, \mathbf{d}) \\ \mathbf{1}^{\mathsf{T}} \end{bmatrix} - \mathbf{U}\right) \odot \mathbf{V}\right|\right|_F^2 . \quad (7)$$

To minimize this objective function, we alternate the minimization w.r.t. **M** and **p** at each iteration. We initialize

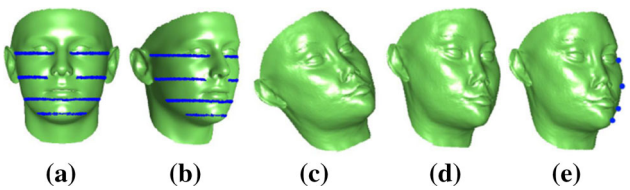


Fig. 3 The landmark marching process for updating vector **d**. (**a**, **b**) show the defined paths of cheek landmarks on the mean shape; (**c**) is the estimated face shape; (**d**) is the estimated face shape by ignoring the roll rotation; and (**e**) shows the locations of landmarks on the cheek

the 3D shape parameter $\mathbf{p} = \mathbf{0}$ and estimate $\mathbf{M}$ first, by $\mathbf{M}^k = \arg\min_{\mathbf{M}} J(\mathbf{M}, \mathbf{p}^{k-1})$,

$$\mathbf{M}^k = \mathbf{U}_V \begin{bmatrix} \mathbf{S}(:, \mathbf{d}_V) \\ \mathbf{1}^\intercal \end{bmatrix}^\intercal \left(\begin{bmatrix} \mathbf{S}(:, \mathbf{d}_V) \\ \mathbf{1}^\intercal \end{bmatrix} \begin{bmatrix} \mathbf{S}(:, \mathbf{d}_V) \\ \mathbf{1}^\intercal \end{bmatrix}^\intercal \right)^{-1}, \tag{8}$$

where $\mathbf{U}_V$ is zero-mean positions (by removing the mean from all the elements) of visible 2D landmarks, $\mathbf{d}_V$ is a vector contains the index of visible landmarks. Given the estimated $\mathbf{M}^k$, we then use the Singular Value Decomposition (SVD) to decompose it to various elements of projection parameter $\mathbf{m}$, i.e, $\mathbf{M}^k = \mathbf{BDQ}^\intercal$. The first diagonal element of $\mathbf{D}$ is scale s and we decompose the rotation matrix $\mathbf{R} = \mathbf{BQ}^\intercal \in \mathbb{R}^{2\times 3}$ to three rotation angles (α, β, γ). Finally, the mean values of $\mathbf{U}$ are translation parameters t_x and t_y.

Then, we estimate $\mathbf{p}^k = \arg\min_{\mathbf{p}} J(\mathbf{M}^k, \mathbf{p})$. Given the orthogonal bases of 3DMM, we choose to compute each element of $\mathbf{p}$ one by one. That is, p^i_{id} is the contribution of i-th identity basis in reconstructing the dense 3D face shape,

$$p^i_{id} = \frac{\mathrm{Tr}\left(\hat{\mathbf{U}}_V^\intercal \hat{\mathbf{U}}_{id_i}\right)}{\mathrm{Tr}\left(\hat{\mathbf{U}}_{id_i}^\intercal \hat{\mathbf{U}}_{id_i}\right)}, \tag{9}$$

where

$$\hat{\mathbf{U}}_V = \mathbf{M}^k \begin{bmatrix} \mathbf{S}(:, \mathbf{d}_V) \\ \mathbf{1}^\intercal \end{bmatrix}, \quad \hat{\mathbf{U}}_{id_i} = \mathbf{M}^k \begin{bmatrix} \mathbf{S}^i_{id}(:, \mathbf{d}_V) \\ \mathbf{1}^\intercal \end{bmatrix}.$$

Here $\hat{\mathbf{U}}_V$ is current residual of position of 2D visible landmarks after subtracting contribution of $\mathbf{M}^k$, and $Tr()$ is the trace function. Once p^i_{id} is computed, we update $\hat{\mathbf{U}}_V$ by subtracting the contribution of i-th basis and continue to compute p^{i+1}_{id}. We alternatively estimate $\mathbf{M}$ and $\mathbf{p}$ until the changes of $\mathbf{M}$ and $\mathbf{p}$ are small enough. After each step of applying Eq. 8 for computing a new estimation of $\mathbf{M}$ and decomposing to its parameters $\mathbf{m}$, we apply the landmark marching algorithm (Algorithm 1) to update the vector $\mathbf{d}$.

3.3 Cascaded CNN Coupled-Regressor

Given a set of N_d training face images and their augmented (i.e., "ground truth") $\mathbf{m}$ and $\mathbf{p}$ representation, we are interested in learning a mapping function that is able to predict $\mathbf{m}$ and $\mathbf{p}$ from the appearance of a face. Clearly this is a complicated non-linear mapping due to the diversity of facial appearance. Given the success of CNN in vision tasks such as pose estimation (Pfister et al. 2015), face detection (Li et al. 2015), and face alignment (Zhang et al. 2014b), we decide to marry the CNN with the cascade regressor framework by learning a series of CNN-based regressors to alternate the estimation of $\mathbf{m}$ and $\mathbf{p}$. To the best of our knowledge, this is the first time CNN is used in 3D face alignment, with the estimation of over 10 landmarks.

For each training image $\mathbf{I}_i$, in addition to the ground truth $\mathbf{m}_i$ and $\mathbf{p}_i$, we also initialize image's representation by, $\mathbf{m}^0_i = h(\bar{\mathbf{m}}, \mathbf{b}_i)$ and $\mathbf{p}^0_i = \mathbf{0}$. Here $\bar{\mathbf{m}}$ is the average of ground truth parameters of projection matrices in the training set, $\mathbf{b}_i$ is a 4-dim vector indicating the bounding box location, and $h(\mathbf{m}, \mathbf{b})$ is a function that modifies the scale and translations of $\mathbf{m}$ based on $\mathbf{b}$.

Thus, at the stage k of the cascaded CNN, we can learn a CNN to estimate the desired update of the projection matrix parameter,

$$\begin{aligned} \Theta^k_m &= \arg\min_{\Theta^k_m} J_\Theta \\ &= \sum_{i=1}^{N_d} ||\Delta\mathbf{m}^k_i - \mathrm{CNN}^k_m(\mathbf{I}_i, \mathbf{U}_i, \mathbf{v}^{k-1}_i; \Theta^k_m)||^2, \end{aligned} \tag{10}$$

where the true projection update is the difference between the current projection matrix parameter and the ground truth, i.e., $\Delta\mathbf{m}^k_i = \mathbf{m}_i - \mathbf{m}^{k-1}_i$, $\mathbf{U}_i$ is current estimated 2D landmarks, computed via Eq. 4, based on $\mathbf{m}^{k-1}_i$ and $\mathbf{d}^{k-1}_i$, and $\mathbf{v}^{k-1}_i$ is estimated landmark visibility at stage $k-1$.

