Robust Pose Invariant Face Recognition Using 3D Thin Plate Spline Spatial Transformer Networks

Submitted in partial fulfillment of the requirements for
the degree of
Doctor of Philosophy
in the
Department of Electrical and Computer Engineering

Chandrasekhar Bhagavatula

B.S., Computer Science, Carnegie Mellon University
M.S., Electrical & Computer Engineering, Carnegie Mellon University

Carnegie Mellon University
Pittsburgh, PA

May, 2018

Acknowledgments

As my PhD comes to a close, I want to thank all of those who have made my research possible and my time enjoyable. First and foremost is my advisor, Prof. Marios Savvides, for the many years of advice and guidance throughout my academic career. From my time as an intern in the lab, as a PhD student, and as a teaching assistant, I have gained immeasurable experience that will no doubt help guide me through the rest of my career path. I also wish to thank my doctoral committee members, Prof. John Dolan, Dr. Nalini Ratha, and Dr. Saad Bedros for taking the time to be a part of the completion of my PhD.

All the members of the CyLab Biometrics Center throughout the years that I have been there deserve thanks for all the help they have provided along the way: Utsav, Keshav, Shreyas, Ramzi, Aaron, Khoa, Nancy, Karanhaar, Raied, and all the masters and undergraduate students along the way. Of particular note to this work, I would like to thank Chenchecn Zhu for his great help in the development of the 3D Thin Plate Spline Spatial Transformer Networks that form the backbone of this thesis. I would also like to thank Dipan Pal, Yutong Zheng, Ran Tao, and Felix Juefei-Xu for their help in training the various face recognition models used to show the effectiveness of our proposed methods. I am sure all of you will go on to do even more impressive research than you have already done and I wish you all the best.

I also must thank the CyLab staff who have always helped us with any problems or last minute requests (of which there were quite a few). So thank you Tina, Michael, Kelley, Brittney, Chelsea, Brigitte, Ivan, and anyone else who I may have forgotten.

My family has always been a source of strength and comfort for me over the years and I feel exceptionally lucky that my parents remained always within

reach, sometimes literally, over my education. My dad has always been like a second advisor during my PhD work, providing guidance whenever I asked for it and my mom has always been a great example of how to stay calm under pressure. I don't think I could have managed the stress of the PhD student's life without them. To both my brothers, Ramu and Madhu, whom have also worked in the same group at some point throughout their careers, I want to say thanks for all the support and I am glad we are all finally finished with our CMU education.

The person I have to thank the most, however, is my loving wife, Theresa. She has stood by me through all the craziness, all the last minute trips to D.C, all the drives to Tampa, all the demos, and everything else while providing an unyielding pillar of support. Even during these last months of my work, when I was staying late in the lab every night to get all my experiments run and my dissertation written, she was handling everything at home while she had an equally busy schedule. With our moving out of our apartment and buying a new house, she took so much off my plate so that I could finish everything I needed to. There are no words to express how much I appreciate all she have done for me. I am so grateful for all of her love, patience, and support and look forward to starting the next phase of our lives together.

This work has been partly funded by Carnegie Mellon University CyLab, National Institute of Justice grant no. 2013-IJ-CX-K005, US Naval Air Systems Command (NAVAIR) grant no. N68335-16-C-0177, NAVMAR Applied Sciences Corporation grant no. NASC-0040-CMU, and Carnegie Mellon University Software Engineering Institute CERT grant no. 6-599B2.

Abstract

In recent years, face recognition has advanced with incredible speed thanks to the advent of deep learning, large scale datasets, and the improvement in GPU computing. While many of these methods claim to be able to match faces from images captured in-the-wild, they still seem to perform poorly when trying to match non-frontal faces to frontal ones which is the practical scenario faced by law enforcement everyday in the processing of criminal cases. Trying to learn these large pose variations implicitly is a very hard problem, both from a deep neural network modeling perspective and from the lack of structured datasets used in training and evaluating these models. As they are often made up of celebrity images found online, they contain a large bias in the types of images present in both the datasets used for training and evaluating new methods. Perhaps the largest bias is in the distribution of the pose of the faces. Most celebrity images are captured from a frontal or near-frontal view which have traditionally been the easiest poses for face recognition. Most importantly, as both training and evaluation datasets share this bias, this has led to artificially high results being reported.

The goal of this thesis is to design a system to be able to take advantage of the large amount of data already available and still be able to perform robust face recognition across large pose variations. We propose that the most efficient way to do this is to transform and reduce the entire pose distribution to just the frontal faces by re-rendering the off-angle faces from a frontal viewpoint. By doing this, the mismatch between the training, evaluation, and real-world multi-modal distributions on pose will be eliminated. To solve this problem we must explicitly understand and model the 3D face structure of faces since faces are not planar objects. This 3D model of the face must be able to be generated

from a single, 2D image since that is all that is usually available in a recognition scenario. This is also the hardest scenario and is often overlooked by the use of temporal fusion to perform some kind of data reconstruction. By improving performance of the models in this worst case scenario, we can always further improve by utilizing temporal information later but maintain a high accuracy on single images.

To achieve this, we first design a new method of 3D facial alignment and modeling from a single 2D image using our 3D Thin Plate Spline Spatial Transformer Networks (3DTPS-STN). We evaluate this method against several previous methods on the Annotated Facial Landmarks in the Wild (AFLW) dataset and the synthetic AFLW2000-3D dataset and show that our method achieves very high performance on these at a much faster speed. We also confirm the intuition that most recognition datasets in use have a heavy bias towards frontal faces using the implicit knowledge of the pose extracted from the 3D modeling. We then show how we can use the 3D models created by the 3DTPS-STN method to frontalize the face from any angle and, by a careful selection of the face region, generate a more stable face image across all poses. We then train a 28 layer ResNet, a common face recognition framework, on these faces and show that this model can outperform all comparable models on the CMU Multi-PIE dataset and also show a detailed analysis on other datasets.

Contents

1 Introduction **1**

1.1 The Challenges of Pose Invariant Face Recognition 2

1.1.1 Face Detection Across Pose 2

1.1.2 Facial Alignment Across Pose 3

1.1.3 Face Recognition Across Pose 7

1.2 Summary of Contributions 10

1.3 Notation and Organization 11

2 Related Works **13**

2.1 Facial Alignment and Modeling 13

2.2 Face Recognition 20

3 Pose Invariant Facial Modeling Using 3D Thin Plate Spline Spatial Transformer Networks **29**

3.1 Spatial Transformer Networks 32

3.2 3D Thin Plate Spline Spatial Transformer Networks 34

3.2.1 Thin Plate Spline Transformers 35

3.2.2 Camera Projection Transformers 39

3.3 2D Landmark Regression 42

3.4 3D Model Regression From 2D Landmarks 44

3.5 Facial Landmarking Experiments . . . 47
3.5.1 Datasets . . . 48
3.5.2 Ablation Experiments . . . 51
3.5.3 Comparison Experiments . . . 52
3.6 Pose Estimation Experiments . . . 53
3.6.1 Datasets . . . 55
3.6.2 Results . . . 57
3.7 Running Speed . . . 59

4 Pose Invariant Face Recognition **63**
4.1 Generating a Pose Invariant Face . . . 63
4.2 Reconstructing the Missing Half . . . 67
4.3 Pose Invariant Face Recognition Experiments . . . 73
4.3.1 Implementation Details . . . 73
4.3.2 MPIE - Pose Variation . . . 76
4.3.3 MPIE - Pose and Expression Variation . . . 80
4.3.4 MPIE - Pose, Illumination, and Expression Variation . . . 83
4.3.5 Experiments on CFP . . . 85

5 Concluding Remarks and Future Work **89**
5.1 Future Work . . . 90

Appendices **94**

A Properties of Camera Geometry **95**
A.1 Formulation of the Camera Projection Matrix . . . 95
A.2 Properties of the Camera Projection Matrix . . . 99
A.3 Determining Self-Occluded Vertexes . . . 102

B Thin Plate Splines **107**

List of Tables

3.1 Ablation study of network architectures on AFLW 51

3.2 Comparison of face alignment methods on AFLW 54

3.3 Comparison of face alignment methods on AFLW2000-3D 54

3.4 Pose estimation error on various datasets 59

4.1 Rank-1 recognition accuracy on the MPIE dataset with only pose variation 80

4.2 Rank-1 recognition accuracy on the MPIE dataset with pose and expression variation 83

4.3 Rank-1 recognition accuracy on the MPIE dataset with pose, illumination, and expression variation 85

4.4 Accuracy on the CFP dataset 86

List of Figures

1.1 Examples from the WIDER FACE dataset 3
1.2 Landmarking schemes 4
1.3 Illustration of landmark traversal 5
1.4 An example of semantically inconsistent landmarks 6
1.5 Pose bias in common recognition datasets 8
1.6 Performance drop across pose for deep networks 9

2.1 The 300W-LP dataset 19
2.2 Small changes in Euclidean space leads to large angular shifts 24
2.3 Pure SoftMax vs. Ring Loss + SoftMax features 26

3.1 Design of traditional Spatial Transformer Network 33
3.2 Design of the 3D Thin Plate Spline Spatial Transformer Network 35
3.3 68 point landmarking scheme 36
3.4 Pre and post regression landmarks 43
3.5 Backprojection of camera rays through image landmarks 45
3.6 Examples of regressed 2D landmarks and 3D models 46
3.7 3D model synthesized from different viewpoints 47
3.8 A few example results of our 3D landmarking 48
3.9 AFLW ground truth landmarks 49
3.10 An example of fake good alignment 50

3.11 AFLW2000-3D ground truth landmarks 51
3.12 Landmarking CED curves on AFLW and AFLW2000-3D 52
3.13 Pose estimation from the estimated camera center 55
3.14 Examples of the FacePix dataset 56
3.15 Examples of the Pointing'04 dataset 58
3.16 Comparison of the 90° pose across datasets 60
3.17 Yaw and pitch errors on the Pointing'04 dataset 61

4.1 Naive sampling artifacts for frontalization 64
4.2 Frontalization of faces with missing regions 66
4.3 Standard deviation of frontalizations 67
4.4 Half-Face frontalizations 68
4.5 GAN architecture used for reconstruction 70
4.6 GAN training data from original PCSO image 71
4.7 GAN reconstructions of PCSO images 72
4.8 GAN reconstructions of MPIE images 72
4.9 FaceResNet-28 layer architecture 74
4.10 CASIA WebFace dataset 75
4.11 ROC curves for Experiment 1 on MPIE 79
4.12 MPIE data with pose and expression variation 81
4.13 MPIE data with pose, expression, and illumination variation 84
4.14 Problems with the CFP dataset 86

5.1 Re-warping of 3D model to remove expression 91
5.2 Synthesized illumination images 92
5.3 Face substitution using images from the MPIE dataset 93

A.1 Ideal pinhole camera model 96

A.2 Illustration of camera projection 99
A.3 Vertex occluded by a surface triangle 103
A.4 Plane created by a surface triangle 104
A.5 Intersection of ray and plane 105
A.6 Smaller sampling region for self-occlusion detection 106

B.1 x-y plane as a thin sheet for TPS warping 108

Chapter 1

Introduction

Unconstrained face recognition has been a long standing problem in the field of computer vision and machine learning. Despite the large amount of research in the area, factors such as pose, expression, illumination, and occlusion still affect face recognition systems to a large degree. Of these factors, perhaps the most pervasive, from a surveillance standpoint, is the problem of pose. Many face recognition systems [1, 2, 3, 4, 5] claim to be able to match "in-the-wild" images and, therefore, can handle pose variations. However, these methods are, more often than not, evaluated on datasets of images collected from the internet such as the Labeled Faces in the Wild (LFW) [6] or the IARPA Janus Benchmark A (IJB-A) [7] datasets. These datasets have an inherent problem when dealing with pose invariant face recognition. Images posted online, especially those of celebrities, are often biased towards a frontal viewpoint as that is what people like to see. This lack of control over the pose of the images and the bias present in the datasets leads to artificially high face recognition performance that cannot be replicated in many scenarios. For instance, in law enforcement, there is often only a non-frontal image of a perpetrator available as, in many cases, the subject is actively trying to avoid looking directly at a camera. No other images of the subject would be available to perform recognition on and the gallery that must be matched to is often a set of frontal mugshot images. In these cases, whatever face

recognition system is in place must be able to recognize the subject at any pose, using only the frontal enrollment image. It is precisely in these cases that many current state-of-the-art face recognition systems would fail. However, this is not seen in many evaluations due to the fact that there is no control over the modes of variation in the testing data. The goal of this dissertation is to provide a new method for face recognition that provides much more tolerance to matching highly off-angle faces to frontal ones.

1.1 The Challenges of Pose Invariant Face Recognition

Posed, or off-angle, faces present challenges at almost every step of the recognition pipeline. Face detection, facial alignment, and face recognition are all affected by the pose of the face in an image. As these are sequential steps in the face recognition pipeline, any error at an earlier step will propagate forward and result in a lower overall accuracy of the system. Therefore, in order to be able to truly match faces at any pose, each of these individual steps has to be able to handle faces at any angle.

