DESIGNING CONVOLUTIONAL NEURAL NETWORKS FOR FACE ALIGNMENT AND ANTI-SPOOFING

By

Amin Jourabloo

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science — Doctor of Philosophy

2019

ABSTRACT

DESIGNING CONVOLUTIONAL NEURAL NETWORKS FOR FACE ALIGNMENT AND ANTI-SPOOFING

By

Amin Jourabloo

Face alignment is the process of detecting a set of fiducial points on a face image, 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. Recently, face alignment works are popular in top vision venues and achieve a lot of attention. In spite of the fruitful prior work and ongoing progress of face alignment, pose-invariant face alignment is still challenging. To address the inherent challenges associated with this problem, we propose pose-invariant face alignment by fitting a dense 3DMM, and integrating estimation of 3D shape and 2D facial landmarks from a single face image in a single CNN. We introduce a new layer, called visualization layer, which is differentiable and allows backpropagation of an error from a later block to an earlier one.

Another application of facial analysis is the face anti-spoofing, which has recently achieved a lot of attention. While face recognition systems serve as a verification portal for various devices (i.e., phone unlock, access control, and transportation security), attackers present face spoofs (i.e., presentation attacks, PA) to the system and attempt to be authenticated as the genuine user. We present our proposed deep models for face anti-spoofing that use the supervision from both the spatial and temporal auxiliary information, for the purpose of robustly detecting face PA from a face video.

This thesis is dedicated to my family,
my parents: *Hassan* and *Kobra*
my brother: *Ahmad*
my sister's family: *Zahra*, *Hamed* and *Samyar*

ACKNOWLEDGMENTS

This dissertation would not have been made possible without the help of many people.

I am very honored to have Dr. Xiaoming Liu as my advisor. His expectation and encouragement have made me achieve more than I could ever have imagined. The time we spent to debug codes, brainstorm, and polish papers has refined my skills in critical thinking, presentation and writing. By setting himself as an example, he has taught me what a good researcher should be like. It is my great pleasure to have the opportunity to work with Dr. Anil K. Jain, Dr. Arun Ross in the Anti-Spoofing lab. As a world-leading researcher, Dr. Jain has inspired many younger generations including me to pursue a Ph.D. Dr. Ross's patience and insightful comments at all presentations have shown me that every researcher desires to be heard.

I am grateful for my labmates, Joseph Roth, Morteza Safdarnejad, Muhammad Jamal Afridi, Yousef Atoum, Xi Yin, Luan Tran, Garrick Brazil, Yaojie Liu, Bangjie Yin, Joel Stehouwer, Adam Terwilliger, Hieu Nguyen, Shengjie Zhu, Masa Hu. The valuable comments in paper review, the willingness to help, the encouragement when I am in a bad mood, and the entertainment together have made it a very pleasant journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction on Face Alignment and Face Anti-Spoofing

1.1 Introduction

Face alignment is the process of detecting a set of fiducial points on a face image, 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 [114], face de-identification [55] and 3D face reconstruction [94].

Due to the importance of face alignment, it has been well studied during past decades [115], with the well-known Active Shape Model [30] and Active Appearance Model (AAM) [70, 78]. Recently, face alignment works are popular in top vision venues and achieve a lot of attention. Despite the continuous improvement on the alignment accuracy, face alignment is still a very challenging problem, due to the non-frontal face *pose*, low image *quality*, *occlusion*, etc. Among all the challenges, we identify the *pose invariant face alignment* as the one deserving substantial research efforts, for a number of reasons. First, face detection has substantially advanced its capability in detecting faces in all poses, including profiles [138], which calls for the subsequent face alignment to handle faces with arbitrary poses. Second, many facial analysis tasks would benefit from the robust alignment of faces at all poses, such as expression recognition and 3D face reconstruc-

tion [94]. Third, there are very few existing approaches that can align a face with any view angle, or have conducted extensive evaluations on face images across $\pm 90^{\circ}$ yaw angles [135, 151], which is a clear *contrast* with the vast face alignment literature [115].

We present our proposed approaches for pose-invariant face alignment in chapters 2 to 4. The core idea of our proposed methods is that instead of estimating 2D landmarks directly, we estimate the 3D shape of the face and by projecting the 3D shape to 2D, we can have the 2D locations of the landmarks.

Another application of facial analysis is face anti-spoofing, which has recently achieved a lot of attention. While face recognition systems serve as a verification portal for various devices (i.e., phone unlock, access control, and transportation security), attackers present face spoofs (i.e., presentation attacks, PA) to the system and attempt to be authenticated as the genuine user. The face PAs include printing the face on paper (print attack), replaying a face video on a digital device (replay attack), and wearing a mask (mask attack). To counteract PA, researchers have developed face anti-spoofing techniques [27, 38, 39, 65] to detect PA *prior to* a face image being recognized. Therefore, face anti-spoofing is vital to ensure that face recognition systems are robust to PA and safe to use.

In chapters 5 and 6, we present our proposed deep models for face anti-spoofing that use the supervisions from both the spatial and temporal auxiliary information, for the purpose of robustly detecting face PA from a face image or a face video.

1.2 Prior work on face alignment

We review prior work on face alignment in seven areas related to the proposed methods: generic face alignment, pose-invariant face alignment, 3D face model fitting, face alignment via deep

learning, sharing information in face alignment and deep learning, convolutional recurrent neural network (CRNN), and visualization in deep learning.

Generic face alignment The first type of face alignment approach is based on Constrained Local Model (CLM), where an early example is ASM [30]. The basic idea is to learn a set of local appearance models, one for each landmark, and the decisions from the local models are fused with a global shape model. There are generative or discriminative [32] approaches in learning the local model, and various approaches in utilizing the shape constraint [4]. While the local models are favored for higher estimation precision, it also creates difficulty for alignment on low-resolution images due to limited local appearance. In contrast, the AAM method [29, 78] and its extension [75, 97] learn a global appearance model, whose similarity to the input image drives the landmark estimation. While AAM is known to have difficulty with unseen subjects [42], the recent development has substantially improved its generalization capability [110]. Motivated by the Shape Regression Machine [140, 146] in the medical domain, cascaded regressor-based methods have been very popular in recent years [26, 111]. On one hand, the series of regressors progressively reduce the alignment error and lead to higher accuracy. On the other hand, advanced feature learning also renders ultra-efficient alignment procedures [56, 93]. Other than the three major types of algorithms, there are also works based on deep learning [142], graph-model [151], and semi-supervised learning [105].

Pose-invariant face alignment The methods of [45, 135, 151] combines face detection, pose estimation and face alignment. By using a 3D shape model with an optimized mixture of parts, [135] is applicable to faces with a large range of poses. In [120], 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 [89]. However, as a 2D landmark-

based approach, it cannot estimate 3D face poses. Occlusion-invariant face alignment, such as RCPR [22], may also be applied to handle large poses since non-frontal faces are one type of occlusions. [109] 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.

3D face model fitting 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 [48], a dense 3D face alignment from videos is proposed. At first, a dense set of 2D landmarks is 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 the estimated 2D landmarks. The authors in [92] aim to make sure that the locations of 2D contour landmarks are consistent with the 3D face shape. In [152], a 3D face model fitting method based on the similarity of frontal view face images is proposed.

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. [99] 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. [144] and CFAN [139]. In [139], a face alignment method based on cascade of stacked auto-encoder (SAE) networks can progressively refine locations of 2D landmarks at each stage. TCDCN [142] 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.

Sharing information in face alignment and deep learning Utilizing different side information in face alignment can improve the alignment accuracy. TCDCN [142] jointly estimates auxil-

iary attributes (e.g., gender, expression) with landmark locations to improve alignment accuracy. In [129], 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. The consensus of occlusion-specific regressors [136] in a Bayesian model is used to share information among different regressors. In [147] 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 [134], or by using the siamese networks [8, 137] to share the weights among branches of the network and make a final decision with combined responses of all branches.

Convolutional recurrent neural network (CRNN) Methods based on CRNNs [107, 116, 122] are the first attempts to combine cascade of regressors with joint optimization, for aligning mostly frontal faces. Their convolutional part extracts features from the whole image [122] or from the patches at the landmark locations [107]. The recurrent part facilitates joint optimization by sharing information among all regressors. Generally, the main drawbacks of CRNNs are: 1) existing CRNN methods are designed for near-frontal face alignment; 2) the CRNN methods share the same CNN at all stages.

Visualization in deep learning Visualization techniques have been used in deep learning to assist in making a relative comparison among the input data and focusing on the region of interest. One category of these methods exploit the deconvolutional and upsampling layers to either expand response maps [67, 87] or represent estimated parameters [132]. Alternatively, various types of feature maps, e.g., heatmaps and Z-Buffering, can represent the current estimation of landmarks and parameters. In [21, 80, 119], 2D landmark heatmaps represent the landmarks' locations. [21] proposes a two step pose invariant alignment based on heatmaps to make more precise estima-

tions. The heatmaps suffer from three drawbacks: 1) lack of the capability to represent objects in details; 2) requirement of one heatmap per landmark due to its weak representation power. 3) they cannot estimate the visibility of landmarks. The Z-Buffer rendered using the estimated 3D face is also used to convey the results of a previous CNN to the next one [147]. However, the Z-Buffer representation is not differentiable, preventing end-to-end training.

1.3 Prior work on face anti-spoofing

We review the prior face anti-spoofing works in three groups: texture-based methods, temporal-based methods, and remote photoplethysmography methods.

Texture-based Methods Since most face recognition systems adopt only RGB cameras, using texture information has been a natural approach to tackling face anti-spoofing. Many prior works utilize hand-crafted features, such as LBP [33, 34, 77], HoG [59, 131], SIFT [84] and SURF [18], and adopt trididitional classifiers such as SVM and LDA. To overcome the influence of illumination variation, they seek solutions in a different input domain, such as HSV and YCbCr color space [16, 17], and Fourier spectrum [65].

As deep learning has proven to be effective in many computer vision problems, there are many recent attempts of using CNN-based features or CNNs in face anti-spoofing [37, 66, 83, 130]. Most of the work treats face anti-spoofing as a simple *binary* classification problem by applying the softmax loss. For example, [66, 83] use CNN as feature extractor and fine-tune from ImageNet-pretrained CaffeNet and VGG-face. The work of [37, 66] feed different designs of the face images into CNN, such as multi-scale faces and hand-crafted features, and directly classify live vs. spoof.

Temporal-based Methods One of the earliest solutions for face anti-spoofing is based on temporal cues such as eye-blinking [82, 83]. Methods such as [58, 98] track the motion of mouth and lip to

detect the face liveness. While these methods are effective to typical paper attacks, they become vulnerable when attackers present a replay attack or a paper attack with eye/mouth portion being cut. There are also methods relying on more general temporal features, instead of the specific facial motion. The most common approach is frame concatenation. Many handcrafted feature-based methods may improve intra-database testing performance by simply concatenating the features of consecutive frames to train the classifiers [16,33,60]. Additionally, there are some work proposing temporal-specific features, e.g., Haralick features [3], motion mag [10], and optical flow [6]. In the deep learning era, Feng *et al*. feed the optical flow map and Shearlet image feature to CNN [37]. In [125], Xu *et al*. propose an LSTM-CNN architecture to utilize temporal information for binary classification. Overall, all prior methods still regard face anti-spoofing as a binary classification problem, and thus they have a hard time to generalize well in the cross-database testing.

Remote Photoplethysmography (rPPG) Remote photoplethysmography (rPPG) is the technique to track vital signals, such as heart rate, without any contact with human skin [14, 35, 91, 108, 118]. Research starts with face videos with no motion or illumination change to videos with multiple variations. In [35], Haan *et al*. estimate rPPG signals from RGB face videos with lighting and motion changes. It utilizes color difference to eliminate the specular reflection and estimate two orthogonal chrominance signals. After applying the Band Pass Filter (BPM), the ratio of the chrominance signals are used to compute the rPPG signal.

The rPPG signal has previously been utilized to tackle face anti-spoofing [69, 81]. In [69], rPPG signals are used for detecting the 3D mask attack, where the live faces exhibit a pulse of heart rate unlike the 3D masks. They use rPPG signals extracted by [35] and compute the correlation features for classification. Similarly, Magdalena *et al*. [81] extract rPPG signals (also via [35]) from three face regions and two non-face regions, for detecting print and replay attacks. Although in replay attacks, the rPPG extractor might still capture the normal pulse, the combination of multiple

regions can differentiate live vs. spoof faces. While the analytic solution to rPPG extraction [35] is easy to implement, we observe that it is sensitive to PIE variations.

1.4 Overview of the thesis

In the second chapter, we propose a novel regression-based approach for *pose-invariant face alignment*, which aims to estimate the 2D and 3D locations of a sparse set of face landmarks, as well as their visibilities in the 2D image, for a face with arbitrary pose (e.g., $\pm 90^{\circ}$ yaw). By extending the popular cascaded regressor for 2D landmark estimation, we learn two fern regressors for each cascade layer, one for predicting the update for the camera projection matrix, and the other for predicting the update for the 3D shape parameter. The learning of two regressors is conducted alternatively with the goal of minimizing the difference between the ground truth updates and the predicted updates.

In the third chapter, 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 [99, 142], our cascaded CNN can produce a substantially larger number (34) of 2D and 3D landmarks. Further, using landmark marching [150], our algorithm can adaptively adjust the 3D landmarks during the fitting, so that the local appearances around cheek landmarks contribute to the fitting process.

In the fourth chapter, we introduce a novel visualization layer. We proposed a CNN architecture which is consist of several blocks, called visualization blocks. This architecture can be considered as a cascade of shallow CNNs. The new layer visualizes the alignment result of a previous visualization block and utilizes it in a later visualization block. It is designed based on several guidelines. Firstly, it is derived from the surface normals of the underlying 3D face model and encodes the rel-

ative pose between the face and camera, partially inspired by the success of using surface normals for 3D face recognition [79]. Secondly, the visualization layer is differentiable, which allows the gradient to be computed analytically and enables end-to-end training. Lastly, a mask is utilized to differentiate between pixels in the middle and contour areas of a face.

The last two chapters contain our proposed methods for the face anti-spoofing problem. In the fifth chapter, we propose a deep model for face anti-spoofing that uses the supervision from both the *spatial* and *temporal auxiliary information*, for the purpose of robustly detecting face presentation attacks (PA) from a face video. These auxiliary information are acquired based on our domain knowledge about the key *differences* between live and spoof faces, which include two perspectives: spatial and temporal. From the spatial perspective, it is known that live faces have face-like depth, e.g., the nose is closer to the camera than the cheek in frontal-view faces, while faces in print or replay attacks have flat or planar depth, e.g., all pixels on the image of a paper have the same depth to the camera. Hence, depth can be utilized as auxiliary information to supervise both live and spoof faces.

From the temporal perspective, it was shown that the normal rPPG signals (i.e., heart pulse signal) are detectable from live, but not spoof, face videos [69,81]. Therefore, we provide different rPPG signals as auxiliary supervision, which guides the network to learn from live or spoof face videos respectively. To enable both supervisions, we design a network architecture with a short-cut connection to capture different scales and a novel non-rigid registration layer to handle the motion and pose change for rPPG estimation.

Finally, in the sixth chapter, we propose a CNN method that, given a spoof face image, it can estimate the spoof noise and reconstruct the original live face. We propose several constraints and supervisions based on our prior knowledge of the spoof noise. First, a live face has no spoof noise. Second, we assume that the spoof noise of a spoof image is ubiquitous, i.e., it exists everywhere in

the spatial domain of the image; and, third, the spoof noise is repetitive, i.e., it is the spatial repetition of certain noise in the image. With such constraints, a novel CNN architecture is presented in the sixth chapter. Given an image, one CNN is designed to synthesize the spoof noise pattern and reconstruct the corresponding live image. In order to examine the reconstructed live image, we train another CNN with auxiliary supervision and a GAN-like discriminator in an end-to-end fashion. These two networks are designed to ensure the quality of the reconstructed image regarding its discriminativeness between live and spoof, and the visual plausibility of the synthesized live image.

1.4.1 Contributions of the thesis

In this section, we list some contributions already made towards pose invariant face alignment:

◇ We propose a pose-invariant face alignment by fitting a dense 3DMM, and integrating estimation of 3D shape and 2D facial landmarks from a single face image.

◇ We introduce 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.

◇ A visualization layer is presented which is differentiable, and allows backpropagation of error from a later block to an earlier one.

Also, we list some of the contributions made toward the face anti-spoofing:

◇ We propose to leverage novel auxiliary information (i.e., depth map and rPPG) to supervise the CNN learning for improved generalization of face anti-spoofing systems.

◇ We propose a novel CNN-RNN architecture for face anti-spoofing which performs end-to-end learning with the depth map and the rPPG signal.

◇ We offer a new perspective for detecting the spoofing face from print attack and replay attack

by inversely decomposing a spoof face image into the live face and the spoofing noise, without having the ground truth of either.

Chapter 2

Pose-Invariant 3D Face Alignment

2.1 Introduction

Motivated by the needs to address the pose variation, and the lack of prior work in handling poses, as shown in Fig. 2.1, this chapter proposes a novel regression-based approach for *pose-invariant face alignment*, which aims to estimate the 2*D and* 3*D locations* of face landmarks, as well as their *visibilities* in the 2D image, for a face with *arbitrary pose* (e.g., $\pm 90^{\circ}$ yaw). By extending the popular cascaded regressor for 2D landmark estimation, we learn two regressors for each cascade layer, one for predicting the update for the camera projection matrix, and the other for predicting the update for the 3D shape parameter. The learning of two regressors is conducted alternatively with the goal of minimizing the difference between the ground truth updates and the predicted updates. By using the 3D surface normals of 3D landmarks, we can automatically estimate the visibilities of their 2D projected landmarks by inspecting whether the transformed surface normal has a positive z coordinate, and these visibilities are dynamically incorporated into the regressor learning such that only the local appearance of visible landmarks contribute to the learning. Finally, extensive experiments are conducted on a large subset of AFLW dataset [57] with a wide range of poses, and the AFW dataset [151], with the comparison with a number of state-of-the-art methods. We demonstrate superior 2D alignment accuracy and quantitatively evaluate the 3D alignment accuracy.

In summary, the main contributions of the proposed pose invariant face alignment are:

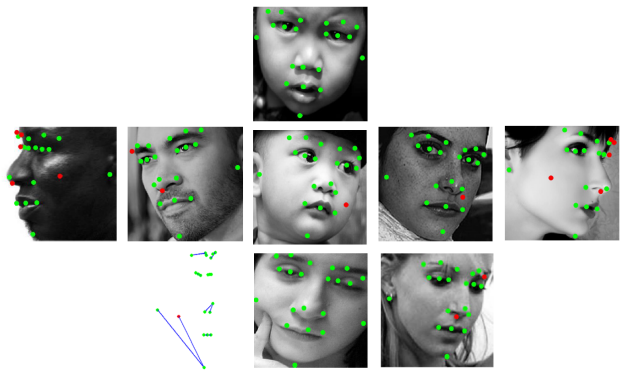


Figure 2.1: Given a face image with an arbitrary *pose*, our proposed algorithm automatically estimates the 2*D locations* and *visibilities* of facial landmarks, as well as 3*D landmarks*. The displayed 3D landmarks are estimated for the image in the center. Green/red points indicate visible/invisible landmarks.

- We propose a face alignment method that can estimate sparse set of 2D/3D landmarks and their visibilities for a face image with an arbitrary pose.
- By integrating with a 3D point distribution model, a cascaded coupled-regressor approach is developed to estimate both the camera projection matrix and the 3D landmarks, where 3D model enables the automatically computed landmark visibilities via surface normal.
- A substantially larger number of non-frontal view face images are utilized in evaluation with demonstrated superior performances than the state of the art.

2.2 Pose-Invariant 3D Face Alignment

This section presents the details of our proposed Pose-Invariant 3D Face Alignment (PIFA) algorithm, with emphasis on the training procedure. As shown in Fig. 2.2, we first learn a 3D Point Distribution Model (3DPDM) [31] from a set of labeled 3D scans, where a set of 2D landmarks on an image can be considered as a projection of a 3DPDM instance (i.e., 3D landmarks). For each 2D training face image, we assume that there exists the manual labeled 2D landmarks and their visibilities, as well as the corresponding 3*D ground truth*– 3D landmarks and the camera projection

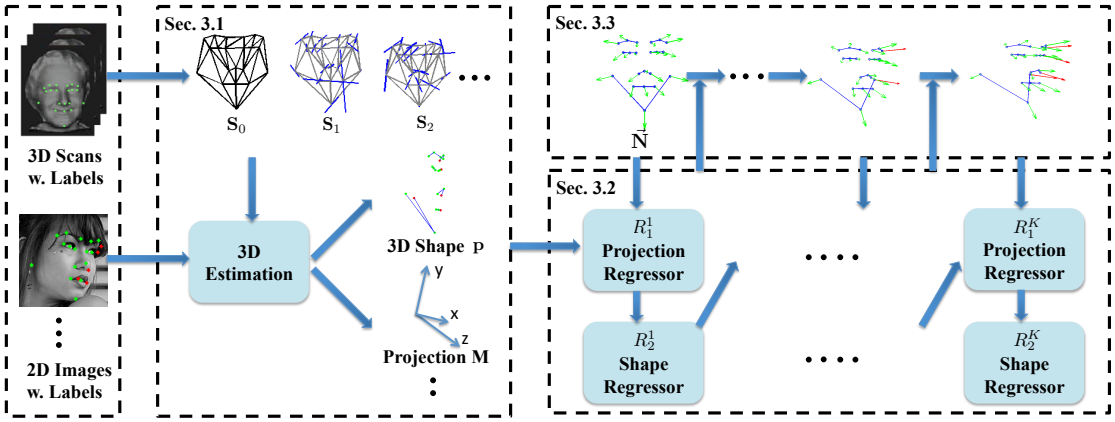


Figure 2.2: Overall architecture of our proposed PIFA method, with three main modules (3D modeling, cascaded coupled-regressor learning, and 3D surface-enabled visibility estimation). Green/red arrows indicate surface normals pointing toward/away from the camera.

matrix. Given the training images and 2D/3D ground truth, we train a cascaded coupled-regressor that is composed of two regressors at each cascade layer, for the estimation of the update of the 3DPDM coefficient and the projection matrix respectively. Finally, the visibilities of the projected 3D landmarks are automatically computed via the domain knowledge of the 3D surface normals, and incorporated into the regressor learning procedure.

2.2.1 3D Face Modeling

Face alignment concerns the 2D face shape, represented by the locations of N 2D landmarks, i.e.,

$$\mathbf{U} = \begin{pmatrix} u_1 & u_2 & \cdots & u_N \\ v_1 & v_2 & \cdots & v_N \end{pmatrix}. \tag{2.1}$$

A 2D face shape $\mathbf{U}$ is a projection of a 3D face shape $\mathbf{S}$, similarly represented by the homogeneous coordinates of N 3D landmarks, i.e.,

$$\mathbf{S} = \begin{pmatrix} x_1 & x_2 & \cdots & x_N \\ y_1 & y_2 & \cdots & y_N \\ z_1 & z_2 & \cdots & z_N \\ 1 & 1 & \cdots & 1 \end{pmatrix}. \tag{2.2}$$

Similar to the prior work [121], a weak perspective model is assumed for the projection,

$$\mathbf{U} = \mathbf{MS}, \tag{2.3}$$

where $\mathbf{M}$ is a 2×4 projection matrix with seven degrees of freedom (yaw, pitch, roll, two scales and 2D translations).

Following the basic idea of 3DPDM [31], we assume a 3D face shape is an instance of the 3DPDM,

$$\mathbf{S} = \mathbf{S}_0 + \sum_{i=1}^{N_s} p_i \mathbf{S}_i, \tag{2.4}$$

where $\mathbf{S}_0$ and $\mathbf{S}_i$ is the mean shape and ith shape basis of the 3DPDM respectively, N_s is the total number of shape bases, and p_i is the ith shape coefficient. Given a dataset of 3D scans with manual labels on N 3D landmarks per scan, we first perform procrustes analysis on the 3D scans to remove the global transformation, and then conduct Principal Component Analysis (PCA) to obtain the $\mathbf{S}_0$ and $\{\mathbf{S}_i\}$ (see the top-left part of Fig. 2.2).

The set of all shape coefficients $\mathbf{p} = (p_1, p_2, \cdots, p_{N_s})$ is termed as the 3*D shape parameter* of an image. At this point, the face alignment for a testing image $\mathbf{I}$ has been converted from the

estimation of $\mathbf{U}$ to the estimation of $\mathbf{P} = \{\mathbf{M}, \mathbf{p}\}$. The conversion is motivated by a few factors. First, without the 3D modeling, it is very difficult to model the out-of-plane rotation, which has a varying number of landmarks depending on the rotation angle and the individual 3D face shape. Second, as pointed out by [121], by only using $\frac{1}{6}$ of the number of the shape bases, 3DPDM can have an equivalent representation power as its 2D counterpart. Hence, using 3D model might lead to a more compact representation of unknown parameters.

Ground truth P Estimating $\mathbf{P}$ for a testing image implies the existence of ground truth $\mathbf{P}$ for each training image. However, while $\mathbf{U}$ can be manually labeled on a face image, $\mathbf{P}$ is normally unavailable unless a 3D scan is captured along with a face image. Therefore, in order to leverage the vast amount of existing 2D face alignment datasets, such as the AFLW dataset [57], it is desirable to estimate $\mathbf{P}$ for a face image and use it as the ground truth for learning.

Given a face image $\mathbf{I}$, we denote the manually labeled 2D landmarks as $\mathbf{U}$ and the landmark visibility as $\mathbf{v}$, an N-dim vector with binary elements indicating visible (1) or invisible (0) landmarks. Note that it is not necessary to label the 2D locations of invisible landmarks. We define the following objective function to estimate $\mathbf{M}$ and $\mathbf{p}$,

$$J(\mathbf{M}, \mathbf{p}) = \left|\left|\left(\mathbf{M}\left(\mathbf{S}_0 + \sum_{i=1}^{N_s} p_i \mathbf{S}_i\right) - \mathbf{U}\right) \odot \mathbf{V}\right|\right|^2, \tag{2.5}$$

where $\mathbf{V} = (\mathbf{v}^\intercal; \mathbf{v}^\intercal)$ is a $2 \times N$ visibility matrix, $\odot$ denotes the element-wise multiplication, and $||\cdot||^2$ is the sum of the squares of all matrix elements. Basically $J(\cdot, \cdot)$ computes the difference between the visible 2D landmarks and their 3D projections. An alternative estimation scheme is utilized, i.e., by assuming $\mathbf{p}^0 = 0$, we estimate $\mathbf{M}^k = \arg\min_{\mathbf{M}} J(\mathbf{M}, \mathbf{p}^{k-1})$, and then $\mathbf{p}^k = \arg\min_{\mathbf{p}} J(\mathbf{M}^k, \mathbf{p})$ iteratively until the changes of $\mathbf{M}$ and $\mathbf{p}$ are small enough. Both minimizations can be efficiently solved in closed forms via least-square error.

2.2.2 Cascaded Coupled-Regressor

For each training image $\mathbf{I}_i$, we now have its ground truth as $\mathbf{P}_i = \{\mathbf{M}_i, \mathbf{p}_i\}$, as well as their initialization, i.e., $\mathbf{M}_i^0 = g(\bar{\mathbf{M}}, \mathbf{b}_i)$, $\mathbf{p}_i^0 = \mathbf{0}$, and $\mathbf{v}_i^0 = \mathbf{1}$. Here $\bar{\mathbf{M}}$ is the average of ground truth projection matrices in the training set, $\mathbf{b}_i$ is a 4-dim vector indicating the bounding box location, and $g(\mathbf{M}, \mathbf{b})$ is a function that modifies the scale and translation of $\mathbf{M}$ based on $\mathbf{b}$. Given a dataset of N_d training images, the question is *how* to formulate an optimization problem to estimate $\mathbf{P}_i$. We decide to extend the successful cascaded regressors framework due to its accuracy and efficiency [26]. The general idea of cascaded regressors is to learn a series of regressors, where the kth regressor estimates the difference between the current parameter $\mathbf{P}_i^{k-1}$ and the ground truth $\mathbf{P}_i$, such that the estimated parameter gradually approximates the ground truth.