Similarly another CNN regressor can be learned to estimate the updates of the shape parameter,

$$\begin{aligned} \Theta^k_p &= \arg\min_{\Theta^k_p} J_\Theta \\ &= \sum_{i=1}^{N_d} ||\Delta\mathbf{p}^k_i - \mathrm{CNN}^k_p\left(\mathbf{I}_i, \mathbf{U}_i, \mathbf{v}^k_i; \Theta^k_p\right)||^2. \end{aligned} \tag{11}$$

Note that $\mathbf{U}_i$ will be re-computed via Eq. 4, based on the updated $\mathbf{m}^k_i$ and $\mathbf{d}^k_i$ by CNN_m.

We use a six-stage cascaded CNN, including CNN^1_m, CNN^2_m, CNN^3_p, CNN^4_m, CNN^5_p, and CNN^6_m. At the first stage, the input layer of CNN^1_m is the entire face region cropped by the initial bounding box, with the goal of roughly estimating the pose of the face. The input for the second to sixth stages is a 114×114 image that contains an array of 19×19 pose-invariant feature patches, extracted from the current estimated 2D landmarks $\mathbf{U}_i$. In our implementation, since we have $N = 34$ landmarks, the last two patches of 114×114 image are filled with zero. Similarly, for invisible 2D landmarks, their corresponding patches will be filled with zeros as well. These feature patches encode sufficient information about the local appearance around the current 2D landmarks, which drives the CNN to optimize the parameters Θ^k_m or Θ^k_p. Also, through concatenation, these feature patches share the information among different landmarks and jointly drive the CNN in parameter estimation. Our input representation can be extended to use a larger number of landmarks and hence a more accurate dense 3D model can be estimated.

Note that since landmark marching is used, the estimated 2D landmarks $\mathbf{U}_i$ include the projection of marched 3D landmarks, i.e., 2D cheek landmarks. As a result, the appearance features around these cheek landmarks are part of the input to CNN as well. This is in sharp contrast to Jourabloo and Liu (2015) where no cheek landmarks participate the regressor learning. Effectively, these additional cheek landmarks serve as constraints to guide how the facial silhouettes at various poses should look like, which is essentially the shape of the 3D face surface.

Another note is that, instead of alternating between the estimation of **m** and **p**, another option is to jointly estimate both parameters in each CNN stage. Experimentally we observed that such a joint estimation schedule leads to a lower accuracy than the alternating scheme, potentially due to the different physical meaning of **m** and **p** and the ambiguity of multiple pairs of **m** and **p** corresponding the same 2D shape. For the alternating scheme, now we present two different CNN architectures, and use the same CNN architecture for all six stages of the cascade.

3.4 Conventional CNN (C-CNN)

The architecture of the first CNN is shown in Fig. 4. It has three convolutional layers where each one is followed by a pooling layer and a batch normalization layer. Then, one fully connected layer and ReLU layer and, at the end of the architecture, it has one fully connected layer and one Euclidian loss (J_Θ) for estimating the projection matrix parameters or 3D shape parameters. We use rectified linear unit (ReLU) (Glorot et al. 2011) as the activation function which enables CNN to achieve the best performance without unsupervised pre-training.

3.5 Mirror CNN (M-CNN)

We deal with two inherent ambiguities when we estimate projection matrix parameter **m** and 3D shape parameter **p**. First, mutiple pairs of **m** and **p** can represent the same 2D face shape. Second, the estimated updates of **m** and **p** are not explicitly related to the face alignment error. In other words,

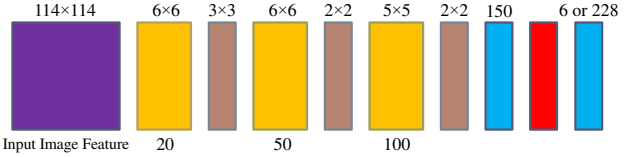


Fig. 4 Architecture of C-CNN (the same CNN architecture is used for all six stages). *Color code* used: *purple* = extracted image feature, *orange* = Conv, *brown* = pooling + batch normalization, *blue* = fully connected layer, *red* = ReLU. The filter size and the number of filters for each layer are shown on the *top* and the *bottom* respectively (Color figure online)

the changes in **m** and **p** are not linearly related to the 2D shape changes. To remedy these ambiguities, we predict 2D shape update simultaneously while estimating the **m** and **p** updates.

We extend the CNN architecture of each cascade stage by encouraging the alignment results of a face image and its mirror to be highly correlated. To this end, we use the idea of mirrorability constraint (Yang and Patras 2015) with two main differences. First, we combine this constraint with the learning procedure rather than using it as a post-processing step. Second, we integrate the mirrorability constraint inside a siamese CNN (Bromley et al. 1993) by sharing the network's weights between the input face image and its mirror image and adding a new loss function.

3.5.1 Mirror Loss

Given the input image and its mirror image with their initial bounding boxes, we use function $h(\bar{\mathbf{m}}, \mathbf{b})$, that modifies the scale and translations of $\bar{\mathbf{m}}$ based on **b**, for initialization. Then, according to the mirror ability constraint, we assume that the estimated update of shape for the input image should be similar to the update of shape for the mirror image with a reordering. This assumption is true when both images are initialized with the *same* landmarks up to a reordering, which is true in all cascade stages. We use the mirror loss to minimize the Euclidian distance of estimated shape update of two images. The mirror loss at stage k is,

$$J_M^k = ||\Delta\hat{\mathbf{U}}^k - C\left(\Delta\hat{\mathbf{U}}_M^k\right)||^2, \tag{12}$$

where $\Delta\hat{\mathbf{U}}^k$ is the input image's shape update, $\Delta\hat{\mathbf{U}}_M^k$ is the mirror image's shape update and $C()$ is a reordering function to indicate landmark correspondence between mirror images.