1.1.1 Face Detection Across Pose

Of all of these steps, face detection is probably the most mature in terms of handling high degrees of pose variation. This is mostly due to the fact that without a strong face detector, most other face related applications cannot work in a real world setting on any medium or large scale. As a result there has been much more work on first creating a face detector capable of finding faces at the more extreme angles. Recently, the creation of the WIDER FACE dataset [8], has helped illuminate the major problems that must be addressed in face detection systems. The dataset contains many images of faces under many different types of degradations such as scale, pose, occlusion, etc as can be seen in Fig. 1.1. The

Figure 1.1: A few examples of images from the WIDER FACE dataset with ground truth bounding boxes annotated in green. This dataset includes many faces that have large pose variation in addition to occlusion, scale, illumination, and expression variation.

work being done on solving the WIDER FACE dataset, [9, 10, 11, 12, 13] just to list a few, has propelled face detection forward and has shown that it is capable of handling many very difficult scenarios. These advances in face detection have also exposed the very real problems in face recognition.

1.1.2 Facial Alignment Across Pose

Often, in the preprocessing of a face for recognition, the face has to be properly aligned. Many traditional recognition algorithms would require the face to be aligned based on some fiducial landmarks on the face such as the eyes, nose, or mouth. However, finding these

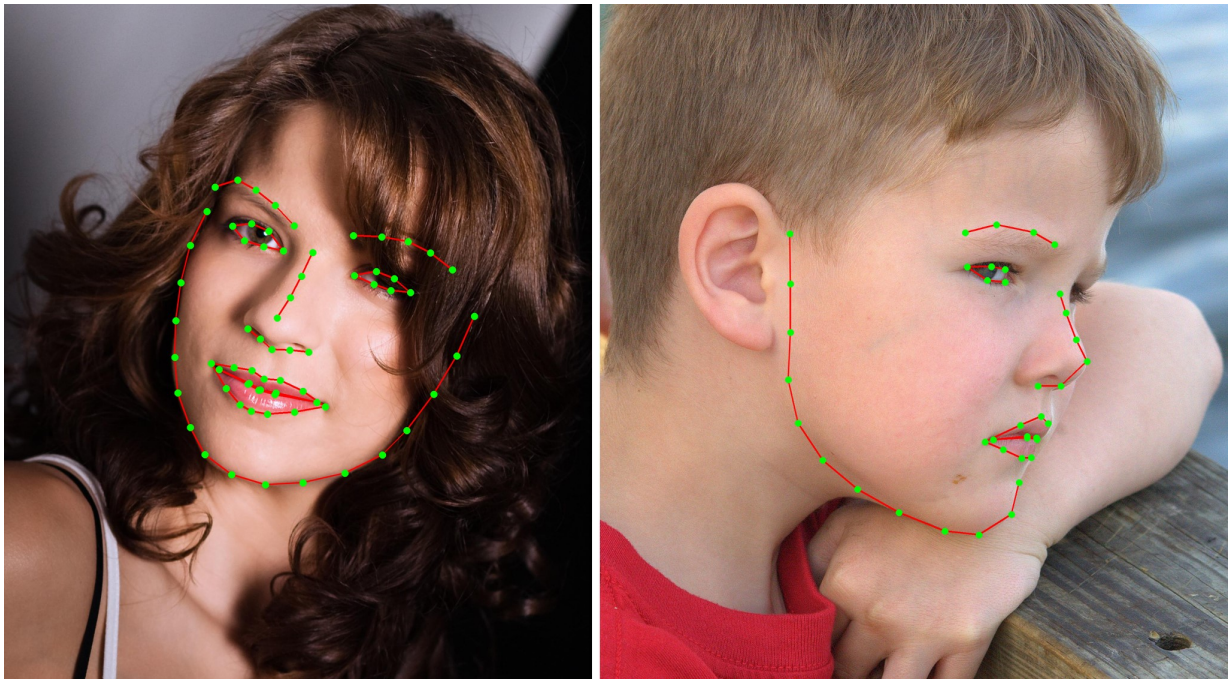

Figure 1.2: The ground truth landmarking schemes from the MENPO dataset [14, 15] on both frontal and profile images. The number of landmarks changes as the angle increases meaning not all images will provide landmarks for both eyes or other keypoints on the face that might be used for normalization.

keypoints on off-angle faces can be very difficult as the points themselves quickly become occluded by the structure of the face. In past facial landmarking techniques, this has not been considered a problem as ground truth data would either give locations of the self-occluded landmarks or, as the pose became extreme enough, would change landmarking schemes to a profile version as can be seen in Fig. 1.2. This raised two problems, however. The first is that the selected points for the self-occluded landmarks would naturally be moved to a visible boundary in the image. This would happen because human labelers needed some visual cue as to where to click the landmark in order to generate consistent sets of landmarks across multiple images and labelers. This visible boundary, however, would move as the pose of the face changed and would result in a different semantic meaning of the landmark at every pose. This "landmark traversal" problem, as seen in Fig. 1.3, was explored in [16] and was shown to be a consistent problem in many landmarking

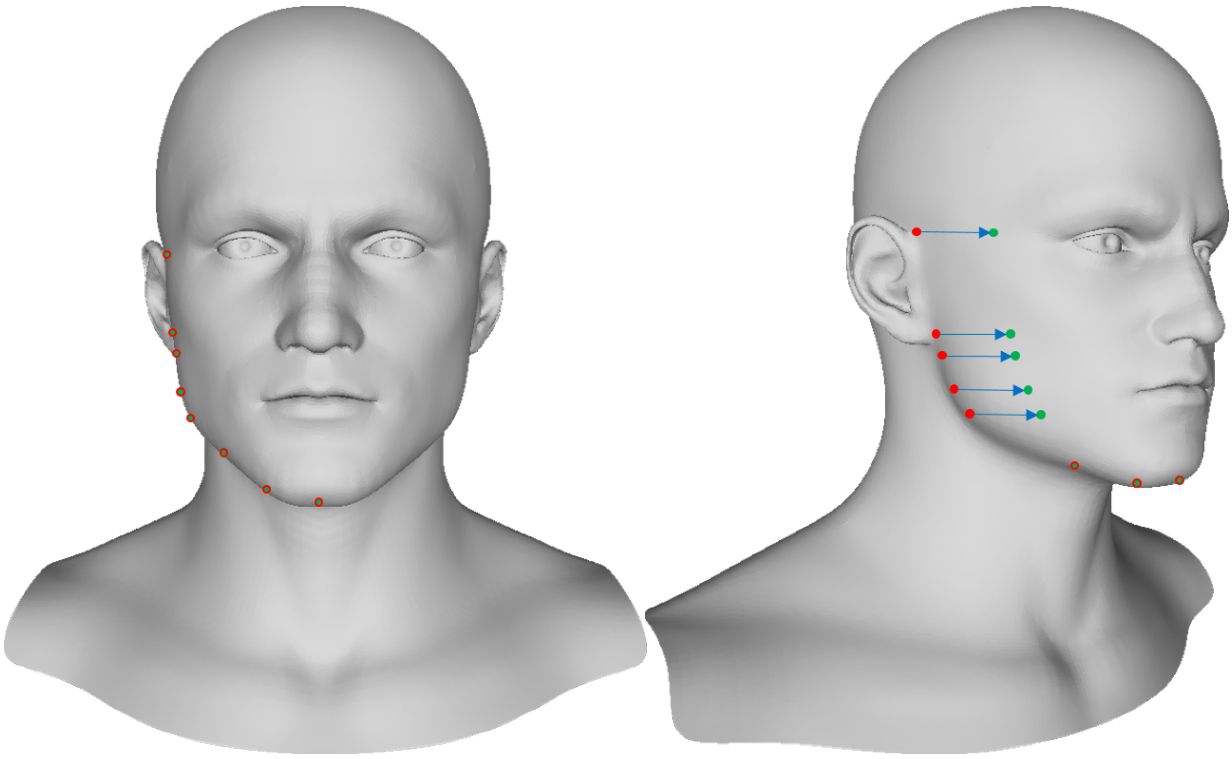

Figure 1.3: The landmarks in green are representative of what is traditionally selected when landmarking a frontal view while the landmarks in red are those selected when landmarking a profile view. Notice how when viewed from a frontal angle, these appear to be the same points but, in actuality, are very different as can be seen in the off-angle view.

datasets. Even relatively recent datasets, such as the MENPO dataset [14, 15], have this inconsistency in the ground truth data. As can be seen in Fig. 1.4, the selected landmarks can have quite a different semantic meaning as the pose changes. While this is not inherently problematic to face recognition, any models making assumptions about the shape and structure of the face would have to take this inconsistency into account, which has often not been the case. For example, there have been many methods in the past, such as the 3D Generic Elastic Models (3D-GEM) [17], that rely on facial landmarks to generate a 3D model of the face from a single 2D image. The 3D-GEM method would deform a generic face shape to an input image based on detected landmarks. While this was shown to be a fairly accurate 3D model, it suffered from the fact that the landmarks it needed could not be consistently found at non-frontal poses. This removed the possibility

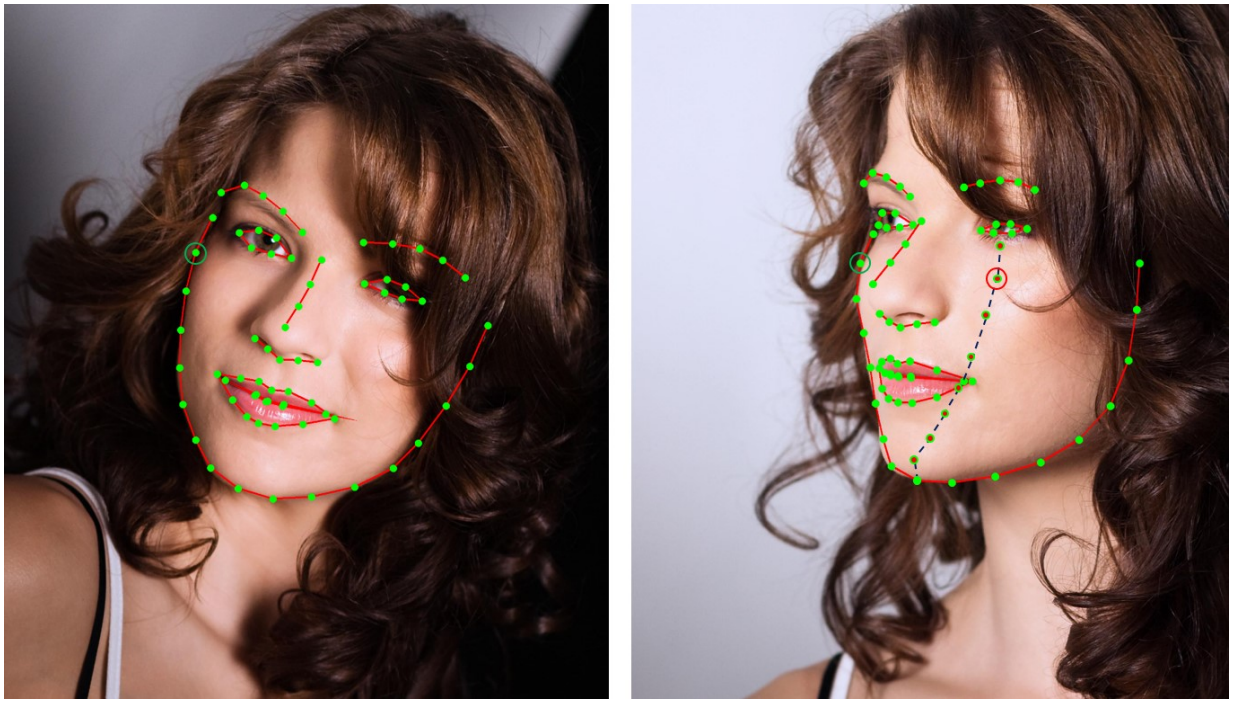

Figure 1.4: An example of how traditional landmarking datasets and methods gave inconsistent landmarks. The ground truth landmarks from the MENPO dataset [14, 15] are shown as solid green points. The same point in the landmarking scheme from the ground truth data is circled in both images in green. In the frontal image, it is where the bottom of the ear joins the head while in the off-angle image, it has moved to somewhere on the cheek. The 2D projection of the jawline on the off-angle image is shown as red points with a green outline. The more correct location for the selected landmark is shown in the red circle.

of using this 3D understanding of the face to improve face recognition for non-frontal faces. The second problem is that any alignment or normalization scheme based on these landmarks would naturally change as the pose changes. A very common approach has been to normalize the face by the eye locations to eliminate scaling and rotation from the face images. However, as the face moves further and further away from a frontal viewpoint, the eyes move closer and closer together in the image. Normalizing the scale of the face based on this distance results in larger and larger faces as the pose changes. Even worse than that, as the pose changes to the more extreme angles, many traditional landmarking systems would stop providing a second eye landmark altogether as can be seen in Fig. 1.2. This means normalizing faces at large angles requires a decision of how to normalize the face

at different poses. Many of the previous approaches to facial alignment and landmarking pushed this decision off to the face recognition system.