Motivated by this general idea, we adopt a cascaded coupled-regressor scheme where two regressors are learned at the kth cascade layer, for the estimation of $\mathbf{M}_i$ and $\mathbf{p}_i$ respectively. Specifically, the first learning task of the kth regressor is,

$$\Theta_1^k = \arg\min_{\Theta_1^k} \sum_{i=1}^{N_d} ||\Delta\mathbf{M}_i^k - R_1^k(\mathbf{I}_i, \mathbf{U}_i, \mathbf{v}_i^{k-1}; \Theta_1^k)||^2, \tag{2.6}$$

where

$$\mathbf{U}_i = \mathbf{M}_i^{k-1} \left(\mathbf{S}_0 + \sum_{i=1}^{N_s} p_i^{k-1} \mathbf{S}_i \right), \tag{2.7}$$

is the current estimated 2D landmarks, $\Delta\mathbf{M}_i^k = \mathbf{M}_i - \mathbf{M}_i^{k-1}$, and $R_1^k(\cdot; \Theta_1^k)$ is the desired regressor with the parameter of Θ_1^k. After Θ_1^k is estimated, we obtain $\Delta\hat{\mathbf{M}}_i = R_1^k(\cdot; \Theta_1^k)$ for all training images and update $\mathbf{M}_i^k = \mathbf{M}_i^{k-1} + \Delta\hat{\mathbf{M}}_i$. Note that this liner updating may potentially break the constraint of the projection matrix. Therefore, we estimate the scales and yaw, pitch, roll angles $(s_x, s_y, \alpha, \beta, \gamma)$ from $\mathbf{M}_i^k$ and compose a new $\mathbf{M}_i^k$ based on these five parameters.

Similarly the second learning task of the kth regressor is,

$$\Theta_2^k = \arg\min_{\Theta_2^k} \sum_{i=1}^{N_d} ||\Delta \mathbf{p}_i^k - R_2^k(\mathbf{I}_i, \mathbf{U}_i, \mathbf{v}_i^k; \Theta_2^k)||^2, \tag{2.8}$$

where $\mathbf{U}_i$ is computed via Eq 2.7 except $\mathbf{M}_i^{k-1}$ is replaced with $\mathbf{M}_i^k$. We also obtain $\Delta\hat{\mathbf{p}}_i = R_2^k(\cdot;\Theta_2^k)$ for all training images and update $\mathbf{p}_i^k = \mathbf{p}_i^{k-1} + \Delta\hat{\mathbf{p}}_i$. This iterative learning procedure continues for K cascade layers.

Learning $R^k(\cdot)$ Our cascaded coupled-regressor scheme does not depend on the particular feature representation or the type of regressors. Therefore, we may define them based on prior work or any future development in features and regressors. Specifically, in this work we adopt the HOG-based linear regressor [126] and the fern regressor [22].

For the linear regressor, we denote a function $f(\mathbf{I}, \mathbf{U})$ to extract HOG features around a small rectangular region of each one of N landmarks, which returns a $32N$-dim feature vector. Thus, we define the regressor function as

$$R(\cdot) = \Theta^\intercal \cdot \text{Diag}^*(\mathbf{v}_i) f(\mathbf{I}_i, \mathbf{U}_i), \tag{2.9}$$

where $\text{Diag}^*(\mathbf{v})$ is a function that duplicates each element of $\mathbf{v}$ 32 times and converts into a diagonal matrix of size $32N$. Note that we also add a constraint, $\lambda||\Theta||^2$, to Eq 2.6 or Eq 2.8 for a more robust least-square solution. By plugging Eq 2.9 to Eq 2.6 or Eq 2.8, the regressor parameter Θ (e.g., a $N_s \times 32N$ matrix for R_2^k) can be easily estimated in the closed form.

For the fern regressor, we follow the training procedure of [22]. That is, we divide the face region into a 3×3 grid. At each cascade layer, we choose 3 out of 9 zones with the least occlusion, computed based on the $\{\mathbf{v}_i^k\}$. For each selected zone, a depth 5 random fern regressor is learned

from the interpolated shape-indexed features selected by the correlation-based method [26] from that zone only. Finally the learned $R(\cdot)$ is a weighted mean voting from the 3 fern regressors, where the weight is inversely proportional to the average amount of occlusion in that zone.

2.2.3 3D Surface-Enabled Visibility

Up to now the only thing that has not been explained in the training procedure is how to estimate the visibility of the projected 2D landmarks, $\mathbf{v}_i$. It is obvious that during the testing we have to estimate $\mathbf{v}$ at each cascade layer for a testing image, since there is no visibility information given. As a result, during the training procedure, we also have to *estimate* $\mathbf{v}$ per cascade layer for each *training image*, rather than using the manually labeled ground truth visibility that is useful for estimating ground truth $\mathbf{P}$ as shown in Eq 2.5.

Depending on the camera projection matrix $\mathbf{M}$, the visibility of each projected 2D landmark may dynamically change along different layers of the cascade (see the top-right part of Fig. 2.2). In order to estimate $\mathbf{v}$, we decide to use the 3D face surface information. We start by assuming every individual has a similar 3D surface normal vector at each of its 3D landmarks. Then, by rotating the surface normal according to the rotation angle indicated by the projection matrix, we know that whether the rotated surface normal is pointing toward the camera (i.e., visible) or away from the camera (i.e., invisible). In other words, the sign of the z-axis coordinates indicates visibility.

By taking a set of 3D scans with manually labeled 3D landmarks, we can compute the landmarks' average 3D surface normals, denoted as a $3 \times N$ matrix $\vec{\mathbf{N}}$. Then we use the following equation to compute the visibility vector,

$$\mathbf{v} = \vec{\mathbf{N}}^{\mathsf{T}} \cdot \left(\frac{\mathbf{m}_1}{||\mathbf{m}_1||} \times \frac{\mathbf{m}_2}{||\mathbf{m}_2||} \right), \tag{2.10}$$

Data: 3D model $\{\{\mathbf{S}\}_{i=0}^{N_s}, \vec{\mathbf{N}}\}$, labeled data $\{\mathbf{I}_i, \mathbf{U}_i, \mathbf{b}_i\}_{i=1}^{N_d}$
Result: Cascaded regressor parameters $\{\Theta_1^k, \Theta_2^k\}_{k=1}^{K}$

/* 3D modeling */
foreach $i = 1, \cdots, N_d$ **do**
    Estimate $\mathbf{M}_i$ and $\mathbf{p}_i$ via Eq. 2.5

/* Initialization */
foreach $i = 1, \cdots, N_d$ **do**
    $\mathbf{p}_i^0 = \mathbf{0}$; ▷ Assuming the mean 3D shape
    $\mathbf{v}_i^0 = \mathbf{1}$; ▷ Assuming all landmarks visible
    $\mathbf{M}_i^0 = g(\bar{\mathbf{M}}, \mathbf{b}_i)$ and $\mathbf{U}_i = \mathbf{M}_i^0 \mathbf{S}_0$

/* Regressor learning */
foreach $k = 1, \cdots, K$ **do**
    Estimate Θ_1^k via Eq 2.6
    Update $\mathbf{M}_i^k$ and $\mathbf{U}_i$ for all images
    Compute $\mathbf{v}_i^k$ via Eq 2.10 for all images
    Estimate Θ_2^k via Eq 2.8 ;
    Update $\mathbf{p}_i^k$ and $\mathbf{U}_i$ for all images .

Figure 2.3: The training procedure of PIFA.

where $\mathbf{m}_1$ and $\mathbf{m}_2$ are the left-most three elements at the first and second row of $\mathbf{M}$ respectively, and $||\cdot||$ denotes the L_2 norm. For fern regressors, $\mathbf{v}$ is a soft visibility within ± 1. For linear regressors, we further compute $\mathbf{v} = \frac{1}{2}(1 + \text{sign}(\mathbf{v}))$, which results in a hard visibility of either 1 or 0. In summary, we present the detailed training procedure in Algorithm 2.3.

Model fitting Given a testing image $\mathbf{I}$ with bounding box $\mathbf{b}$ and its initial parameter $\mathbf{M}^0 = g(\bar{\mathbf{M}}, \mathbf{b})$ and $\mathbf{p}^0 = \mathbf{0}$, we can apply the learned cascaded coupled-regressor for face alignment. Basically we iteratively use $R_1^k(\cdot; \Theta_1^k)$ to compute $\Delta\hat{\mathbf{M}}$, update $\mathbf{M}^k$, compute $\mathbf{v}^k$, use $R_2^k(\cdot; \Theta_2^k)$ to compute $\Delta\hat{\mathbf{p}}$, and update $\mathbf{p}^k$. Finally the estimated 3D landmarks are $\hat{\mathbf{S}} = \mathbf{S}_0 + \sum_i p_i^K \mathbf{S}_i$, and the estimated 2D landmarks are $\hat{\mathbf{U}} = \mathbf{M}^K \hat{\mathbf{S}}$. Note that $\hat{\mathbf{S}}$ carries the individual 3D shape information of the subject, but not necessary in the same pose as the 2D testing image.

2.3 Experimental Results

Datasets The goal of this work is to advance the capability of face alignment on **in-the-wild faces with all possible view angles**, which is the type of images we desire when selecting experimental datasets. However, very few publicly available datasets satisfy this characteristic, or have been extensively evaluated in prior work (see Tab. 2.1). Nevertheless, we identify three datasets for our experiments.

AFLW dataset [57] contains $\sim 25,000$ in-the-wild face images, each image annotated with the *visible* landmarks (up to 21 landmarks), and a bounding box. Based on our estimated $\mathbf{M}$ for each image, we select a subset of $5,200$ images where the numbers of images whose absolute yaw angles within $[0°, 30°]$, $[30°, 60°]$, $[60°, 90°]$ are roughly $\frac{1}{3}$ each. To have a more *balanced distribution* of the left vs. right view faces, we take the odd indexed images among $5,200$ (i.e., 1st, 3rd), flip them horizontally, and use them to replace the original images. Finally, a random partition leads to $3,901$ and $1,299$ images for training and testing respectively. As shown in Tab. 2.1, among the methods that test on all poses, we have the largest number of testing images.

AFW dataset [151] contains 205 images and in total 468 faces with different poses within $\pm 90°$. Each image is labeled with *visible* landmarks (up to 6), and a face bounding box. We only use AFW for testing.

Since we are also estimating 3D landmarks, it is important to test on a dataset with *ground*

Table 2.1: The comparison of face alignment algorithms in pose handling (estimation errors may have different definitions).

Method	3D landmark	Visibility	Pose-related database	Pose range	Training face #	Testing face #	Landmark #	Estimation errors
RCPR [22]	No	Yes	COFW	frontal w. occlu.	1,345	507	19	8.5
CoR [136]	No	Yes	COFW; LFPW-O; Helen-O	frontal w. occlu.	1,345;468;402	507;112;290	19;49;49	8.5
TSPM [151]	No	No	AFW	all poses	2,118	468	6	11.1
CDM [135]	No	No	AFW	all poses	1,300	468	6	9.1
OSRD [123]	No	No	MVFW	$< \pm 40°$	2,050	450	68	N/A
TCDCN [142]	No	No	AFLW, AFW	$< \pm 60°$	10,000	3,000;$\sim$ 313	5	8.0;8.2
PIFA	Yes	Yes	AFLW, AFW	all poses	3,901	1,299;468	21,6	6.5;8.6

truth, rather than estimated, 3D landmark locations. We find BP4D-S database [141] to be the best for this purpose, which contains pairs of 2D images and 3D scans of spontaneous facial expressions from 41 subjects. Each pair has semi-automatically generated 83 2D and 83 3D landmarks, and the pose. We apply a random perturbation on 2D landmarks (to mimic imprecise face detection) and generate their enclosed bounding box. With the goal of selecting as many non-frontal view faces as possible, we choose a subset where the numbers of faces whose yaw angle within $[0^\circ, 10^\circ]$, $[10^\circ, 20^\circ]$, $[20^\circ, 30^\circ]$ are 100, 500, and 500 respectively. We randomly select half of $1,100$ images for training and the rest for testing, with disjoint subjects.

Experiment setup Our PIFA approach needs a 3D model of $\{\mathbf{S}\}_{i=0}^{N_s}$ and $\vec{\mathbf{N}}$. Using the BU-4DFE database [133] that contains 606 3D facial expression sequences from 101 subjects, we evenly sample 72 scans from each sequence and gather a total of 72×606 scans. Based on the method in Sec. 2.2.1, the resultant model has $N_s = 30$ for AFLW and AFW, and $N_s = 200$ for BP4D-S.

During the training and testing, for each image with a bounding box, we place the mean 2D landmarks (learned from the training set) on the image such that the landmarks on the boundary are within the four edges of the box. For training with linear regressors, we set $K = 10$, $\lambda = 120$, while $K = 75$ for fern regressors.

Evaluation metric 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 have two ways of computing the landmark estimation errors: 1) Mean Average Pixel Error (MAPE) [135], which is the average of the estimation errors for visible landmarks, i.e.,

$$\text{MAPE} = \frac{1}{\sum_i^{N_t} |\mathbf{v}_i|_1} \sum_{i,j}^{N_t,N} \mathbf{v}_i(j) ||\hat{\mathbf{U}}_i(:,j) - \mathbf{U}_i(:,j)||, \tag{2.11}$$

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.,

$$\text{NME} = \frac{1}{N_t}\sum_{i}^{N_t}(\frac{1}{d_i|\mathbf{v}_i|_1}\sum_{j}^{N}\mathbf{v}_i(j)||\hat{\mathbf{U}}_i(:,j)-\mathbf{U}_i(:,j)||), \tag{2.12}$$

where d_i is the square root of the face bounding box size, as used by [135]. Note that normally d_i is the inter-eye distance in prior face alignment work dealing with near-frontal faces.

Given the ground truth 3D landmarks $\mathbf{S}_i$ and estimated landmarks $\hat{\mathbf{S}}_i$, we first estimate the global rotation, translation and scale transformation so that the transformed $\mathbf{S}_i$, denoted as $\mathbf{S}'_i$, has the minimum distance to $\hat{\mathbf{S}}_i$. We then compute the MAPE via Eq 2.11 except replacing $\mathbf{U}$ and $\hat{\mathbf{U}}_i$ with $\mathbf{S}'_i$ and $\hat{\mathbf{S}}_i$, and $\mathbf{v}_i = \mathbf{1}$. Thus the MAPE only measures the error due to non-rigid shape deformation, rather than the pose estimation.

Choice of baseline methods 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. In this work, we select three recent works as baseline methods: 1) CDM [135] is a CLM-type method and the first one claimed to perform pose-free face alignment, which has exactly the same objective as ours. On AFW it also outperforms the other well-known TSPM method [151] that can handle all pose faces. 2) TCDCN [142] is a powerful deep learning-based method published in the most recent ECCV. Although it only estimates 5 landmarks for up to $\sim 60^\circ$ yaw, it represents the recent development in face alignment. 3) RCPR [22] is a regression-type method that represents the occlusion-invariant face alignment. Although it is an earlier work than CoR [136], we choose it due to its superior performance on the large COFW dataset (see Tab. 1 of [136]). It can be seen that these three baselines not only are most relevant to our focus on pose-invariant face alignment, but also well represent the major categories of existing face alignment algorithms based on [115].

Table 2.2: The NME(%) of three methods on AFLW.

N_t	PIFA	CDM	RCPR
1,299	**6.52**		7.15
783	**6.08**	8.65	

Table 2.3: The comparison of four methods on AFW.

N_t	N	Metric	PIFA	CDM	RCPR	TCDCN
468	6	MAPE	**8.61**	9.13		
313	5	NME	9.42		9.30	**8.20**

Comparison on AFLW Since the source code of RCPR is publicly available, we are able to perform the training and testing of RCPR on our specific AFLW partition. We use the available executable of CDM to compute its performance on our test set. We strive to provide the same setup to the baselines as ours, such as the initial bounding box, regressor learning, etc. For our PIFA method, we use the fern regressor. Because CDM integrates face detection and pose-free face alignment, no bounding box was given to CDM and it successfully detects and aligns 783 out of 1,299 testing images. Therefore, to compare with CDM, we evaluate the NME on the *same* 783 testing images. As shown in Tab. 2.2, our PIFA shows superior performance to both baselines. Although TCDCN also reports performance on a subset of 3,000 AFLW images within $\pm 60^\circ$ yaw, it is evaluated with 5 landmarks, based on NME when d_i is the inter-eye distance. Hence, without the source code of TCDCN, it is difficult to have a fair comparison on our subset of AFLW images (e.g., we can not define d_i as the inter-eye distance due to profile view faces). On the 1,299 testing images, we also test our method with linear regressors, and achieve a NME of 7.50, which shows the strength of fern regressors.

Comparison on AFW Unlike our specific subset of AFLW, the AFW dataset has been evaluated by all three baselines, but different metrics are used. Therefore, the results of the baselines in Tab. 2.3 are from the published papers, instead of executing the testing code. One note is that from the TCDCN paper [142], it appears that all 5 landmarks are visible on all displayed images and

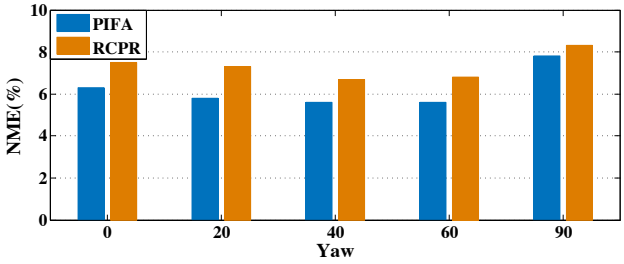


Figure 2.4: The NME of five pose groups for two methods.

no visibility estimation is shown, which might suggest that TCDCN was evaluated on a subset of AFW with up to $\pm 60^\circ$ yaw. Hence, we select the total of 313 out of 468 faces within this pose range and test our algorithm. Since it is likely that our subset could differ to [142], please take this into consideration while comparing with TCDCN. Overall, our PIFA method still performs comparably among the four methods. This is especially encouraging given the fact that TCDCN utilizes a substantially larger training set of $10,000$ images - more than two times of our training set. Note that in addition to Tab. 2.2 and 2.3, our PIFA also has other benefits as shown in Tab. 2.1. E.g., we have 3D and visibility estimation, while RCPR has no 3D estimation and TCDCN does not have visibility estimation.

Estimation error across poses Just like pose-invariant face recognition studies the recognition rate across poses [71, 72], we also like to study the performance of face alignment across poses. As shown in Fig. 2.4, based on the estimated projection matrix $\mathbf{M}$ and its yaw angles, we partition all testing images of AFLW into five bins, each around a specific yaw angle. Then we compute the NME of testing images within each bin, for our method and RCPR. We can observe that the profile view images have in general larger NME than near-frontal images, which shows the challenge of pose-invariant face alignment. Further, the improvement of PIFA over RCPR is consistent across most of the poses.

Estimation error across landmarks We are also interested in the estimation error across various

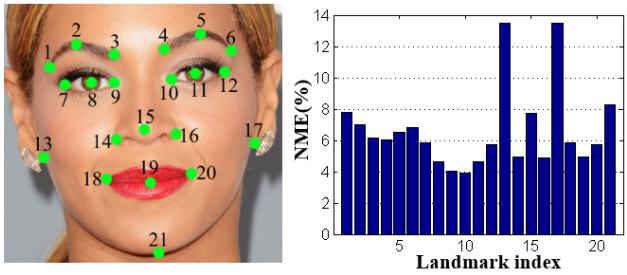


Figure 2.5: The NME of each landmark for PIFA.

Table 2.4: Efficiency of four methods in FPS.

PIFA	CDM	RCPR	TCDCN
3.0	0.2	3.0	**58.8**

landmarks, under a wide range of poses. Hence, for the AFLW test set, we compute the NME of each landmark for our method. As shown in Fig. 2.5, the two eye regions have the least amount of error. The two landmarks under the ears have the most error, which is consistent with the intuition. These observations also align well with prior face alignment study on near-frontal faces.

3D landmark estimation By performing the training and testing on the BP4D-S dataset, we can evaluate the MAPE of 3D landmark estimation, with exemplar results shown in Fig. 2.6. Since there are limited 3D alignment work and many of which do not perform quantitative evaluation, such as [43], we are not able to find another method as the baseline. Instead, we use the 3D mean shape, $\mathbf{S}_0$, as a baseline and compute its MAPE with respect to the ground truth 3D landmarks $\mathbf{S}_i$ (after global transformation). We find that the MAPE of $\mathbf{S}_0$ baseline is 5.02, while our method has 4.75. Although our method offers a better estimation than the mean shape, this shows that 3D face alignment is still a very challenging problem. We hope the effort to quantitatively measure the 3D estimation error, which is more difficult than its 2D counterpart, will encourage more research activities to address this challenge.

Computational efficiency Based on the efficiency reported in the publications of baseline methods, we compare the computational efficiency of four methods in Tab. 2.4. Only TCDCN is mea-

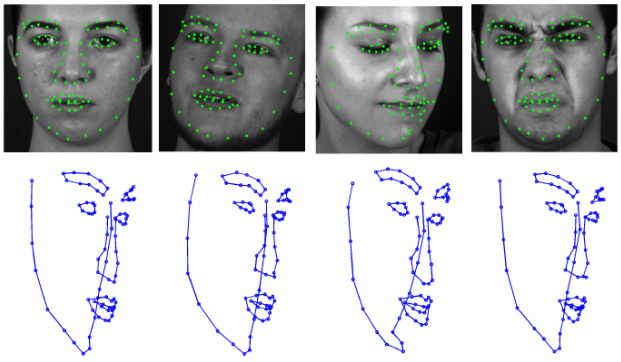


Figure 2.6: 2D and 3D alignment results of the BP4D-S dataset.

sured based on the C implementation while other three are all based on Matlab implementation. It can be observed that TCDCN is the most efficient one. Consider that we estimate both 2D and 3D landmarks, at 3 FPS our unoptimized implementation is reasonably efficient. In our algorithm, the most computational demanding part is feature extraction, while estimating the updates for the projection matrix and 3D shape parameter has closed-form solutions and is very efficient.

Qualitative results We now show the qualitative face alignment results for images in two datasets. As shown in Fig. 2.7, despite the large pose range of $\pm 90^{\circ}$ yaw, our algorithm does a good job of aligning the landmarks, and correctly predict the landmark visibilities. These results are especially impressive if you consider the same mean shape (2D landmarks) is used as the initialization of all testing images, which has very large deformations with respect to their final landmark estimation.

2.4 Summary

Motivated by the fast progress of face alignment technologies and the need to align faces at all poses, this chapter draws attention to a relatively less explored problem of face alignment robust to poses variation. To this end, we propose a novel approach to tightly integrate the powerful cascaded regressor scheme and the 3D face model. The 3D model not only serves as a compact constraint,

Figure 2.7: Testing results of AFLW (top) and AFW (bottom). As shown in the top row, we initialize face alignment by placing a 2D mean shape in the given bounding box of each image. Note the *disparity* between the initial landmarks and the final estimated ones, as well as the diversity in pose, illumination and resolution among the images. Green/red points indicate visible/invisible estimated landmarks.

but also offers an automatic and convenient way to estimate the visibilities of 2D landmarks - a key for successful pose-invariant face alignment. As a result, for a 2D image, our approach estimates the locations of 2D and 3D landmarks, as well as their 2D visibilities. We conduct an extensive experiment on a large collection of all-pose face images and compare with three state-of-the-art methods. While superior 2D landmark estimation has been shown, the performance on 3D landmark estimation indicates the future direction to improve this line of research.

Chapter 3

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

3.1 Introduction

In the previous chapter, we propose the PIFA method which can estimate the locations of a sparse set of 3D landmark points. In this chapter, we extend PIFA in a number of ways:

First, 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, we propose to use Convolutional Neural Networks (CNN) as the regressor in the cascaded framework, to learn the mapping. The main advantage of CNN over the fern regression trees (in the previous chapter) is that it does not depend on hand-crafted feature extraction methods. The CNN can learn and extract more meaningful, generalizable and abstract features by hierarchical representation. This property is more important in pose-invariant face alignment because a change in the head pose (frontal to side-view) makes a considerable difference in the face images.

While most prior work on CNN-based face alignment estimate no more than six 2D landmarks per image [99, 142], our cascaded CNN can produce a substantially larger number (34) of 2D and 3D landmarks. Further, using landmark marching [150], our algorithm can adaptively adjust the

Figure 3.1: 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.

3D landmarks during the fitting, so that the local appearances around cheek landmarks contribute to the fitting process.

Third, we propose two novel pose-invariant local features, as the input layer for CNN learning. We utilize the dense 3D face model as an oracle to build dense feature *correspondence* across various poses and expressions. 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 [20]. Although this mirrorability constraint was an alignment accuracy measure used in post-processing [129], we integrate it directly in CNN learning.

These algorithm designs collectively lead to the extended 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 [61] and AFW [151], 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.

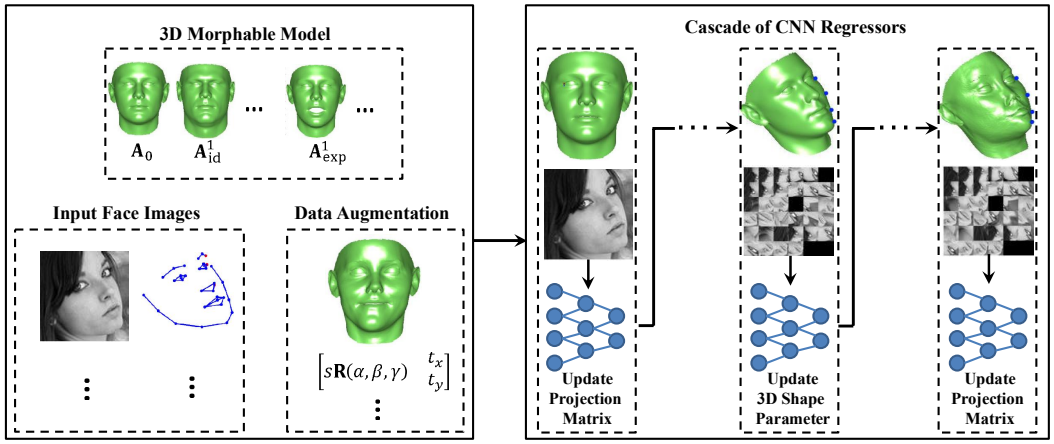


Figure 3.2: The overall process of the proposed method.

3.2 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 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 [52]. Fig. 3.2 shows the overall process of the proposed method.

3.2.1 3D Morphable Model

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

$$\mathbf{S} = \mathbf{S}_0 + \sum_{i=1}^{N_{id}} p_{id}^i \mathbf{S}_{id}^i + \sum_{i=1}^{N_{exp}} p_{exp}^i \mathbf{S}_{exp}^i, \tag{3.1}$$

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 [86] and the face wearhouse as the expression bases [25]. 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,

$$\mathbf{S} = \begin{pmatrix} x_1 & x_2 & \cdots & x_Q \\ y_1 & y_2 & \cdots & y_Q \\ z_1 & z_2 & \cdots & z_Q \end{pmatrix}. \tag{3.2}$$

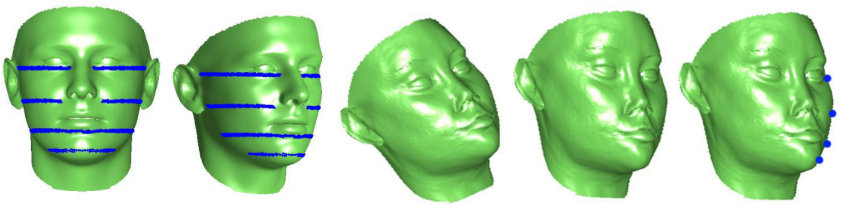
Any 3D face model will be projected onto a 2D image where the face shape may be represented as a sparse set of N landmarks, on the facial fiducial points. We denote x and y coordinates of these 2D landmarks as a matrix $\mathbf{U}$,

$$\mathbf{U} = \begin{pmatrix} u_1 & u_2 & \cdots & u_N \\ v_1 & v_2 & \cdots & v_N \end{pmatrix}. \tag{3.3}$$

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

$$\mathbf{U} = s\mathbf{R}\mathbf{S}(:, \mathbf{d}) + \mathbf{t}, \tag{3.4}$$

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 [48, 121].