3.5.2 Mirror CNN Architecture

The new CNN architecture follows the siamese network (Bromley et al. 1993) with two branches whose weights are shared. Figure 5 shows the architecture of the M-CNN. The top and bottom branches are feeded with the extracted input feature from a training image and its mirror respectively. Each branch has two convolutional layers and two layers of locally connected layers. The locally connected layer (Taigman et al. 2014) is similar to convolutional layer and learns a set of filters for various regions of its input. The locally connected layers are spatial location dependent, which is a correct assumption for our extracted image feature at each stage. After each of these layers, we have one pooling and batch normalization layers. At the end in the top branch, after a fully connected layer, batch normalization, ReLU and dropout layers, we have two fully connected layers, one for

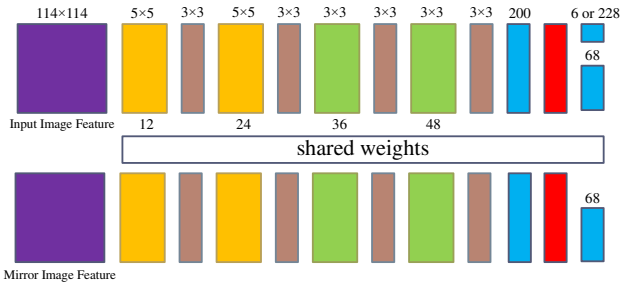


Fig. 5 Architecture of the M-CNN (the same CNN architecture is used for all six stages). *Color code* used: *purple* = extracted image feature, *orange* = Conv, *brown* = pooling + batch normalization, *green* = locally connected layer, *blue* = fully connected layer, *red* = batch normalization + ReLU + dropout. The filter size and the number of filters of each layer are shown on the *top* and the *bottom* of the *top* branch respectively (Color figure online)

estimating the update of parameters (J_Θ) and the other one for estimating the update of 2D shape via the loss (J_U),

$$J_U^k = ||\Delta \mathbf{U}^k - \Delta \hat{\mathbf{U}}^k||^2. \tag{13}$$

In the bottom branch, we only have one loss (J_{MU}) for estimating the update of 2D shape in the mirror image. In total, we have four loss functions, one for the updates of $\mathbf{m}$ or $\mathbf{p}$, two for the 2D shape updates of two images respectively, and one mirror loss. We minimize the total loss at stage k,

$$J_T^k = J_\Theta^k + \lambda_1 J_U^k + \lambda_2 J_{MU}^k + \lambda_3 J_M^k, \tag{14}$$

where λ_1 to λ_3 are weights for loss functions. Despite M-CNN appears more complicated to be trained than C-CNN, their testing are the same. That is, the only useful result at each cascade stage of M-CNN is the estimated update of the $\mathbf{m}$ or $\mathbf{p}$, which is also passed to the next stage and initialize the input image features. In other words, the mirror images and estimated $\Delta\mathbf{U}$ in both images only serve as constraints in training, and are neither needed nor used in testing.

3.6 Visibility and 2D Appearance Features

One notable advantage of employing a dense 3D shape model is that more advanced 2D features, which might be only possible because of the 3D model, can be extracted and contribute to the cascaded CNN learning. In this work, these 2D features refer to the 2D landmark visibility and the appearance patch around each 2D landmark.

In order to compute the visibility of each 2D landmark, we leverage the basic idea of examining whether the 3D surface normal of the corresponding 3D landmark is pointing to the camera or not, under the current camera projection matrix (Jourabloo and Liu 2015). Instead of using the average 3D surface normal for all humans, we extend it by using person-specific 3D surface normal. Specifically, given the

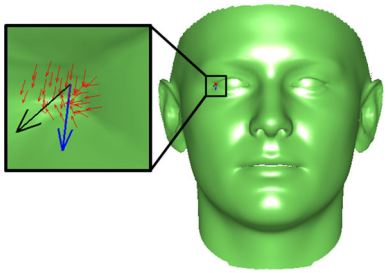


Fig. 6 The person-specific 3D surface normal as the average of normals around a 3D landmark (*black arrow*). Notice the relatively noisy surface normal of the 3D "left eye corner" landmark (*blue arrow*) (Color figure online)

current estimated 3D shape $\mathbf{S}$, we compute the 3D surface normals for a set of sparse vertexes around the 3D landmark of interest, and the average of these 3D normals is denoted as $\mathbf{N}$. Figure 6 illustrates the advantage of using the average 3D surface normal. Given $\mathbf{N}$, we compute,

$$\mathbf{v} = \mathbf{N}^\mathsf{T} \cdot (\mathbf{R}_1 \times \mathbf{R}_2), \tag{15}$$

where $\mathbf{R}_1$ and $\mathbf{R}_2$ are the first two rows of $\mathbf{R}$. If $\mathbf{v}$ is positive, the 2D landmark is considered as visible and its 2D appearance feature will be part of the input for CNN. Otherwise, it is invisible and the corresponding feature will be zero for CNN. Note that this method does not estimate occlusion due to other objects such as hairs.

In addition to visibility estimation, a 3D shape model can also contribute in generating advanced appearance features as the input layer for CNN. Specifically, we aim to extract a pose-invariant appearance patch around each estimated 2D landmark, and the array of these patches will form the input layer. In Yang et al. (2015), a similar feature extraction is proposed by putting different scales of input image together and forming a big image as the appearance feature. We now describe two proposed approaches to extract an appearance feature, i.e., a 19×19 patch, for the nth 2D landmark.

Piecewise affine-warped feature (PAWF) Feature correspondence is always very important for any visual learning, as evident by the importance of eye-based rectification to face recognition (Shan et al. 2004). Yet, due to the fact that a 2D face is a projection of 3D surface with an arbitrary view angle, it is hard to make sure that a local patch extracted from this 2D image corresponds to the patch from another image, even both patches are centered at the ground truth locations of the same nth 2D landmark. Here, "correspond" means that the patches cover the exactly same local region of faces anatomically. However, with a dense 3D shape model in hand, we may extract local patches across different subjects and poses with anatomical correspondence. These correspondences across subjects and poses facilitate CNN to learn the appearance variation induced by misalignment, rather than subjects or poses.

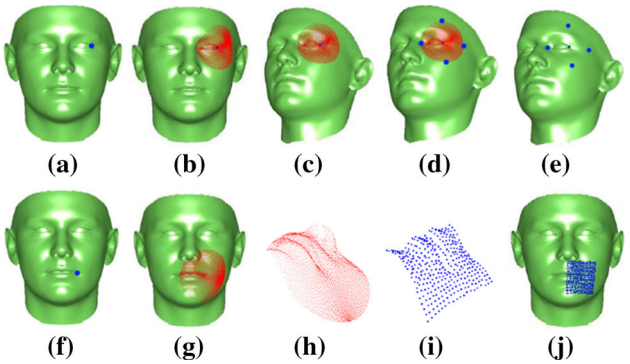


Fig. 7 Feature extraction process, (**a**–**e**) PAWF for the landmark on the *right side* of the *right eye*, (**f**–**j**) D3PF for the landmark on the *right side* of the lip

In an offline procedure, we first search for T vertexes on the mean 3D shape $\mathbf{S}_0$ that are the most closest to the nth landmark (Fig. 7b). Second, we rotate the T vertexes such that the 3D surface normal of the nth landmark points toward the camera (Fig. 7c). Third, among the T vertexes we find four "neighborhood vertexes", which have the minimum and maximum x and y coordinates, and denote the four vertex IDs as a 4-dim vector $\mathbf{d}_p^{(n)}$ (Fig. 7d). The first row of Fig. 7 shows the process of extracting PAWF for right landmark of the right eye.