1.1.3 Face Recognition Across Pose

Feature extraction for face matching is the last step in any face recognition pipeline and is thrown off by any errors encountered in the previous steps. Any undetected faces would not be processed and badly detected faces would fall so far outside the expected distribution of faces seen in training that the recognition system would have no chance of performing well. Even on those off-angle faces that were detected, facial alignment failed or changed schemes as the pose changed, causing the need for recognition models that could handle very different kinds of input, both in style and dimension. Even if detection and landmarking worked perfectly, the problem of data made robust face recognition across large pose variations nearly impossible. Many of the widely used face recognition datasets today, such as the CASIA WebFace [18], IARPA Janus Benchmark-A (IJB-A) [7], Labeled Faces-in-the-Wild (LFW) [6], or the Janus Challenge Set 3 (a super-set of the IJB-A dataset also known as CS3), all suffer from a lack of data. Though these datasets are thought to be large and well representative of the range of faces that should be recognized, they all suffer from the fact that they are collected from online sources and, most often, are pictures of celebrities. As a result, there is, most likely, a bias present in these datasets as celebrity photos are, most often, frontal images taken at high publicity events. Even images posted to social media by everyday people tend to be frontal as the photos are taken when they are posing for them. Of course, this is only an intuition as these datasets do not have labeled pose estimates for each photo. However, we can estimate the pose distribution of various datasets to see the likely bias present. Fig. 1.5 shows how the estimated pose distribution of these datasets is highly biased towards a near-frontal face. This bias means that there is a lack of data for both training and testing face recognition models across pose. Some of the older face

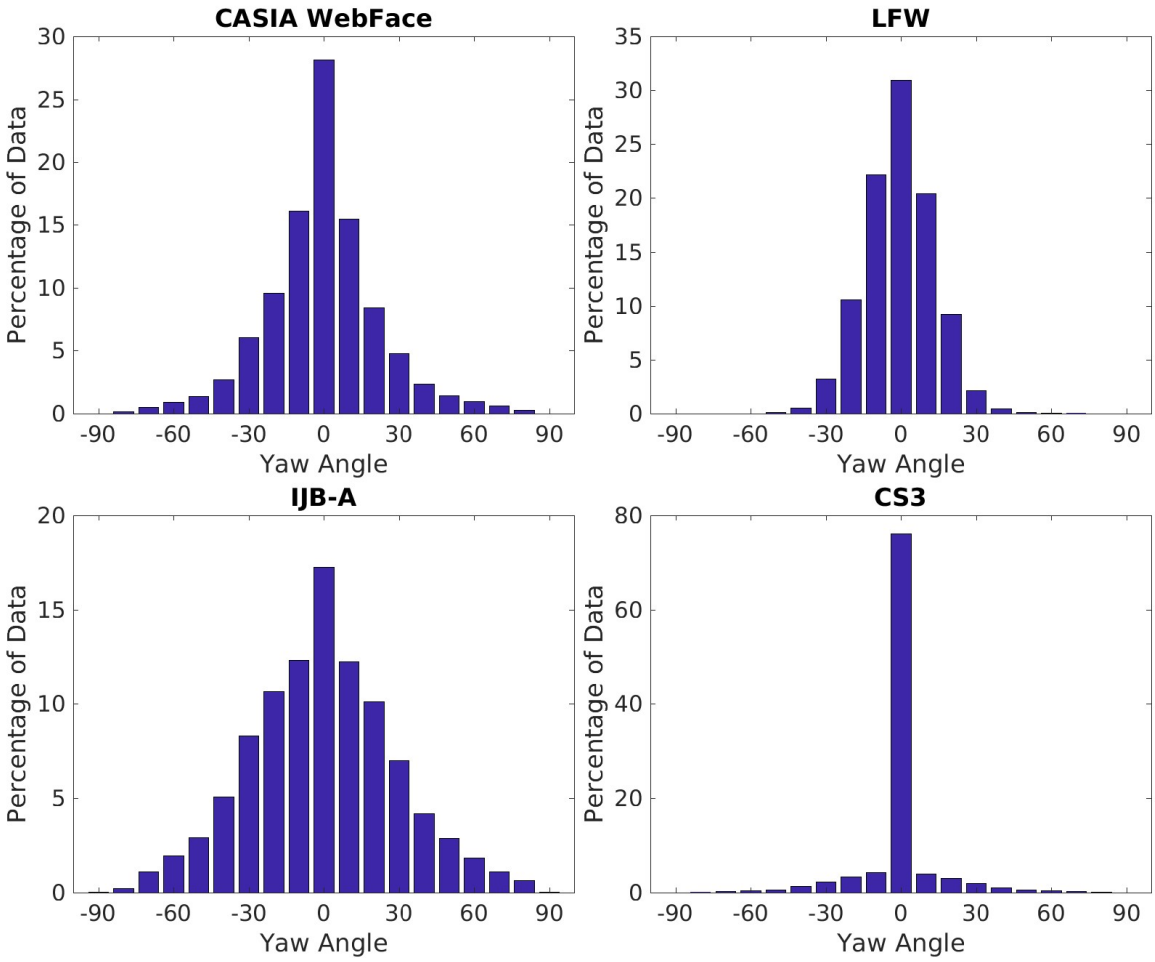

Figure 1.5: The estimated pose distribution for the CASIA WebFace, LFW, IJB-A, and CS3 datasets. It s clear that these datasets have a bias towards frontal or near-frontal faces. In the $\pm 30^{\circ}$ range, there is $\approx 89\%$ of the CASIA data, $\approx 99\%$ of the LFW data, $\approx 78\%$ of the IJB-A data, and $\approx 95\%$ of the CS3 data. The details of the pose estimator used can be found in Section 3.5.

recognition datasets, such as the CMU Multi-PIE (MPIE) [19, 20] dataset, have a uniform pose, or at least more uniform, distribution on the pose. To see how badly face recognition models perform across large pose variations, we take the FaceResNet-28 layer architecture developed by Wen *et al.* [21] and train it on the CASIA-WebFace dataset. While this is not the largest dataset available, it provides a good baseline training dataset that many groups still use. This model is able to achieve a 96.86% accuracy on the LFW dataset and yet, when we evaluate it on the MPIE dataset as seen in Fig. 1.6, we can see how

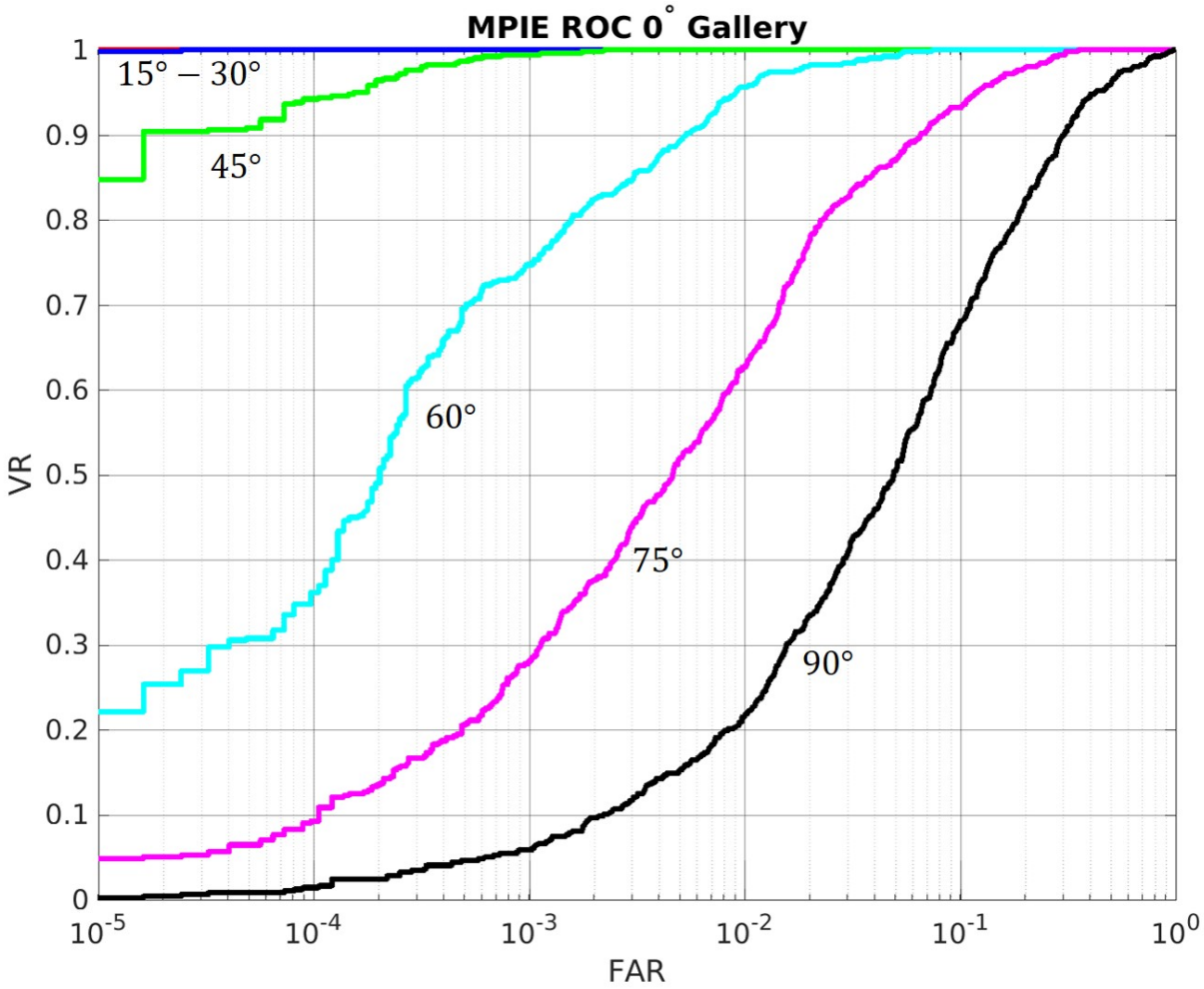

Figure 1.6: The Receiver Operating Characteristic (ROC) curves for the FaceResNet-28 model on the MPIE dataset. The gallery consists of the frontal images and the probe set contains the images at each pose grouping separately. All images were neutral illumination and expression to see the effect of pose only. Only the subjects from Session 1 were used to ensure no differences due to capture time as well.

quickly the accuracy drops as the pose increases. Out to ±30°, exactly where the vast majority of the training data lies, the model is able to do extremely well. However, at ±45°, the accuracy starts to drop at the lower False Accept Rates (FARs). Past ±45°, the Verification Rates (VRs) drop to unacceptable levels, especially given the relative ease of this dataset with illumination, expression, and time controlled in the images. However, there is not a large scale dataset that contains a good distribution of pose for training deep networks or other methods that require large amounts of training data. One of the goals

of this dissertation is to develop a method by which the current available data can be used to create recognition models that generalize well over large pose variation.

1.2 Summary of Contributions

In summary, the main contributions of this work are as follows

- We develop a new method of simultaneously fitting a 3D model to an input 2D face image at any pose and extracting accurate landmarks that maintain a semantic meaning over all poses. This is done using our newly developed 3D Thin Plate Spline Spatial Transformer Networks (3DTPS-SPT) to both model the camera projection parameters and the Thin Plate Spline warping parameters that model the face structure in both 2D and 3D. As a byproduct of this modeling, we are also able to estimate the pose of the face and we show that the resulting pose estimate is fairly accurate out to $\pm 75°$.
- We analyze the pose distribution of commonly used training and testing datasets for face recognition and show that there is a heavy bias in these datasets. This bias leads to poor pose tolerance in face recognition systems as any uniform distribution on pose causes a mismatch between the training and testing distributions.
- We propose to frontalize all the faces in both the training and testing datasets to remove the effect of the mismatched pose distributions. We show that the resulting frontalized faces are very stable on one half of the face image depending on the sign of the pose. By cropping the frontalized faces to only the visible half-face, we generate a much more stable input over all poses for recognition purposes.
- We use the frontalized half-faces to train a Generative Adversarial Network (GAN) to reconstruct the missing half of each face.
- We show the effectiveness of our methods as a preprocessing step by training a

popular deep architecture for face recognition on these frontalized half-faces, the reconstructions, and the original whole faces to compare how they perform on the recognition task. We run several evaluation experiments on the CMU Multi-PIE dataset in order to control for pose, illumination, and expression. This allows us to see the effect of the pre-processing step. We also run an evaluation on the Celebrities Frontal-Profile in the Wild (CFP) dataset to see how our method performs on real-world images.

1.3 Notation and Organization

The rest of this dissertation is organized in the following manner. Chapter 2 outlines the key related works in facial alignment and modeling as well as pose invariant face recognition and how these works motivate our approach. In Chapter 3, we develop our new approach to 3D facial alignment and modeling through the use of our 3DTPS-SPT networks. We first give a small overview of the original Spatial Transformer Network (STN) and how it relates to our network. We then go into detail of how each new module in the network is formulated and derive both the forward and backward pass equations for training the network. We also explain how to use the results of the network to refine the 3D model of the face to ensure it is textured properly from the input image. We run a series of facial landmarking experiments and compare to many other prior works to show the efficacy of this approach. Lastly, we run a small set of pose estimation experiments to show the estimated pose is within a reasonable error. In Chapter 4, we discuss how to generate a pose invariant face from the pose varying inputs by only sampling the visible half of the face at each pose. We also show how this half-face can be used to train a network to reconstruct the missing regions of the face. Lastly, we run a series of experiments using the original faces and the faces generated using the methods outlined earlier in the chapter to evaluate the efficacy of these techniques in achieving pose invariant face recognition.

Appendix A goes into detail about the formulation of the camera projection matrix and the various properties used in Chapter 3. Similarly, Appendix B details how the Thin Plate Spline function used in Chapter 3 is created and how to find the parameters of the function.