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

Figure 3.3: 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.

At this point, we can represent any 2D face shape as the projection of a 3D face shape. In other words, the projection parameter **m** and shape parameter **p** can uniquely represent a 2D face shape. Therefore, the face alignment problem amounts to estimating **m** and **p**, given a face image. Estimating **m** and **p** instead of estimating **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 [121], 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 Eqn. 3.4 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 Eqn. 3.4, 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 [24, 92, 150]. In this paper, we leverage the landmark marching method proposed in [150].

Specifically, we define a set of *paths* each storing the indexes of vertexes that are not only the

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

Figure 3.4: Landmark marching $g(\mathbf{S}, \mathbf{m})$.

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 $\mathbf{S}$, by ignoring the roll rotation γ, we rotate $\mathbf{S}$ 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 $\mathbf{d}$ to make sure these vertexes are selected in the projection of Eqn. 3.4. This landmark marching process is summarized in Algorithm 3.4 as a function $\mathbf{d} \leftarrow g(\mathbf{S}, \mathbf{m})$. Note that when the face is approximately of profile view ($|\beta| > 70\,^\circ$), we do not apply landmark marching since the marched landmarks would overlap with the existing 2D landmarks on the middle of nose and mouth. Fig. 3.3 shows the defined set of pathes on the 3D shape of face and one example of applying Algorithm 3.4 for updating vector $\mathbf{d}$.

3.2.2 Data Augmentation

Given that the projection matrix parameter $\mathbf{m}$ and shape parameter $\mathbf{p}$ are the representation of a face shape, we should have a collection of face images with ground truth $\mathbf{m}$ and $\mathbf{p}$ so that the learning algorithm can be applied. However, while $\mathbf{U}$ can be manually labeled on a face image, $\mathbf{m}$ and $\mathbf{p}$ 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 [61], 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:

$$J(\mathbf{m},\mathbf{p}) = ||(s\mathbf{R}\mathbf{S}(:,g(\mathbf{S},\mathbf{m})) + \mathbf{t} - \mathbf{U}) \odot \mathbf{V}||_F^2, \tag{3.5}$$

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.

3.2.2.1 Optimization

For convenient optimization of Eqn. 3.5, we redefine all projection parameters as a projection matrix, i.e.,

$$\mathbf{M} = \begin{bmatrix} s\mathbf{R} & \begin{matrix} t_x \\ t_y \end{matrix} \end{bmatrix} \in \mathbb{R}^{2\times 4}. \tag{3.6}$$

Also, we denote $\mathbf{d} = g(\mathbf{S},\mathbf{m})$ in Eqn. 3.5 by assuming it is a constant given the currently estimated **m** and **p**. We then rewrite Eqn. 3.5 as,

$$J(\mathbf{M},\mathbf{p}) = \left|\left|\left(\mathbf{M}\begin{bmatrix} \mathbf{S}(:,\mathbf{d}) \\ \mathbf{1}^\intercal \end{bmatrix} - \mathbf{U}\right) \odot \mathbf{V}\right|\right|_F^2. \tag{3.7}$$

To minimize this objective function, we alternate the minimization w.r.t. $\mathbf{M}$ and $\mathbf{p}$ at each iteration. We initialize the 3D shape parameter $\mathbf{p} = \mathbf{0}$ and estimate $\mathbf{M}$ first, by $\mathbf{M}^k = \arg\min_{\mathbf{M}} J(\mathbf{M}, \mathbf{p}^{k-1})$,

$$\mathbf{M}^k = \mathbf{U}_V \begin{bmatrix} \mathbf{S}(:, \mathbf{d}_V) \\ \mathbf{1}^\intercal \end{bmatrix}^\intercal \left(\begin{bmatrix} \mathbf{S}(:, \mathbf{d}_V) \\ \mathbf{1}^\intercal \end{bmatrix} \begin{bmatrix} \mathbf{S}(:, \mathbf{d}_V) \\ \mathbf{1}^\intercal \end{bmatrix}^\intercal \right)^{-1}, \tag{3.8}$$

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_{id}^i is the contribution of i-th identity basis in reconstructing the dense 3D face shape,

$$p_{id}^i = \frac{\mathrm{Tr}\left(\hat{\mathbf{U}}_V^\intercal \hat{\mathbf{U}}_{id_i}\right)}{\mathrm{Tr}\left(\hat{\mathbf{U}}_{id_i}^\intercal \hat{\mathbf{U}}_{id_i}\right)}, \tag{3.9}$$

where

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

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_{id}^i is computed, we update $\hat{\mathbf{U}}_V$ by subtracting the contribution of i-th basis and continue to compute p_{id}^{i+1}. 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 Eqn. 3.8 for computing a new estimation of $\mathbf{M}$ and decomposing to its parameters $\mathbf{m}$, we apply the landmark marching algorithm (Algorithm 3.4) to update the vector $\mathbf{d}$.

3.2.3 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 [88], face detection [64], and face alignment [142], 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, $\mathbf{m}_i^0 = h(\bar{\mathbf{m}}, \mathbf{b}_i)$ and $\mathbf{p}_i^0 = \mathbf{0}$. 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,

$$\Theta_m^k = \arg\min_{\Theta_m^k} J_\Theta = \sum_{i=1}^{N_d} ||\Delta\mathbf{m}_i^k - \mathrm{CNN}_m^k(\mathbf{I}_i, \mathbf{U}_i, \mathbf{v}_i^{k-1}; \Theta_m^k)||^2, \tag{3.10}$$

where the true projection update is the difference between the current projection matrix parameter and the ground truth, i.e., $\Delta\mathbf{m}_i^k = \mathbf{m}_i - \mathbf{m}_i^{k-1}$, $\mathbf{U}_i$ is current estimated 2D landmarks, computed via Eqn. 3.4, based on $\mathbf{m}_i^{k-1}$ and $\mathbf{d}_i^{k-1}$, and $\mathbf{v}_i^{k-1}$ is estimated landmark visibility at stage $k-1$.

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

$$\Theta_p^k = \arg\min_{\Theta_p^k} J_\Theta = \sum_{i=1}^{N_d} ||\Delta \mathbf{p}_i^k - \text{CNN}_p^k(\mathbf{I}_i, \mathbf{U}_i, \mathbf{v}_i^k; \Theta_p^k)||^2. \tag{3.11}$$

Note that $\mathbf{U}_i$ will be re-computed via Eqn. 3.4, based on the updated $\mathbf{m}_i^k$ and $\mathbf{d}_i^k$ by CNN_m.

We use a six-stage cascaded CNN, including CNN_m^1, CNN_m^2, CNN_p^3, CNN_m^4, CNN_p^5, and CNN_m^6. At the first stage, the input layer of CNN_m^1 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_m^k or Θ_p^k. 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 [52] 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, in stead of alternating between the estimation of $\mathbf{m}$ and $\mathbf{p}$, another option is to jointly estimate both parameters in each CNN stage. Experimentally we observed that such

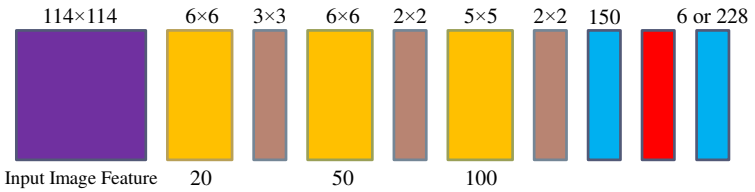


Figure 3.5: Architecture of C-CNN (the same CNN architecture is used for all six stages). Color code used: purple = extracted image feature, orange = Conv, brown = pooling + batch normalization, blue = fully connected layer, red = ReLU. The filter size and the number of filters for each layer are shown on the top and the bottom respectively.

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.

3.2.4 Conventional CNN (C-CNN)

The architecture of the first CNN is shown in Fig. 3.5. 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) [41] as the activation function which enables CNN to achieve the best performance without unsupervised pre-training.

3.2.5 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, 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 $\mathbf{m}$ and $\mathbf{p}$ 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 [129] 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 [20] by sharing the network's weights between the input face image and its mirror image and adding a new loss function.

3.2.5.1 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 $\mathbf{b}$, 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,

$$J_M^k = ||\Delta\hat{\mathbf{U}}^k - C\left(\Delta\hat{\mathbf{U}}_M^k\right)||^2, \tag{3.12}$$

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.

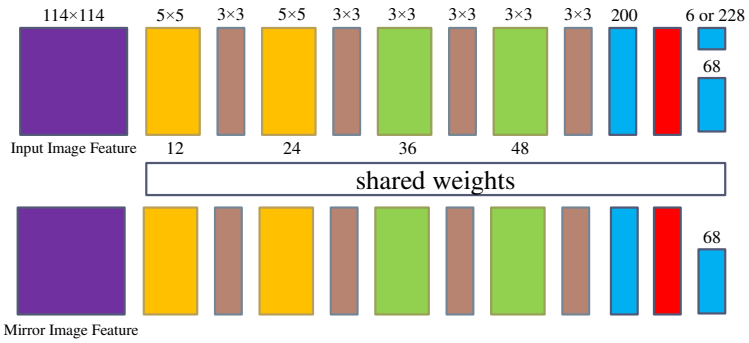


Figure 3.6: Architecture of the M-CNN (the same CNN architecture is used for all six stages). Color code used: purple = extracted image feature, orange = Conv, brown = pooling + batch normalization, green = locally connected layer, blue = fully connected layer, red = batch normalization + ReLU + dropout. The filter size and the number of filters of each layer are shown on the top and the bottom of the top branch respectively.

3.2.5.2 Mirror CNN Architecture

The new CNN architecture follows the siamese network [20] with two branches whose weights are shared. Fig. 3.6 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 [102] 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 estimating the update of parameters (J_Θ) and the other one for estimating the update of 2D shape via the loss (J_U),

$$J_U^k = ||\Delta \mathbf{U}^k - \Delta \hat{\mathbf{U}}^k||^2. \tag{3.13}$$

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 **m** or **p**, two for the 2D shape updates of two images respectively, and one mirror loss. We minimize the total loss at stage k,

$$J_T^k = J_\Theta^k + \lambda_1 J_U^k + \lambda_2 J_{MU}^k + \lambda_3 J_M^k, \tag{3.14}$$

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 **m** or **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.

3.2.6 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 [52]. Instead of using the average 3D surface normal for all humans, we extend it by using person-specific 3D surface normal. Specifically, given the current estimated 3D shape **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 $\vec{\mathbf{N}}$. Fig. 3.7

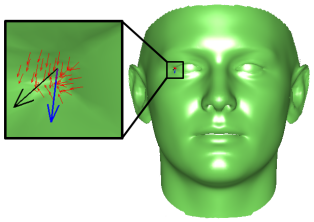


Figure 3.7: The person-specific 3D surface normal as the average of normals around a 3D landmark (black arrow). Notice the relatively noisy surface normal of the 3D "left eye corner" landmark (blue arrow).

illustrates the advantage of using the average 3D surface normal. Given $\vec{\mathbf{N}}$, we compute,

$$\mathbf{v} = \vec{\mathbf{N}}^{\mathsf{T}} \cdot (\mathbf{R}_1 \times \mathbf{R}_2), \tag{3.15}$$

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 [128], 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. 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

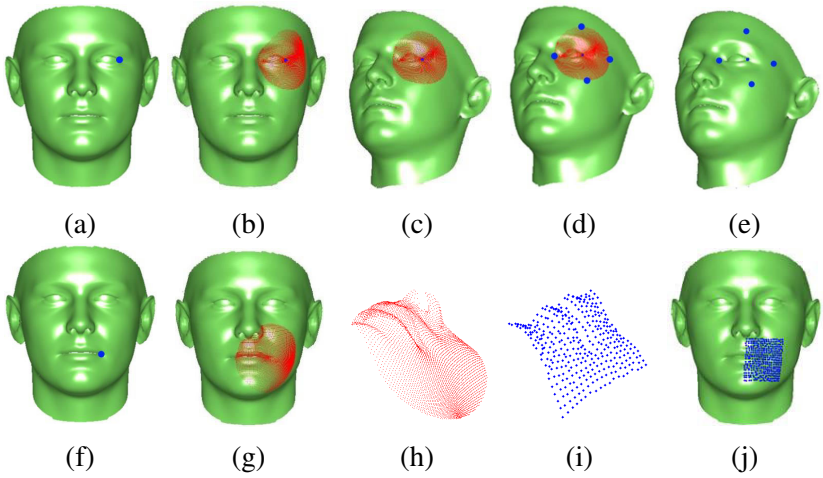


Figure 3.8: Feature extraction process, (a-e) PAWF for the landmark on the right side of the right eye, (f-j) D3PF for the landmark on the right side of the lip.

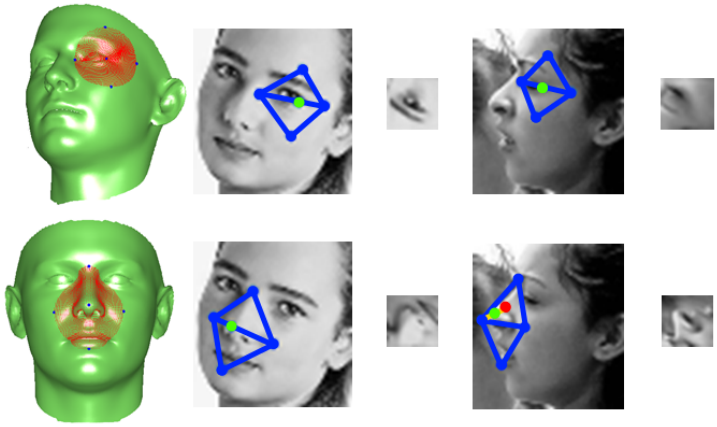


Figure 3.9: Examples of extracting PAWF. When one of the four neighborhood points (red point in the bottom-right) is invisible, it connects to the 2D landmark (green point), extends the same distance further, and generate a new neighborhood point. This helps to include the background context around the nose.

from another image, even both patches are centered at the ground truth locations of the same *n*th 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.

In an offline procedure, we first search for T vertexes on the mean 3D shape $\mathbf{S}_0$ that are the

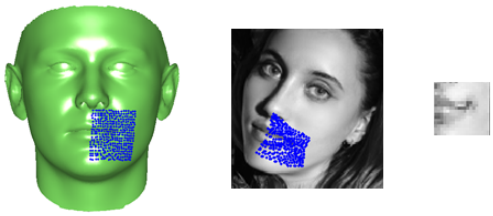


Figure 3.10: Example of extracting D3PF.

most closest to the nth landmark (Fig. 3.8(b)). Second, we rotate the T vertexes such that the 3D surface normal of the nth landmark points toward the camera (Fig. 3.8(c)). 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. 3.8(d)). The first row of Fig. 3.8 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 [78].

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. 3.9.

Direct 3D projected feature (D3PF) Both D3PF and PAWF start with the T vertexes surrounding the nth 3D landmark (Fig. 3.8(g)). 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. 3.8(i)). The second row of Fig. 3.8 shows the process of extracting D3PF. Similar to

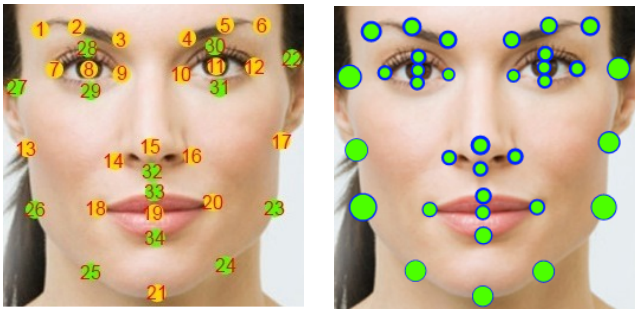
PAWF, we can now project the set of 3D vertexes $\mathbf{S}(:,\mathbf{d}_d^{(n)})$ to the 2D image and extract a 19×19 patch via bilinear-interpolation, as shown in Fig. 3.10. We also estimate the visibilities of the 3D vertexes $\mathbf{S}(:,\mathbf{d}_d^{(n)})$ via their surface normals, and zero will 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.

3.2.7 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 Eqn. 3.1, 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.

3.3 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)

Figure 3.11: (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.

3.3.1 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 [61] 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 [52], a subset of AFLW is selected to have a balanced distribution of yaw angles, including $3,901$ images for training and $1,299$ images for testing. We use the same subset and manually label 13 additional landmarks for all $5,200$ images. We call these $3,901$ images as the *base* training set. The definition of original landmarks and added landmarks is shown in Fig. 3.11(a). 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. 3.17). These randomly generated bounding boxes mimic the imprecise face detection window and will be used for both training and testing.

AFW dataset [151] 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 [86] for representing identity variation and the $N_{exp} = 29$ bases of face wearhouse [25] 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 $1,035$ subjects of LFPW dataset [7] by the method of [150], 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. We compare the proposed method with two methods on AFLW: 1) PIFA [52] 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 [22] 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 [52], 2) CDM [135] is a method based on Constrained Local Model (CLM) and the first one claimed to perform pose-free face alignment, 3) TSPM [151] 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 [113] with a constant learning rate of $1e{-}4$, with ten epochs for training each CNN and a batch size of 100.

Table 3.1: 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

For the M-CNN architecture, we use the Caffe toolbox [49] 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 Eqn. 3.14 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 $5,000$, and $3,000$ 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) [135], which is the average of the estimation errors for visible landmarks, i.e.,

$$\text{MAPE} = \frac{1}{\sum_i^{N_t} |\mathbf{v}_i|_1} \sum_{i,j}^{N_t,N} \mathbf{v}_i(j) ||\hat{\mathbf{U}}_i(:,j) - \mathbf{U}_i(:,j)||, \tag{3.16}$$

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.,

$$\text{NME} = \frac{1}{N_t} \sum_i^{N_t} (\frac{1}{d_i |\mathbf{v}_i|_1} \sum_j^N \mathbf{v}_i(j) ||\hat{\mathbf{U}}_i(:,j) - \mathbf{U}_i(:,j)||), \tag{3.17}$$

where d_i is the square root of the face bounding box size [52]. The eye-to-eye distance is not used in NME since it is not well defined in large poses such as profile.

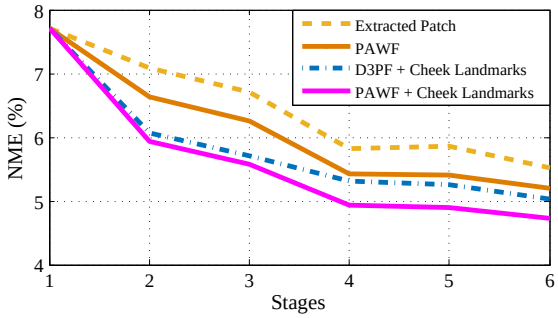


Figure 3.12: Errors on AFLW testing set after each stages of CNN for different feature extraction methods with the C-CNN architecture and the base training set. The initial error is 25.8%.

3.3.2 Comparison Experiments

Feature extraction methods To show the advantages of the proposed features, Table 3.1 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 [139,147]. 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 3.1, 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. Fig. 3.12 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.

Table 3.2: The NME (%) of three methods on AFLW with the base training set.

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

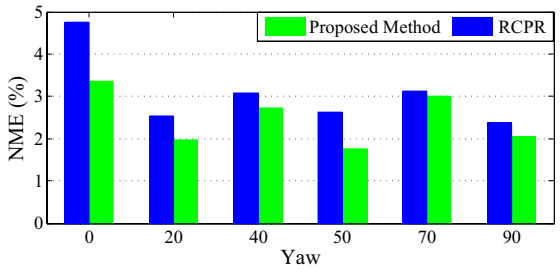


Figure 3.13: Comparison of NME for each pose with the C-CNN architecture and the base training set.

CNN is known for demanding a large training set, while the 3,901-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 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 3.2. The proposed method can achieve better results than the two baselines. The error comparison for each landmark is shown in Fig. 3.11(b). 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. Fig. 3.13 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

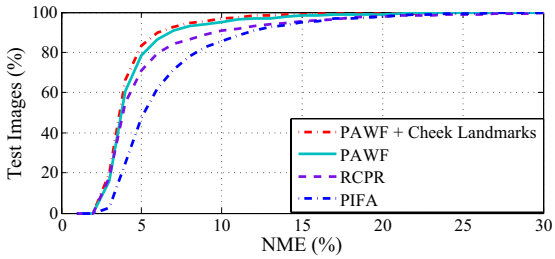


Figure 3.14: The comparison of CED for different methods with the C-CNN architecture and the base training set.

Figure 3.15: 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.

the NME distribution, the Cumulative Errors Distribution (CED) diagrams of various methods are shown in Fig. 3.14. The improvement seems to be over all NME values, and is especially larger around lower NMEs ($\leq 8\%$). We use the t-SNE toolbox [112] 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. 3.15). This figure shows that the first stage CNN can model the distribution of face poses.

Table 3.3: 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

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 3.3. 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 Table 3.2 and Table 3.3 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 images. Although the method in [150] 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. 3.6 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 till 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 3.4. For PIFA, CDM and TSPM, we show the error reported in their papers. Again we see the consistent improvement of our proposed method (with both architectures) over the baseline methods.

Table 3.4: 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

Comparison of two CNN toolboxes We utilize two toolboxes for our implementations. We use the MatConvNet toolbox [113] to implement the C-CNN architecture (Fig. 3.5). 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 [49] to implement the M-CNN architecture (Fig. 3.6). Based on our experiments on the AFLW test set, there are noticeable difference between the testing results of these two toolboxes.

Table 3.5 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

Table 3.5: The six-stage NMEs of implementing C-CNN and M-CNN architectures with different training data sets and CNN toolboxes. The initial error is 25.8%.

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

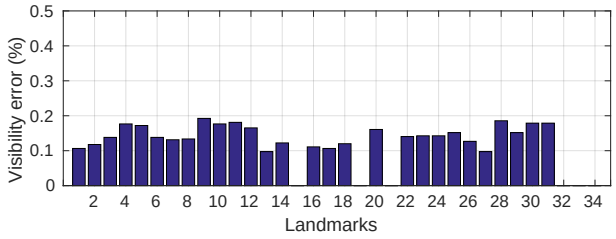


Figure 3.16: The distribution of visibility errors for each landmark. For six landmarks on the horizontal center of the face, their visibility errors are zeros since they are always visible.

the visibility error for each landmark, their distribution is shown in Fig. 3.16.

Qualitative results Some examples of alignment results for the proposed method on AFLW and AFW datasets are shown in Fig. 3.17. The result of the proposed method at each stage is shown in Fig. 3.18.

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.

Figure 3.17: 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.

3.4 Summary

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.

Figure 3.18: 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.

Chapter 4

Pose-Invariant Face Alignment with a Single CNN

4.1 Introduction

In the previous chapter, we propose our PIFA method based on cascade of CNN regressors and 3D Morphable Model. The cascade of the regressors is the dominant technology for pose-invariant face alignment [68, 147, 148]. Despite the recent success, the cascade of CNNs, when applied to the large pose face images, still suffers from the following drawbacks.

Lack of end-to-end training: It is a consensus that end-to-end training is desired for CNN [23, 47]. However, in the existing methods, the CNNs are typically trained independently at each cascade stage. Sometimes even multiple CNNs are applied independently at each stage. For example, locations of different landmark sets are estimated by various CNNs and combined via a separate fusing module [99]. Therefore, these CNNs can not be jointly optimized and might lead to a sub-optimal solution.

Hand-crafted feature extraction: Since the CNNs are trained independently, feature extraction is required to utilize the result of a previous CNN and provide input to the current CNN. Simple feature extraction methods are used, e.g., extracting patches based on 2D or 3D face shapes without considering other factors including pose and expression [99, 139] . Normally, the cascade of CNNs is a collection of shallow CNNs where each one has less than five layers. Hence, this framework

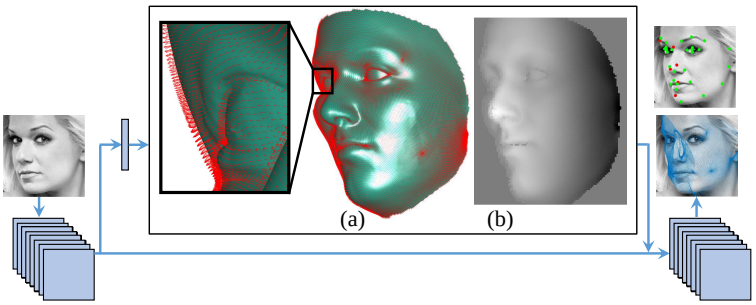


Figure 4.1: For the purpose of learning an end-to-end face alignment model, our novel visualization layer reconstructs the 3D face shape (a) from the estimated parameters inside the CNN and synthesizes a 2D image (b) via the surface normal vectors of visible vertexes.

can not extract *deep* features by building upon the extracted features of early-stage CNNs.

Slow training speed: Training a cascade of CNNs is usually time-consuming for two reasons. Firstly, the CNNs are trained sequentially, one after another. Secondly, feature extraction is required between two consecutive CNNs.

To address these issues, we introduce a novel layer, as shown in Figure 4.1. Our CNN architecture consists of several blocks, which are called visualization blocks. This architecture can be considered as a cascade of shallow CNNs. The new layer visualizes the alignment result of a previous visualization block and utilizes it in a later visualization block. It is designed based on several guidelines. Firstly, it is derived from the surface normals of the underlying 3D face model and encodes the relative pose between the face and camera, partially inspired by the success of using surface normals for 3D face recognition [79]. Secondly, the visualization layer is differentiable, which allows the gradient to be computed analytically and enables end-to-end training. Lastly, a mask is utilized to differentiate between pixels in the middle and contour areas of a face.

Benefiting from the design of the visualization layer, our method has the following advantages and contributions:

◇ The proposed method allows a block in the CNN to utilize the extracted features from previ-

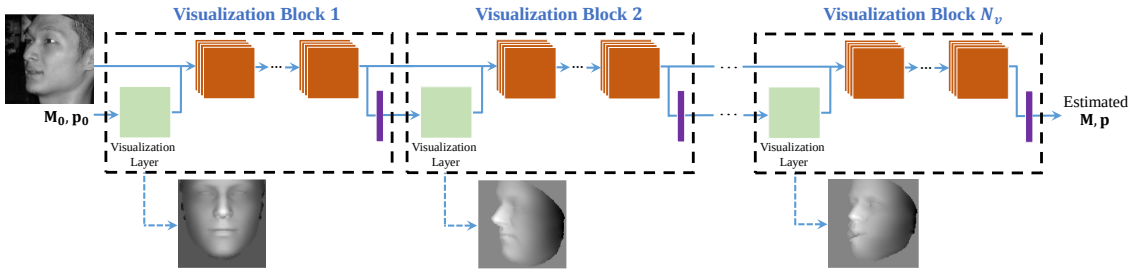


Figure 4.2: The proposed CNN architecture. We use green, orange, and purple to represent the visualization layer, convolutional layer, and fully connected layer, respectively. Please refer to Figure 4.3 for the details of the visualization block.

ous blocks and extract deeper features. Therefore, extraction of hand-crafted features is no longer necessary.

◇ The visualization layer is differentiable, allowing for backpropagation of an error from a later block to an earlier one. To the best of our knowledge, this is the first method for pose-invariant face alignment, that utilizes only one single CNN and allows end-to-end training.

◇ The proposed method converges faster during the training phase compared to the cascade of CNNs. Therefore, the training time is dramatically reduced.

4.2 3D Face Alignment with Visualization Layer

Given a single face image with an arbitrary pose, our goal is to estimate the 2D landmarks with their visibility labels by fitting a 3D face model. Towards this end, we propose a CNN architecture with end-to-end training for model fitting, as shown in Figure 4.2. In this section, we will first describe the underlying 3D face model used in this work, followed by our CNN architecture and the visualization layer.