During the CNN learning, for the nth landmark of ith image, we project the four neighborhood vertexes onto the ith image and obtain four neighborhood points, $\mathbf{U}_i^{(n)} = s\mathbf{RS}(:, \mathbf{d}_p^{(n)}) + \mathbf{t}$, based on the current estimated projection parameter $\mathbf{m}$. Across all 2D face images, $\mathbf{U}_i^{(n)}$ correspond to the same face vertexes anatomically. Therefore, we warp the imagery content within these neighborhood points to a 19×19 patch by using the piecewise affine transformation (Matthews and Baker 2004).

This novel feature representation can be well extracted in most cases, except for cases such as the nose tip at the profile view. In such cases, the projection of the nth landmark is outside the region specified by the neighborhood points, where one of the neighborhood points is invisible due to occlusion. When this happens, we change the location of the invisible point by using its relative distance to the projected landmark location, as shown in Fig. 8.

Direct 3D projected feature (D3PF) Both D3PF and PAWF start with the T vertexes surrounding the nth 3D landmark (Fig. 7g). Instead of finding four neighborhood vertexes as in PAWF, D3PF overlays a 19×19 grid covering the T vertexes, and stores the vertexes of the grid points in $\mathbf{d}_d^{(n)}$ (Fig. 7i). The second row of Fig. 7 shows the process of extracting D3PF. Similar to PAWF, we can now project the set of 3D vertexes $\mathbf{S}(:, \mathbf{d}_d^{(n)})$ to the 2D image and extract a 19×19 patch via bilinear-interpolation, as shown in Fig. 9. We also estimate the visibilities of the 3D vertexes $\mathbf{S}(:, \mathbf{d}_d^{(n)})$ via their surface normals, and zero will be placed in the patch for invisible ones. For D3PF, every pixel in the patch will be corresponding to the same pixel in the patches of other images, while for PAWF, this is true only for the four neighborhood points.

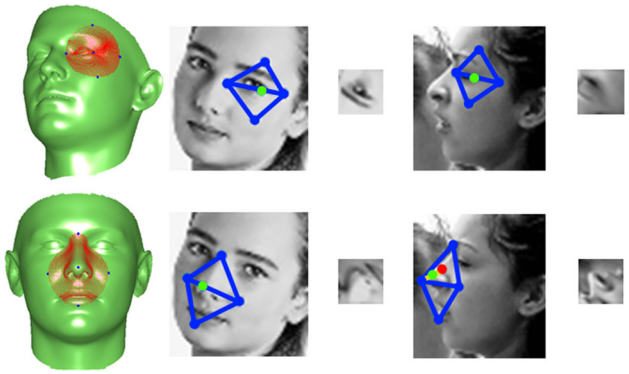


Fig. 8 Examples of extracting PAWF. When one of the four neighborhood points (*red point* in the *bottom-right*) is invisible, it connects to the 2D landmark (*green point*), extends the same distance further, and generate a new neighborhood point. This helps to include the background context around the nose (Color figure online)

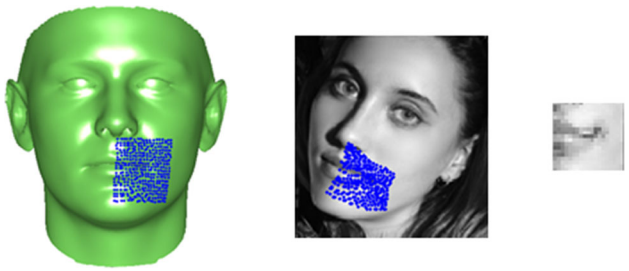


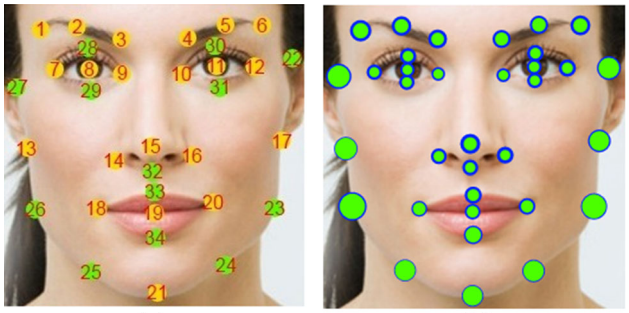
Fig. 9 Example of extracting D3PF

3.7 Testing

The testing part of both C-CNN and M-CNN are the same. Given a testing image $\mathbf{I}$ and its initial parameter $\mathbf{m}^0$ and $\mathbf{p}^0$, we apply the learned cascaded CNN coupled-regressor for face alignment. Basically we iteratively use $R_m^k(\cdot; \Theta_m^k)$ to compute $\Delta\hat{\mathbf{m}}$, update $\mathbf{m}^k$, use $R_p^k(\cdot; \Theta_p^k)$ to compute $\Delta\hat{\mathbf{p}}$, and update $\mathbf{p}^k$. Finally the dense 3D shape is constructed via Eq. 1, and the estimated 2D landmarks are $\hat{\mathbf{U}} = s\mathbf{R}\hat{\mathbf{S}}(:, \mathbf{d}) + \mathbf{t}$. Note that we apply the feature extraction procedure one time for each CNN stage.

4 Experimental Results

In this section, we design experiments to answer the following questions: (1) What is the performance of proposed method on challenging datasets in comparison to the state-of-the-art methods? (2) How do different feature extraction methods perform in pose-invariant face alignment? (3) What is the performance of proposed method with different CNN architectures and with different deep learning toolboxes?



(a) (b)

Fig. 10 **a** AFLW original (*yellow*) and added landmarks (*green*), **b** Comparison of mean NME of each landmark for RCPR (*blue*) and proposed method (*green*). The radius of circles is determined by the mean NME multipled with the face bounding box size (Color figure online)

4.1 Experimental Setup

Databases Given that this work focus on pose-invariant face alignment, we choose two publicly available face datasets with labeled landmarks and a *wide* range of poses.

AFLW database (Köstinger et al. 2011) is a large face dataset with 25K face images. Each image is manually labeled with up to 21 landmarks, with a visibility label for each landmark. In Jourabloo and Liu (2015), a subset of AFLW is selected to have a balanced distribution of yaw angles, including 3901 images for training and 1299 images for testing. We use the same subset and manually label 13 additional landmarks for all 5200 images. We call these 3901 images as the *base* training set. The definition of original landmarks and added landmarks is shown in Fig. 10a. Using ground truth landmarks of each image, we find the tightest bounding box, expand it by 10% of its size, and add 10% noise to the top-left corner, width and height of the bounding box (examples in the 1st row of Fig. 16). These randomly generated bounding boxes mimic the imprecise face detection window and will be used for both training and testing.