Throughout this work, the following notation is used. Any new notation is explained as it appears.

m	A scalar value
$\mathbf{m}$	A column vector
$\mathbf{M}$	A matrix

Chapter 2

Related Works

2.1 Facial Alignment and Modeling

There has been a long history of work in trying to develop the most accurate facial landmarking system for recognition, as well as many other applications. One of the first methods developed for facial landmarking was the Active Shape Models (ASM) developed by Cootes *et al.* [22]. The ASM method refines an initial set of landmarks by looking in a small local neighborhood around each point to find the best new landmark location. This is often done by matching some feature extracted from the area to a learned template for that landmark. The resulting landmarks are then projected onto some shape basis that has been learned from training data, usually through a Principal Component Analysis (PCA). This forces the landmarks into a reasonable face shape. This process is iterated either for a set number of steps or until the landmarks reach some convergence criterion. While the ASM approach relies on a shape basis to regularize the landmarks, the Active Appearance Model (AAM) [23] also imposes a regularization based on the global texture of the region. AAM approaches either use independent shape and texture subspaces or a joint subspace for both. In either case, an iterative process is still used to find the best set of landmarks according to the model. While the addition of a texture basis showed

improvements in many alignment tasks such as aligning MRI scans, the task of finding facial landmark points that depend more on local structure was better suited for ASM techniques [24]. This is generally because AAMs are based on a global texture and, therefore, degrade faster in the presence of factors such as pose, illumination, and expression. Unfortunately, these are exactly the factors to which facial landmarking must be tolerant. In both of these approaches, the key factor is how well the subspaces learned from the training data can model unseen images. While these subspaces are intended to restrict the shape or texture to the set of reasonable face shapes or images, it is possible that some strange combination of basis elements will result in a very strange looking shape. Seshadri *et al.* [25] proposed a Modified Active Shape Model (MASM) in which a regularization was done on the coefficients of the subspace projection in order to ensure realistic face shapes were extracted. This was done by restricting the projection coefficients to be within three standard deviations of the mean projection coefficient. In this way, no individual basis element could fall outside a likely distribution and badly influence the final shape. While this may have restricted the subspace from modeling a few faces well, it was shown to perform more accurately than the traditional ASM approach.

All of these methods require a proper initialization from a face detector to find an accurate set of landmarks. Zhu and Ramanan [26] approached the landmarking problem in a different fashion. They saw the problem of facial alignment and face detection as an interconnected problem and proposed to solve both at the same time with a single method. By using a mixture of trees to model how the landmarks deform and using this information to drive the decision of whether a region was a face or not, they were able to achieve very impressive results on both face detection and landmarking at the time. However, their model relied on enumerating all the possible views of a face to fit the best model to a region as the number of landmarks changed in different poses. Xiong and De la Torre in their work on the Supervised Descent Method (SDM) [27] and its extension, the Global Supervised Descent Method (GSDM) [28], decided to focus more on alignment

in video. The SDM algorithm solves a non-linear optimization by using descent directions learned from the training data, thus reducing the complexity of the solution. The main difference between SDM and GSDM is that the objective function in GSDM was divided into multiple regions of similar gradient directions. It then constructed a separate cascaded shape regressor for each region to try and solve the non-linear optimization problem more efficiently. Both of these methods take advantage of the temporal correlation between samples when processing video. By using one frame's landmarks as the initialization to the next, both SDM and GSDM were able to achieve very accurate and stable landmarks over large pose variation. However, in many applications, only a single image is available and no landmarks can be tracked from a good pose to deal with large viewing angles.

With the rapid progress being made in many areas using Convolutional Neural Networks (CNNs), it was only natural that CNNs would be applied to facial alignment. Many CNN-based methods have achieved impressive results in facial alignment over the past few years. The advantage of these CNN based models over the previous approaches is that a CNN based method, and deep learning in general, does not rely on hand crafted or pre-selected features such as Histograms of Oriented Gradient (HoG) or Scale-Invariant Feature Transform (SIFT) features. Instead, the model learns what are the best features to use for a specific task. This allows the model to be much more tuned to the problem at hand than models using a pre-selected group of features. Previously, deep learning based approaches were not feasible, either due to hardware constraints or the lack of large scale amounts of data. However, this has changed over the past few years with more powerful graphics cards being used in training these models as well as a push for larger and larger datasets. Several of these CNN approaches actually aim to do more than just facial alignment in the same model in what is commonly referred to as multi-task learning. Zhang *et al.* [29] used this idea of multi-task learning in creating their Task-Constrained Deep Convolutional Networks (TCDCN). The TCDCN model is trained by first pre-training it on a sparse set of landmarks on the face with the additional tasks of classifying various attributes about

the face such as the gender, presence of facial hair, or a rough pose estimate for example. This addition of the attributes is intended to force the network to understand how these attributes affect landmark localization by making it explicit in the training step. Once a model is pre-trained on a sparse set of landmarks, the convolutional kernels are transferred to a new network that aims to find the dense set of landmarks that are of interest. This network is trained on the dense set of landmarks only so that it focuses on the landmarking as the final task and is not concerned with errors in the attribute classification. This idea of multi-task learning is very similar to Zhu and Ramanan's work [26] and has also been applied to joint face detection and landmarking in CNN based models.

Zhang *et al.* [30] merged these two tasks in their Multitask Cascaded Convolutional Networks (MTCCN). In a similar fashion to the traditional Viola-Jones detectors [31], in which a series of Haar cascades are used to progressively reject detections until only the most confident regions remain, the MTCCN model cascades several stages of CNNs to generate accurate detections and landmarks. The first stage generates a large set of proposal bounding boxes for face detection while the second stage takes all of these proposed regions as separate inputs and tries to reject the incorrect regions. A final CNN is used to select the final set of faces in the image as well as generate a set of sparse landmarks for each detected region. Ranjan *et al.* [32], in their HyperFace framework, do not use the idea of cascading networks but instead add many more tasks to the network. The goal of the HyperFace network is to simultaneously detect faces, find landmarks, estimate landmark visibility, estimate the three orientation angles (pitch, yaw, and roll), and classify the gender of the face. The HyperFace network has achieved some impressive results and has shown that there is a lot of redundant information in many of these tasks that can be shared and potentially reduce computation and memory consumption for these networks.

While all of these approaches have shown steady improvement in the area of facial alignment, they still do not address the main problem of landmark traversal when aligning off-angle faces. Realistically, in order to find the 2D locations that have the same meaning

across all poses, an understanding of the 3D structure of the face has to be embedded, either explicitly or implicitly, somewhere in the model. This often means estimating a 3D model of the face from the input image. While estimating a 3D model from images is not a new problem, the task of modeling objects from a single image has always posed a challenge. This is, of course, due to the ambiguous nature of images where depth information is removed. With the recent success of deep learning and especially CNNs in extracting salient information from images, there have been many explorations into how to best use CNNs for modeling objects in 3 dimensions. Many of these approaches are aimed creating a depth estimation for natural images [33, 34, 35, 36, 37]. While the results on uncontrolled images are impressive, the fact that these models are very general means they tend to suffer when applied to specific objects, such as faces. In fact, many times, the depth estimate for faces in the scene tend to be fairly flat. Thankfully, since we are only interested in modeling faces, the resulting estimated 3D model can be made much more accurate.

One of the earliest methods in modeling the 3D structure of the faces was the 3D-GEM method developed by Heo and Savvides [17]. This method was based on a very simple observation that the relative depth of various points on the face did not change very much between people once they were aligned in the x and y dimensions. By localizing a set of keypoints on the image, a generic depth map could be deformed to fit the input points and a depth estimate could be generated for the input image. The resulting model could be used to synthesize new viewpoints of the face for later use in recognition [38]. However, this method could only be run on frontal face images as there was no inherent understanding of how the depth map should change as the pose of the face changes. Hassner *et al.* [39] proposed using a similar generic 3D structure to try and estimate the camera projection parameters for the image using the landmarks on the image and the 3D model. By doing this, they claimed to be able to fit the model to a face at any pose and frontalize (re-render it from a frontal viewpoint) it for use in improving recognition across pose. The missing regions are filled using a symmetry constraint while visually obvious regions, such

as the eyes, are excluded. However, the results they show are only on the LFW dataset which, as was shown in Fig. 1.5, is almost entirely made up of near-frontal faces. Even the visualizations shown for the corrections are only from near-frontal faces where there is very little normalization that needs to be done. Both of these methods require that the landmarks on the face be given as an input to the method and do not attempt to address the landmark traversal problem. Zhu *et al.* [16] were one of the first to try and directly incorporate the landmark traversal into their modeling approach. With an input pose estimate, they can determine how each of the landmarks would move along the contours of the face structure to find a new 3D correspondence for the 2D landmarks that were found on the image. In order to get this pose estimate, the pitch and yaw rotations needed to best match the original 2D-3D correspondences are solved for. However, since these correspondences are incorrect, this process must be done iteratively until convergence. All of these methods still require another method to provide a set of landmarks to be able to generate the 3D model of the face.

Instead of requiring the landmarks as an input, many methods are now trying to simultaneously estimate the 3D structure of the face and the 2D landmarks. To be able to estimate the 3D structure of the face and the corresponding 2D projections on the image with any CNN based model, a large amount of training data is needed that has both images with 2D landmarks and 3D models. Such a dataset did not exist for a long time as collecting 3D scans of faces was very time consuming and required specialized equipment. This meant that capturing data could not be done in the wild and resulting datasets would be highly controlled. Jourabloo and Liu [40] attempted to address this problem in their Pose Invariant 3D Face Alignment (PIFA) method by fitting a 3D model to training data by only fitting on those points visible in the image. This allowed for the generation of a 3D model associated with every training image and, as a result, an estimate of the invisible landmark locations. By training a cascade of regressors on the newly created training data, they were able to generate impressive initial results that estimated the invisible landmarks

Figure 2.1: One subject from the 300W-LP dataset. The original image (top left) has a 3D model fit to it and is then synthetically rotated towards 90°. The image is flipped and rotated towards −90° as well. Images are never rotated towards 0° to avoid having to fill missing regions of the face.

in each image. However, their model was only trained on a small number of faces as each image had to have visibility estimates for the landmarks so that the estimated 3D shape would be accurate. Most datasets do not provide this information and thus the amount of available data was limited. Zhu *et al.* [41] were able to alleviate this problem somewhat with their creation of the 300W-LP dataset which consists of the images from several landmarking datasets combined. By fitting 3D models to these images by hand, the images could be synthesized at many different views to provide a large amount of data of subjects at different poses, as can be seen in Fig. 2.1, along with the 3D shape and its 2D projection on each image. The main advantage this dataset has over other synthesized datasets is that it maintains the background of the image to a large degree resulting in more realistic looking images. By using this dataset, Zhu *et al.* [41] created their 3D Dense Face Alignment (3DDFA) technique which aims to map the projected coordinates of the 3D shape to the image. The 3D shape is created by using a 3D Morphable Model (3DMM) to generate a candidate shape. This requires a set of coefficients (the parameters of the 3DMM) to generate the shape. The 3DDFA technique assumes an initialization of the coefficients and

then uses the projected coordinates of the shape along with the input image to predict an update to the coefficients. This can be done iteratively until the coefficients converge though the authors say that, in practice, 3 iterations seems to be enough. 3DDFA was one of the first CNN-based techniques to demonstrate high accuracy on 3D facial alignment and was able to generate a good solution to the landmark traversal problem. Most recently, Bulat and Tzimiropoulos [42] created a new, large scale dataset of 3D landmarks from 2D images for the purpose of evaluating 3D landmarking methods. By utilizing stacked hourglass networks [43], they created a model to generate 2D projections of 3D landmarks from only traditional 2D landmarks, called 2D-to-3D Face Alignment Networks (2D-to-3D FAN). While this does allow them to generate "3D" landmarks from many of the large scale, traditional landmarking datasets, it introduces a natural bias to any evaluation on these datasets. As is the case with any synthetically generated data, there is only a benefit to reaching a certain point of accuracy before there is a worry of over-fitting to potential bias in the synthesis procedure. Unfortunately, Bulat and Tzimiropoulos [42] do not evaluate any of their 3D landmarking models on any human labeled data or any non-synthetic datasets, making comparison to such a method difficult.

2.2 Face Recognition

There is an equally long history of work in face recognition as there has been in facial alignment. With the very rapid progress being made in face recognition using deep learning based methods, it is important to understand what advances have been made. Some of the earlier and promising results to come out of the field focused heavily on the importance of the data used to train these models. The DeepID [44] and DeepID2 [45] networks, were some of the first to show a large leap in performance on the LFW dataset and and achieving over 97% accuracy. Both of these networks would not be considered very deep by today's standards but were able to achieve very impressive results nonetheless. This

was due, in no small part, to the use of the CelebFaces+ dataset [44] in the training of the networks. This dataset contained 202,599 face images of 10,177 identities, a large departure from the traditional face recognition datasets which only had several hundred identities. Very soon after, Taigman *et al.* [46] were able to improve the recognition rates to almost human levels of performance by using the Social Face Classification (SFC) dataset from Facebook. This dataset contained fewer subject with only 4,030 identities but had a total of 4.4 million labeled faces, almost 22 times the amount of images used by DeepID and DeepID2. Lately, however, many methods have seem to have hit diminishing return from larger and larger datasets. As a result, there has been a lot of focus on improving various parts of the deep networks, such as the architecture or the loss function used. Perhaps the most well-known deep architecture is the model developed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton known as AlexNet [47]. This model was a fairly basic combination of convolutional layers and a few fully connected layers. However, by splitting the model intelligently, they were able to train it on relatively low memory GPUs, at least by today's standards, on a large amount of data and achieve previously unheard of performance on the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC-2012) [48]. This architecture has been used in many works since as a baseline for what deep learning is capable of on a specific problem. Only a few years later, Simonyan and Zisserman [49] achieved the state-of-the-art performance on the ILSVRC-2014 challenge by exploring the effect of the depth of their network on recognition performance. By expanding their network to $16-19$ layers, they were able to dramatically improve the performance over the AlexNet benchmark. This architecture, commonly known as VGG16, is another often used as a benchmark in many image-based machine learning tasks from detection [10, 50, 51, 52], image caption generation [53, 54], and other image recognition tasks [55, 56, 57, 58]. This trend towards deeper and deeper models soon hit a wall as it was observed by He *et al.* [59] that naively increasing the depth of a model could often lead to lower accuracy. They hypothesized that deeper networks could not model identity mappings between layers easily

and therefore, extra depth would always force the network to change its decision surface. However, by introducing a simple skip connection, in which the input to a "residual block" is added to the output of the block, learning identity mappings becomes much easier. The network merely has to 0 out all the weights in those layers in order to maintain the input at the end of a residual block. Experimentally, this has been key in allowing networks to progress into much deeper territory and has resulted in another leap in performance in many areas, including the face recognition task we are interested in.