4.2.1 3D and 2D Face Shapes

We use the 3D Morphable Model (3DMM) to represent the 3D shape of a face $\mathbf{S}_p$ as a linear combination of mean shape $\mathbf{S}_0$, identity bases $\mathbf{S}^I$ and expression bases $\mathbf{S}^E$:

$$\mathbf{S}_p = \mathbf{S}_0 + \sum_k^{N_I} p_k^I \mathbf{S}_k^I + \sum_k^{N_E} p_k^E \mathbf{S}_k^E. \tag{4.1}$$

We use vector $\mathbf{p} = [\mathbf{p}^I, \mathbf{p}^E]$ to indicate the 3D shape parameters, where $\mathbf{p}^I = [p_0^I, \cdots, p_{N_I}^I]$ are the identity parameters and $\mathbf{p}^E = [p_0^E, \cdots, p_{N_E}^E]$ are the expression parameters. We use the Basel 3D face model [86], which has 199 bases, as our identity bases and the face wearhouse model [25] with 29 bases as our expression bases. Each 3D face shape consists of a set of Q 3D vertexes:

$$\mathbf{S}_p = \begin{pmatrix} x_1^p & x_2^p & \dots & x_Q^p \\ y_1^p & y_2^p & \dots & y_Q^p \\ z_1^p & z_2^p & \dots & z_Q^p \end{pmatrix}. \tag{4.2}$$

The 2D face shapes are the projection of 3D shapes. In this work, we use the weak perspective projection model with 6 degrees of freedoms, i.e., one for scale, three for rotation angles and two for translations, which projects the 3D face shape $\mathbf{S}_p$ onto 2D images to obtain the 2D shape $\mathbf{U}$:

$$\mathbf{U} = f(\mathbf{P}) = \mathbf{M} \begin{pmatrix} \mathbf{S}_p(:, \mathbf{b}) \\ \mathbf{1} \end{pmatrix}, \tag{4.3}$$

where

$$\mathbf{M} = \begin{bmatrix} m_1 & m_2 & m_3 & m_4 \\ m_5 & m_6 & m_7 & m_8 \end{bmatrix}, \tag{4.4}$$

and

$$\mathbf{U} = \begin{pmatrix} x_1^t & x_2^t & \dots & x_N^t \\ y_1^t & y_2^t & \dots & y_N^t \end{pmatrix}. \tag{4.5}$$

Here $\mathbf{U}$ is a set of N 2D landmarks, $\mathbf{M}$ is the camera projection matrix. With misuse of notation, we define the target parameters $\mathbf{P} = \{\mathbf{M}, \mathbf{p}\}$. The N-dim vector $\mathbf{b}$ includes 3D vertex indexes which are semantically corresponding to 2D landmarks. We denote $\mathbf{m}_1 = [m_1\ m_2\ m_3]$ and $\mathbf{m}_2 = [m_5\ m_6\ m_7]$ as the first two rows of the scaled rotation component, while m_4 and m_8 are the translations.

Equation 4.3 establishs the relationship, or equivalency, between 2D landmarks $\mathbf{U}$ and $\mathbf{P}$, i.e., 3D shape parameters $\mathbf{p}$ and the camera projection matrix $\mathbf{M}$. Given that almost all the training images for face alignment have only 2D labels, i.e., $\mathbf{U}$, we preform a data augmentation step similar to [53] to compute their corresponding $\mathbf{P}$. Given an input image, our goal is to estimate the parameter $\mathbf{P}$, based on which the 2D landmarks and their visibilities can be naturally derived.

4.2.2 Proposed CNN Architecture

Our CNN architecture resembles the cascade of CNNs, while each “shallow CNN" is defined as a visualization block. Inside each block, a visualization layer based on the latest parameter estimation serves as a bridge between consecutive blocks. This design enables us to address the drawbacks of typical cascade of CNNs in Section 4.1. We now describe the visualization block and CNN architecture, and dive into the details of the visualization layer in Section 4.2.3.

Visualization Block Figure 4.3 shows the structure of our visualization block. The visualization layer generates a feature map based on the latest parameter $\mathbf{P}$ (details in Section 4.2.3). Each convolutional layer is followed by a batch normalization (BN) layer and a ReLU layer. It extracts deeper features based on the features provided by the previous visualization block and the visualization layer output. Between the two fully connected layers, the first one is followed by a ReLU

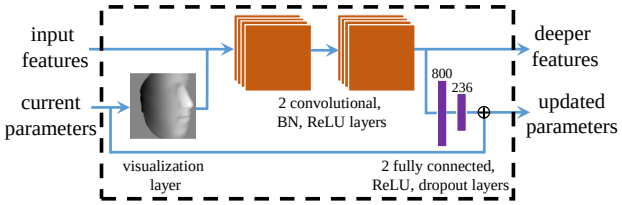


Figure 4.3: A visualization block consists of a visualization layer, two convolutional layers and two fully connected layers.

layer and a dropout layer, while the second one simultaneously estimates the update of $\mathbf{M}$ and $\mathbf{p}$, denoted $\Delta\mathbf{P}$. The outputs of the visualization block are deeper features and the new estimation of the parameters ($\Delta\mathbf{P}+\mathbf{P}$). As shown in Figure 4.3, the top part of the visualization block focuses on learning deeper features, while the bottom part utilizes those features to estimate the parameters in a ResNet-like structure [44]. During the backward pass of the training phase, the visualization block backpropagates the loss through both of its inputs to adjust the convolutional and the fully connected layers in previous blocks. This allows the block to extract better features for the next block and improve the overall parameter estimation.

CNN Architecture The proposed architecture consists of several connected visualization blocks as shown in Figure 4.2. The inputs include an image and an initial estimation of the parameter $\mathbf{P}^0$. The outputs is the final estimation of the parameter. Due to the joint optimization of all visualization blocks through backpropagation, the proposed architecture is able to converge with substantially fewer epochs during training, compared to the typical cascade of CNNs.

Loss Functions Two types of loss functions are employed in our CNN architecture. The first one is an Euclidean loss between the estimation and the target of the parameter update, with each parameter weighted separately:

$$\mathbf{E}_P^i = (\Delta\mathbf{P}^i - \Delta\bar{\mathbf{P}}^i)^T \mathbf{W} (\Delta\mathbf{P}^i - \Delta\bar{\mathbf{P}}^i), \tag{4.6}$$

where $\mathbf{E}_P^i$ is the loss, $\Delta\mathbf{P}^i$ is the estimation and $\Delta\bar{\mathbf{P}}^i$ is the target (or ground truth) at the i-th visualization block. The diagonal matrix $\mathbf{W}$ contains the weights. For each element of the shape parameter $\mathbf{p}$, its weight is the inverse of the standard deviation that was obtained from the data used in 3DMM training. To compensate the relative scale among the parameters of $\mathbf{M}$, we compute the ratio r between the average of scaled rotation parameters and average of translation parameters in the training data. We set the weights of the scaled rotation parameters of $\mathbf{M}$ to $\frac{1}{r}$ and the weights of the translation of $\mathbf{M}$ to 1. The second type of loss function is the Euclidean loss on the resultant 2D landmarks:

$$\mathbf{E}_S^i = \|f(\mathbf{P}^i + \Delta\mathbf{P}^i) - \bar{\mathbf{U}}\|^2, \tag{4.7}$$

where $\bar{\mathbf{U}}$ is the ground truth 2D landmarks, and $\mathbf{P}^i$ is the input parameter to the i-th block, i.e., the output of the $i-1$-th block. $f(\cdot)$ computes 2D landmark locations using the currently updated parameters via Equation 4.3. For backpropagation of this loss function to the parameter $\Delta\mathbf{P}$, we use the chain rule to compute the gradient.

$$\frac{\partial \mathbf{E}_S^i}{\partial \Delta\mathbf{P}^i} = \frac{\partial \mathbf{E}_S^i}{\partial f}\frac{\partial f}{\partial \mathbf{P}^i}.$$

For the first three visualization blocks, the Euclidean loss on the parameter updates (Equation 4.6) is used, while the Euclidean loss on 2D landmarks (Equation 4.7) is applied to the last three blocks. The first three blocks estimate parameters to roughly align 3D shape to the face image and the last three blocks leverage the good initialization to estimate the parameters and the 2D landmark locations more precisely.

4.2.3 Visualization Layer

Several visualization techniques have been explored for facial analysis. In particular, Z-Buffering, which is widely used in prior works [12, 13], is a simple and fast 2D representation of the 3D shape. However, this representation is not differentiable. In contrast, our visualization is based on surface normals of the 3D face, which describes surface's orientation in a local neighbourhoods. It has been successfully utilized for different facial analysis tasks, e.g., 3D face reconstruction [95] and 3D face recognition [79].

In this work, we use the z coordinate of surface normals of each vertex, transformed with the pose. It is an indicator of "frontability" of a vertex, i.e., the amount that the surface normal is pointing towards the camera. This quantity is used to assign an intensity value at its projected 2D location to construct the visualization image. The frontability measure $\mathbf{g}$, a Q dimensional vector, can be computed as,

$$\mathbf{g} = \max\left(\mathbf{0}, \frac{(\mathbf{m}_1 \times \mathbf{m}_2)}{\|\mathbf{m}_1\|\|\mathbf{m}_2\|}\mathbf{N}_0\right), \tag{4.8}$$

where $\times$ is the cross product, and $\|.\|$ denotes the L_2 norm. The $3 \times Q$ matrix $\mathbf{N}_0$ is the surface normal vectors of a 3D face shape. To avoid the high computational cost of calculating the surface normals after each shape update, we approximate $\mathbf{N}_0$ with the surface normals of the mean 3D face. Note that both the face shape and pose are still continuously updated across visualization blocks, and are used to determine the projected 2D location. Hence, this approximation would only slightly affect the intensity values. To transform the surface normal based on the pose, we apply the estimation of the scaled rotation matrix ($\mathbf{m}_1$ and $\mathbf{m}_2$) to the surface normals computed from the mean face. The value is then truncated with the lower bound of 0 (Equation 4.8).

The pixel intensity of a visualized image $\mathbf{V}(u,v)$ is computed as the weighted average of the frontability measures within a local neighbourhood:

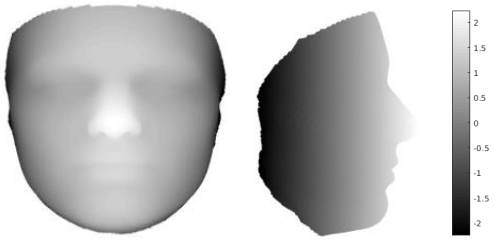


Figure 4.4: The frontal and side views of the mask **a** that has positive values in the middle and negative values in the contour area.

$$\mathbf{V}(u,v) = \frac{\sum_{q\in\mathbb{D}(u,v)} \mathbf{g}(q)\mathbf{a}(q)w(u,v,x_q^t,y_q^t)}{\sum_{q\in\mathbb{D}(u,v)} w(u,v,x_q^t,y_q^t)}, \tag{4.9}$$

where $\mathbb{D}(u,v)$ is the set of indexes of vertexes whose 2D projected locations are within the local neighborhood of the pixel (u,v). (x_q^t,y_q^t) is the 2D projected location of q-th 3D vertex. The weight w is the distance metric between the pixel (u,v) and the projected location (x_q^t,y_q^t),

$$w(u,v,x_q^t,y_q^t) = \exp\left(-\frac{(u-x_q^t)^2+(v-y_q^t)^2}{2\sigma^2}\right). \tag{4.10}$$

The Q-dim vector **a** is a mask with positive values for vertexes in the middle area of the face and negative values for vertexes around the contour area of the face:

$$\mathbf{a}(q) = \exp\left(-\frac{(x^n-x_q^p)^2+(y^n-y_q^p)^2+(z^n-z_q^p)^2}{2\sigma_n^2}\right), \tag{4.11}$$

where (x^n,y^n,z^n) is the vertex coordinate of the nose tip. **a** is pre-computed and normalized for zero-mean and unit standard deviation. The mask is utilized to discriminate between the middle and contour areas of the face. A visualization of the mask is provided in Figure 4.4.

Since the human face is a 3D object, visualizing it at an arbitrary view angle requires the estimation of the visibility of each 3D vertex. To avoid the computationally expensive visibility

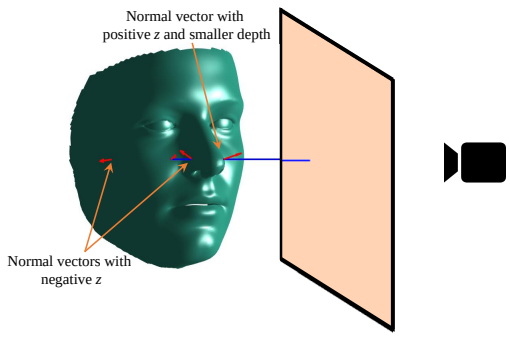


Figure 4.5: An example with four vertexes projected to a same pixel. Two of them have negative values in z component of their normals (red arrows). Between the other two with positive values, the one with the smaller depth (closer to the image plane) is selected.

test via rendering, we adopt two strategies for approximation. Firstly, we prune the vertexes whose frontability measures $\mathbf{g}$ equal 0, i.e., the vertexes pointing against the camera. Secondly, if multiple vertexes projects to a same image pixel, we keep only the one with the smallest depth values. An illustration is provided in Figure 4.5.

Backpropagation To allow backpropagation of the loss functions through the visualization layers, we compute the derivative of $\mathbf{V}$ with respect to the elements of the parameters $\mathbf{M}$ and $\mathbf{p}$. Firstly, we compute the partial derivatives, $\frac{\partial \mathbf{g}}{\partial m_k}$, $\frac{\partial w(u,v,x_i^t,y_i^t)}{\partial m_k}$ and $\frac{\partial w(u,v,x_i^t,y_i^t)}{\partial p_j}$. Then the derivatives of $\frac{\partial \mathbf{V}}{\partial m_k}$ and $\frac{\partial \mathbf{V}}{\partial p_j}$ can be computed based on Equation. 4.9.

4.3 Experimental Results

We evaluate our method on two challenging PIFA datasets, AFLW and AFW, both qualitatively and quantitatively, as well as the near-frontal face dataset of 300W. Furthermore, we conduct experiments on different CNN architectures to validate our visualization layer design.

Implementation details Our implementation is built upon the Caffe toolbox [49]. In all of the experiments, we use six visualization blocks (N_v) with two convolutional layers (N_c) and fully

connected layers in each block (Figure 4.3). Details of the network structure are provided in Table 4.1.

Instead of using the sequentially pretrain strategy [127], we perform the joint end-to-end training from scratch. To better estimate the parameter update in each block and to increase the effectiveness when using visualization blocks, we set the weight of the loss function in the first visualization block to 1 and linearly increase the weights by one for each later block. This strategy helps the CNN to pay more attention to the landmark loss used in later blocks. Backpropagation of loss functions in the last blocks would have more impact in the first block, and the last block can adopt itself more quickly to the changes in the first block.

In the training phase, we set the weight decay to 0.005, the momentum to 0.99, the initial learning rate to $1e{-}6$. Besides, we decrease the learning rate to $5e{-}6$ and $1e{-}7$ after 20 and 29 epochs. In total, the training phase is continued for 33 epochs for all experiments.

4.3.1 Quantitative Evaluations on AFLW and AFW

The AFLW dataset [61] is very challenging with large-pose face images ($\pm 90^\circ$ yaw). We use the subset with $3,901$ training images and $1,299$ testing images released by [53]. All face images in this subset are labeled with 34 landmarks and a bounding box. The AFW dataset [151] contains 205 images with 468 faces. Each image is labeled with at most 6 landmarks with visibility labels, as well as a bounding box. AFW is used only for testing in our experiments. The bounding boxes

Table 4.1: The number and size of convolutional filters in each visualization block. For all blocks, the two fully connected layers have the same length of 800 and 236.

Block #	1	2	3	4	5,6
Conv. layers	12 (5×5) 16 (5×5)	20 (3×3) 24 (3×3)	28 (3×3) 32 (3×3)	36 (3×3) 40 (3×3)	40 (3×3) 40 (3×3)

Table 4.2: NME (%) of four methods on the AFLW dataset.

Proposed method	Extended-PIFA [53]	PIFA	RCPR
4.45	4.72	8.04	6.26

Table 4.3: NME (%) of the proposed method at each visualization block on AFLW dataset. The initial NME is 25.8%.

Block #	1	2	3	4	5	6
NME	9.26	6.77	5.51	4.98	4.60	4.45

in both datasets are used as the initilization for our algorithm, as well as the baselines. We crop the region inside the bounding box and normalize it to 114×114. Due to the memory constraint of GPUs, we have a pooling layer in the first visualization block after the first convolutional layer to decrease the size of feature maps to half. The input to the subsequent visualization blocks is of 57×57. To augment the training data, we generate 20 different variations for each training image by adding noise to the location, width and height of the bounding boxes.

For quantitative evaluations, we use two conventional metrics. The first one is Mean Average Pixel Error (MAPE) [135], which is the average of the pixel errors for the visible landmarks. The other one is Normalized Mean Error (NME), i.e., the average of the normalized estimation error of visible landmarks. The normalization factor is the square root of the face bounding box size [52], instead of the eye-to-eye distance in the frontal-view face alignment.

We compare our method with several state-of-the-art LFPA approaches. On AFLW, we compare with Extended-PIFA [53], PIFA [52] and RCPR [22] with the NME metric. Table 4.2 shows that our proposed method achieves a higher accuracy than the alternatives. The heatmap-based method, named CALE [21], reported an NME of 2.96%, but suffers from several disadvantages. To demonstrate the capabilities of each visualization block, the NME computed using the estimated

P after each block is shown in Table 4.3. If a higher alignment speed is desirable, it is possible to skip the last two visualization blocks with a reasonable NME. On the AFW dataset, comparisons are conducted with Extended-PIFA [53], PIFA [52], CDM [135] and TSPM [151] with the MAPE metric. The results in Table 4.4 again show the superiority of the proposed method.

Some examples of alignment results of the proposed method on AFLW and AFW datasets are shown in Figure 4.9. Three examples of visualization layer output at each visualization block are shown in Figure 4.10.

4.3.2 Evaluation on 300W dataset

While our main goal is PIFA, we also evaluate on the most widely used near frontal 300W dataset [96]. 300W containes $3,148$ training and 689 testing images, which are divided into common and challenging sets with 554 and 135 images, respectively. Table 4.5 shows the NME (normalized by the interocular distance) of the evaluated methods. The most relevant method is 3DDFA [147], which also estimates **M** and **p**. Our method outperforms it on both the common and challenging sets. Methods that do not employ shape constraints, e.g., via 3DMM, generally have higher freedom and could achieve slightly better accuracy on frontal face cases. Nonetheless, they are typically less robust in more challenging cases. Another comparison is with MDM [107] via the failure rate using a threshold of 0.08. The failure rates are 16.83% (ours) versus 6.80% (MDM) with 68 landmarks, and 8.99% (ours) versus 4.20% (MDM) with 51 landmarks.

Table 4.4: MAPE of five methods on the AFW dataset.

Proposed method	Extended-PIFA [53]	PIFA	CDM	TSPM
6.27	7.43	8.61	9.13	11.09

Table 4.5: The NME of different methods on 300W dataset.

Method	Common	Challenging	Full
ESR [26]	5.28	17.00	7.58
RCPR [22]	6.18	17.26	8.35
SDM [124]	5.57	15.40	7.50
LBF [93]	4.95	11.98	6.32
CFSS [147]	4.73	9.98	5.76
RCFA [116]	4.03	9.85	5.32
RAR [122]	4.12	8.35	4.94
3DDFA [147]	6.15	10.59	7.01
3DDFA+SDM	5.53	9.56	6.31
Proposed method	5.43	9.88	6.30

4.3.3 Analysis of the Visualization Layer

We perform four sets of experiments to study the properties of the visualization layer and network architectures.

Influence of visualization layers To analyze the influence of the visualization layer in the testing phase, we add 5% noise to the fully connected layer parameters of each visualization block, and compute the alignment error on the AFLW test set. The NMEs are [4.46, 4.53, 4.60, 4.66, 4.80, 5.16] when each block is modified seperately. This analysis shows that the visualized images have more influence on the later blocks, since imprecise parameters of early blocks could be compensated by later blocks. In another experiment, we train the network without any visualization layer. The final NME on AFLW is 7.18% which shows the importance of visualization layers in guiding the network training.

Advantage of deeper features We train three CNN architectures shown in Figure 4.6 on AFLW. The inputs of the visualization block in the first architecture are the input images $\mathbf{I}$, feature maps $\mathbf{F}$ and the visualization image $\mathbf{V}$. The inputs of the second and the third architectures are $\{\mathbf{F}, \mathbf{V}\}$ and $\{\mathbf{I}, \mathbf{V}\}$, respectively. The NME of each architecture is shown in Table 4.6. While the first one

Table 4.6: The NME (%) of three architectures with different inputs (**I**: Input image, **V**: Visualization, **F**: Feature maps).

Architecture a (**I**, **F**, **V**)	Architecture b (**F**, **V**)	Architecture c (**I**, **V**)
4.45	4.48	5.06

performs the best, the substantial lower performance of the third one demonstrates the importance of deeper features learned across blocks.

At the first convolutional layer of each visualization block, we compute the average of the filter weights, across both the kernel size and the number of maps. The averages for these three types of input features are shown in Figure 4.7. As observed, the weights decrease across blocks, leading to a more precise estimation of small-scale parameter updates. Considering the number of filters in Table 4.1, the total impact of feature maps are higher than the other two inputs in all blocks. This again shows the importance of deeper features in guiding the network to estimate parameters. Furthermore, the average of the visualization filter is higher than that of the input image filter, demonstrating stronger influence of the proposed visualization during training.

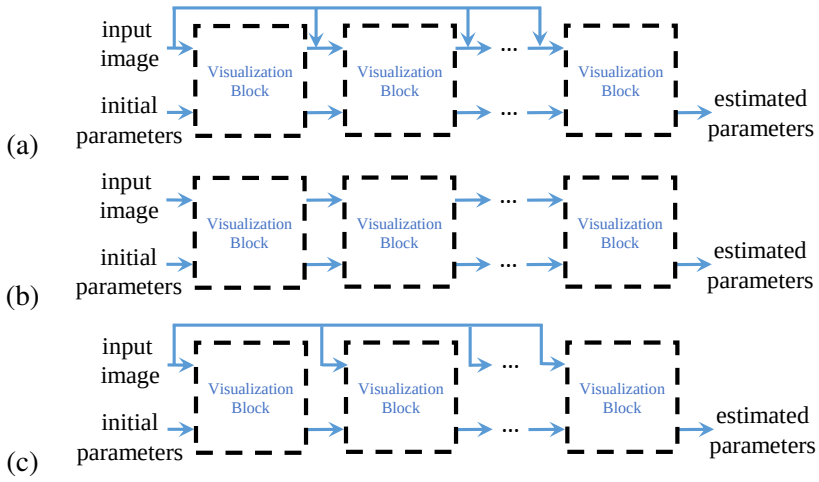


Figure 4.6: Architectures of three CNNs with different inputs.

Table 4.7: NME (%) when different masks are used.

Mask 1	Mask 2	No Mask
4.45	4.49	5.31

Advantage of using masks To show the advantage of using the mask in the visualization layer, we conduct an experiment with different masks. Specifically, we define another mask for comparison as shown in Figure 4.8. It has five positive areas, i.e., the eyes, nose tip and two lip corners. The values are normalized to zero-mean and unit standard deviation. Compared to the original mask in Fig. 4.4, this mask is more complicated and conveys more information about the informative facial areas to the network. Moreover, to show the necessity of using the mask, we also test using visualization layers without any mask. The NMEs of the trained networks with different masks are shown in Table 4.7. Comparison between the first and the third columns shows the benefit of using the mask , by differentiating the middle and contour areas of the face. By comparing the first and second columns, we can see that utilizing more complicated mask does not further improve the result, indicating the original mask provides sufficient information for its purpose.

Different numbers of blocks and layers Given the total number of 12 convolutional layers in our network, we can partition them into visualization blocks of various sizes. To compare their

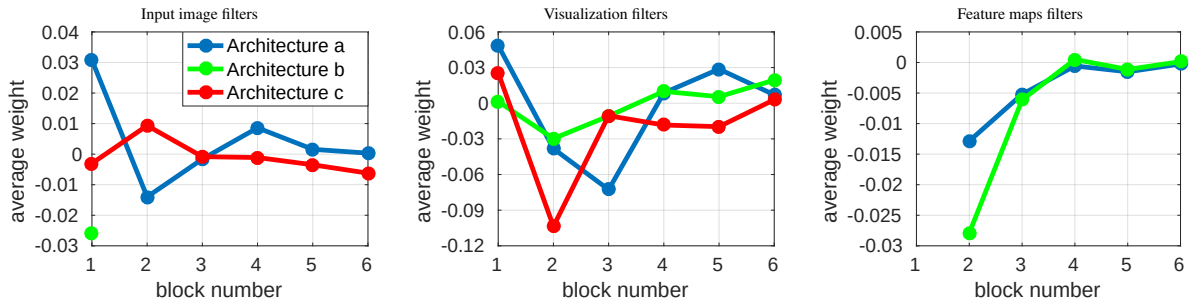


Figure 4.7: The average of filter weights for input image, visualization and feature maps in three architectures of Figure 4.6. The y-axis and x-axis show the average and the block index, respectively.

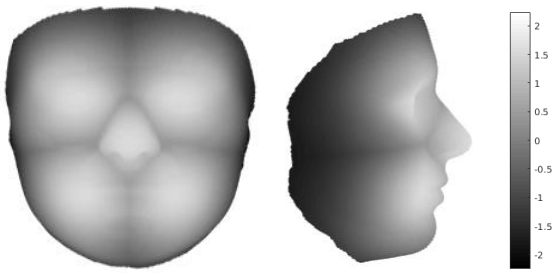


Figure 4.8: Mask 2, a different designed mask with five positive areas on the eyes, top of the nose and sides of the lip.

performance, we train two additional CNNs. The first one consists of 4 visualization blocks, with 3 convolutional layers in each. The other comes with 3 block and 4 convolutional layers per block. Hence, all three architectures have 12 convolutional layers in total. The NME of these architectures are shown in Table 4.8. Similar to [22], it shows that the number of regressors is important for face alignment and we can potentially achieve a higher accuracy by increasing the number of visualization blocks.

4.3.4 Time complexity

Compared to the cascade of CNNs, one of the main advantages of end-to-end training a single CNN is the reduced training time. The proposed method needs 33 epochs which takes around 2.5 days. With the same training and testing data sets, [53] requires 70 epochs for each CNN. With a total of six CNNs, it needs around 7 days. Similarly, the method in [147] needs around 12 days to train three CNNs, each one with 20 epochs, despite using different training data. Compared

Table 4.8: NME (%) when using different numbers of visualization blocks (N_v) and convolutional layers (N_c).

$N_v = 6$, $N_c = 2$	$N_v = 4$, $N_c = 3$	$N_v = 3$, $N_c = 4$
4.45	4.61	4.83

Figure 4.9: Results of alignment on AFLW and AFW datasets, green landmarks show the estimated locations of visible landmarks and red landmarks show estimated locations of invisible landmarks. First row: provided bounding box by AFLW with initial locations of landmarks, Second: estimated 3D dense shapes, Third: estimated landmarks, Fourth to sixth: estimated landmarks for AFLW, Seventh: estimated landmarks for AFW.

to [53], our method reduces the training time by more than half. The testing speed of proposed method is 4.3 FPS on a Titan X GPU. It is much faster than the 0.6 FPS speed of [53] and is similar to the 4 FPS speed of [122].