AFW dataset (Zhu and Ramanan 2012) contains 468 faces in 205 images. Each face image is manually labeled with up to 6 landmarks and has a visibility label for each landmark. For each face image a detected bounding box is provided, and will be used as initialization. Given the small number of images, we only use this dataset for testing.

We use the $N_{id} = 199$ bases of Basel Face Model (Paysan et al. 2009) for representing identity variation and the $N_{exp} = 29$ bases of face wearhouse (Cao et al. 2014b) for representing expression variation. In total, there are 228 bases representing 3D face shapes with 53, 215 vertexes.

Synthetic training data Unlike conventional face alignment, one of the main challenges in pose-invariant face alignment is the limited training images. There are only two publicly available face databases with a wide poses, along with landmark labeling. Therefore, utilizing synthetic face images is an efficient way to supply more images into the training set. Specifically, we add 16, 556 face images with various poses, generated from 1035 subjects of LFPW dataset (Belhumeur et al. 2011) by the method of Zhu et al. (2015b), to the base training set. We call this new training set as the *extended* training set.

Baseline selection Given the explosion of face alignment work in recent years, it is important to choose appropriate baseline methods so as to make sure the proposed method advances the state of the art. We select the most recent pose-invariant face alignment methods for comparing with the proposed method, according to Table 1. We compare the proposed method with two methods on AFLW: (1) PIFA (Jourabloo and Liu 2015) is a pose-invariant face alignment method which aligns faces of arbitrary poses with the assistant of a sparse 3D point distribution model, (2) RCPR (Burgos-Artizzu et al. 2013) is a method based on cascade of regressors that represents the occlusion-invariant face alignment. For comparison on AFW, we select three methods: (1) PIFA (Jourabloo and Liu 2015), (2) CDM (Yu et al. 2013) is a method based on Constrained Local Model (CLM) and the first one claimed to perform pose-free face alignment, (3) TSPM (Zhu and Ramanan 2012) is based on a mixtures of trees with a shared pool of parts and can handle face alignment for large pose face images. It can be seen that these baselines are most relevant to our focus on pose-invariant face alignment.

Parameter setting For implementing the proposed methods, we use two different deep learning toolboxes. For implementing the C-CNN architecture, we use the MatConvNet toolbox (Vedaldi and Lenc 2015) with a constant learning rate of 1e−4, with ten epochs for training each CNN and a batch size of 100. For the M-CNN architecture, we use the Caffe toolbox (Jia et al. 2014) with a learning rate of 1e−7 and the step learning rate policy with a drop rate of 0.9, in 70 epochs at each stage and a batch size of 100. We set the weight parameters of the total loss λ_1 to λ_3 in Eq. 14 to 1. For RCPR, we use the parameters reported in its paper, with 100 iterations and 15 boosted regressors. For PIFA, we use 200 iterations and 5 boosted regressors. For PAWF and D3PF, at the second stage T is 5000, and 3000 for the other stages. According to our empirical evaluation, six stages of CNN are sufficient for convergence of fitting process.

Evaluation metrics Given the ground truth 2D landmarks $\mathbf{U}_i$, their visibility $\mathbf{v}_i$, and estimated landmarks $\hat{\mathbf{U}}_i$ of N_t testing images, we use two conventional metrics for measuring the error of up to 34 landmarks: (1) Mean Average Pixel Error (MAPE) (Yu et al. 2013), which is the average of the estimation errors for visible landmarks, i.e.,

$$\text{MAPE} = \frac{1}{\sum_i^{N_t} |\mathbf{v}_i|_1} \sum_{i,j}^{N_t,N} \mathbf{v}_i(j) ||\hat{\mathbf{U}}_i(:, j) - \mathbf{U}_i(:, j)||, \quad (16)$$

where $|\mathbf{v}_i|_1$ is the number of visible landmarks of image $\mathbf{I}_i$, and $\mathbf{U}_i(:, j)$ is the jth column of $\mathbf{U}_i$. (2) Normalized Mean Error (NME), which is the average of the normalized estimation error of visible landmarks, i.e.,

$$\text{NME} = \frac{1}{N_t} \sum_i^{N_t} \left(\frac{1}{d_i |\mathbf{v}_i|_1} \sum_j^N \mathbf{v}_i(j) ||\hat{\mathbf{U}}_i(:, j) - \mathbf{U}_i(:, j)|| \right), \quad (17)$$

where d_i is the square root of the face bounding box size (Jourabloo and Liu 2015). The eye-to-eye distance is not used in NME since it is not well defined in large poses such as profile.

4.2 Comparison Experiments

Feature extraction methods To show the advantages of the proposed features, Table 3 compares the accuracy of the proposed method on AFLW with 34 landmarks, with various feature presentation (i.e., the input layer for CNN^2 to CNN^6). For this experiment, we use the C-CNN architecture with the base training set. The "Extracted Patch" refers to extracting a constant size (19×19) patch centered by an estimated 2D landmark, from a face image normalized using the bounding box, which is a baseline feature widely used in conventional 2D alignment methods (Zhu et al. 2015a; Zhang et al. 2014a). For the feature "+Cheek Landmarks", additional up to four 19×19 patches of the contour landmarks, which are invisible for non-frontal faces, will be replaced with patches of the cheek landmarks, and used in the input layer of CNN learning. The PAWF can achieve higher accuracy than the D3PF. By comparing Column 1 and 3 of Table 3, it shows that extracting features from cheek landmarks are effective in acting as additional visual cues for the cascaded CNN regressors. The combination of using the cheek landmarks and extracting PAWF achieves the highest accuracy, which will be used in the remaining experiments. Figure 11 shows the errors on AFLW testing set after each stages of CNN for different feature extraction methods. There is no difference in the errors of the first stage CNN because it uses the global appearance in the bounding box, rather than the array of local features.