While one avenue of research has been on the architectures used in these deep networks, another avenue pursued by many groups has been to investigate the loss functions used to guide the training of the network. These two approaches are equally important as without enough complexity in the architecture, the network may not be able to model all variations needed to achieve good results but without the correct supervision signal (i.e. loss function), the network will not learn to generalize from the training data. While the traditional loss used in classification tasks has been the SoftMax loss, which transforms the output layer of the network into probabilities of each class in the training data, newer loss functions have shown great promise and improvements over it. One of the major problems with learning a SoftMax loss is that it only constrains the feature space learned to ensure the training data is separated. However, there is no weight on how separated the different classes are. This can lead to very small margins between classes which, in turn, can lead to more misclassifications. Schroff *et al.* [60] tried to solve this problem by forcing the network to learn to generate features that were close within a class and well separated from any features from other classes. This was done by training on triplets of training samples, one query sample, one sample from the same class, and one sample from a different class. The triplet loss function penalized large within class distances and small between class distances in the feature space from this triplet. This forced the network to learn to extract features that were good for classification according to the distance metric used. This metric learning resulted in features that were much more stable across variations in the images

and lowered errors in face recognition by a great deal on the LFW and YouTube Faces database [61]. Unfortunately, determining how best to sample triplets from the training data was an open question as continuously sampling easy cases would not lead to a well separated feature space. Additionally, the need for sampling triplets meant the complexity of the training sample space increased from $\mathcal{O}(n)$ to $\mathcal{O}(n^3)$ which, with the large datasets available, was often not feasible in academic settings.

Wen *et al.* [21] realized that a similar feature space might be learned by penalizing large distances of samples to the average feature for that class, also called the class center, while, at the same time, penalizing small distances between class centers themselves in a new loss known as center loss. This would force the features for each class to a compact space while being well separated from the rest of the classes and would not require complex sampling of the training data. Unfortunately, to know the class centers as the feature space changes would require the features for the entire training dataset at each iteration, which is computationally infeasible. Instead, Wen *et al.* [21] maintain the current class centers at each step and update them based on the features in each batch of data used in training. While some centers may not update in every batch, this approximation led to a similarly separated feature space as the triplet loss but in a much easier to learn fashion. While both triplet loss and center loss ideally would learn a well separated feature space for any distance metric, they focused on the Euclidean distance between samples. Liu *et al.* [62] decided, instead, to focus on the angular distance between features and the hyperplanes separating the classes. These hyperplanes are the learned weights in the final fully connected layer of the deep network. The inner product of each feature with the hyperplane that happens in the last layer does not compute the angular distance unless the hyperplanes have a unit magnitude. This observation led Liu *et al.* [62] to enforce a unit magnitude on the weights in the last layer of their network. Once this was done, a regular SoftMax loss would naturally impose an angular distance loss on the resulting space. Empirically, they were able to show that this led to better generalization for face

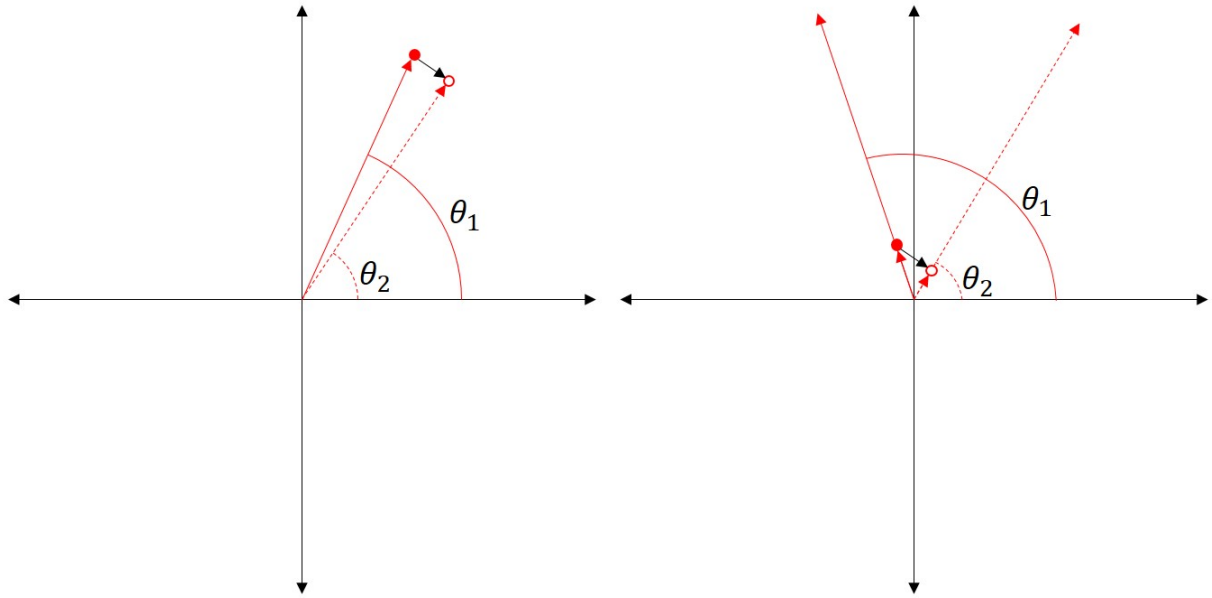

Figure 2.2: A small Euclidean shift, represented by the black arrow, in a large magnitude feature leads to a small shift in the angle as seen on the left (i.e $\theta_2 - \theta_1$ is small). However, the same change when applied to a small magnitude feature, as seen on the right, can lead to a very large change in angle ($\theta_2 - \theta_1$ is large). This means small magnitude features may not be robust to noise.

recognition on large scale testing datasets such as the MegaFace challenge [63]. While this was an impressive result, the angular distance that was desired was not robust to noise in the features as they were not constrained to a unit magnitude. As a result, a small shift in a larger norm feature would affect the angular distance less than the same shift in a smaller norm feature as seen in Fig. 2.2. This was happening because there was no normalization on the magnitude of the features themselves. In the testing scenarios, these features were often normalized after the network had been run on an image but this would not alleviate the problem as the original features were still unstable at low magnitude norms.

Ranjan *et al.* [1] proposed a simple solution to this problem. They added a constraint to the SoftMax loss function that the L_2 norm of the feature had to be some α, resulting in the L_2 constrained SoftMax loss. To satisfy this constraint, they add in a normalization layer that converts the features to unit norm and then a scaling layer that allows it to scale the unit norm features to some learned magnitude, α. While this technique did improve

performance as features tended to be spread further out from 0, and therefore have a larger norm, the actual features extracted in this fashion were not normalized to a specific magnitude by the network as can be seen in Fig. 3b of [1]. This is due to the fact that the normalization layer is applied after the features have already been extracted in the prior parts of the network. In fact, there is no incentive for the network to extract well behaved features in the sense of their magnitude as it will always be re-scaled afterwards. Zheng *et al.* [64] identified this problem and imposed a loss directly on the norm features themselves. Imposing a hard equality on the norm of features would be a non-convex problem to solve so they, instead, use a convex relaxation of the norm constraint and only penalize the features based on how far they are from the desired magnitude. Since this loss is imposed on the features directly, the network learns to extract normalized features and avoids the problem of small magnitude features. This results in all the features being on, or close to, a hypersphere of some radius as seen in Fig. 2.3. Since this loss does not inform classification at all, it is used in conjunction with other loss functions, such as the traditional SoftMax loss. This ring-like appearance in 2D is where the ring loss function derives its name. All of these approaches are intended to be as general as possible and do not attempt to create a method specific to the problems we are trying to solve. While this can be useful in many scenarios, using domain knowledge can improve results dramatically.

To that extent, there have been a few methods developed specifically for trying to solve pose invariant face recognition. Yim *et al.* [65] train a multi-task network to take an input image at an arbitrary pose and synthesize any viewpoint of the face. This is done by taking in the desired pose as an additional input to the network and showing that face at that pose as the desired reconstruction during the training phase. Since this requires ground truth data of the face at each desired pose, the MPIE dataset is used for training the network. By also imposing a loss on the distance between the extracted feature from the reconstruction and the ground truth image, a form of identity preservation was added into the training of the network. This allows them to extract what should be a pose

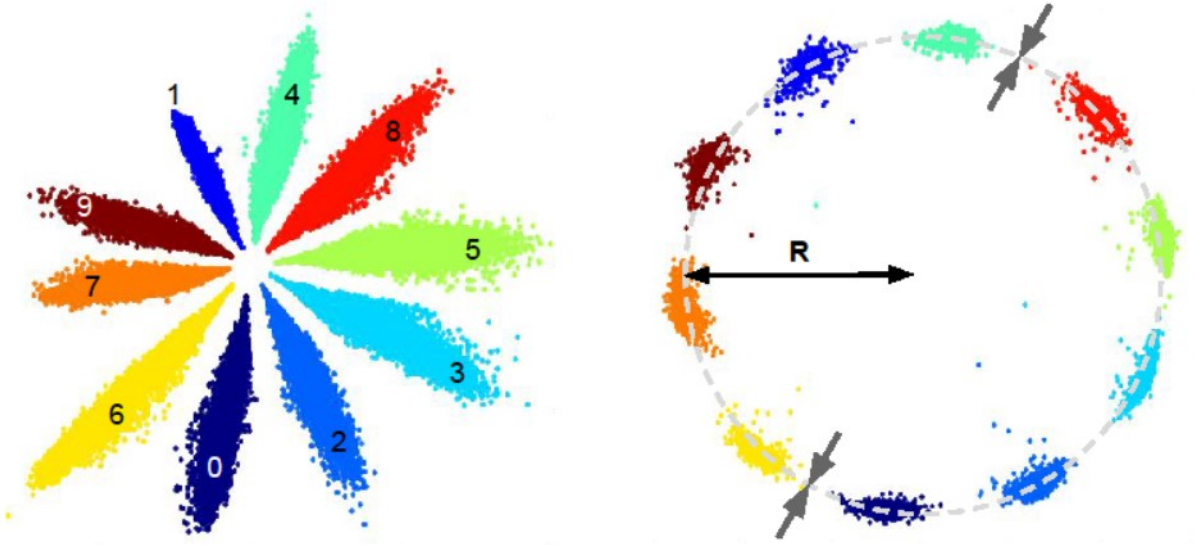

Figure 2.3: Example 2D features extracted from the MNIST dataset using only a SoftMax loss (left) and the Ring loss with a SoftMax loss (right). The Ring loss features are much more consistent in magnitude and avoid the center region of the feature space where small Euclidean shifts can cause large angular distance changes.

invariant feature from an intermediate layer in the network for face recognition. However, the resulting model was only able to generate decent synthesized images out to $\pm 45°$ but, inside that range, generated relatively good images. This limitation on the pose does seem to indicate that the network is not very capable of hallucinating large unseen portions of the face while preserving the identity of the subject. In a similar fashion, Tran *et al.* [66] try to enforce a feature representation that is disentangled from the pose factor by training a generative model to generate both the image and the pose information. Their Disentangled Representation learning-Generative Adversarial Network (DR-GAN) allows for both single image and multi image inputs to be able to generate a pose invariant feature. When dealing with image sets of the same subject, each is passed through the encoder portion of the DR-GAN and a feature vector as well as a weight term are created. These weighting factors allow the individual features to be fused into a single feature vector that can be passed through the decoder with a specified pose parameter to generate the subject at the desired pose. Peng *et al.* [67] also approach the problem by trying

to disentangle pose from the feature but attempt to do so directly on the feature space without the need for reconstructing the input image. Instead, the feature space is separated into a set of dimensions representing the identity and a set of dimensions representing non-identifying information (i.e. pose, landmark locations, etc.). In order to fully decouple the identity and non-identity features, the 300W-LP dataset is used to sample pairs of the same identity, one at a frontal view and one at a non-frontal view. The non-identity features are concatenated with the identity features from each image to generate two new sets of features. These features are passed through a set of fully connected layers to transform them to the "reconstructed" features. By imposing a loss on the distance between these reconstructed features and the features originally extracted from the frontal image, the identity and non-identity features are better disentangled. Masi *et al.* [68] train a feature extraction network for each pose by synthesizing the training data at several specified poses. These different models are used in conjunction to match across pose. In order to get the different models to generate similar features at each pose, the frontal model is trained first. Then as the angle each model is meant to handle increases, each model is fine-tuned from the previous model (i.e. 40° from 0° and 75° from 40°). Then any input image is classified according to a pose estimate and is passed to the appropriate model for feature extraction. All of these approaches try to address pose invariance by training across many synthesized images at different poses but these methods still rely on the network to handle the fact that a pose varying input signal is being given for recognition. The hope is that the network will learn to ignore the pose factor in the input images which means some capacity of the network must be dedicated to this. Since this isn't made explicit, it becomes difficult to tell how much of the network is handling pose versus the recognition of faces.