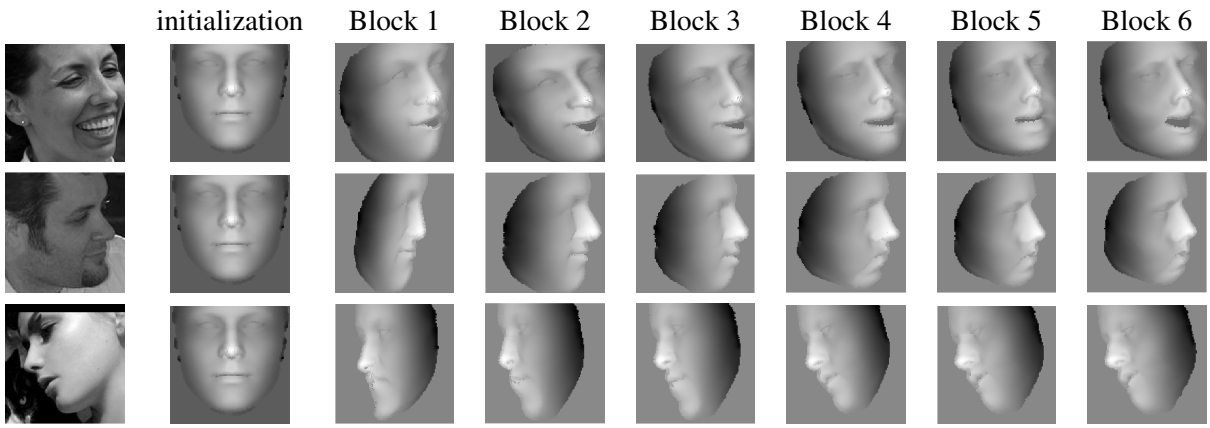


Figure 4.10: Three examples of outputs of visualization layer at each visualization block. The first row shows that the proposed method recovers the expression of the face gracefully, the third row shows the visualizations of a face with a more challenging pose.

4.4 Summary

We propose a pose-invariant face alignment method with end-to-end training in a single CNN. The key is a differentiable visualization layer, which is integrated to the network and enables joint optimization by backpropagating the error from a later visualization blocks to early ones. It allows the visualization block to utilize the extracted features from previous blocks and extract deeper features, without extracting hand-crafted features. In addition, the proposed method converges faster during the training phase compared to the cascade of CNNs. Through extensive experiments, we demonstrate the superior results of the proposed method over the state-of-the-art approaches.

Chapter 5

Learning Deep Models for Face Anti-Spoofing: Binary or Auxiliary Supervision

5.1 Introduction

With the increasing influence of smart devices in our daily lives, people are seeking for secure and convenient ways to access their personal information. Biometrics, such as face, fingerprint, and iris, are widely utilized for person authentication due to their intrinsic distinctiveness and convenience to use. Face, as one of the most popular modalities, has received increasing attention in the academia and industry in the recent years (e.g., iPhone X). However, the attention also brings a growing incentive for hackers to design biometric presentation attacks (PA), or spoofs, to be authenticated as the genuine user. Due to the almost no-cost access to the human face, the spoof face can be as simple as a printed photo paper (i.e., print attack) and a digital image/video (i.e., replay attack), or as complicated as a 3D Mask and facial cosmetic makeup. With proper handling, those spoofs can be visually very close to the genuine user's live face. As a result, these call for the need of developing robust face anti-spoofing algorithms.

RGB image and video are the standard input to face anti-spoofing systems since the majority

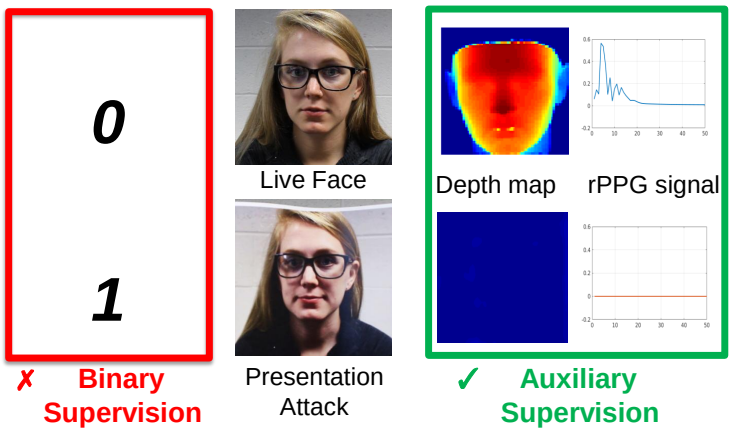


Figure 5.1: Conventional CNN-based face anti-spoof approaches utilize the binary supervision, which may lead to overfitting given the enormous solution space of CNN. This work designs a novel network architecture to leverage two auxiliary information as supervision: the depth map and rPPG signal, with the goals of improved generalization and explainable decisions during inference.

of face recognition systems adopt RGB cameras as the sensor. Researchers start the texture-based anti-spoofing approaches by feeding handcrafted features to binary classifiers [18, 33, 34, 59, 77, 84, 131]. Later in the deep learning era, several Convolutional Neural Networks (CNN) approaches utilize softmax loss as the supervision [37, 66, 83, 130]. It appears almost all prior work regard the face anti-spoofing problem as merely a *binary* (live vs. spoof) classification problem.

There are two main issues in learning deep anti-spoofing models with binary supervision. First, there are different levels of image degradation, namely *spoof patterns*, comparing a spoof face to a live one, which consist of skin detail loss, color distortion, moiré pattern, shape deformation and spoof artifacts (e.g., reflection) [65, 84]. A CNN with softmax loss might discover *arbitrary* cues that are able to separate the two classes, such as screen bezel, but not the *faithful* spoof patterns. When those cues disappear during testing, these models would fail to distinguish spoof vs. live faces and result in poor generalization. Second, during the testing, models learnt with binary supervision will only generate a binary decision without *explanation* or *rationale* for the decision. In the pursuit of Explainable Artificial Intelligence [1], it is desirable for the learnt

model to generate the spoof patterns that support the final binary decision.

To address these issues, as shown in Fig. 5.1, we propose a deep model that uses the supervision from both the *spatial and temporal auxiliary information* rather than binary supervision, for the purpose of robustly detecting face PA from a face video. These auxiliary information are acquired based on our domain knowledge about the key *differences* between live and spoof faces, which include two perspectives: spatial and temporal. From the spatial perspective, it is known that live faces have face-like depth, e.g., the nose is closer to the camera than the cheek in frontal-view faces, while faces in print or replay attacks have flat or planar depth, e.g., all pixels on the image of a paper have the same depth to the camera. Hence, depth can be utilized as auxiliary information to supervise both live and spoof faces. From the temporal perspective, it was shown that the normal rPPG signals (i.e., heart pulse signal) are detectable from live, but not spoof, face videos [69, 81]. Therefore, we provide different rPPG signals as auxiliary supervision, which guides the network to learn from live or spoof face videos respectively. To enable both supervisions, we design a network architecture with a short-cut connection to capture different scales and a novel non-rigid registration layer to handle the motion and pose change for rPPG estimation.

Furthermore, similar to other vision problems, data plays a significant role in training the anti-spoofing models. As we know, camera/screen quality is a critical factor to the quality of spoof faces. Existing face anti-spoofing databases, such as NUAA [104], CASIA [143], Replay-Attack [28], and MSU-MFSD [117], were collected $3-5$ years ago. Given the fast development pace of consumer electronics, the types of equipment (e.g., cameras and spoofing mediums) used in those data collection are outdated compared to the ones nowadays, regarding the resolution and imaging quality. More recent MSU-USSA [84] and OULU databases [19] have subjects with fewer variations in facial poses, illuminations, expressions (PIE). The lack of necessary variations would make it hard to learn an effective model. Given the clear need for more advanced

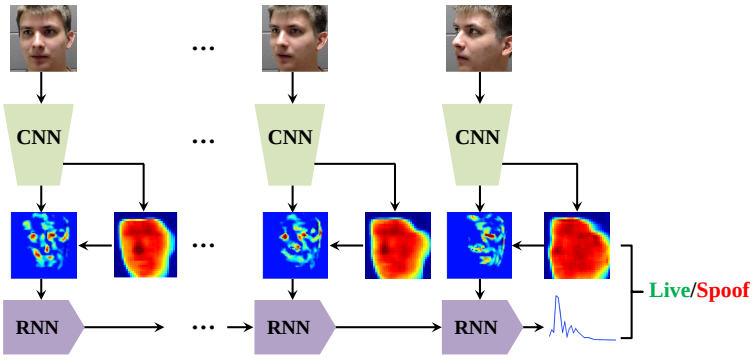


Figure 5.2: The overview of the proposed method.

databases, we collect a face anti-spoofing database for training and evaluation, named Spoof in the Wild Database (SiW). SiW database consists of 299 subjects, 6 spoofing mediums, and 4 sessions covering variations such as PIE, distance-to-camera, etc. SiW covers much larger variations than previous databases, as detailed in Tab. 5.1 and Sec. 5.3.

The main contributions of this work include:

◇ We propose to leverage novel auxiliary information (i.e., depth map and rPPG) to supervise the CNN learning for improved generalization.

◇ We propose a novel CNN-RNN architecture for end-to-end learning the depth map and rPPG signal.

◇ We release a new database that contains variations of PIE, and other practical factors. We achieve the state-of-the-art performance for face anti-spoofing.

5.2 Face Anti-Spoofing with Deep Network

The main idea of the proposed approach is to guide the deep network to focus on the *known spoof patterns* across spatial and temporal domains, rather than to extract any cues that could separate two classes but are not generalizable. As shown in Fig. 5.2, the proposed network combines CNN and RNN architectures in a coherent way. The CNN part utilizes the depth map supervision to

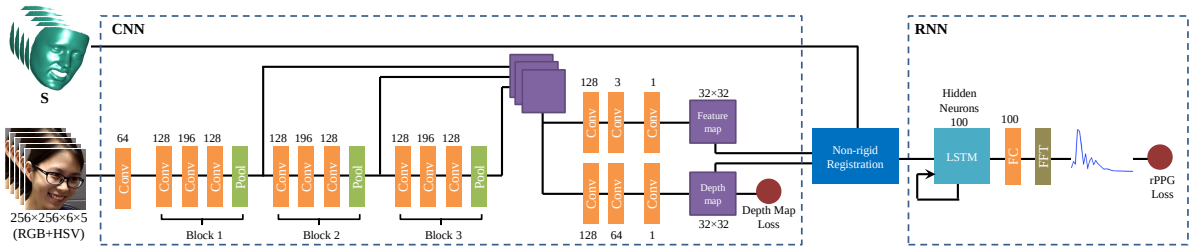


Figure 5.3: The proposed CNN-RNN architecture. The number of filters are shown on top of each layer, the size of all filters is 3×3 with stride 1 for convolutional and 2 for pooling layers. *Color code* used: *orange*=convolution, *green*=pooling, *purple*=response map.

discover subtle texture property that leads to distinct depths for live and spoof faces. Then, it feeds the estimated depth and the feature maps to a novel *non-rigid registration* layer to create aligned feature maps. The RNN part is trained with the aligned maps and the rPPG supervision, which examines temporal variability across video frames.

5.2.1 Depth Map Supervision

Depth maps are a representation of the 3D shape of the face in a 2D image, which shows the face location and the depth information of different facial areas. This representation is more informative than binary labels since it indicates one of the fundamental differences between live faces, and print and replay PA. We utilize the depth maps in the depth loss function to supervise the CNN part. The pixel-based depth loss guides the CNN to learn a mapping from the face area within a receptive field to a labeled depth value – a scale within $[0,1]$ for live faces and 0 for spoof faces.

To estimate the depth map for a 2D face image, given a face image, we utilize the state-of-the-art dense face alignment (DeFA) methods [54, 74] to estimate the 3D shape of the face. The frontal dense 3D shape $\mathbf{S}_F \in \mathbb{R}^{3\times Q}$, with Q vertices, is represented as a linear combination of identity bases $\{\mathbf{S}_{id}^i\}_{i=1}^{N_{id}}$ and expression bases $\{\mathbf{S}_{exp}^i\}_{i=1}^{N_{exp}}$,

$$\mathbf{S}_F = \mathbf{S}_0 + \sum_{i=1}^{N_{id}} \alpha_{id}^i \mathbf{S}_{id}^i + \sum_{i=1}^{N_{exp}} \alpha_{exp}^i \mathbf{S}_{exp}^i, \tag{5.1}$$

where $\alpha_{id} \in \mathbb{R}^{199}$ and $\alpha_{ext} \in \mathbb{R}^{29}$ are the identity and expression parameters, and $\alpha = [\alpha_{id}, \alpha_{exp}]$ are the shape parameters. We utilize the Basel 3D face model [86] and the facewearhouse [25] as the identity and expression bases.

With the estimated pose parameters $\mathbf{P} = (s, \mathbf{R}, \mathbf{t})$, where $\mathbf{R} \in \mathbb{R}^{3\times3}$ is a rotation matrix, $\mathbf{t} \in \mathbb{R}^3$ is a 3D translation, and s is a scale, we align the 3D shape $\mathbf{S}$ to the 2D face image:

$$\mathbf{S} = s\mathbf{R}\mathbf{S}_F + \mathbf{t}. \tag{5.2}$$

Given the challenge of estimating the *absolute* depth from a 2D face, we normalize the z values of 3D vertices in $\mathbf{S}$ to be within $[0, 1]$. That is, the vertex closest to the camera (e.g., nose) has a depth of one, and the vertex furthest away has the depth of zero. Then, we apply the Z-Buffer algorithm [149] to $\mathbf{S}$ for projecting the normalized z values to a 2D plane, which results in an estimated "ground truth" 2D depth map $\mathbf{D} \in \mathbb{R}^{32\times32}$ for a face image.

5.2.2 rPPG Supervision

rPPG signals have recently been utilized for face anti-spoofing [69, 81]. The rPPG signal provides temporal information about face liveness, as it is related to the changes in the intensities of facial skins over time. These intensity changes are highly correlated with the blood flow. The traditional method [35] for extracting rPPG signals has three drawbacks. First, it is sensitive to pose and expression variations, since it becomes harder to *track* a specific face area for measuring intensity changes. Second, it is also sensitive to the changes in illumination, since the extra lighting affects the amount of reflected light from the skin surface. Third, for the purpose of anti-spoof, the rPPG signal extracted from spoof videos might not be sufficiently *distinguishable* to the signal of live videos.

One novelty aspect of our approach is that, instead of computing the rPPG signal via [35], our RNN part learns to estimate the rPPG signal. This eases the signal estimation from face videos with PIE variations, and also leads to more discriminative rPPG signals, as different rPPG supervisions are provided to live vs. spoof videos. We assume that the videos of the same subject under different PIE conditions have the *same* ground truth rPPG signal. This assumption is valid since the heart beat is similar for the videos of the same subject that are captured in a short span of time (< 5 minutes). The rPPG signal extracted from the constrained videos (i.e., no PIE variation) are used as the "ground truth" supervision in the rPPG loss function for *all* live videos of the same subject. This consistent supervision helps the CNN and RNN parts to be robust to the PIE changes.

In order to extract the rPPG signal from a face video without PIE, we apply the DeFA [74] to each frame and estimate the dense 3D face shape. We utilize the estimated 3D shape to track a face region. For a tracked region, we compute two orthogonal chrominance signals $\mathbf{x}_f = 3\mathbf{r}_f - 2\mathbf{g}_f$, $\mathbf{y}_f = 1.5\mathbf{r}_f + \mathbf{g}_f - 1.5\mathbf{b}_f$ where $\mathbf{r}_f, \mathbf{g}_f, \mathbf{b}_f$ are the bandpass filtered versions of the $\mathbf{r}, \mathbf{g}, \mathbf{b}$ channels with the skin-tone normalization. We utilize the ratio of the standard deviation of the chrominance signals $\gamma = \frac{\sigma(\mathbf{x}_f)}{\sigma(\mathbf{y}_f)}$ to compute blood flow signals [35]. We calculate the signal $\mathbf{p}$ as:

$$\mathbf{p} = 3(1 - \frac{\gamma}{2})\mathbf{r}_f - 2(1 + \frac{\gamma}{2})\mathbf{g}_f + \frac{3\gamma}{2}\mathbf{b}_f. \tag{5.3}$$

By applying FFT to $\mathbf{p}$, we obtain the rPPG signal $\mathbf{f} \in \mathbb{R}^{50}$, which shows the magnitude of each frequency.

5.2.3 Network Architecture

Our proposed network consists of two deep networks. First, a CNN part evaluates each frame separately and estimates the depth map and feature map of each frame. Second, a recurrent neural

network (RNN) part evaluates the temporal variability across the feature maps of a sequence.

5.2.3.1 CNN Network

We design a Fully Convolutional Network (FCN) as our CNN part, as shown in Fig. 5.3. The CNN part contains multiple blocks of three convolutional layers, pooling and resizing layers where each convolutional layer is followed by one exponential linear layer and batch normalization layer. Then, the resizing layers resize the response maps after each block to a pre-defined size of 64×64 and concatenate the response maps. The bypass connections help the network to utilize extracted features from layers with different depths similar to the ResNet structure [44]. After that, our CNN has two branches, one for estimating the depth map and the other for estimating the feature map.

The first output of the CNN is the estimated depth map of the input frame $\mathbf{I} \in \mathbb{R}^{256\times256}$, which is supervised by the estimated "ground truth" depth $\mathbf{D}$,

$$\Theta_D = \arg\min_{\Theta_D} \sum_{i=1}^{N_d} ||\text{CNN}_D(\mathbf{I}_i;\Theta_D) - \mathbf{D}_i||_1^2, \tag{5.4}$$

where Θ_D is the CNN parameters and N_d is the number of training images. The second output of the CNN is the feature map, which is fed into the non-rigid registration layer.

5.2.3.2 RNN Network

The RNN part aims to estimate the rPPG signal $\mathbf{f}$ of an input sequence with N_f frames $\{\mathbf{I}_j\}_{j=1}^{N_f}$. As shown in Fig. 5.3, we utilize one LSTM layer with 100 hidden neurons, one fully connected layer, and an FFT layer that converts the response of fully connected layer into the Fourier domain. Given the input sequence $\{\mathbf{I}_j\}_{j=1}^{N_f}$ and the "ground truth" rPPG signal $\mathbf{f}$, we train the RNN to minimize the ℓ_1 distance of the estimated rPPG signal to "ground truth" $\mathbf{f}$,

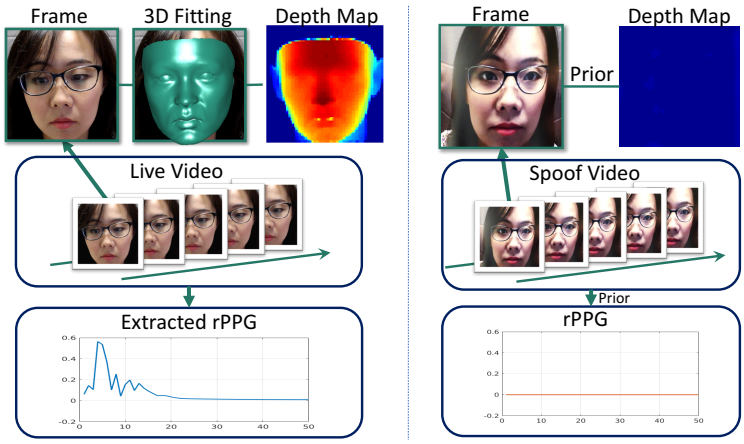


Figure 5.4: Example ground truth depth map and rPPG signals.

$$\Theta_R = \arg\min_{\Theta_R} \sum_{i=1}^{N_s} ||\text{RNN}_R([\{\mathbf{F}_j\}_{j=1}^{N_f}]_i; \Theta_R) - \mathbf{f}_i||_1^2, \tag{5.5}$$

where Θ_R is the RNN parameters, $\mathbf{F}_j \in \mathbb{R}^{32\times32}$ is the frontalized feature map (details in Sec. 5.2.4), and N_s is the number of sequences.

5.2.3.3 Implementation Details

Ground Truth Data Given a set of live and spoof face videos, we provide the ground truth supervision for the depth map $\mathbf{D}$ and rPPG signal $\mathbf{f}$. We follow the procedure in Sec. 5.2.1 to compute "ground truth" data for live videos. For spoof videos, we set the ground truth depth maps to a plain surface, i.e., zero depth. Similarly, we follow the procedure in Sec. 5.2.2 to compute the "ground truth" rPPG signal from a patch on the forehead, for one live video of each subject without PIE variation. Also, we normalize the norm of estimated rPPG signal such that $\|\mathbf{f}\|_2 = 1$. For spoof videos, we consider the rPPG signals are zero. Fig. 5.4 shows examples of the ground truth depth map and rPPG signal.

Note that, while the term "depth" is used here, our estimated depth is different to the conventional depth map in computer vision. It can be viewed as a "pseudo-depth" and serves the purpose of providing discriminative auxiliary supervision to the learning process. The same perspective

applies to the supervision based on pseudo-rPPG signal.

Training Strategy Our proposed network combines the CNN and RNN parts for end-to-end training. The desired training data for the CNN part should be from diverse subjects, so as to make the training procedure more stable and increase the generalizability of the learnt model. Meanwhile, the training data for the RNN part should be long sequences to leverage the temporal information across frames. These two preferences can be contradictory to each other, especially given the limited GPU memory. Hence, to satisfy both preferences, we design a two-stream training strategy. The first stream satisfies the preference of the CNN part, where the input includes face images $\mathbf{I}$ and the ground truth depth maps $\mathbf{D}$. The second stream satisfies the RNN part, where the input includes face sequences $\{\mathbf{I}_j\}_{j=1}^{N_f}$, the ground truth depth maps $\{\mathbf{D}_j\}_{j=1}^{N_f}$, the estimated 3D shapes $\{\mathbf{S}_j\}_{j=1}^{N_f}$, and the corresponding ground truth rPPG signals $\mathbf{f}$. During training, our method alternates between these two streams to converge to a model that minimizes both the depth map and rPPG losses. Note that even though the first stream only updates the weights of the CNN part, the back propagation of the second stream updates the weights of both CNN and RNN parts in an end-to-end manner.

Testing To provide a classification score, we feed the testing sequence to our network and compute the estimated depth map $\hat{\mathbf{D}}$ of the last frame and the rPPG signal $\hat{\mathbf{f}}$. To avoid overfitting due to utilizing binary loss function in the network, we compute the final score as:

$$score = ||\hat{\mathbf{f}}||_2^2 + \lambda ||\hat{\mathbf{D}}||_2^2, \tag{5.6}$$

where λ is a constant weight for combining the two outputs of the network.

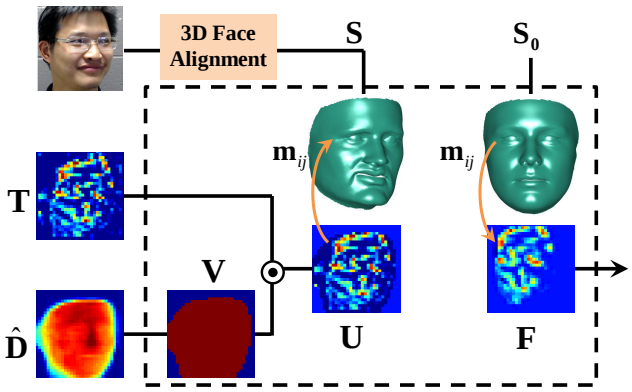


Figure 5.5: The non-rigid registration layer.

5.2.4 Non-rigid Registration Layer

We design a new non-rigid registration layer to prepare data for the RNN part. This layer utilizes the estimated dense 3D shape to align the activations or feature maps from the CNN part. This layer is important to ensure that the RNN tracks and learns the changes of the activations for the *same facial area* across time, as well as across all subjects.

As shown in Fig. 5.5, this layer has three inputs: the feature map $\mathbf{T} \in \mathbb{R}^{32\times32}$, the depth map $\hat{\mathbf{D}}$ and the 3D shape $\mathbf{S}$. Within this layer, we first threshold the depth map and generate a binary mask $\mathbf{V} \in \mathbb{R}^{32\times32}$:

$$\mathbf{V} = \hat{\mathbf{D}} \geq threshold. \tag{5.7}$$

Then, we compute the inner product of the binary mask and the feature map $\mathbf{U} = \mathbf{T} \odot \mathbf{V}$, which essentially utilizes the depth map as a visibility indicator for each pixel in the feature map. If the depth value for one pixel is less than the threshold, we consider that pixel to be invisible. Finally, we frontalize $\mathbf{U}$ by utilizing the estimated 3D shape $\mathbf{S}$,

$$\mathbf{F}(i,j) = \mathbf{U}(\mathbf{S}(\mathbf{m}_{ij},1),\mathbf{S}(\mathbf{m}_{ij},2)), \tag{5.8}$$

Table 5.1: The comparison of our collected dataset with available datasets for the face anti-spoofing.

Dataset	Year	# of subj.	# of sess.	# of live/attack vid. (V), ima. (I)	Pose range	Different expres.	Extra light.	Display devices	Spoof attacks
NUAA [104]	2010	15	3	5105/7509 (I)	Frontal	No	Yes	-	Print
CASIA-MFSD [143]	2012	50	3	150/450 (V)	Frontal	No	No	iPad	Print, Replay
Replay-Attack [28]	2012	50	1	200/1000 (V)	Frontal	No	Yes	iPhone 3GS, iPad	Print, 2 Replay
MSU-MFSD [117]	2015	35	1	110/330 (V)	Frontal	No	No	iPad Air, iPhone 5S	Print, 2 Replay
MSU-USSA [84]	2016	1140	1	1140/9120 (I)	$[-45°, 45°]$	Yes	Yes	MacBook, Nexus 5, Nvidia Shield Tablet	2 print, 6 Replay
Oulu-NPU [19]	2017	55	3	1980/3960 (V)	Frontal	No	Yes	Dell 1905FP, Macbook Retina	2 Print, 2 Replay
SiW	2018	165	4	1320/3300 (V)	$[-90°, 90°]$	Yes	Yes	iPad Pro, iPhone 7S, Galaxy S8, Asus MB168B	2 Print, 4 Replay

where $\mathbf{m} \in \mathbb{R}^K$ is the pre-defined list of K indexes of the face area in $\mathbf{S_0}$, and $\mathbf{m}_{ij}$ in corresponding index for the pixel i, j. We utilize $\mathbf{m}$ to project the masked activation values $\mathbf{U}$ to the frontalized image $\mathbf{F}$.

This non-rigid registration layer has three main contributions to the proposed network architecture:

$\diamond$ By applying the non-rigid registration, the input data are aligned and the RNN can compare the feature maps without concerning about the facial pose or expression. In other words, it can learn the temporal changes in the activations of the feature maps for the same facial area.

$\diamond$ The non-rigid registration removes the background area in the feature map. Hence the background area would not participate in RNN learning, although the background information is already utilized in the layers of the CNN part.

$\diamond$ For spoof faces, the depth maps are likely to be closer to zero. Hence, the inner product with the depth maps substantially weakens the activations in the feature maps, which makes it convenient for the RNN to output zero rPPG signals. Likewise, the back propagation from the rPPG loss also encourages the CNN part to generate zero depth maps for either all frames, or one pixel location in majority of the frames within an input sequence.

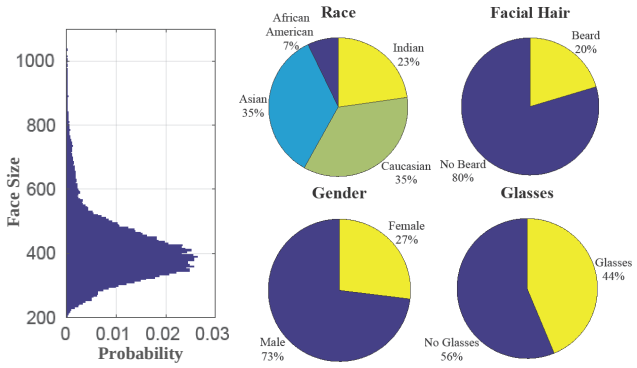


Figure 5.6: The statistics of the subjects in the SiW database. Left side: The histogram shows the distribution of the face sizes.

5.3 Collection of Face Anti-Spoofing Database

With the fast advancement of imaging sensor technology, the existing anti-spoofing systems could become vulnerable to emerging high-quality spoof mediums. One way to make the system robust to these attacks is to collect new high-quality databases. In responding to this need, we collect a new face anti-spoofing database named Spoof in the Wild (SiW) database, which has multiple advantages over previous datasets as shown in Tab. 5.1. First, it contains substantially more live subjects with diverse races, e.g., 5 times of the subjects of Oulu-NPU. Note that MSU-USSA is constructed based on existing images of celebrities without capturing live face videos. Second, live videos are captured with two high-quality cameras (Canon EOS T6, Logitech C920 webcam) with different PIE variations.