CNN is known for demanding a large training set, while the 3901-image AFLW training set is relatively small from CNN's perspective. However, our CNN-based regressor is still able to learn and align well on unseen images. We

Table 3 NME (%) of the proposed method with different features with the C-CNN architecture and the base training set

PAWF + Cheek landmarks	D3PF + Cheek landmarks	PAWF	Extracted patch
4.72	5.02	5.19	5.51

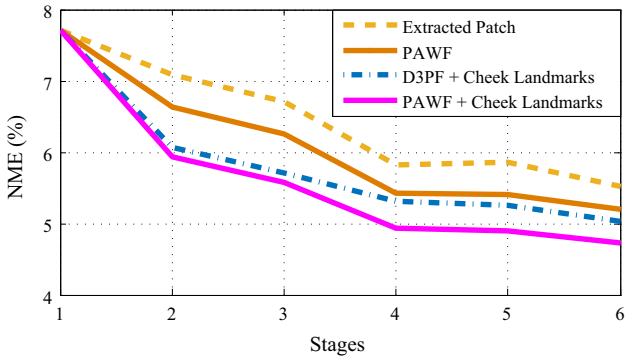


Fig. 11 Errors on AFLW testing set after each stages of CNN for different feature extraction methods with the C-CNN architecture and the base training set. The initial error is 25.8%

Table 4 The NME (%) of three methods on AFLW with the base training set

Proposed method (C-CNN)	PIFA	RCPR
4.72	8.04	6.26

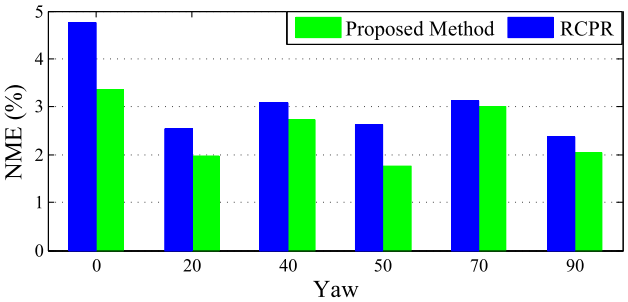


Fig. 12 Comparison of NME for each pose with the C-CNN architecture and the base training set

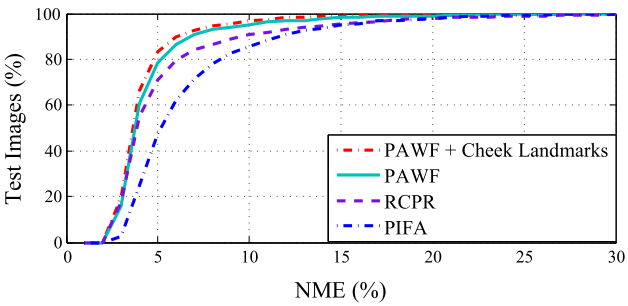


Fig. 13 The comparison of CED for different methods with the C-CNN architecture and the base training set

Fig. 14 Result of the proposed method after the first stage CNN. This image shows that the first stage CNN can model the distribution of face poses. The *right-view* faces are at the *top*, the *frontal-view* faces are at the *middle*, and the *left-view* faces are at the *bottom*

Table 5 The NME (%) of three methods on ALFW with extended training set and Caffe toolbox

Proposed method (M-CNN)	Proposed method (C-CNN)	RCPR
4.52	5.38	7.04

attribute this fact to the effective appearance features proposed in this work, i.e., the superior feature correspondence enabled by the dense face model reduces CNN's demand for massive training data.

Experiments on AFLW dataset We compare the proposed method with the two most related methods for aligning faces with arbitrary poses. For both RCPR and PIFA, we use their source code to perform training on the base training set. The NME of the three methods on the AFLW testing set are shown in Table 4. The proposed method can achieve better results than the two baselines. The error comparison for each landmark is shown in Fig. 10b. As expected, the contour landmarks have relatively higher errors and the proposed method has lower errors than RCPR across all of the landmarks.

By using the ground truth landmark locations of the test images, we divide all test images to six subsets according to the estimated yaw angle of each image. Figure 12 compares the proposed method with RCPR. The proposed method can achieve better results across different poses, and more importantly, is more robust or has less variation across poses. For the detailed comparison on the NME distribution, the Cumulative Errors Distribution (CED) diagrams of various methods are shown in Fig. 13. The improvement seems to be over all NME values, and is especially larger around lower NMEs ($\leq$8%). We use the t-SNE toolbox (Maaten and Hinton 2008) to apply dimension reduction on the output of ReLU layer in the first stage CNN. The output of each test image is reduced to a two-dimensional point and all test images are plotted based on their location of the points (Fig. 14). This figure shows that the first stage CNN can model the distribution of face poses.

Experiments on the AFLW dataset with M-CNN We use the extended training set and the mirror CNN architecture (M-CNN) for this experiment. We report the NME results of three methods in Table 5. The M-CNN architecture, which incorporates the mirror constraint during the learning, achieves approximately 16% reduction of error over the C-CNN architecture implemented with the Caffe toolbox. This shows the effectiveness of the mirrorability constraint in the new architecture.

The comparison of Tables 4 and 5 shows that the accuracy of the RCPR method is lower with the extended training set than with the base training set. We attribute this to the low quality of the side parts of the synthesized large-pose face

Table 6 The MAPE of six methods on AFW

Proposed method (M-CNN + PAWF)	Proposed method (C-CNN + PAWF)	Proposed method (C-CNN + D3PF)	PIFA	CDM	TSPM
6.52	7.43	7.83	8.61	9.13	11.09

Table 7 The six-stage NMEs of implementing C-CNN and M-CNN architectures with different training data sets and CNN toolboxes

Sett.	Toolbox	Method/data	S-1	S-2	S-3	S-4	S-5	S-6
1	MatConvNet	C-CNN/Base set	7.68	5.93	5.58	4.94	4.89	4.72
2	Caffe		8.75	6.32	6.15	5.55	5.53	5.44
3		M-CNN/Base set	7.18	6.06	5.83	5.08	4.91	4.76
4		C-CNN/Extended set	8.44	6.78	6.60	5.75	5.70	5.38
5		M-CNN/Extended set	7.41	6.16	5.80	4.76	4.67	4.52

The initial error is 25.8%

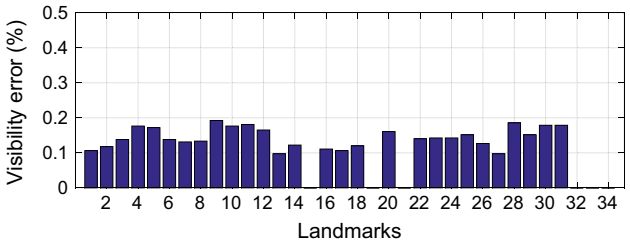


Fig. 15 The distribution of visibility errors for each landmark. For six landmarks on the horizontal center of the face, their visibility errors are zeros since they are always visible

images. Although the method in Zhu et al. (2015b) can synthesize side-view face images, the synthesized images could have some artifacts on the side part of the face. These artifacts make it hard for the local fern features-based RCPR method to simultaneously estimate the location and the visibility of landmarks.