Another way of trying to ensure that a pose-invariant feature is extracted is to input a pose-invariant image to the network. This requires the face to be normalized to a specific pose, usually frontal, regardless of the input image. While there have been previous

methods, such as [39] and [16] as elaborated in Section 2.1, they have focused on reconstructing the missing regions of the face somehow. These missing regions are the result of self-occlusion from the 3D structure of the face. However, whether the self-occluded regions are filled or not, those areas of the face end up being dependent on the pose of the face itself and therefore are generating a pose varying image. This becomes especially relevant at the extreme poses where a large portion of the face becomes self-occluded. In Chapter 4, we do explore reconstructing the missing regions of the face for recognition purposes but also propose a "half-face" frontalization for recognition. This has the benefit of only showing the network frontal faces without pose variation while, at the same time, removes, or at least severely reduces, the variation due to self occlusion. In this way, when matching a frontal image to any pose, the same half of the face extracted will be visible and will generate a very stable input to the network.

Chapter 3

Pose Invariant Facial Modeling Using 3D Thin Plate Spline Spatial Transformer Networks

When modeling the projective geometry of cameras, it is necessary to distinguish between points and lines in the world and points and lines at infinity. This is needed to model where parallel lines intersect in the image even though they do not actually intersect in the real world. This point of intersection is a real point on the image but represented by a point at infinity in the world space. In order to distinguish these points, homogeneous coordinates are used. A homogeneous representation of a 2D or 3D point is created by appending a 1 to the end of the coordinates unless the point is at infinity, in which case, a 0 is appended to it. The relationship between a point in the world and where it appears in an image is well known and defined by the camera projection equation

$$\mathbf{p}_c \widetilde{=} \mathbf{M}\mathbf{p}_w \tag{3.1}$$

where $\mathbf{p}_c$ is the homogeneous 2D point in the camera coordinate system, $\mathbf{p}_w$ is the homogeneous 3D point in the world coordinate system, and $\mathbf{M}$ is the 3×4 camera projection matrix. This relationship is only defined up to scale due to the ambiguity of scale present in projective geometry, hence the $\widetilde{=}$ instead of a hard equality. The camera projection matrix has only 11 degrees of freedom since it is only defined up to scale as well. Given a set of 2D and 3D correspondences, it is possible to find the $\mathbf{M}$ matrix that relates them through solving a system of linear equations. All of this is explained in more detail in Appendix A.

However, in most scenarios, both the camera projection parameters, $\mathbf{M}$, and the 3D points, $\mathbf{p}_w$, are unknown as only the image is given as an input. Through the use of many images of the same object, both of these can be determined using only correspondences between the 2D points. In a face recognition scenario, however, we do not have the luxury of always having multiple images of the same subject to create a 3D model from. In fact, we often need to first match several images together from different acquisitions in order to generate a 3D model which requires a pose-invariant face matching system in the first place. Therefore, we need to devise a method of determining both the camera parameters and the 3D structure of a face from only the input image. Since we are dealing with faces only, we know the general structure we should be looking for (*i.e.* the mean face shape). Additionally, deep learning techniques have proven that they are capable of finding functions to generate desired parameters from well structured inputs even when it is not obvious how to do so. The combination of these two facts allow us to devise our 3D Thin Plate Spline Spatial Transformer Networks (3DTPS-STN) to generate a 3D structure of the face from a single 2D input.

The question then becomes, how to parameterize the 3D model in order to allow a deep network to estimate the 3D structure of the face. One possibility that has been explored for quite some time is to use a 3D Morphable Model (3DMM) to represent the face. At the heart of it, a 3DMM is a Principal Component Analysis (PCA) subspace of

3D shapes and possibly textures of an object of interest. In the case of faces, the most popular 3DMM in use is the Basel Face Model (BFM) [69], mostly due to it being easily accessible to those in academic circles. Many previous approaches to 3D facial modeling have used the BFM in their approaches. In fact, two recent approaches [40, 41] have used the BFM as an underlying component in their 3D facial alignment methods. However, the BFM is only created from a set of 100 male and 100 female scans. As any basis can only recreate combinations of the underlying samples, this can severely limit the capability of these models to fit outlier faces or expressions not seen before. Although there has been recent efforts to generate more accurate 3DMMs [70], neither the data nor the model is available to researchers in the field of biometrics.

In order to be able to model any face in the wild, we instead use a Thin Plate Spline (TPS) [71] warping to model any possible face shape. Using a TPS warp has one major advantage over a 3DMM. In order to project a given 3D shape onto a 3DMM basis, a set of correspondence points between the given shape and the basis is needed so that the shape and the basis can be aligned. A TPS warp also requires a set of correspondence points between the given shape and a mean shape that is being warped. However, the TPS warp will guarantee that the warped mean shape will exactly match the points on the given shape while determining a smooth transformation for the rest of the shape. In cases where we have correspondence points, such as eye or nose locations in 3D, we want to make sure any shape we generate follows these constraints. This is not generally possible with a 3DMM. Therefore, we can model any possible shape with the TPS given enough correspondence points. Of course, this means it is possible that a TPS warp may model an impossible face, unlike a 3DMM. However, as long as the correspondence points are modeling a realistic face shape, this will not happen. It is also the case that, since we are only dealing with faces, the true shape is not very different from a mean shape (*i.e.* faces do not dramatically change shapes unlike when dealing with general objects). For these reasons, we model a face shape as a 3D TPS warp of a mean face shape. The exact details

of this modeling are described in Section 3.2.1.

3.1 Spatial Transformer Networks

Traditional CNNs either align their inputs before hand based on some localization or assume the network will somehow learn to deal with unaligned inputs through exposure to many variations in alignment. Jaderberg *et al.* [72], instead, decided to approach the alignment problem as another learnable module within the network itself called Spatial Transformer Networks (STN). In this way, the problem at hand, recognition in most cases, can actually drive the alignment process. Additionally, the alignment can be done at any intermediate layer in the network to achieve aligned intermediate feature maps for, potentially, higher accuracy. Jaderberg *et al.* [72] use a deep network to estimate the parameters of either an affine transformation or a 2D TPS transformation. These parameters are then used to generate a new sampling grid which can then be used to generate the transformed image. The STN itself is made up of three components: the localization network, the grid generator, and the sampler as can be seen in Fig. 3.1. The localization network can be any combination of learnable layers but must output as many parameters as are needed for the specific type of alignment or warping function being used. These parameters are then fed into the grid generator, a layer that must be specifically coded for each type of warping function that is to be used. The grid generator takes in these parameters and applies the warping function to a fixed set of sampling coordinates, a uniform grid in general. The result is a new set of sampling coordinates with respect to the input to the localization network. When the input is sampled from those locations, an aligned and localized version of the input is created. This can be passed into later layers of the network so that the classification portions of the network only have to deal with aligned data. The original STN was developed for both affine transformations and 2D TPS transformations. We take this approach one step further by applying it to 3D alignment from 2D images using 3D

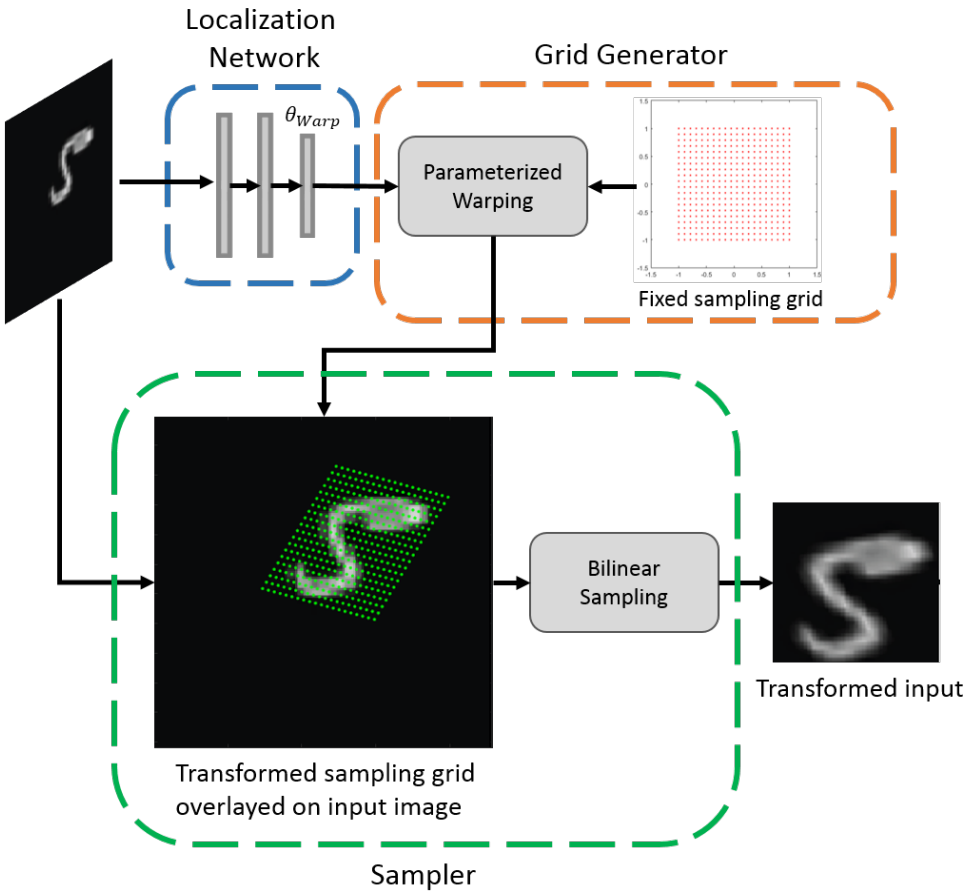

Figure 3.1: Design of the traditional Spatial Transformer Network as shown in [72]. The input image or feature map is passed through some number of layers making up the localization network (blue). The parameters are then passed into a grid generator (orange), which warps the fixed grid based on the specified transformation function. The input is then sampled according to this new sampling grid with a bilinear sampler (green) to generate the aligned output.

TPS warping functions. However, unlike Jaderberg *et al.* [72], we estimate the TPS parameters directly whereas they first estimate a set of points in the image and use those to solve for the TPS parameters. The process of estimating TPS parameters from a set of correspondences is shown in Appendix B.

3.2 3D Thin Plate Spline Spatial Transformer Networks

Whereas the original STN operated purely on 2D transformations of the input, we have extended the idea to make use of a 3D interpretation of the face so that a true 3D normalization can be achieved from a single input image. As stated at the beginning of this chapter, we model the face as a TPS deformation from a mean 3D shape while also using the camera projection equation in Eqn. 3.1 to model how the 3D shape is being viewed in the image. The 3DTPS-STN architecture is detailed in Fig. 3.2. As with the traditional STN, this architecture contains a localization network for each set of parameters that need to be estimated. In this case, they are the TPS parameters, which localize the 3D shape in 3D space, and the camera projection parameters, which localize the 3D shape in the image space. The TPS parameters are used along with a generic 3D mesh of a face to generate a warped 3D mesh that is specific to the subject in the image. This new mesh is used, along with the camera projection parameters, to generate the set of coordinates in the image space from which the 3D mesh can be textured.