SiW provides live and spoof videos from 299 subjects. For each subject, we have 8 live and 16 spoof videos, in total $7,170$ videos. Some statistics of the subjects are shown in Fig. 5.6. The live videos are collected in four sessions. In Session 1, the subject moves his head with varying distances to the camera. In Session 2, the subject changes the yaw angle of the head within $[-90°, 90°]$, and makes different face expressions. In Sessions $3, 4$, the subject repeats the Sessions $1, 2$, while the collector moving the point light source around the face from different orientations.

Figure 5.7: Examples of the live and spoof attack videos in the SiW database. The first row shows a live subject with different PIE. The second row shows different types of the spoof attacks.

The live videos from the Canon EOS T6 and Logitech C920 webcam are in the resolution of $1,920 \times 1,080$. We provide two print and four replay video attacks for each subject in SiW. Some examples of the live and spoof videos are shown in Fig. 5.7. To generate different qualities of print attacks, we capture a high-resolution image ($5,184 \times 3,456$) for each subject and use it to make a high-quality print attack. Also, we extract a frontal-view frame from a live video for lower-quality print attack. We print the images with an HP color LaserJet M652 printer. The print attack videos are captured by holding printed images still and warping them in front of the cameras. To generate high-quality replay attack videos, we select four spoof mediums: Samsung Galaxy S8, iPhone 7, iPad Pro, and PC screens. For each subject, we randomly select two videos from the four high-quality live videos to display in the spoof mediums.

5.4 Experimental Results

5.4.1 Experimental Setup

Databases We evaluate our method on multiple databases to demonstrate its generalizability. We utilize SiW and Oulu databases [19] as new high-resolution databases and perform intra and cross testing between them. Also, we use the CASIA-MFSD [143] and Replay-Attack [28] databases

Table 5.2: TDR at different FDRs, cross testing on Oulu Protocol 1.

FDR	1%	2%	10%	20%
Model 1	8.5%	18.1%	71.4%	81.0%
Model 2	40.2%	46.9%	78.5%	93.5%
Model 3	39.4%	42.9%	67.5%	87.5%
Model 4	**45.8%**	**47.9%**	**81%**	**94.2%**

Table 5.3: ACER of our method at different N_f, on Oulu Protocol 2.

Test \ Train	5	10	20
5	4.16%	4.16%	3.05%
10	4.02%	3.61%	2.78%
20	4.10%	3.67%	2.98%

for cross testing and comparing with the state of the art.

Parameter setting The proposed method is implemented in TensorFlow [2] with a constant learning rate of 3e$-$3, and 10 epochs of the training phase. The batch size of the CNN stream is 10 and that of the CNN-RNN stream is 2 with N_f being 5. We randomly initialize our network by using a normal distribution with zero mean and std of 0.02. We set λ in Eq. 5.6 to 0.015 and *threshold* in Eq. 5.7 to 0.1.

Evaluation metrics To compare with prior works, we report our results with the following metrics: Attack Presentation Classification Error Rate $APCER$ [46], Bona Fide Presentation Classification Error Rate $BPCER$ [46], $ACER = \frac{APCER+BPCER}{2}$ [46], and Half Total Error Rate $HTER$. The $HTER$ is half of the summation of the False Rejection Rate (FRR) and the False Acceptance Rate (FAR).

Table 5.4: The intra-testing results on four protocols of Oulu.

Prot.	Method	APCER (%)	BPCER (%)	ACER (%)
1	CPqD	2.9	10.8	6.9
	GRADIANT	**1.3**	12.5	6.9
	Proposed method	1.6	**1.6**	**1.6**
2	MixedFASNet	9.7	2.5	6.1
	Proposed method	**2.7**	2.7	2.7
	GRADIANT	3.1	**1.9**	**2.5**
3	MixedFASNet	5.3 ± 6.7	7.8 ± 5.5	6.5 ± 4.6
	GRADIANT	$\mathbf{2.6 \pm 3.9}$	5.0 ± 5.3	3.8 ± 2.4
	Proposed method	2.7 ± 1.3	$\mathbf{3.1 \pm 1.7}$	$\mathbf{2.9 \pm 1.5}$
4	Massy_HNU	35.8 ± 35.3	$\mathbf{8.3 \pm 4.1}$	22.1 ± 17.6
	GRADIANT	$\mathbf{5.0 \pm 4.5}$	15.0 ± 7.1	10.0 ± 5.0
	Proposed method	9.3 ± 5.6	10.4 ± 6.0	$\mathbf{9.5 \pm 6.0}$

5.4.2 Experimental Comparison

5.4.2.1 Ablation Study

Advantage of proposed architecture We compare four architectures to demonstrate the advantages of the proposed loss layers and non-rigid registration layer. *Model* 1 has an architecture similar to the CNN part in our method (Fig. 5.3), except that it is extended with additional pooling layers, fully connected layers, and softmax loss for binary classification. *Model* 2 is the CNN part in our method with a depth map loss function. We simply use $||\hat{\mathbf{D}}||_2$ for classification. *Model* 3 contains the CNN and RNN parts without the non-rigid registration layer. Both of the depth map and rPPG loss functions are utilized in this model. However, the RNN part would process unregistered feature maps from the CNN. *Model* 4 is the proposed architecture.

We train all four models with the live and spoof videos from 20 subjects of SiW. We compute the cross-testing performance of all models on Protocol 1 of Oulu database. The TDR at different FDR are reported in Tab. 5.2. *Model* 1 has a poor performance due to the binary supervision. In comparison, by only using the depth map as supervision, *Model* 2 achieves substantially better performance. However, after adding the RNN part with the rPPG supervision, our proposed *Model* 4 can further the performance improvement. By comparing *Model* 4 and 3, we can see the advantage

of the non-rigid registration layer. It is clear that the RNN part cannot use feature maps directly for tracking the changes in the activations and estimating the rPPG signals.

Advantage of longer sequences To show the advantage of utilizing longer sequences for estimating the rPPG, we train and test our model when the sequence length N_f is 5, 10, or 20, using intra-testing on Oulu Protocol 2. From Tab. 5.3, we can see that by increasing the sequence length, the ACER decreases due to more reliable rPPG estimation. Despite the benefit of longer sequences, in practice, we are limited by the GPU memory size, and forced to decrease the image size to 128×128 for all experiments in Tab. 5.3. Hence, we set N_f to be 5 with the image size of 256×256 in subsequent experiments, due to importance of higher resolution (e.g, a lower *ACER* of 2.5% in Tab. 5.4 is achieved than 4.16%).

5.4.2.2 Intra Testing

We perform intra testing on Oulu and SiW databases. For Oulu, we follow the four protocols [15] and report their *APCER*, *BPCER* and *ACER*. Tab. 5.4 shows the comparison of our proposed method and the best two methods for *each* protocol respectively, in the face anti-spoofing competition [15]. Our method achieves the lowest *ACER* in 3 out of 4 protocols. We have slightly worse *ACER* on Protocol 2. To set a baseline for future study on SiW, we define three protocols for SiW. The Protocol 1 deals with variations in face pose and expression. We train using the first 60 frames of the training videos that are mainly frontal view faces, and test on all testing videos. The Protocol 2 evaluates the performance of cross spoof medium of replay attack. The Protocol 3 evaluates the performance of cross PA, i.e., from print attack to replay attack and vice versa. Tab. 5.5 shows the protocol definition and our performance of each protocol.

Table 5.5: The intra-testing results on three protocols of SiW.

Prot.	Subset	Subject #	Attack	APCER (%)	BPCER (%)	ACER (%)
1	Train	90	First 60 Frames	3.58	3.58	3.58
	Test	75	All			
2	Train	90	3 display	0.57 ± 0.69	0.57 ± 0.69	0.57 ± 0.69
	Test	75	1 display			
3	Train	90	print (display)	8.31 ± 3.81	8.31 ± 3.80	8.31 ± 3.81
	Test	75	display (print)			

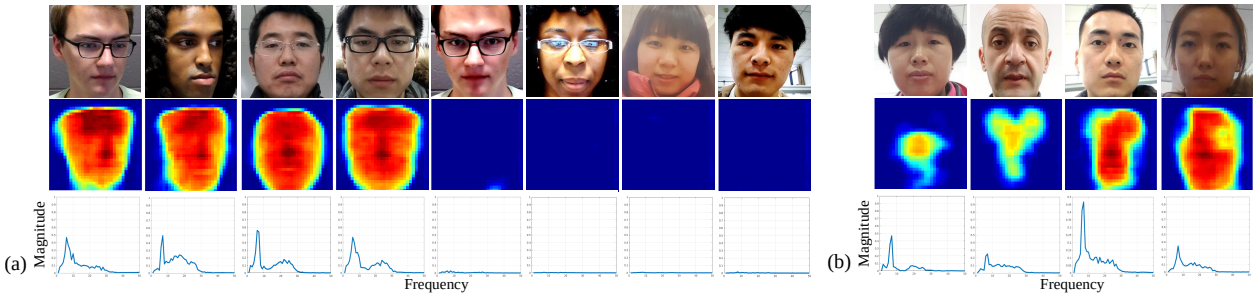


Figure 5.8: (a) 8 successful anti-spoofing examples and their estimated depth maps and rPPG signals. (b) 4 failure examples: the first two are live and the other two are spoof. Note our ability to estimate discriminative depth maps and rPPG signals.

5.4.2.3 Cross Testing

To demonstrate the generalization of our method, we perform multiple cross-testing experiments. Our model is trained with live and spoof videos of 80 subjects in SiW, and test on all protocols of Oulu. The *ACER* on Protocol 1-4 are respectively: 10.0%, 14.1%, $13.8 \pm 5.7\%$, and $10.0 \pm 8.8\%$. Comparing these cross-testing results to the *intra-testing* results in [15], we are ranked sixth on the average *ACER* of four protocols, among the 15 participants of the face anti-spoofing competition. Especially on Protocol 4, the hardest one among all protocols, we achieve the *same ACER* of 10.0% as the top performer. This is a notable result since cross testing is known to be substantially harder than intra testing, and yet our cross-testing result is comparable with the top intra-testing performance. This demonstrates the generalization ability of our learnt model.

Furthermore, we utilize the CASIA-MFSD and Replay-Attack databases to perform cross testing between them, which is widely used as a cross-testing benchmark. Tab. 5.6 compares the

Table 5.6: Cross testing on CASIA-MFSD vs. Replay-Attack.

Method	Train	Test	Train	Test
	CASIA-MFSD	Replay Attack	Replay Attack	CASIA-MFSD
Motion [34]	50.2%		47.9%	
LBP [34]	55.9%		57.6%	
LBP-TOP [34]	49.7%		60.6%	
Motion-Mag [11]	50.1%		47.0%	
Spectral cubes [90]	34.4%		50.0%	
CNN [130]	48.5%		45.5%	
LBP [16]	47.0%		39.6%	
Colour Texture [17]	30.3%		37.7%	
Proposed method	**27.6%**		**28.4%**	

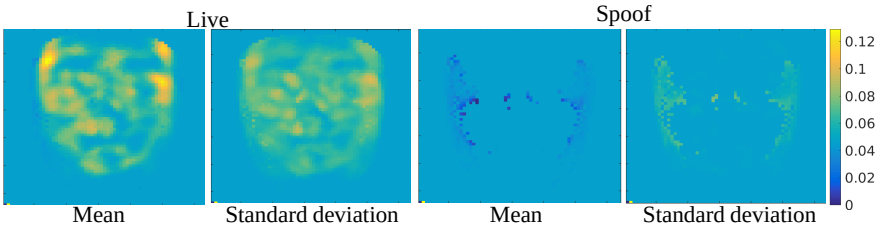


Figure 5.9: Mean/Std of frontalized feature maps for live and spoof.

cross-testing *HTER* of different methods. Our proposed method reduces the cross-testing errors on the Replay-Attack and CASIA-MFSD databases by 8.9% and 24.6% respectively, relative to the previous SOTA.

5.4.2.4 Visualization and Analysis

In the proposed architecture, the frontalized feature maps are utilized as input to the RNN part and are supervised by the rPPG loss function. The values of these maps can show the importance of different facial areas to rPPG estimation. Fig. 5.9 shows the mean and standard deviation of frontalized feature maps, computed from $1,080$ live and spoof videos of Oulu. We can see that the side areas of forehead and cheek have higher influence for rPPG estimation.

While the goal of our system is to detect PAs, our model is trained to estimate the auxiliary information. Hence, in addition to anti-spoof, we also like to evaluate the accuracy of auxiliary

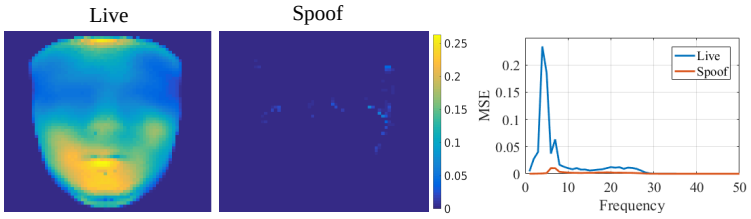


Figure 5.10: The MSE of estimating depth maps and rPPG signals.

information estimation. For this purpose, we calculate the accuracy of estimating depth maps and rPPG signals, for testing data in Protocol 2 of Oulu. As shown in Fig. 5.10, the accuracy for both estimation in spoof data is high, while that of the live data is relatively lower. Note that the depth estimation of the mouth area has more errors, which is consistent with the fewer activations of the same area in Fig. 5.9. Examples of successful and failure cases in estimating depth maps and rPPG signals are shown in Fig. 5.8.

Finally, we conduct statistical analysis on the failure cases, since our system can determine potential causes using the auxiliary information. With Proctocol 2 of Oulu, we identify 31 failure cases (2.7% *ACER*). For each case, we calculate whether anti-spoofing using its depth map or rPPG signal would fail if that information alone is used. In total, $\frac{29}{31}$, $\frac{13}{31}$, and $\frac{11}{31}$ samples fail due to depth map, rPPG signals, or both. This indicates the future research direction.

5.5 Summary

This chapter identifies the importance of auxiliary supervision to deep model-based face anti-spoofing. The proposed network combines CNN and RNN architectures to jointly estimate the depth of face images and rPPG signal of face video. We introduce the SiW database that contains more subjects and variations than prior databases. Finally, we experimentally demonstrate the superiority of our method.

Chapter 6

Face De-Spoofing: Anti-Spoofing via Noise Modeling

6.1 Introduction

The print and the replay attacks are the the most common spoofs types and they have been well studied previously, from different perspectives. The cue-based methods aim to detect liveness cues [82, 83] (e.g., eye blinking, head motion) to classify live videos. But these methods can be fooled by video replay attacks. The texture-based methods attempt to compare texture difference between live and spoof faces, using pre-defined features such as LBP [33, 34], HOG [59, 131]. Similar to texture-based methods, CNN-based methods [66, 83, 130] design a unified process of feature extraction and classification. With a softmax loss based binary supervision, they have the risk of overfitting on the training data. Regardless of the perspectives, almost all the prior works treat face anti-spoofing as a *black box* binary classification problem. In contrast, in this chapter, we propose to open the black box by modeling the process of how a spoof image is generated from its original live image.

Our approach is motivated by the classic image de-X problems, such as image de-noising and de-blurring [36, 51, 62, 85]. In image de-noising, the corrupted image is regarded as a degradation from the additive noise, e.g., salt-and-pepper noise and white Gaussian noise. In image de-blurring, the uncorrupted image is degraded by motion, which can be described as a process of convolution.

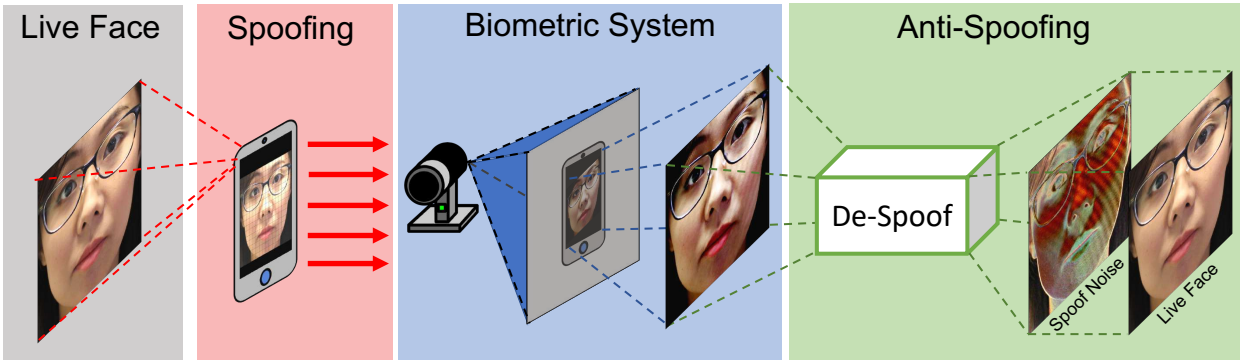


Figure 6.1: The illustration of face spoofing and anti-spoofing processes. De-spoofing process aims to estimate a spoof noise from a spoof face and reconstruct the live face. The estimated spoof noise should be discriminative for face anti-spoofing.

Similarly, in face anti-spoofing, the spoof image can be viewed as a re-rendering of the live image but with some “special" noise from the spoof medium and the environment. Hence, the natural question is, *can we recover the underlying live image when given a spoof image, similar to image de-noising*?

Yes. This chapter shows “how" to do this. We call the process of decomposing a spoof face to the spoof noise pattern and a live face as *Face De-spoofing*, shown in Fig. 6.1. Similar to the previous de-X works, the degraded image $\mathbf{x} \in \mathbb{R}^m$ can be formulated as a function of the original image $\hat{\mathbf{x}}$, the degradation matrix $\mathbf{A} \in \mathbb{R}^{m \times m}$ and an additive noise $\mathbf{n} \in \mathbb{R}^m$.

$$\mathbf{x} = \mathbf{A}\hat{\mathbf{x}} + \mathbf{n} = \hat{\mathbf{x}} + (\mathbf{A} - \mathbb{I})\hat{\mathbf{x}} + \mathbf{n} = \hat{\mathbf{x}} + N(\hat{\mathbf{x}}), \tag{6.1}$$

where $N(\hat{\mathbf{x}}) = (\mathbf{A} - \mathbb{I})\hat{\mathbf{x}} + \mathbf{n}$ is the image-dependent noise function. Instead of solving $\mathbf{A}$ and $\mathbf{n}$, we decide to estimate $N(\hat{\mathbf{x}})$ directly since it is more solvable under the deep learning framework [40, 63, 100, 101, 145]. Essentially, by estimating $N(\hat{\mathbf{x}})$ and $\hat{\mathbf{x}}$, we aim to peel off the spoof noise and reconstruct the original live face. Likewise, if given a live face, face de-spoofing model should return itself plus *zero* noise. Note that our face de-spoofing is designed to handle paper attack,

replay attack and possibly make-up attack, but our experiments are limited to the first two PAs. The benefits of face de-spoofing are twofold: 1) it reverses, or undoes, the spoofing generation process, which helps us to model and visualize the spoof noise pattern of different spoof mediums. 2) the spoof noise itself is discriminative between live and spoof images and hence is useful for face anti-spoofing.

While face de-spoofing shares the same challenges as other image de-X problems, it has a few distinct difficulties to conquer:

No Ground Truth: Image de-X works often use synthetic data where the original undegraded image could be used as ground truth for supervised learning. In contrast, we have no access to $\hat{\mathbf{x}}$, which is the corresponding live face of a spoof face image.

No Noise Model: There is no comprehensive study and understanding about the spoof noise. Hence it is not clear how we can constrain the solution space to *faithfully* estimate the spoof noise pattern.

Diverse Spoof Mediums: Each type of spoofs utilizes different spoof mediums for generating spoof images. Each spoof medium represents a specific type of noise pattern.

To address these challenges, we propose several constraints and supervisions based on our prior knowledge and the conclusions from a case study (in Section 6.2.1). Given that a live face has no spoof noise, we impose the constraint that $N(\hat{\mathbf{x}})$ of a live image is *zero*. Based on our study, we assume that the spoof noise of a spoof image is ubiquitous, i.e., it exists everywhere in the spatial domain of the image; and is repetitive, i.e., it is the spatial repetition of certain noise in the image. The repetitiveness can be encouraged by maximizing the high-frequency magnitude of the estimated noise in the Fourier domain.

With such constraints and auxiliary supervisions proposed in [73], a novel CNN architecture is presented in this paper. Given an image, one CNN is designed to synthesize the spoof noise pattern

and reconstruct the corresponding live image. In order to examine the reconstructed live image, we train another CNN with auxiliary supervision and a GAN-like discriminator in an end-to-end fashion. These two networks are designed to ensure the quality of the reconstructed image regarding its discriminativeness between live and spoof, and the visual plausibility of the synthesized live image.

To summarize, the main contributions of this work include:

◇ We offer a new perspective for detecting the spoofing face from print attack and replay attack by inversely decomposing a spoof face image into the live face and the spoofing noise, without having the ground truth of either.

◇ A novel CNN architecture is proposed for face de-spoofing, where appropriate constraints and auxiliary supervisions are imposed.

◇ We demonstrate the value of face de-spoofing by its contribution to face anti-spoofing and the visualization of the spoof noise patterns.

6.2 Face De-spoofing

In this section, we start with a case study of spoof noise pattern, which demonstrates a few important characteristics of the noise. This study motivates us to design the novel CNN architecture that will be presented in Sec. 6.2.2.1.

6.2.1 A Case Study of Spoof Noise Pattern

The core task of face de-spoofing is to estimate the spoofing-relevant noise pattern in the given face image. Despite the strength of using a CNN model, we are still facing the challenge of learning *without* the ground truth of the noise pattern. To address this challenge, we would like to first carry

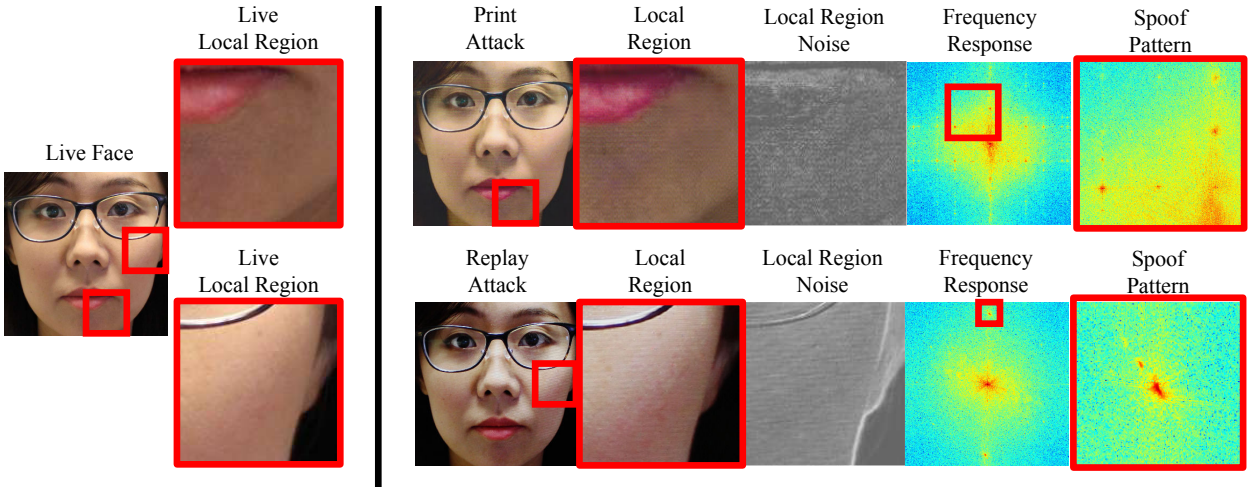


Figure 6.2: The illustration of the spoof noise pattern. **Left:** live face and its local regions. **Right:** Two registered spoofing faces from print attack and replay attack. For each sample, we show the local region of the face, intensity difference to the live image, magnitude of 2D FFT, and the local peaks in the frequency domain that indicates the spoof noise pattern. Best viewed electronically.

out a case study on the noise pattern with the objectives of answering the following questions: 1) is Eqn. 6.1 a good modeling of the spoof noise? 2) what characteristics does the spoof noise hold?

Let us denote a genuine face as $\hat{\mathbf{I}}$. By using printed paper or video replay on digital devices, the attacker can manufacture a spoof image $\mathbf{I}$ from $\hat{\mathbf{I}}$. Considering no non-rigid deformation between $\mathbf{I}$ and $\hat{\mathbf{I}}$, we summarize the degradation from $\hat{\mathbf{I}}$ to $\mathbf{I}$ as the following steps:

1. **Color distortion:** Color distortion is due to a narrower color gamut of the spoof medium (e.g. LCD screen or Toner Cartridge). It is a projection from the original color space to a tinier color subspace. This noise is dependent on the color intensity of the subject, and hence it may apply as a degradation matrix to the genuine face $\mathbf{I}$ during the degradation.

2. **Display artifacts:** Spoof mediums often use several nearby dots/sensors to approximate one pixel's color, and they may also display the face differently than the original size. Approximation and down-sampling procedure would cause a certain degree of high-frequency information loss, blurring, and pixel perturbation. This noise may also apply as a degradation matrix due to its subject dependence.

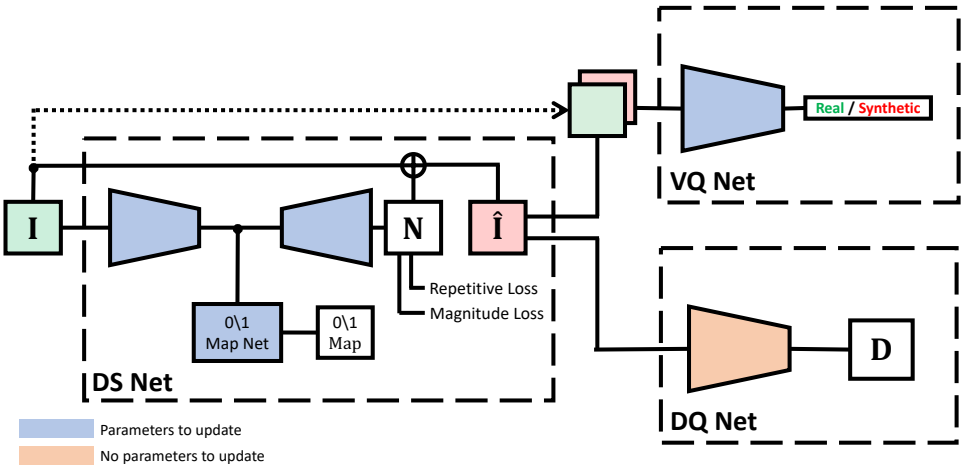


Figure 6.3: The proposed network architecture.

3. **Presenting artifacts:** When presenting the spoof medium to the camera, the medium interacts with the environment and brings several artifacts, including reflection and transparency of the surface. This noise may apply as an additive noise.

4. **Imaging artifacts:** Imaging lattice patterns such as screen pixels on the camera's sensor array (e.g. CMOS and CCD) would cause interference of light. This effect leads to aliasing and creates moiré pattern, which appears in replay attack and some print attack with strong lattice artifacts. This noise may apply as an additive noise.

These four steps show that the spoof image $\mathbf{I}$ can be generated via applying degradation matrices and additive noises to $\hat{\mathbf{I}}$, which is basically conveyed by Eqn. 6.1. As expressed by Eqn. 6.1, the spoof image is the summation of the live image and image-dependent noise. To further validate this model, we show an example in Fig. 6.2. Given a high-quality live image, we carefully produce two spoof images via print and replay attack, with minimal non-rigid deformation. After each spoof image is registered with the live image, the live image becomes the *ground truth* live image if we would perform de-spoofing on the spoof image. This allows us to compute the difference between the live and spoof images, which is the noise pattern $N(\hat{\mathbf{I}})$. To analyze its frequency properties, we

perform FFT on the spoof noise and show the 2D shifted magnitude response.

In both spoof cases, we observe a high response in the low-frequency domain, which is related to color distortion and display artifacts. In print attack, *repetitive* noise in Step 3 leads to a few “peak" responses in the high-frequency domain. Similarly, in the replay attack, visible moiré pattern reflects as several spurs in the low-frequency domain, and the lattice pattern that causes the moiré pattern is represented as peaks in the high-frequency domain. Moreover, spoof patterns are uniformly distributed in the image domain due to the uniform texture of the spoof mediums. And the high response of the repetitive pattern in the frequency domain exactly demonstrates that it appears widely in the image and thus can be viewed as ubiquitous.