In our proposed method, we arrange the extracted PAWF patches in a spatial array and use it as the input to CNN. An alternative CNN input is to assign the extracted PAWF patches to different channels and construct a $19 \times 19 \times 34$ input datum. To evaluate its performance, considering the change of the input size, we modify the CNN architecture in Fig. 5 by removing the first, the third and the fourth pooling layers. The NME of M-CNN with the extended training set is 4.91%, which shows that arranging the PAWF patches as a large image is still superior.

Experiments on AFW dataset The AFW dataset contains faces of all pose ranges with labels of 6 landmarks. We report the MAPE for six methods in Table 6. For PIFA, CDM and TSPM, we show the error reported in their papers. Again

Fig. 16 The results of the proposed method on AFLW and AFW. The *green*/*red*/*yellow dots* show the visible/invisible/cheek landmarks, respectively. *First row* initial landmarks for AFLW, *Second* estimated 3D dense shapes, *Third* estimated landmarks, *Forth* and *Fifth* estimated landmarks for AFLW, *Sixth* estimated landmarks for AFW. Notice that despite the discrepancy between the diverse face poses and constant *front-view* landmark initialization (*top row*), our model can adaptively estimate the pose, fit a dense model and produce the 2D landmarks as a byproduct (Color figure online)

Fig. 17 The result of the proposed method across stages, with the extracted features (*1st and 3rd rows*) and alignment results (*2nd and 4th rows*). Note the changes of the landmark position and visibility (the *blue arrow*) over stages

we see the consistent improvement of our proposed method (with both architectures) over the baseline methods.

Comparison of two CNN toolboxes We utilize two toolboxes for our implementations. We use the MatConvNet toolbox (Vedaldi and Lenc 2015) to implement the C-CNN architecture (Fig. 4). However, the MatConvNet toolbox has limited ability in defining different branches for CNN, which is required to train a siamese network. Therefore, we use the Caffe toolbox (Jia et al. 2014) to implement the M-CNN architecture (Fig. 5). Based on our experiments on the AFLW test set, there are noticeable difference between the testing results of these two toolboxes.

Table 7 shows the detailed comparison of the C-CNN and M-CNN architectures with different settings. The Settings 1 and 2 compare the implementations of the C-CNN architecture on the AFLW training set, using the MatConvNet and Caffe toolboxes respectively. It shows the superior accuracy of the MatConvNet implementation in all stages, even when the extended training set is provided in the Setting 4. These different testing results of two toolboxes might be due to two reasons. One is that, the implementation of the basic building blocks, the optimization, and the default parameters could be different on the two toolboxes. The other is the random initialization of network parameters. The comparison of the Settings 2 and 3 shows the superiority of the M-CNN architecture. The Setting 5 includes our final result with the M-CNN architecture and the extended training set.

Landmark visibility estimation For evaluating the accuracy of our visibility prediction, we utilize the ground truth 3D shape of the test images and compute the visibility label of landmarks due to the self occlusion. We define the "visibility error" as the metric, which is the average of the ratios between the number of incorrectly estimated visibility labels and the total number of landmarks per image. The proposed method achieves a visibility error of 4.1%. If we break down the visibility error for each landmark, their distribution is shown in Fig. 15.

Qualitative results Some examples of alignment results for the proposed method on AFLW and AFW datasets are shown in Fig. 16. The result of the proposed method at each stage is shown in Fig. 17.

Time complexity The speeds of proposed method with PAWF and D3PF are 0.6 and 0.26 FPS respectively, with the Matlab implementation. The most time consuming part in the proposed method is feature extraction which consumes 80% of the total time. We believe this can be substantially improved with C coding and parallel feature extraction. Note that the speed of C-CNN and M-CNN architectures are the same because we only compute response of the top branch of M-CNN in the testing phase.

5 Conclusions and Future Directions

We propose a method to fit a 3D dense shape to a face image with large poses by combining cascade CNN regressors and the 3D Morphable Model (3DMM). We propose two types of pose invariant features and one new CNN architecture for boosting the accuracy of face alignment. Also, we estimate the location of landmarks on the cheek, which also drives the 3D face model fitting. Finally, we achieve the state-of-the-art performance on two challenging face alignment with large poses.

There are many interesting directions to further improve pose-invariant face alignment. The first direction is to investigate the importance of each landmark for correct pose estimation, and find the best landmarks for estimating projection matrix parameters and the best ones for estimating 3D shape parameters. The second direction is to investigate combining the cascaded CNN regressors to a single deep CNN regressor. Currently, the CNNs are trained sequentially for each stage. Combining CNNs and incorporating results of previous CNNs, which lead to extraction of deeper features, are promising directions to improve pose-invariant face alignment.

References

Amberg, B., Knothe, R., & Vetter, T. (2008). Expression invariant 3D face recognition with a morphable model. In *FG* (pp. 1–6).

Belhumeur, P.N., Jacobs, D.W., Kriegman, D., & Kumar, N. (2011). Localizing parts of faces using a consensus of exemplars. In *CVPR* (pp. 545–552).

Bell, S., & Bala, K. (2015). Learning visual similarity for product design with convolutional neural networks. *ACM Transactions on Graphics*, *34*(4), 98.

Bromley, J., Bentz, J.W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., et al. (1993). Signature verification using a siamese time delay neural network. *International Journal Pattern Recognition*, *7*(04), 669–688.

Burgos-Artizzu, X.P., Perona, P., & Dollár, P. (2013). Robust face landmark estimation under occlusion. In *ICCV* (pp. 1513–1520).

Cao, C., Hou, Q., & Zhou, K. (2014). Displaced dynamic expression regression for real-time facial tracking and animation. *ACM Transactions on Graphics (TOG)*, *33*(4), 43.

Cao, X., Wei, Y., Wen, F., & Sun, J. (2014). Face alignment by explicit shape regression. *International Journal of Computer Vision*, *107*(2), 177–190.

Cao, C., Weng, Y., Zhou, S., Tong, Y., & Zhou, K. (2014). Facewarehouse: A 3D facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics*, *20*(3), 413–425.

Cootes, T., Taylor, C., & Lanitis, A. (1994) Active shape models: Evaluation of a multi-resolution method for improving image search. In *BMVC* vol. 1, (pp. 327–336).

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings artificial intelligence and statistics (AISTATS)* (pp. 315–323).

Hsu, G.S., Chang, K.H., & Huang, S.C. (2015). Regressive tree structured model for facial landmark localization. In *ICCV* (pp. 3855–3861)

Jeni, L.A., Cohn, J.F., & Kanade, T. (2015). Dense 3D face alignment from 2d videos in real-time. In *FG* (vol. 1, pp. 1–8)

Jeni, L.A., Tulyakov, S., Yin, L., Sebe, N., & Cohn, J.F. (2016). The first 3D face alignment in the wild (3DFAW) challenge. In *ECCV* (pp. 511–520).