While this initial set of coordinates can be used for both facial landmarking and 3D modeling, the results can be improved by an additional regression step. The parameters estimated by both localization networks operate very much on a global image scale to generate the 3D shape and the 2D projections of said shape. However, when dealing with texturing the shape, we want to ensure that the locations of high texture variance, such as the eyes, are localized very accurately. Otherwise, small localization errors in these areas can lead to models with the eyes textured outside of the eye socket or parts of the background being mistakenly put onto the model. These areas need to operate on a much more local region in order to ensure they are localized very accurately. Therefore, we take a subset of the 3D shape, in our case the 68 landmarks commonly used in facial alignment

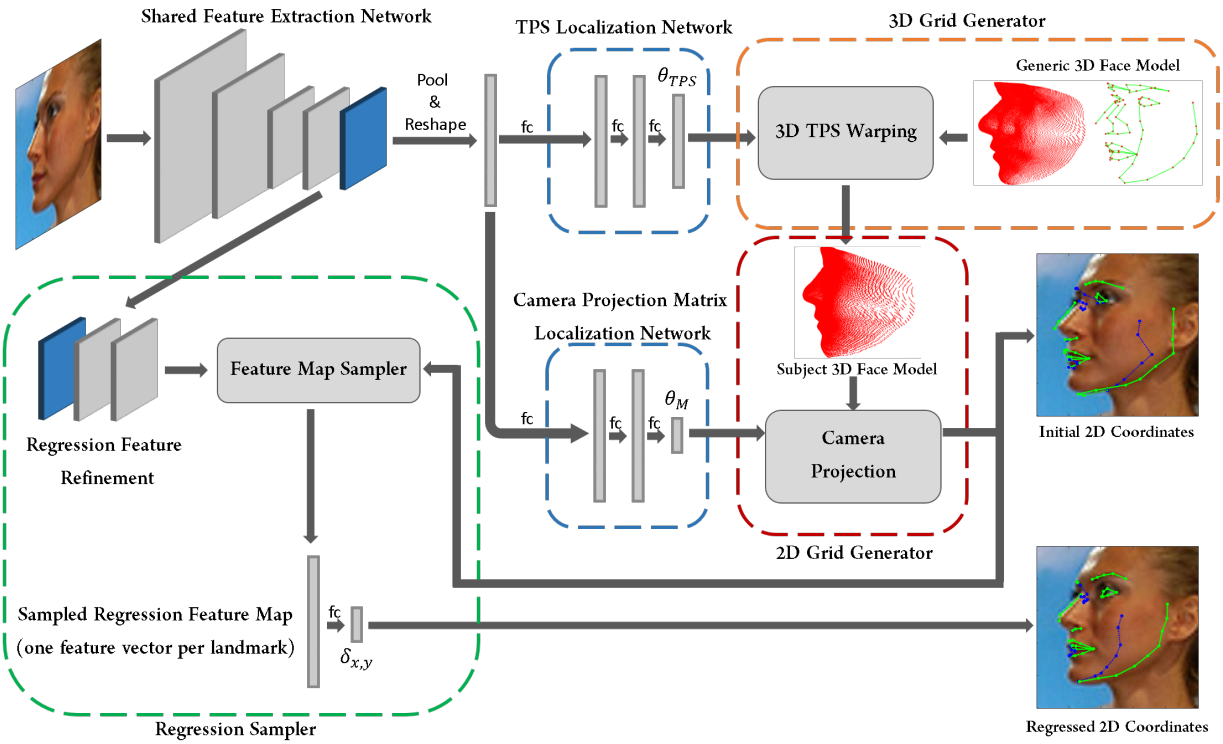

Figure 3.2: Design of the 3D TPS Spatial Transformer Network for facial alignment. Because a 3D model and an estimate of the camera position are found in the output of the network, visibility of landmarks can also be determined. Visible landmarks are shown in green while non-visible landmarks are shown in blue. The input to the regression sampler (in blue) is the same as the feature map extracted from the shared feature extraction network (blue).

methods shown in Fig. 3.3, and refine the 2D locations of those points to generate a final 2D localization. Each part of this architecture is detailed further in the following sections of this chapter.

3.2.1 Thin Plate Spline Transformers

In order to model the face shape as a 3D TPS warp of a mean shape, we have to create a new module that will output the new 3D shape given a set of TPS parameters. Additionally, we have to derive the backpropagation rules for such module in order to be able to learn the rest of the network. The TPS module itself has no learnable parameters but the gradients from later layers have to be passed onto earlier layers appropriately. The details on the

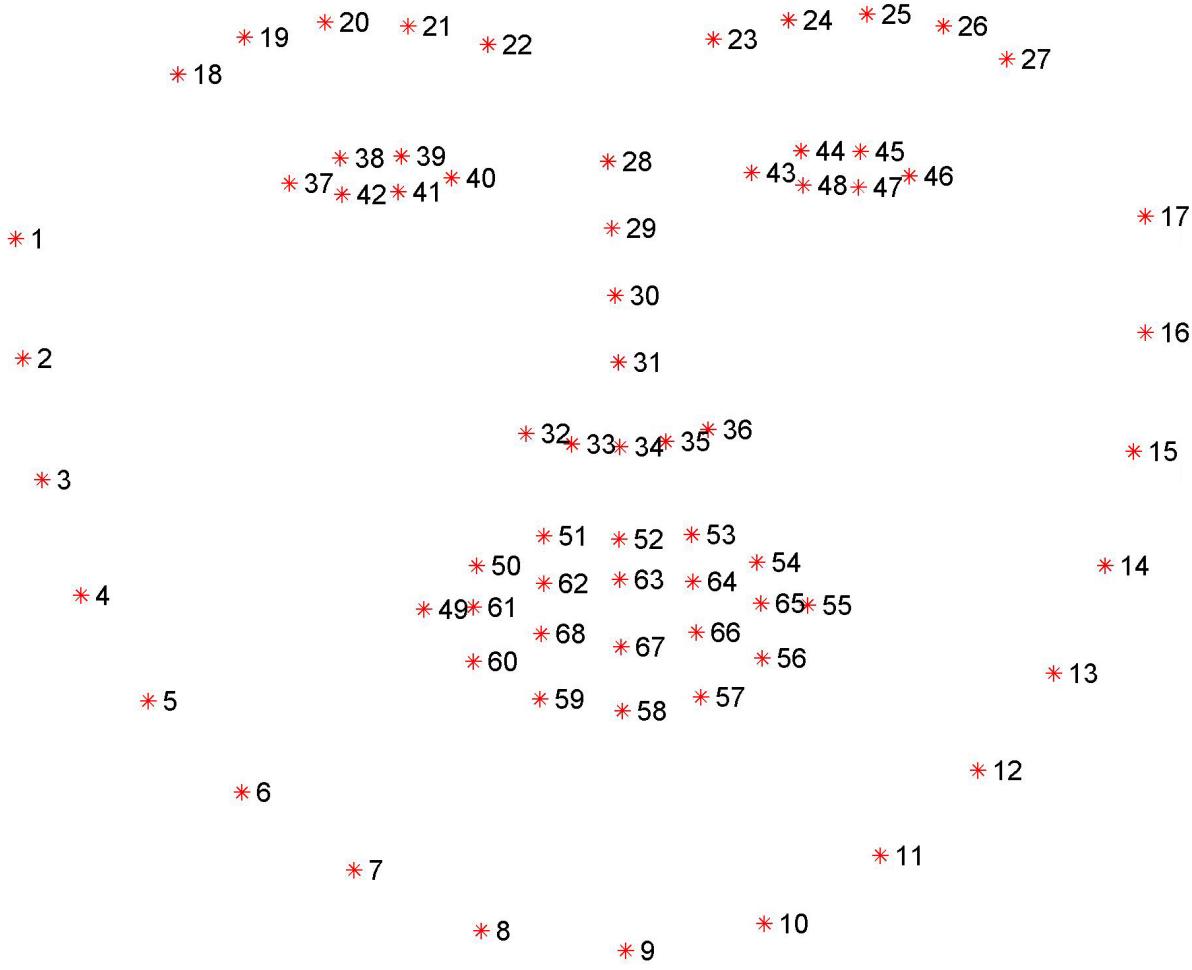

Figure 3.3: The traditional 68 point landmarking scheme adopted by many databases for evaluating facial alignment.

forward and backward pass through the 3D TPS warping module are shown below.

Forward Pass on TPS Parameters

A 3D TPS function is of the form

$$f_{\Delta_x}(x,y,z) = \begin{bmatrix} b_{1x} \\ b_{2x} \\ b_{3x} \\ b_{4x} \end{bmatrix}^T \begin{bmatrix} 1 \\ x \\ y \\ z \end{bmatrix} + \sum_{j=1}^{n} w_{jx} U\left(\left|\left(x_j, y_j, z_j\right) - (x,y,z)\right|\right) \tag{3.2}$$
$$s.t. \sum_{j=1}^{n} w_{jx} = 0,\ \sum_{j=1}^{n} w_{jx} x_j = 0,\ \sum_{j=1}^{n} w_{jx} y_j = 0,\ \sum_{j=1}^{n} w_{jx} z_j = 0$$

where b_{1x}, b_{2x}, b_{3x}, b_{4x}, and w_{jx} are the parameters of the function, $\mathbf{c}_j = (x_j, y_j, z_j)$ is the j^{th} control point used in determining the function parameters, and $U(r) = r^2 \log r$. The parameters of the TPS function in Eqn. 3.2 (b_{1x}, b_{2x}, b_{3x}, and all w_{jx}) are normally learned by setting up a system of linear equations using the known control points, $\mathbf{c}_j$ and the corresponding points in the warped 3D object. The function finds the change in a single coordinate, the change in the x-coordinate in the case of Eqn. 3.2. Similarly, one such function is created for each dimension, i.e. $f_{\Delta_x}(x,y,z)$, $f_{\Delta_y}(x,y,z)$, and $f_{\Delta_z}(x,y,z)$. The 3D TPS module would then take in the parameters for all three of these functions as input and output the newly transformed points on a 3D structure as

$$\mathbf{O} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} f_{\Delta_x}(x,y,z) \\ f_{\Delta_y}(x,y,z) \\ f_{\Delta_z}(x,y,z) \end{bmatrix} + \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{3.3}$$

This means that the 3D TPS module must have all of the 3D vertices of the generic model and the control points on the generic model as fixed parameters specified from the start. This will allow the module to warp the specified model by the warps specified by the TPS parameters. However, if we need to ensure that the w_j parameters for each function adhere

to the constraints of the TPS equation, the network cannot estimate the parameters directly as there would be no way to guarantee this. Instead, we can see that the parameters must lie in the null space of the matrix

$$\mathbf{P} = \begin{bmatrix} \mathbf{c}_1 & \mathbf{c}_2 & \cdots & \mathbf{c}_n \\ 1 & 2 & \cdots & 1 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \\ z_1 & z_2 & \cdots & z_n \\ 1 & 1 & \cdots & 1 \end{bmatrix} \tag{3.4}$$

Since all the $\mathbf{c}_i$ points are chosen on the generic model before training of the network, $\mathbf{P}$ is a fixed matrix. As long as we have at least 5 points, a null space will exist for this matrix and a basis for the null space, $\mathbf{B}$ such that $\mathcal{N}(\mathbf{P}) = span(\mathbf{B})$ can be found. Any solution for the w parameters must be some linear combination of this basis in order to lie in $\mathcal{N}(\mathbf{P})$. Therefore, we can have the network estimate a set of coefficients for this basis to generate the w parameters separately from the b parameters for each TPS function. In practice, however, it is simpler to relax the constraints of this problem and let the network directly estimate all the TPS parameters for the 3D modeling since we only use it to find an initialization for a set of 2D landmarks. The refinement process on the 2D landmarks breaks the relationship between the 3D and 2D points anyway, meaning a very accurate 3D shape is not needed in the intermediate steps. Once the final set of 2D landmarks are found, the 3D shape can be refined to fit these landmarks and the TPS constraints can be imposed as shown in Section 3.4.

Backpropagation on TPS Parameters

In order for end-to-end learning on the TPS parameters to be performed, the gradient of the loss of the network with respect to the input parameters must be computed. This will allow a gradient descent learning to be performed on this module. As is normal in

neural networks, backpropagation can be used, meaning only the gradient of the output with respect to the input parameters needs to be computed and the chain rule can be applied to compute the final gradient. Since each 3D vertex in the generic model will give one 3D vertex as an output, it is easier to compute the gradient on one of these points, $\mathbf{p}_i = (x_i, y_i, z_i)$, first. This can be shown to be

$$\frac{\delta \mathbf{O}}{\delta \boldsymbol{\theta}_{\Delta_x}} = \begin{bmatrix} 1 & 0 & 0 \\ x_i & 0 & 0 \\ y_i & 0 & 0 \\ z_i & 0 & 0 \\ U\left(|\mathbf{c}_1 - (x_i, y_i, z_i)\,|\right) & 0 & 0 \\ \vdots & \vdots & \vdots \\ U\left(|\mathbf{c}_n - (x_i, y_i, z_i)\,|\right) & 0 & 0 \end{bmatrix}^T \tag{3.5}$$

where $\boldsymbol{\theta}_{\Delta_x}$ are the parameters of f_{Δ_x}. Similarly, the gradients for $\boldsymbol{\theta}_{\Delta_y}$ and $\boldsymbol{\theta}_{\Delta_z}$ are the same with only the non-zeros values in either the second or third row, respectively. The final gradient of the loss with respect to the parameters can be computed as

$$\frac{\delta L}{\delta \boldsymbol{\theta}_{\Delta_x}} = \frac{\delta L}{\delta \mathbf{O}} \frac{\delta \mathbf{O}}{\delta \boldsymbol{\theta}_{\Delta_x}} \tag{3.6}$$

The gradient can be computed for every point and added together to get the final gradient for each set of parameters that can be used to update previous layers of the network.

3.2.2 Camera Projection Transformers

Once a 3D shape has been estimated for the input image, we need to determine how it should be projected to the image space in order to texture it appropriately. As stated at the beginning of this chapter, this relationship is modeled by the camera projection

equation, Eqn. 3.1. The details of the forward and backward pass through this portion of the network are detailed below. As with the TPS layer, there are no parameters to learn in this module but we have to pass gradients back to both the camera parameter localization network as well as the TPS module. Therefore, we need a backpropagation rule for the camera parameters, $\mathbf{M}$, and the 3D points $\mathbf{p}_w$.