Under this ideal registration, the comparison between live and spoof images provides us a basic understanding of the spoof noise pattern. It is a type of texture that has the characteristics of **repetitive** and **ubiquitous**. Based on this modeling and noise characteristics, we design a network to estimate the noise *without* the access to the precisely registered ground truth live image, as this case study has.

6.2.2 De-Spoof Network

6.2.2.1 Network Overview

Figure 6.3 shows the overall network architecture of our proposed method. It consists of three parts: De-Spoof Net (DS Net), Discriminative Quality Net (DQ Net), and Visual Quality Net (VQ Net). DS Net is designed to estimate the spoof noise pattern $\mathbf{N}$ (i.e. the output of $N(\hat{\mathbf{I}})$) from the input image $\mathbf{I}$. The live face $\hat{\mathbf{I}}$ then can be reconstructed by subtracting the estimated noise $\mathbf{N}$ from the input image $\mathbf{I}$. This reconstructed image $\hat{\mathbf{I}}$ should be both visually appealing and indeed live, which will be safeguarded by the DQ Net and VQ Net respectively. All networks can be trained in

Table 6.1: The network structure of DS Net, DQ Net and VQ Net. Each convolutional layer is followed by an exponential linear unit (ELU) and batch normalization layer. The input image size for DS Net is $256 \times 256 \times 6$. All the convolutional filters are 3×3. 0\1 Map Net is the bottom-left part, i.e., conv1-10, conv1-11, and conv1-12.

DS Net (Encoder Part)			DS Net (Decoder Part)			DQ Net			VQ Net		
Layer	Chan./Stri.	Outp. Size	Layer	Chan./Stri.	Outp. Size	Layer	Chan./Stri.	Outp. Size	Layer	Chan./Stri.	Outp. Size
Input			**Input**			**Input**			**Input**		
image			pool1-1+pool1-2+pool1-3			{image,live}			{image,live}		
conv1-0	24/1	256	resize	-/-	256	conv3-0	64/1	256			
conv1-1	20/1	256	conv2-1	28/1	256	conv3-1	128/1	256	conv4-1	24/2	256
conv1-2	25/1	256	conv2-2	24/1	256	conv3-2	196/1	256	conv4-2	20/2	256
conv1-3	20/1	256				conv3-3	128/1	256	pool4-1	-/2	128
pool1-1	-/2	128				pool3-1	-/2	128			
conv1-4	20/1	128	conv2-3	20/1	256	conv3-4	128/1	128	conv4-3	20/1	128
conv1-5	25/1	128	conv2-4	20/1	256	conv3-5	196/1	128	conv4-4	16/1	128
conv1-6	20/1	128				conv3-6	128/1	128	pool4-2	-/2	64
pool1-2	-/2	64				pool3-2	-/2	64			
conv1-7	20/1	64	conv2-5	20/1	256	conv3-7	128/1	64	conv4-5	12/1	64
conv1-8	25/1	64	conv2-6	16/1	256	conv3-8	196/1	64	conv4-6	6/1	64
conv1-9	20/1	64				conv3-9	128/1	64	pool4-3	-/2	32
pool1-3	-/2	32				pool3-3	-/2	32			
short-cut connection						**short-cut connection**			**vectorize**		
pool1-1+pool1-2+pool1-3						pool3-1+pool3-2+pool3-3					1024
conv1-10	28/1	32	conv2-7	16/1	256	conv3-10	128/1	32	fc4-1	1/1	100
conv1-11	16/1	32	conv2-8	6/1	256	conv3-11	64/1	32	dropout	-	0.2%
conv1-12	1/1	32	live	(image - conv2-8)		conv3-12	1/1	32	fc4-2	1/1	2

an end-to-end fashion. The details of the network structure are shown in Tab. 6.1.

As the core part, DS Net is designed as an encoder-decoder structure with the input $\mathbf{I} \in \mathbb{R}^{256\times256\times6}$. Here the 6 channels are RGB + HSV color space, following the suggestion in [5]. In the encoder part, we first stack 10 convolutional layers with 3 pooling layers. Inspired by the residual network [44], we follow by a short-cut connection: concatenating the responses from *pool*1-1, *pool*1-2 with *pool*1-3, and then sending them to *conv*1-10. This operation helps us to pass the feature responses from different scales to the later stages and ease the training procedure. Going through 3 more convolution layers, the responses $\mathbf{F} \in \mathbb{R}^{32\times32\times32}$ from *conv*1-12 are the feature representation of the spoof noise patterns. The higher magnitudes the responses have, the more spoofing-perceptible the input is.

Out from the encoder, the feature representation $\mathbf{F}$ is fed into the decoder to reconstruct the spoof noise pattern. $\mathbf{F}$ is directly resized to the input spatial size 256×256. It introduces no extra

grid artifacts, which exist in the alternative approach of using a deconvolutional layer. Then, we pass the resized **F** to several convolutional layers to reconstruct the noise pattern **N**. According to Eqn. 6.1, the reconstructed live image can be retrieved by: $\hat{\mathbf{x}} = \mathbf{x} - N(\hat{\mathbf{x}}) = \mathbf{I} - \mathbf{N}$.

Each convolutional layer in the DS Net is equipped with exponential linear unit (ELU) and batch normalization layers. To supervise the training of DS Net, we design multiple loss functions: losses from DQ Net and VQ Net for the image quality, 0\1 map loss, and noise property losses. We introduce these loss functions in Sec. 6.2.3-6.2.4.

6.2.3 DQ Net and VQ Net

While we do not have the ground truth to supervise the estimated spoof noise pattern, it is possible to supervise the reconstructed live image, which implicitly guides the noise estimation. To estimate a good-quality spoof noise, the reconstructed live image should be quantitatively and visually recognized as live. For this purpose, we propose two networks in our architecture: Discriminative Quality Net (DQ Net) and Visual Quality Net (VQ Net). The VQ Net aims to guarantee the reconstructed live face is photorealistic. The DQ Net is proposed to guarantee the reconstructed face would indeed be considered as live, based on the judgment of a pre-trained face anti-spoofing network. The details of our proposed architecture are shown in Tab. 6.1.

6.2.3.1 Discriminative Quality Net

We follow the state-of-the-art network architecture of face anti-spoofing [73] to build our DQ Net. It is a fully convolutional network with three filter blocks and three additional convolutional layers. Each block consists of three convolutional layers and one pooling layer. The feature maps after each pooling layer are resized and stacked to feed into the following convolutional layers. Finally, DQ Net is supervised to estimate the pseudo-depth **D** of an input face, where **D** for the live face is

the depth of the face shape and **D** for the spoof face is a zero map as a flat surface. We adopt the 3D face alignment algorithm in [74] to estimate the face shape and render the depth via Z-Buffering.

Similar to the previous work [50], DQ Net is pre-trained to obtain the semantic knowledge of live faces and spoofing faces. And during the training of DS Net, the parameters of DQ Net are fixed. Since the reconstructed images $\hat{\mathbf{I}}$ are live images, the corresponding pseudo-depth **D** should be the depth of the face shape. The backpropagation of the error from DQ Net guides the DS Net to estimate the spoof noise pattern which should be subtracted from the input image,

$$J_{DQ} = \left\| \text{CNN}_{DQ}(\hat{\mathbf{I}}) - \mathbf{D} \right\|_1, \tag{6.2}$$

where CNN_{DQ} is a fixed network and **D** is the depth of the face shape.

6.2.3.2 Visual Quality Net

We deploy a GAN to verify the visual quality of the estimated live image $\hat{\mathbf{I}}$. Given both the real live image $\mathbf{I}_{\textbf{live}}$ and the synthesized live image $\hat{\mathbf{I}}$, VQ Net is trained to distinguish between $\mathbf{I}_{\textbf{live}}$ and $\hat{\mathbf{I}}$. Meanwhile, DS Net tries to reconstruct photorealistic live images where the VQ Net would classify them as non-synthetic (or real) images. The VQ Net consists of 6 convolutional layers and a fully connected layer with a 2D vector as the output, which represents the probability of the input image to be real or synthetic. In each iteration during the training, the VQ Net is evaluated with two batches, in the first one, the DS Net is fixed and we update the VQ Net,

$$J_{VQ_{train}} = -\mathbb{E}_{\mathbf{I}\in\mathscr{R}} \log(\text{CNN}_{VQ}(\mathbf{I})) - \mathbb{E}_{\mathbf{I}\in\mathscr{S}} \log(1 - \text{CNN}_{VQ}(\text{CNN}_{DS}(\mathbf{I}))), \tag{6.3}$$

where $\mathscr{R}$ and $\mathscr{S}$ are the sets of real and synthetic images respectively. In the second batch, the VQ Net is fixed and the DS Net is updated,

$$J_{VQ_{test}} = -\mathbb{E}_{\mathbf{I}\in\mathscr{S}} \log(\text{CNN}_{VQ}(\text{CNN}_{DS}(\mathbf{I}))). \tag{6.4}$$

6.2.4 Loss functions

The main challenge for spoof modeling is the lack of the ground truth for the spoof noise pattern. Since we have concluded some properties about the spoof noise in Sec. 6.2.1, we can leverage them to design several novel loss functions to constrain the convergence space. First, we introduce magnitude loss to enforce the spoof noise of the live image to be zero. Second, zero\one map loss is used to demonstrate the ubiquitousness of the spoof noise. Third, we encourage the repetitiveness property of spoof noise via repetitive loss. We describe three loss functions as the following:

6.2.4.1 Magnitude Loss

The spoof noise pattern for the live images is zero. The magnitude loss can be utilized to impose the constraint for the estimated noise. Given the estimated noise $\mathbf{N}$ and reconstructed live image $\hat{\mathbf{I}} = \mathbf{I} - \mathbf{N}$ of an original live image $\mathbf{I}$, we have,

$$J_m = \|\mathbf{N}\|_1 . \tag{6.5}$$

Zero\One Map Loss: To learn discriminative features in the encoder layers, we define a sub-task in the DS Net to estimate a zero-map for the live faces and an one-map for the spoof. Since this is a per *pixel* supervision, it is also a constraint of ubiquitousness on the noise. Moreover, 0\1 map

enables the receptive field of each pixel to cover a local area, which helps to learn generalizable features for this problem. Formally, given the extracted features $\mathbf{F}$ from an input face image $\mathbf{I}$ in the encoder, we have,

$$J_z = \left\| \mathrm{CNN}_{01map}(\mathbf{F};\Theta) - \mathbf{M} \right\|_1, \tag{6.6}$$

where $\mathbf{M} \in \mathbf{0}^{32\times32}$ or $\mathbf{M} \in \mathbf{1}^{32\times32}$ is the zero\one map label.

6.2.4.2 Repetitive Loss

Based on the previous discussion, we assume the spoof noise pattern to be repetitive, because it is generated from the repetitive spoof medium. To encourage the repetitiveness, we convert the estimated noise $\mathbf{N}$ to the Fourier domain and compute the maximum value in the high-frequency band. The existence of high peak is indicative of the repetitive pattern. We would like to maximize this peak for spoof images, but minimize it for live images, as the following loss function:

$$J_r = \begin{cases} -\max(H(\mathscr{F}(\mathbf{N}),k)), & \mathbf{I} \in Spoof \\ \|\max(H(\mathscr{F}(\mathbf{N}),k))\|_1, & \mathbf{I} \in Live \end{cases},$$

where $\mathscr{F}$ is the Fourier transform operator, H is an operator for masking the low-frequency domain of an image, i.e., setting a $k \times k$ region in the center of the shifted 2D Fourier response to zero.

Finally, the total loss function in our training is the weighted summation of the aforementioned loss functions and the supervisions for the image qualities,

$$J_T = J_z + \lambda_1 J_m + \lambda_2 J_r + \lambda_3 J_{DQ} + \lambda_4 J_{VQ_{test}}, \tag{6.7}$$

where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are the weights. During the training, we alternate between optimizing Eqn. 6.7 and Eqn. 6.3.

6.3 Experimental Results

6.3.1 Experimental Setup

Databases We evaluate our work on three face anti-spoofing databases, with print and replay attacks: Oulu-NPU [19], CASIA-MFSD [143] and Replay-Attack [28]. Oulu-NPU [19] is a high-resolution database, considering many real-world variations. Oulu-NPU also includes 4 testing protocols: Protocol 1 evaluates on the illumination variation, Protocol 2 examines the influence of different spoof medium, Protocol 3 inspects the effect of different camera devices and Protocol 4 contains all the challenges above, which is close to the scenario of cross testing. CASIA-MFSD [143] contains videos with resolution 640×480 and 1280×720. Replay-Attack [28] includes videos of 320×240. These two databases are often used for cross testing [83].

Parameter setting We implement our method in Tensorflow [2]. Models are trained with the batch size of 6 and the learning rate of 3e$-$5. We set the $k = 64$ in the repetitive loss and set λ_1 to λ_4 in Eqn. 6.7 as $3, 0.005, 0.1$ and 0.016, respectively. DQ Net is trained separately and remains fixed during the update of DS Net and VQ Net, but all sub-networks are trained with the same and respective data in each protocol.

Evaluation metrics To compare with previous methods, we use Attack Presentation Classification Error Rate (*APCER*) [46], Bona Fide Presentation Classification Error Rate (*BPCER*) [46] and, $ACER = (APCER + BPCER)/2$ [46] for the intra testing on Oulu-NPU, and Half Total Error Rate (*HTER*) [9], half of the summation of FAR and FRR, for the cross testing between CASIA-MFSD and Replay-Attack.

Table 6.2: The accuracy of different outputs of the proposed architecture and their fusions.

Method	0\1 map	Spoof noise	Depth map	Fusion (Spoof noise, Depth map)		Fusion of all three outputs	
				Maximum	Average	Maximum	Average
APCER	2.50	1.70	1.66	1.70	1.27	1.70	1.27
BPCER	2.52	1.70	1.68	1.73	1.73	1.73	1.73
ACER	2.51	1.70	1.67	1.72	1.50	1.72	1.50

6.3.2 Ablation Study

Using Oulu-NPU Protocol 1, we perform three studies on the effect of score fusing, the importance of each loss function, and the influence of image resolution and blurriness.

Different fusion methods In the proposed architecture, three outputs can be utilized for classification: the norms of either the 0\1 map, the spoof noise pattern or the depth map. Because of the discriminativeness enabled by our learning, we can simply use a rudimentary classifier like L-1 norm. Note that a more advance classifier is applicable and would likely lead to higher performance. Table 6.2 shows the performance of each output and their fusion with maximum and average. It shows that the fusion of spoof noise and depth map achieves the best performance. However, adding the 0\1 map scores do not improve the accuracy since it contains the same information as the spoof noise. Hence, for the rest of experiments, we report performance from the average fusion of the spoof noise $\mathbf{N}$ and the depth map $\hat{\mathbf{D}}$, i.e., $score = (\|\mathbf{N}\|_1 + \left\|\hat{\mathbf{D}}\right\|_1)/2$.

Advantage of each loss function We have three main loss functions in our proposed architecture. To shows the effect of each loss function, we train a network with each loss excluded one by one. By disabling the magnitude loss, the 0\1 map loss and the repetitive loss, we obtain the ACERs 5.24, 2.34 and 1.50, respectively. To further validate the repetitive loss, we perform an experiment on high-resolution images by changing the network input to the cheek region of the original 1080P resolution. The ACER of the network with the repetitive loss is 2.92 but the network without cannot converge.

Resolution and blurriness As shown in the ablation study of repetitive loss, the image qual-

Table 6.3: ACER of the proposed method with different image resolutions and blurriness. To create blurry images, we apply Gaussian filters with different kernel sizes to the input images.

Metric \ Resolution	256×256	128×128	64×64
APCER	1.27	2.27	5.24
BPCER	1.73	3.36	5.30
ACER	1.50	3.07	5.27

Metric \ Blurriness	1×1	3×3	5×5	7×7	9×9
APCER	1.27	2.29	3.12	3.95	4.79
BPCER	1.73	2.50	3.33	4.16	5.00
ACER	1.50	2.39	3.22	4.06	4.89

Table 6.4: The intra testing results on 4 protocols of Oulu-NPU.

Protocol	Method	APCER (%)	BPCER (%)	ACER (%)
1	CPqD [15]	2.9	10.8	6.9
	GRADIANT [15]	1.3	12.5	6.9
	Auxiliary [73]	1.6	**1.6**	1.6
	Ours	**1.2**	1.7	**1.5**
2	MixedFASNet [15]	9.7	2.5	6.1
	Ours	4.2	4.4	4.3
	Auxiliary [73]	2.7	2.7	2.7
	GRADIANT	**3.1**	**1.9**	**2.5**
3	MixedFASNet	5.3 ± 6.7	7.8 ± 5.5	6.5 ± 4.6
	GRADIANT	$\mathbf{2.6 \pm 3.9}$	5.0 ± 5.3	3.8 ± 2.4
	Ours	4.0 ± 1.8	3.8 ± 1.2	3.6 ± 1.6
	Auxiliary [73]	2.7 ± 1.3	$\mathbf{3.1 \pm 1.7}$	$\mathbf{2.9 \pm 1.5}$
4	Massy_HNU [15]	35.8 ± 35.3	8.3 ± 4.1	22.1 ± 17.6
	GRADIANT	$\mathbf{5.0 \pm 4.5}$	15.0 ± 7.1	10.0 ± 5.0
	Auxiliary [73]	9.3 ± 5.6	10.4 ± 6.0	9.5 ± 6.0
	Ours	5.1 ± 6.3	$\mathbf{6.1 \pm 5.1}$	$\mathbf{5.6 \pm 5.7}$

ity is critical for achieving a high accuracy. The spoof noise pattern may not be detected in the low-resolution or motion-blurred images. The testing results on different image resolutions and blurriness are shown in Tab. 6.3. These results validate that the spoof noise pattern is less discriminative for the lower-resolution or blurry images, as the high-frequency part of the input images contains most of the spoof noise pattern.

6.3.3 Experimental Comparison

To show the performance of our proposed method, we present our accuracy in the intra testing of Oulu-NPU and the cross testing on CASIA and Replay-Attack.

6.3.3.1 Intra Testing

We compare our intra testing performance on all 4 protocols of Oulu-NPU. Table 6.4 shows the comparison of our method and the best 3 out of 18 previous methods [15, 73]. Our proposed method achieves promising results on all protocols. Specifically, we outperform the previous state of the art by a large margin in Protocol 4, which is the most challenging protocol, and similar to cross testing.

6.3.3.2 Cross Testing

We perform cross testing between CASIA-MFSD [143] and Replay-Attack [28]. As shown in Tab. 6.5, our method achieves the competitive performance on the cross testing from CASIA-MFSD to Replay-Attack. However, we achieve a worse HTER compared to the best performing methods from Replay Attack to CASIA-MFSD. We hypothesize the reason is that images of CASIA-MFSD are of much higher resolution than those of Replay Attack. This shows that the model trained with higher-resolution data can generalize well on lower-resolution testing data, but not the other way around. This is one limitation of the proposed method, and worthy further research.

6.3.4 Qualitative Experiments

6.3.4.1 Spoof medium classification

The estimated spoof noise pattern of the test images can be used for clustering them into different groups and each group represents one spoof medium. To visualize the results, we use t-SNE [76] for dimension reduction. The t-SNE projects the noise $\mathbf{N} \in \mathbb{R}^{256\times256\times6}$ to 2 dimensions by best preserving the KL divergence distance. Fig. 6.4 shows the distributions of the testing videos on

Table 6.5: The HTER of different methods for the cross testing between the CASIA-MFSD and the Replay-Attack databases. We mark the top-2 performances in bold.

Method	Train	Test	Train	Test
	CASIA MFSD	Replay Attack	Replay Attack	CASIA MFSD
Motion [34]	50.2%		47.9%	
LBP-TOP [34]	49.7%		60.6%	
Motion-Mag [11]	50.1%		47.0%	
Spectral cubes [90]	34.4%		50.0%	
CNN [130]	48.5%		45.5%	
LBP [16]	47.0%		39.6%	
Colour Texture [17]	30.3%		**37.7**%	
Auxiliary [73]	**27.6**%		**28.4**%	
Ours	**28.5**%		41.1%	

Oulu-NPU Protocol 1. The left image shows that the noise of live is well-clustered, and the noise of spoof is subject dependent, which is consistent with our noise assumption. To obtain a better visualization, we utilize the high pass filter to extract the high-frequency information of noise pattern for dimension reduction. The right image shows that the high frequency part has more subject independent information about the spoof type and can be utilized for classification of the spoof medium.

To further show the discriminative power of the estimated spoof noise, we divide the testing set of Protocol 1 to training and testing parts and train an SVM classifier for spoof medium classification. We train two models, a three-class classifier (live, print and display) and a five-class classifier (live, print1, print2, display1 and display2), and they achieve the classification accuracy of 82.0% and 54.3% respectively, shown in Tab. 6.6. Most classification errors of the five-class model are within the same spoof medium. This result is noteworthy given that no label of spoof medium type is provided during the learning of the spoof noise model. Yet the estimated noise actually carries appreciable information regarding the medium type; hence we can observe reasonable results of spoof medium classification. This demonstrates that the estimated noise contains spoof medium

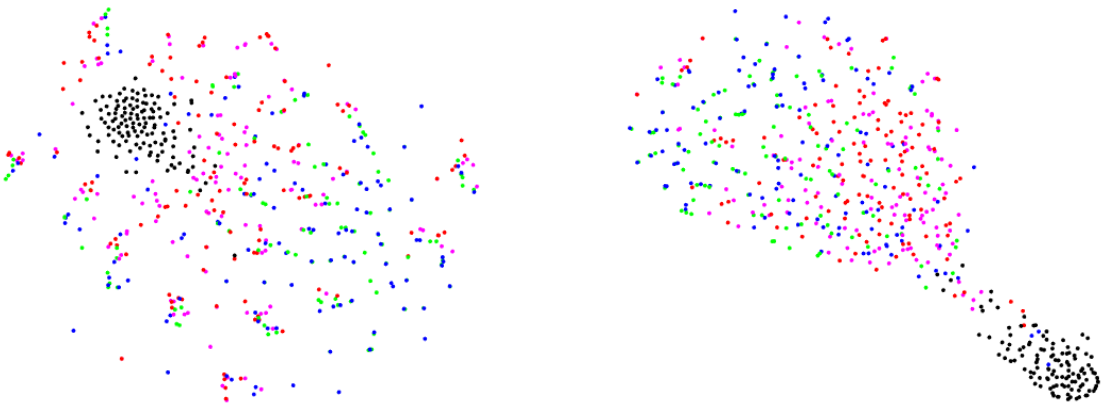


Figure 6.4: The 2D visualization of the estimated spoof noise for test videos on Oulu-NPU Protocol 1. Left: the estimated noise, Right: the high-frequency band of the estimated noise, *Color code* used: *black*=live, *green*=printer1, *blue*=printer2, *magenta*=display1, *red*=display2.

Table 6.6: The confusion matrices of spoof mediums classification based on spoof noise pattern.

Actual \ Predicted	live	print	display
live	59	1	0
print	0	88	32
display	13	8	99

Actual \ Predicted	live	print1	print2	display1	display2
live	59	0	1	0	0
print1	0	41	2	11	6
print2	0	34	11	9	6
display1	10	6	0	13	31
display2	8	7	0	6	39

information and indeed we are moving toward estimating the faithful spoof noise residing in each spoof image. In the future, if the performance of spoof medium classification improves, this could bring new impact to applications such as forensic.

6.3.4.2 Successful and failure cases

We show several success and failure cases in Fig. 6.5-6.6. Fig. 6.5 shows that the estimated spoof noises are similar within each medium but different from the other mediums. We suspect that the yellowish color in the first four columns is due to the stronger color distortion in the paper attack. The fifth row shows that the estimated noise for the live images is nearly zero. For the failure cases, we only have a few false positive cases. The failures are due to undesired noise estimation which will motivate us for further research.

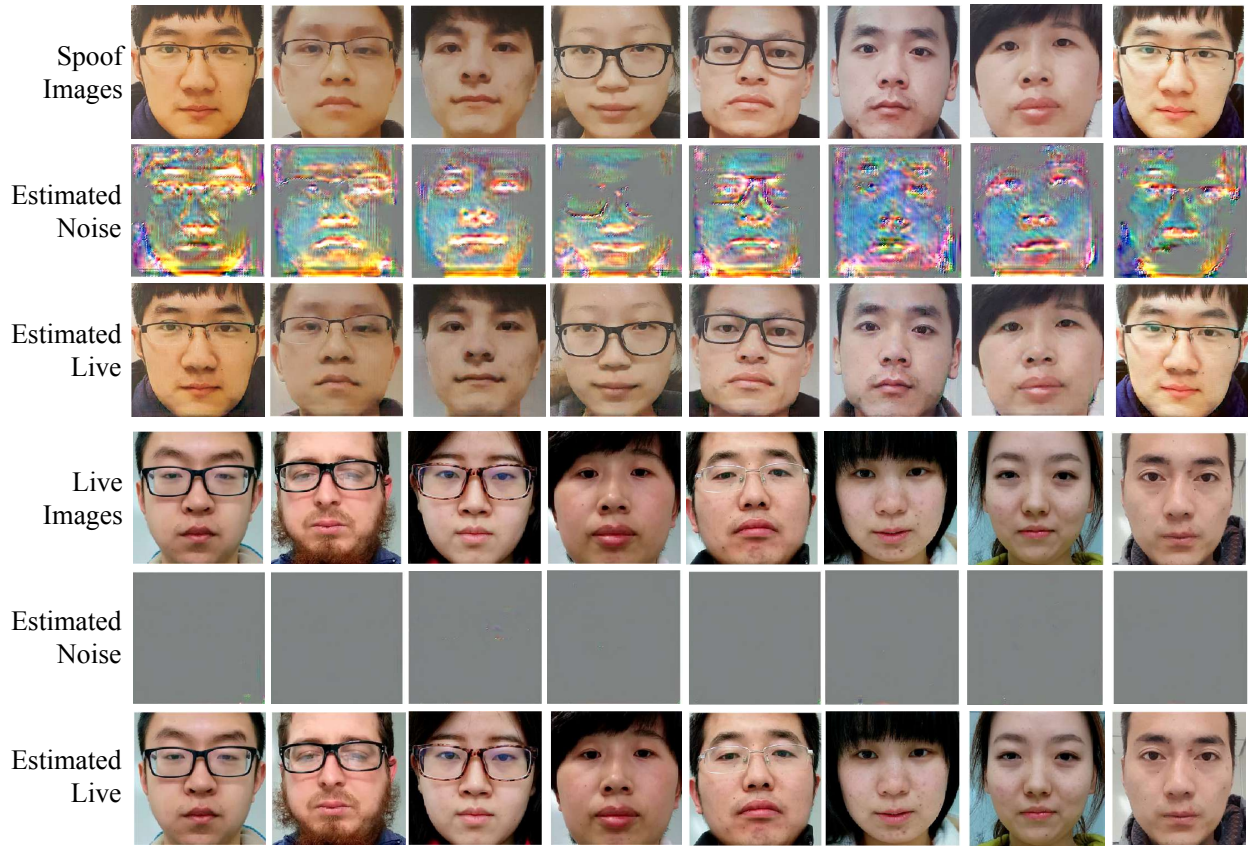


Figure 6.5: The visualization of input images, estimated spoof noises and estimated live images for test videos of Protocol 1 of Oulu-NPU database. The first four columns in the first row are paper attacks and the second four are the replay attacks. For a better visualization, we magnify the noise by 5 times and add the value with 128, to show both positive and negative noise.