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S. & Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*, (2014) (pp. 675–678).

Jourabloo, A., & Liu, X. (2015). Pose-invariant 3D face alignment. In *ICCV* (pp. 3694–3702).

Jourabloo, A., & Liu, X. (2016). Large-pose face alignment via cnn-based dense 3D model fitting. In *CVPR* (pp. 4188–4196).

Jourabloo, A., Yin, X., & Liu, X. (2015). Attribute preserved face de-identification. In *ICB* (pp. 278–285).

Köstinger, M., Wohlhart, P., Roth, P.M., & Bischof, H. (2011). Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization. In *ICCVW* (pp. 2144–2151).

Li, H., Lin, Z., Shen, X., Brandt, J., & Hua, G. (2015) A convolutional neural network cascade for face detection. In *CVPR* (pp. 5325–5334).

Liu, X. (2009). Discriminative face alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(11), 1941–1954.

Liu, X. (2010). Video-based face model fitting using adaptive active appearance model. *Journal of Image Vision Computing*, *28*(7), 1162–1172.

Matthews, I., & Baker, S. (2004). Active appearance models revisited. *International Journal of Computer Vision*, *60*(2), 135–164.

Paysan, P., Knothe, R., Amberg, B., Romdhani, S., & Vetter, T. (2009). A 3D face model for pose and illumination invariant face recognition. In *AVSS* (pp. 296–301).

Pfister, T., Simonyan, K., Charles, J., & Zisserman, A. (2015). Deep convolutional neural networks for efficient pose estimation in gesture videos. In *ACCV* (pp. 538–552).

Phillips, P.J., Moon, H., Rizvi, S., Rauss, P.J., et al. (2000). The FERET evaluation methodology for face-recognition algorithms. *IEEE Transactions Pattern Analysis and Machine Intelligence*, *22*(10), 1090–1104.

Qu, C., Monari, E., Schuchert, T., & Beyerer, J. (2015) Adaptive contour fitting for pose-invariant 3D face shape reconstruction. In *BMVC* (pp. 1–12).

Roth, J., Tong, Y., & Liu, X. (2015). Unconstrained 3D face reconstruction. In *CVPR* (pp. 2606–2615).

Roth, J., Tong, Y., & Liu, X. (2016). Adaptive 3D face reconstruction from unconstrained photo collections. In *CVPR* (pp. 4197–4206).

Saragih, J.M., Lucey, S., & Cohn, J. (2009). Face alignment through subspace constrained mean-shifts. In *ICCV* (pp. 1034–1041).

Shan, S., Chang, Y., Gao, W., Cao, B., & Yang, P. (2004). Curse of mis-alignment in face recognition: Problem and a novel mis-alignment learning solution. In *FG* (pp. 314–320).

Sun, Y., Wang, X., & Tang, X. (2013). Deep convolutional network cascade for facial point detection. In *CVPR* (pp. 3476–3483).

Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. In *CVPR* (pp. 1701–1708).

Tulyakov, S., & Sebe, N. (2015) Regressing a 3D face shape from a single image. In *ICCV* (pp. 3748–3755).

Tzimiropoulos, G. (2015) Project-out cascaded regression with an application to face alignment. In *CVPR* (pp. 3659–3667).

Valstar, M., Martinez, B., Binefa, X., & Pantic, M. (2010) Facial point detection using boosted regression and graph models. In *CVPR* pp. 2729–2736.

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*, 2579–2605.

Vedaldi, A., & Lenc, K. (2015). MatConvNet—convolutional neural networks for matlab. In *ACM MM, (2015)* (pp. 689–692).

Wagner, A., Wright, J., Ganesh, A., Zhou, Z., Mobahi, H., & Ma, Y. (2012). Toward a practical face recognition system: Robust alignment and illumination by sparse representation. *IEEE Transactions Pattern Analysis Machine Intelligence*, *34*(2), 372–386.

Wang, N., Gao, X., Tao, D., & Li, X. (2014). Facial feature point detection: A comprehensive survey. arXiv preprint arXiv:1410.1037.

Wu, Y., & Ji, Q. (2015) Robust facial landmark detection under significant head poses and occlusion. In *ICCV* (pp. 3658–3666).

Xiao, J., Baker, S., Matthews, I., & Kanade, T. (2004). Real-time combined 2D+3D active appearance models. In *CVPR* (vol. 2, pp. 535–542).

Yang, H., & Patras, I. (2015). Mirror, mirror on the wall, tell me, is the error small? In *CVPR* (pp. 4685–4693).

Yang, B., Yan, J., Lei, Z., & Li, S.Z. (2015). Convolutional channel features. In *ICCV* (pp. 82–90).

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In *NIPS* (pp. 3320–3328).

Yu, X., Huang, J., Zhang, S., Yan, W., & Metaxas, D.N. (2013). Pose-free facial landmark fitting via optimized part mixtures and cascaded deformable shape model. In *ICCV* (pp. 1944–1951).

Yu, X., Lin, Z., Brandt, J., & Metaxas, D.N. (2014). Consensus of regression for occlusion-robust facial feature localization. In *ECCV* (pp. 105–118).

Zagoruyko, S., & Komodakis, N. (2015). Learning to compare image patches via convolutional neural networks. In *CVPR* (pp. 4353–4361).

Zhang, Z., Luo, P., Loy, C.C., & Tang, X. (2014). Facial landmark detection by deep multi-task learning. In *ECCV* (pp. 94–108).

Zhang, J., Shan, S., Kan, M., & Chen, X. (2014). Coarse-to-fine auto-encoder networks (CFAN) for real-time face alignment. In *ECCV* (pp. 1–16).

Zhang, J., Zhou, S.K., Comaniciu, D., & McMillan, L. (2008). Conditional density learning via regression with application to deformable shape segmentation. In *CVPR* (pp. 1–8).

Zhou, E., Fan, H., Cao, Z., Jiang, Y., & Yin, Q. (2013). Extensive facial landmark localization with coarse-to-fine convolutional network cascade. In *ICCVW* (pp. 386–391).

Zhu, X., & Ramanan, D. (2012). Face detection, pose estimation, and landmark localization in the wild. In *CVPR* (pp. 2879–2886).

Zhu, X., Lei, Z., Yan, J., Yi, D., & Li, S.Z. (2015). High-fidelity pose and expression normalization for face recognition in the wild. In *CVPR* (pp. 787–796).

Zhu, S., Li, C., Change Loy, C., & Tang, X. (2015). Face alignment by coarse-to-fine shape searching. In *CVPR* (pp. 4998–5006).

Zhu, X., Yan, J., Yi, D., Lei, Z., & Li, S.Z. (2015). Discriminative 3D morphable model fitting. In *FG* (pp. 1–8).