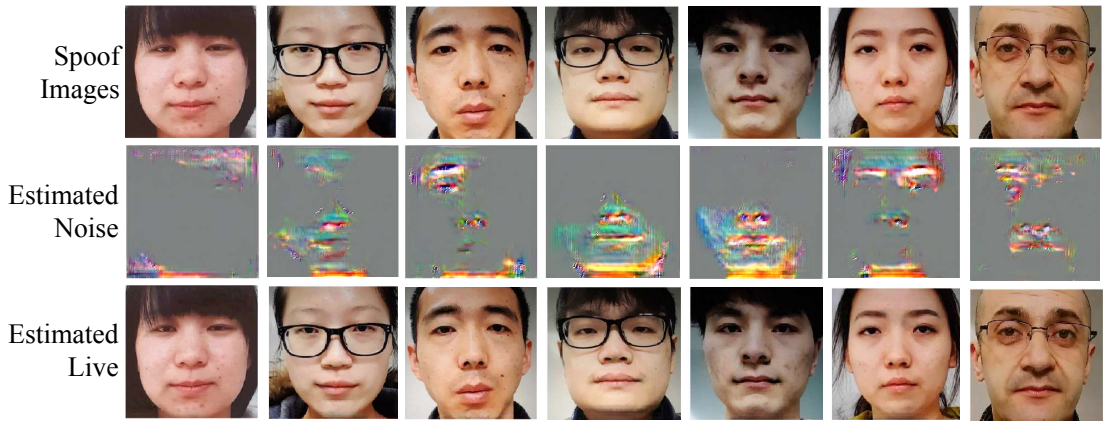


Figure 6.6: The failure cases for converting the spoof images to the live ones.

6.4 Summary

This chapter introduces a new perspective for solving the face anti-spoofing by inversely decomposing a spoof face into the live face and the spoof noise pattern. A novel CNN architecture with

multiple appropriate supervisions is proposed. We design loss functions to encourage the pattern of the spoof images to be ubiquitous and repetitive, while the noise of the live images should be zero. We visualize the spoof noise pattern which can help to have a deeper understanding of the added noise by each spoof medium. We evaluate the proposed method on multiple widely-used face anti-spoofing databases.

Chapter 7

Conclusions and Future Work

Face alignment is an important research topic because it has many applications in face recognition, face tracking, expression estimation, augmented and virtual reality, etc. By utilizing deep learning methods, face alignment systems improved dramatically and they can be used in many commercial applications for face tracking and augmented reality tasks e.g., Snapchat and Facebook.

In chapter 2 to 4, we propose three face alignment methods with the same idea of converting the 2D face alignment to 3D face alignment and reconstructing the 3D shape of the face. We present our advantage to perform face alignment for large pose face images by utilizing the estimated 3D shape. These three methods are an extended version of each other, and the accuracy and the speed of our pose-invariant face alignment are improved in each method.

The face is one of the most popular biometric modalities that can be used for access control and authentication. Although face recognition systems can achieve very high identification accuracy, if we want to use face recognition systems in practice we need to have a robust face anti-spoofing system

In chapter 5 and 6, we propose our CNN-based face anti-spoofing methods that use the supervisions from both the spatial and temporal auxiliary information for the purpose of robustly detecting face PA from a face video.

7.1 Limitations

Pose-invariant Face Alignment: The main limitation of the proposed PIFA methods is that the estimated 3D shape of face does not contain the detailed changes in the 3D shape e.g., wrinkles. This is due to the limited numbers of ground truth 2D landmarks which are utilized during training as supervision. Tran *et. al* [106] show that utilizing unsupervised loss functions based on the texture reconstruction is helpful to have more detailed 3D shape estimation.

Face Anti-spoofing: The main challenge of anti-spoofing methods for all of the Biometric modalities is unknown spoof attack (spoof medium). The anti-spoofing methods are mainly constrained with the spoof medium data which used for training them. Making the anti-spoofing methods more generalizable and robust to new spoof medium is the main limitation of the proposed methods.

7.2 Future Work

Detailed 3D Shape for Face The estimated 3D shapes, by our proposed methods, are limited to the 3DMM bases which we utilize to represent the 3D shape. In order to have more powerful bases to represent the 3D shapes, we can learn the bases from the data and make a more detailed reconstruction by incorporating unsupervised loss functions. In [106], a CNN based method proposed to learn a set of non-linear bases for representing the 3D shape of the face. Similarly, Tan *et. al* [103] utilize variational auto-encoders for representing the 3D meshes in order to have more flexibility for representing the 3D shape and allowing non-linear deformations.

General Object Anti-Spoofing: The idea of face anti-spoofing can be extended for detecting the spoof image for all objects. The general object anti-spoofing has applications for detecting the spoof images in online shopping websites e.g., eBay and Amazon. The general anti-spoofing is much more challenging than the face anti-spoof due to various materials and different texture of

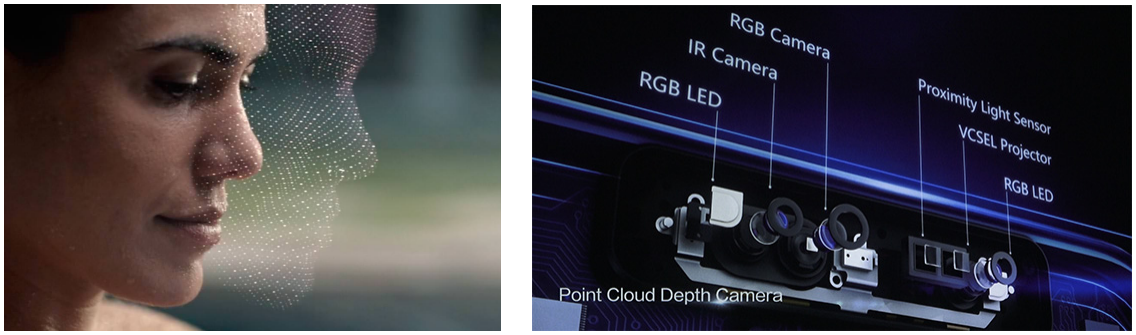


Figure 7.1: Left: A representation of the estimated point cloud in iPhone X. Right: The hardware technology in Huawei P11 for capturing the point cloud.

objects. The goal is to detect the print attack and the replay attack for general object anti-spoofing.

3**D Point Cloud Classification:** The 3D point cloud classification has several applications in computer vision. Recently, with the advancement of hardware technology, we can have sensors on the phone for capturing the 3D point cloud. In iPhone X, the point cloud data is utilized for face anti-spoofing (Left figure in Fig. 7.1). Similarly, other cell phone companies are utilizing point cloud for face anti-spoofing. For example, the right figure in Fig. 7.1 shows the hardware technology in Huawei P11 for capturing the point cloud. These data can be utilized for detecting objects and measuring the length of objects and the distance to the objects.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Explainable Artificial Intelligence (XAI) (https://www.darpa.mil/program/explainable-artificial-intelligence).

[2] M. Abadi, A. Agarwal, and et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[3] A. Agarwal, R. Singh, and M. Vatsa. Face anti-spoofing using Haralick features. IEEE, 2016.

[4] A. Asthana, S. Zafeiriou, S. Cheng, and M. Pantic. Robust discriminative response map fitting with constrained local models. pages 3444–3451. IEEE, 2013.

[5] Y. Atoum, Y. Liu, A. Jourabloo, and X. Liu. Face anti-spoofing using patch and depth-based cnns. IEEE, 2017.

[6] W. Bao, H. Li, N. Li, and W. Jiang. A liveness detection method for face recognition based on optical flow field. In *IASP*. IEEE, 2009.

[7] P. N. Belhumeur, D. W. Jacobs, D. Kriegman, and N. Kumar. Localizing parts of faces using a consensus of exemplars. pages 545–552. IEEE, 2011.

[8] S. Bell and K. Bala. Learning visual similarity for product design with convolutional neural networks. 34(4):98, 2015.

[9] S. Bengio and J. Mariéthoz. A statistical significance test for person authentication. In *Proceedings of Odyssey 2004: The Speaker and Language Recognition Workshop*, 2004.

[10] S. Bharadwaj, T. Dhamecha, M. Vatsa, and R. Singh. Face anti-spoofing via motion magnification and multifeature videolet aggregation. 2014.

[11] S. Bharadwaj, T. I. Dhamecha, M. Vatsa, and R. Singh. Computationally efficient face spoofing detection with motion magnification. IEEE, 2013.

[12] V. Blanz and T. Vetter. A morphable model for the synthesis of 3D faces. In *ACM SIGGRAPH*, pages 187–194, 1999.

[13] V. Blanz and T. Vetter. Face recognition based on fitting a 3D morphable model. 25(9):1063–1074, 2003.

[14] S. Bobbia, Y. Benezeth, and J. Dubois. Remote photoplethysmography based on implicit living skin tissue segmentation. pages 361–365, 2016.

[15] Z. Boulkenafet. A competition on generalized software-based face presentation attack detection in mobile scenarios. IEEE, 2017.

[16] Z. Boulkenafet, J. Komulainen, and A. Hadid. Face anti-spoofing based on color texture analysis. IEEE, 2015.

[17] Z. Boulkenafet, J. Komulainen, and A. Hadid. Face spoofing detection using colour texture analysis. 11(8):1818–1830, 2016.

[18] Z. Boulkenafet, J. Komulainen, and A. Hadid. Face antispoofing using speeded-up robust features and fisher vector encoding. *IEEE Signal Processing Letters*, 24(2):141–145, 2017.

[19] Z. Boulkenafet, J. Komulainen, L. Li, X. Feng, and A. Hadid. OULU-NPU: A mobile face presentation attack database with real-world variations. IEEE, 2017.

[20] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah. Signature verification using a "siamese" time delay neural network. 7(04):669–688, 1993.

[21] A. Bulat and G. Tzimiropoulos. Convolutional aggregation of local evidence for large pose face alignment. 2016.

[22] X. P. Burgos-Artizzu, P. Perona, and P. Dollár. Robust face landmark estimation under occlusion. pages 1513–1520, 2013.

[23] H. Caesar, J. Uijlings, and V. Ferrari. Region-based semantic segmentation with end-to-end training. pages 381–397, 2016.

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

[25] C. Cao, Y. Weng, S. Zhou, Y. Tong, and K. Zhou. Facewarehouse: a 3D facial expression database for visual computing. 20(3):413–425, 2014.

[26] X. Cao, Y. Wei, F. Wen, and J. Sun. Face alignment by explicit shape regression. 107(2):177–190, 2014.

[27] G. Chetty and M. Wagner. Multi-level liveness verification for face-voice biometric authentication. In *BC*, 2006.

[28] I. Chingovska, A. Anjos, and S. Marcel. On the effectiveness of local binary patterns in face anti-spoofing. IEEE, 2012.

[29] T. Cootes, G. Edwards, and C. Taylor. Active appearance models. 23(6):681–685, June 2001.

[30] T. Cootes, C. Taylor, and A. Lanitis. Active shape models: Evaluation of a multi-resolution method for improving image search. volume 1, pages 327–336, 1994.

[31] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models — their training and application. 61(1):38–59, Jan 1995.

[32] D. Cristinacce and T. Cootes. Boosted regression active shape models. volume 2, pages 880–889, 2007.

[33] T. de Freitas Pereira, A. Anjos, J. M. De Martino, and S. Marcel. LBP-TOP based countermeasure against face spoofing attacks. Springer, 2012.

[34] T. de Freitas Pereira, A. Anjos, J. M. De Martino, and S. Marcel. Can face anti-spoofing countermeasures work in a real world scenario? IEEE, 2013.

[35] G. de Haan and V. Jeanne. Robust pulse rate from chrominance-based rPPG. *IEEE Trans. Biomedical Engineering*, 60(10):2878–2886, 2013.

[36] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. Springer, 2014.

[37] L. Feng, L.-M. Po, Y. Li, X. Xu, F. Yuan, T. C.-H. Cheung, and K.-W. Cheung. Integration of image quality and motion cues for face anti-spoofing: a neural network approach. *Journal of Visual Communication and Image Representation*, 38:451–460, 2016.

[38] R. W. Frischholz and U. Dieckmann. BiolD: a multimodal biometric identification system. *J. Computer*, 33(2):64–68, 2000.

[39] R. W. Frischholz and A. Werner. Avoiding replay-attacks in a face recognition system using head-pose estimation. In *AMFGW*, pages 234–235, 2003.

[40] M. Gharbi, G. Chaurasia, S. Paris, and F. Durand. Deep joint demosaicking and denoising. 35(6):191, 2016.

[41] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proc. Artificial Intelligence and Statistics (AISTATS)*, pages 315–323, 2011.

[42] R. Gross, I. Matthews, and S. Baker. Generic vs. person specific active appearance models. 23(11):1080–1093, Nov. 2005.

[43] L. Gu and T. Kanade. 3D alignment of face in a single image. volume 1, pages 1305–1312, 2006.

[44] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. IEEE, 2016.

[45] G.-S. Hsu, K.-H. Chang, and S.-C. Huang. Regressive tree structured model for facial landmark localization. pages 3855–3861, 2015.

[46] ISO/IEC JTC 1/SC 37 Biometrics. Information technology biometric presentation attack detection part 1: Framework. international organization for standardization (https://www.iso.org/obp/ui/iso), 2016.

[47] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. pages 2017–2025, 2015.

[48] L. A. Jeni, J. F. Cohn, and T. Kanade. Dense 3D face alignment from 2d videos in real-time. volume 1, pages 1–8, 2015.

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

[50] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. Springer, 2016.

[51] A. Jourabloo, A. Feghahati, and M. Jamzad. New algorithms for recovering highly corrupted images with impulse noise. *Scientia Iranica*, 19(6):1738–1745, 2012.

[52] A. Jourabloo and X. Liu. Pose-invariant 3D face alignment. pages 3694–3702, 2015.

[53] A. Jourabloo and X. Liu. Large-pose face alignment via cnn-based dense 3D model fitting. 2016.

[54] A. Jourabloo and X. Liu. Pose-invariant face alignment via cnn-based dense 3D model fitting. pages 1–17, 2017.

[55] A. Jourabloo, X. Yin, and X. Liu. Attribute preserved face de-identification. 2015.

[56] V. Kazemi and J. Sullivan. One millisecond face alignment with an ensemble of regression trees. pages 1867–1874, 2014.

[57] M. Koestinger, P. Wohlhart, P. M. Roth, and H. Bischof. Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization. In *First IEEE International Workshop on Benchmarking Facial Image Analysis Technologies*, 2011.

[58] K. Kollreider, H. Fronthaler, M. I. Faraj, and J. Bigun. Real-time face detection and motion analysis with application in liveness assessment. 2(3):548–558, 2007.

[59] J. Komulainen, A. Hadid, and M. Pietikainen. Context based face anti-spoofing. IEEE, 2013.

[60] J. Komulainen, A. Hadid, M. Pietikäinen, A. Anjos, and S. Marcel. Complementary countermeasures for detecting scenic face spoofing attacks. 2013.

[61] M. Köstinger, P. Wohlhart, P. M. Roth, and H. Bischof. Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization. pages 2144–2151, 2011.

[62] K. Kulkarni, S. Lohit, P. Turaga, R. Kerviche, and A. Ashok. Reconnet: Non-iterative reconstruction of images from compressively sensed measurements. IEEE, 2016.

[63] S. Lefkimmiatis. Non-local color image denoising with convolutional neural networks. IEEE, 2017.

[64] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. pages 5325–5334, 2015.

[65] J. Li, Y. Wang, T. Tan, and A. K. Jain. Live face detection based on the analysis of fourier spectra. In *Biometric Technology for Human Identification*. SPIE, 2004.

[66] L. Li, X. Feng, Z. Boulkenafet, Z. Xia, M. Li, and A. Hadid. An original face anti-spoofing approach using partial convolutional neural network. In *Image Processing Theory Tools and Applications (IPTA), 2016 6th International Conference on*. IEEE, 2016.

[67] Y. Li, B. Sun, T. Wu, Y. Wang, and W. Gao. Face detection with end-to-end integration of a convnet and a 3D model. 2016.

[68] F. Liu, D. Zeng, Q. Zhao, and X. Liu. Joint face alignment and 3D face reconstruction. pages 545–560, 2016.

[69] S. Liu, P. C. Yuen, S. Zhang, and G. Zhao. 3D mask face anti-spoofing with remote photoplethysmography. pages 85–100, 2016.

[70] X. Liu. Discriminative face alignment. 31(11):1941–1954, 2009.

[71] X. Liu and T. Chen. Pose-robust face recognition using geometry assisted probabilistic modeling. volume 1, pages 502–509, 2005.

[72] X. Liu, J. Rittscher, and T. Chen. Optimal pose for face recognition. volume 2, pages 1439–1446, 2006.

[73] Y. Liu, A. Jourabloo, and X. Liu. Learning deep models for face anti-spoofing: Binary or auxiliary supervision. IEEE, 2018.

[74] Y. Liu, A. Jourabloo, W. Ren, and X. Liu. Dense face alignment. IEEE, 2017.

[75] S. Lucey, R. Navarathna, A. B. Ashraf, and S. Sridharan. Fourier Lucas-Kanade algorithm.

35(6):1383–1396, 2013.

[76] L. v. d. Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[77] J. Määttä, A. Hadid, and M. Pietikäinen. Face spoofing detection from single images using micro-texture analysis. IEEE, 2011.

[78] I. Matthews and S. Baker. Active appearance models revisited. 60(2):135–164, 2004.

[79] H. Mohammadzade and D. Hatzinakos. Iterative closest normal point for 3D face recognition. 35(2):381–397, 2013.

[80] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. pages 483–499, 2016.

[81] E. M. Nowara, A. Sabharwal, and A. Veeraraghavan. Ppgsecure: Biometric presentation attack detection using photopletysmograms. pages 56–62, 2017.

[82] G. Pan, L. Sun, Z. Wu, and S. Lao. Eyeblink-based anti-spoofing in face recognition from a generic webcamera. IEEE, 2007.

[83] K. Patel, H. Han, and A. K. Jain. Cross-database face antispoofing with robust feature representation. In *Chinese Conference on Biometric Recognition*. Springer, 2016.

[84] K. Patel, H. Han, and A. K. Jain. Secure face unlock: Spoof detection on smartphones. 11(10):2268–2283, 2016.

[85] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. IEEE, 2016.

[86] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter. A 3D face model for pose and illumination invariant face recognition. pages 296–301, 2009.

[87] X. Peng, R. S. Feris, X. Wang, and D. N. Metaxas. A recurrent encoder-decoder network for sequential face alignment. pages 38–56. Springer, 2016.

[88] T. Pfister, K. Simonyan, J. Charles, and A. Zisserman. Deep convolutional neural networks for efficient pose estimation in gesture videos. pages 538–552, 2015.

[89] P. J. Phillips, H. Moon, S. Rizvi, P. J. Rauss, et al. The FERET evaluation methodology for face-recognition algorithms. 22(10):1090–1104, 2000.

[90] A. Pinto, H. Pedrini, W. R. Schwartz, and A. Rocha. Face spoofing detection through visual codebooks of spectral temporal cubes. 24(12):4726–4740, 2015.

[91] L.-M. Po, L. Feng, Y. Li, X. Xu, T. C.-H. Cheung, and K.-W. Cheung. Block-based adaptive ROI for remote photoplethysmography. *J. Multimedia Tools and Applications*, pages 1–27, 2017.

[92] C. Qu12, E. Monari, T. Schuchert, and J. Beyerer21. Adaptive contour fitting for pose-invariant 3d face shape reconstruction. 2015.

[93] S. Ren, X. Cao, Y. Wei, and J. Sun. Face alignment at 3000 fps via regressing local binary features. pages 1685–1692, 2014.

[94] J. Roth, Y. Tong, and X. Liu. Unconstrained 3D face reconstruction. pages 2606–2615, 2015.

[95] J. Roth, Y. Tong, and X. Liu. Adaptive 3D face reconstruction from unconstrained photo collections. 2016.

[96] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic. 300 faces in-the-wild challenge: The first facial landmark localization challenge. pages 397–403, 2013.

[97] E. Sánchez-Lozano, F. De la Torre, and D. González-Jiménez. Continuous regression for non-rigid image alignment. pages 250–263. Springer, 2012.

[98] R. Shao, X. Lan, and P. C. Yuen. Deep convolutional dynamic texture learning with adaptive channel-discriminability for 3D mask face anti-spoofing. In *IJCB*, 2017.

[99] Y. Sun, X. Wang, and X. Tang. Deep convolutional network cascade for facial point detection. pages 3476–3483, 2013.

[100] Y. Tai, J. Yang, and X. Liu. Image super-resolution via deep recursive residual network. IEEE, 2017.

[101] Y. Tai, J. Yang, X. Liu, and C. Xu. Memnet: A persistent memory network for image restoration. IEEE, 2017.

[102] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. pages 1701–1708, 2014.

[103] Q. Tan, L. Gao, Y.-K. Lai, and S. Xia. Variational autoencoders for deforming 3d mesh models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5841–5850, 2018.

[104] X. Tan, Y. Li, J. Liu, and L. Jiang. Face liveness detection from a single image with sparse low rank bilinear discriminative model. pages 504–517, 2010.

[105] Y. Tong, X. Liu, F. W. Wheeler, and P. Tu. Automatic facial landmark labeling with minimal

supervision. 2009.

[106] L. Tran and X. Liu. Nonlinear 3d face morphable model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7346–7355, 2018.

[107] G. Trigeorgis, P. Snape, M. A. Nicolaou, E. Antonakos, and S. Zafeiriou. Mnemonic descent method: A recurrent process applied for end-to-end face alignment. 2016.

[108] S. Tulyakov, X. Alameda-Pineda, E. Ricci, L. Yin, J. F. Cohn, and N. Sebe. Self-adaptive matrix completion for heart rate estimation from face videos under realistic conditions. pages 2396–2404, 2016.

[109] S. Tulyakov and N. Sebe. Regressing a 3D face shape from a single image. pages 3748–3755, 2015.

[110] G. Tzimiropoulos and M. Pantic. Optimization problems for fast AAM fitting in-the-wild. pages 593–600. IEEE, 2013.

[111] M. Valstar, B. Martinez, X. Binefa, and M. Pantic. Facial point detection using boosted regression and graph models. pages 2729–2736. IEEE, 2010.

[112] L. Van der Maaten and G. Hinton. Visualizing data using t-SNE. 9(2579-2605), 2008.

[113] A. Vedaldi and K. Lenc. MatConvNet – convolutional neural networks for matlab. In *ACM MM, (2015)*, pages 689–692, 2015.

[114] A. Wagner, J. Wright, A. Ganesh, Z. Zhou, H. Mobahi, and Y. Ma. Toward a practical face recognition system: Robust alignment and illumination by sparse representation. 34(2):372–386, 2012.

[115] N. Wang, X. Gao, D. Tao, and X. Li. Facial feature point detection: A comprehensive survey. *arXiv preprint arXiv:1410.1037*, 2014.

[116] W. Wang, S. Tulyakov, and N. Sebe. Recurrent convolutional face alignment. 2016.

[117] D. Wen, H. Han, and A. Jain. Face Spoof Detection with Image Distortion Analysis. 10(4):746–761, 2015.

[118] B.-F. Wu, Y.-W. Chu, P.-W. Huang, M.-L. Chung, and T.-M. Lin. A motion robust remote-PPG approach to driver's health state monitoring. pages 463–476, 2016.

[119] J. Wu, T. Xue, J. J. Lim, Y. Tian, J. B. Tenenbaum, A. Torralba, and W. T. Freeman. Single image 3D interpreter network. 2016.

[120] Y. Wu and Q. Ji. Robust facial landmark detection under significant head poses and occlusion. pages 3658–3666, 2015.

[121] J. Xiao, S. Baker, I. Matthews, and T. Kanade. Real-time combined 2D+3D active appearance models. volume 2, pages 535–542, 2004.

[122] S. Xiao, J. Feng, J. Xing, H. Lai, S. Yan, and A. Kassim. Robust facial landmark detection via recurrent attentive-refinement networks. pages 57–72, 2016.

[123] J. Xing, Z. Niu, J. Huang, W. Hu, and S. Yan. Towards multi-view and partially-occluded face alignment. pages 1829–1836. IEEE, 2014.

[124] X. Xiong and F. De la Torre. Supervised descent method and its applications to face alignment. pages 532–539, 2013.

[125] Z. Xu, S. Li, and W. Deng. Learning temporal features using LSTM-CNN architecture for face anti-spoofing. In *IAPR Asian Conference*. IEEE, 2015.

[126] J. Yan, Z. Lei, D. Yi, and S. Z. Li. Learn to combine multiple hypotheses for accurate face alignment. pages 392–396. IEEE, 2013.

[127] Z. Yan, H. Zhang, R. Piramuthu, V. Jagadeesh, D. DeCoste, W. Di, and Y. Yu. Hd-cnn: hierarchical deep convolutional neural networks for large scale visual recognition. pages 2740–2748, 2015.

[128] B. Yang, J. Yan, Z. Lei, and S. Z. Li. Convolutional channel features. pages 82–90, 2015.

[129] H. Yang and I. Patras. Mirror, mirror on the wall, tell me, is the error small? pages 4685–4693, 2015.

[130] J. Yang, Z. Lei, and S. Z. Li. Learn convolutional neural network for face anti-spoofing. *arXiv preprint arXiv:1408.5601*, 2014.

[131] J. Yang, Z. Lei, S. Liao, and S. Z. Li. Face liveness detection with component dependent descriptor. IEEE, 2013.

[132] J. Yang, S. E. Reed, M.-H. Yang, and H. Lee. Weakly-supervised disentangling with recurrent transformations for 3D view synthesis. pages 1099–1107, 2015.

[133] L. Yin, X. Chen, Y. Sun, T. Worm, and M. Reale. A high-resolution 3D dynamic facial expression database. 2008.

[134] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? pages 3320–3328, 2014.

[135] X. Yu, J. Huang, S. Zhang, W. Yan, and D. N. Metaxas. Pose-free facial landmark fitting via optimized part mixtures and cascaded deformable shape model. pages 1944–1951, 2013.

[136] X. Yu, Z. Lin, J. Brandt, and D. N. Metaxas. Consensus of regression for occlusion-robust

facial feature localization. pages 105–118, 2014.

[137] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. pages 4353–4361, 2015.

[138] C. Zhang and Z. Zhang. A survey of recent advances in face detection. Technical report, Tech. rep., Microsoft Research, 2010.

[139] J. Zhang, S. Shan, M. Kan, and X. Chen. Coarse-to-fine auto-encoder networks (cfan) for real-time face alignment. pages 1–16. 2014.

[140] J. Zhang, S. Zhou, D. Comaniciu, and L. McMillan. Conditional density learning via regression with application to deformable shape segmentation. 2008.

[141] X. Zhang, L. Yin, J. F. Cohn, S. Canavan, M. Reale, A. Horowitz, P. Liu, and J. M. Girard. BP4D-spontaneous: a high-resolution spontaneous 3D dynamic facial expression database. 32(10):692 – 706, 2014.

[142] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Facial landmark detection by deep multi-task learning. pages 94–108. 2014.

[143] Z. Zhang, J. Yan, S. Liu, Z. Lei, D. Yi, and S. Z. Li. A face antispoofing database with diverse attacks. IEEE, 2012.

[144] E. Zhou, H. Fan, Z. Cao, Y. Jiang, and Q. Yin. Extensive facial landmark localization with coarse-to-fine convolutional network cascade. pages 386–391, 2013.

[145] R. Zhou, R. Achanta, and S. Süsstrunk. Deep residual network for joint demosaicing and super-resolution. *arXiv preprint arXiv:1802.06573*, 2018.

[146] S. Zhou and D. Comaniciu. Shape regression machine. pages 13–25, 2007.

[147] S. Zhu, C. Li, C. Change Loy, and X. Tang. Face alignment by coarse-to-fine shape searching. pages 4998–5006, 2015.

[148] S. Zhu, C. Li, C. C. Loy, and X. Tang. Unconstrained face alignment via cascaded compositional learning. 2016.

[149] X. Zhu, Z. Lei, X. Liu, H. Shi, and S. Z. Li. Face alignment across large poses: A 3D solution. pages 146–155, 2016.

[150] X. Zhu, Z. Lei, J. Yan, D. Yi, and S. Z. Li. High-fidelity pose and expression normalization for face recognition in the wild. pages 787–796, 2015.

[151] X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. pages 2879–2886, 2012.

[152] X. Zhu, J. Yan, D. Yi, Z. Lei, and S. Z. Li. Discriminative 3D morphable model fitting. pages 1–8, 2015.