Forward Pass on Camera Projection Parameters

Since Eqn. 3.1 is only defined up to scale, the final output of this module will have to divide out the scale factor. By first rewriting the camera projection matrix as

$$\mathbf{M} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ a_5 & a_6 & a_7 & a_8 \\ a_9 & a_{10} & a_{11} & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{m}_1^T \\ \mathbf{m}_2^T \\ \mathbf{m}_3^T \end{bmatrix} \tag{3.7}$$

where a_i is the i^{th} element of $\mathbf{a}$, the final output of the camera projection module can be written as

$$\mathbf{O} = \begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} \frac{\mathbf{m}_1^T \mathbf{p}_w}{\mathbf{m}_3^T \mathbf{p}_w} \\ \frac{\mathbf{m}_2^T \mathbf{p}_w}{\mathbf{m}_3^T \mathbf{p}_w} \end{bmatrix} \tag{3.8}$$

Backpropagation on Camera Projection Parameters

In order to perform backpropagation, the derivative of the projected coordinates with respect to $\mathbf{a}$ must be computed. The gradient with respect to each of the rows of $\mathbf{M}$ can

be shown to be

$$\frac{\delta \mathbf{O}}{\delta \mathbf{m}_1^T} = \begin{bmatrix} \frac{\mathbf{p}_w^T}{\mathbf{m}_3^T \mathbf{p}_w} \\ \mathbf{0} \end{bmatrix} \quad \frac{\delta \mathbf{O}}{\delta \mathbf{m}_2^T} = \begin{bmatrix} \mathbf{0} \\ \frac{\mathbf{p}_w^T}{\mathbf{m}_3^T \mathbf{p}_w} \end{bmatrix}$$
$$\frac{\delta \mathbf{O}}{\delta \mathbf{m}_3^T} = \begin{bmatrix} \frac{-\mathbf{p}_w^T \left(\mathbf{m}_1^T \mathbf{p}_w\right)}{\left(\mathbf{m}_3^T \mathbf{p}_w\right)^2} \\ \frac{-\mathbf{p}_w^T \left(\mathbf{m}_2^T \mathbf{p}_w\right)}{\left(\mathbf{m}_3^T \mathbf{p}_w\right)^2} \end{bmatrix} \tag{3.9}$$

Using the chain rule, the gradient of the loss of the network with respect to the input can be found as

$$\frac{\delta L}{\delta \mathbf{a}} = \begin{bmatrix} \left(\frac{\delta L}{\delta \mathbf{O}} \frac{\delta \mathbf{O}}{\delta \mathbf{m}_1^T}\right)^T \\ \left(\frac{\delta L}{\delta \mathbf{O}} \frac{\delta \mathbf{O}}{\delta \mathbf{m}_2^T}\right)^T \\ \left(\frac{\delta L}{\delta \mathbf{O}} \frac{\delta \mathbf{O}}{\delta \mathbf{m}_3^T}\right)^T \end{bmatrix} \tag{3.10}$$

Since $\mathbf{M}$ is only defined up to scale, the last element of $\mathbf{M}$ can be defined to be a constant which means that only the first 11 elements of this gradient are used to actually perform the backpropagation on $\mathbf{a}$. Since $\mathbf{M}$ relates many pairs of 2D and 3D points, the gradient is computed for every pair and added together to give the final gradient that is used for updating $\mathbf{a}$.

Backpropagation on 3D Coordinates

In order to make use of the TPS warped 3D points in the camera projection module of the transformer network, the module must take in as input the warped coordinates. This means that such a module would also have to do backpropagation on the 3D coordinates as well as the camera projection parameters. Taking the derivative of the output in Eqn. 3.8 with respect to the 3D point, $\mathbf{p}_w$ results in

$$\frac{\delta \mathbf{O}}{\delta \mathbf{p}_w} = \begin{bmatrix} \frac{\mathbf{m}_1^T}{\mathbf{m}_3^T \mathbf{p}_w} - \frac{\mathbf{m}_1^T \mathbf{p}_w}{\left(\mathbf{m}_3^T \mathbf{p}_w\right)^2} \mathbf{m}_3^T \\ \frac{\mathbf{m}_2^T}{\mathbf{m}_3^T \mathbf{p}_w} - \frac{\mathbf{m}_2^T \mathbf{p}_w}{\left(\mathbf{m}_3^T \mathbf{p}_w\right)^2} \mathbf{m}_3^T \end{bmatrix} \tag{3.11}$$

However, since $\mathbf{p}_w$ is in homogeneous coordinates and only the gradient with respect to the x, y, and z coordinates are needed, the actual gradient becomes

$$\frac{\delta \mathbf{O}}{\delta \mathbf{p}'_w} = \begin{bmatrix} \frac{\mathbf{m}_1'^T}{\mathbf{m}_3^T \mathbf{p}_w} - \frac{\mathbf{m}_1^T \mathbf{p}_w}{\left(\mathbf{m}_3^T \mathbf{p}_w\right)^2} \mathbf{m}_3'^T \\ \frac{\mathbf{m}_2'^T}{\mathbf{m}_3^T \mathbf{p}_w} - \frac{\mathbf{m}_2^T \mathbf{p}_w}{\left(\mathbf{m}_3^T \mathbf{p}_w\right)^2} \mathbf{m}_3'^T \end{bmatrix} \tag{3.12}$$

where

$$\mathbf{p}'_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} \quad \mathbf{m}'_i = \begin{bmatrix} m_{i_1} \\ m_{i_3} \\ m_{i_3} \end{bmatrix} \tag{3.13}$$

and m_{i_j} is the j^{th} element of $\mathbf{m}_i$. This gradient is computed for every 3D point independently and used in the chain rule to compute

$$\frac{\delta L}{\delta \mathbf{p}_w} = \frac{\delta L}{\delta \mathbf{O}} \frac{\delta \mathbf{O}}{\delta \mathbf{p}_w} \tag{3.14}$$

which can then be used to perform backpropagation on each $\mathbf{p}_w$.

3.3 2D Landmark Regression

After this initial alignment of the face to the image, we have a 3D model and associated 2D locations. However, we can see in Fig. 3.4 that the results are not as accurate as we would like. This is, most likely, due to the fact that the network is using information from the entire input image to determine both the TPS warping parameters and the camera projection parameters. This leads to a good initialization of the 2D alignment but the points are not specifically weighting any of the local information around the projected point. In many of the previous methods of landmarking, it has been shown that local information is very important for accurate alignment in 2D. In order to further improve the landmark accuracy, we extend our network with a landmark refinement stage. This stage

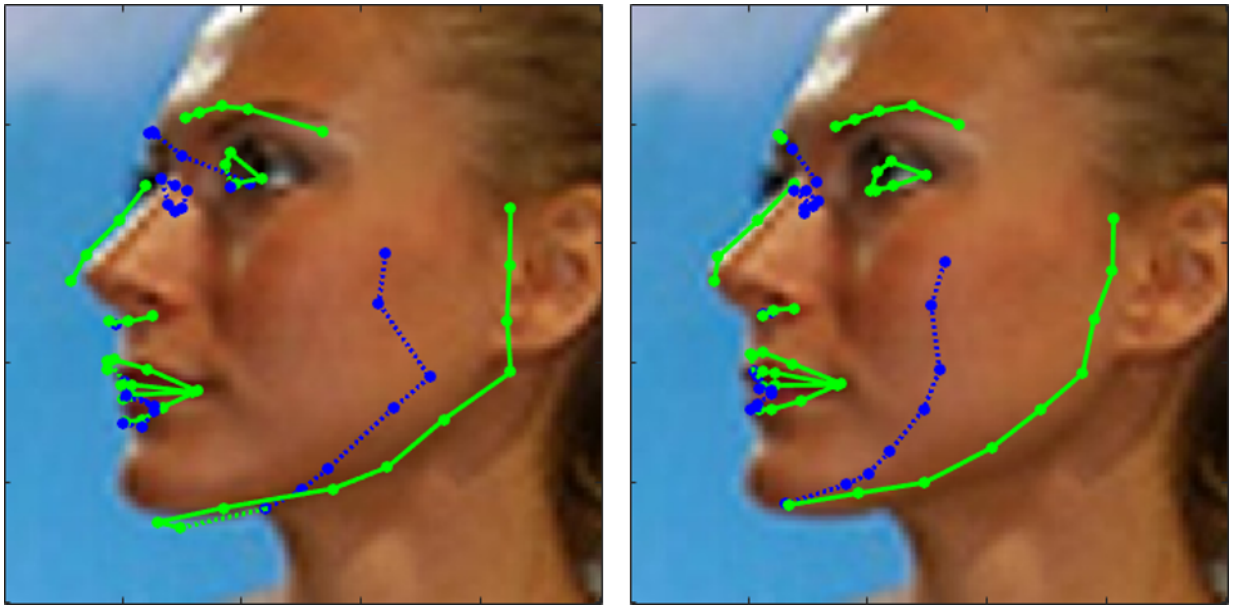

Figure 3.4: The landmarks from the scheme in Fig. 3.3 before regression (left) and after regression (right). Notice how the initial landmarks do not align very precisely to areas with highly descriptive local patches, such as the corners of the eye. After regression, these landmarks are much more accurately aligned. Landmarks determined to be self-occluded by the 3D geometry of the scene are in blue while visible landmarks are in green.

treats the projected 2D coordinates from the previous stage as initial points and estimates the offsets for each point. To extract the feature vector for each point, a 3×3 convolution layer is attached on top of the last convolution layer in the base model. This allows for a local feature to be extracted. However, a single 3×3 convolution layer is probably not complex enough to find good feature for landmark regression. Therefore, several 1×1 convolution layers are added afterwards for more complexity without increasing the spatial support of the feature, resulting in a feature map with D channels. Then each initial point is projected onto this feature map and its D-dimensional feature vector is extracted along the channel direction. The initial points are often not aligned with the grids on the feature map and so their feature vectors are sampled with a bilinear interpolation. This bilinear sampler is the same one used in the traditional SPT and detailed in [72]. Given the feature vector for each landmark, it goes through a fully-connected (FC) layer to output the offsets, i.e. δ_x and δ_y. Then the offsets are added to the coordinates of the initial

location. For each landmark we use an independent FC layer. We don't share the FC layer for all landmarks because each landmark should have a unique behavior of offsets. For example, the center of the eye may move left after regression whereas the corner of the eye may move right. Also, sometimes two initial landmarks may be projected to the same location due to a certain pose. We want them to move to different locations even when they have the same feature vector.

3.4 3D Model Regression From 2D Landmarks

Once the 2D landmark regression has been performed, we have a more accurate set of landmarks. For 2D facial alignment, this is enough. However, when dealing with 3D facial alignment or modeling, we need to have a relationship between the 3D and 2D spaces. Before the regression, this relationship was modeled by the camera projection parameters that we estimated in the network. After the regression, however, this relationship has been broken. We still have the initial 3D model that was generated as a result of the TPS warping inside the network. Using this and the refined 2D landmarks, we could re-estimate the camera projection parameters. Unfortunately, this is an over-constrained problem and we would have to settle for the closest solution. This would result in the 3D model projecting to potentially different locations than the landmarks specify meaning that any textures for the 3D model that we extract would be incorrect. Especially sensitive locations, such as the eyes, would be very noticeably wrong on a 3D model once it was rendered at different poses. This would cause many problems for any tasks relying on a good 3D model for later tasks.

Instead, we can use the already estimated camera projection parameters and refine the 3D model to fit the new 2D landmarks. Since we know the projective relationship between 3D and 2D points, we can find the ray that goes through the 2D landmark into 3D space as shown in Fig. 3.5. The details of how to find this ray are given in Appendix

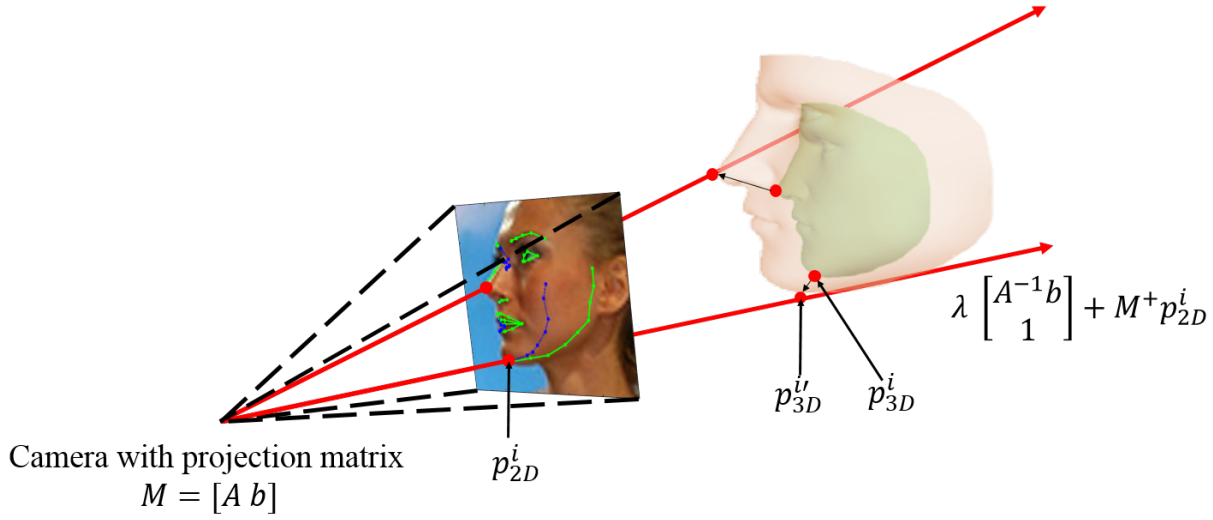

Figure 3.5: Backprojection of rays through image landmarks. The closest points are found for each ray-landmark pair to use as new 3D coordinates for the face model. The original model (green) is warped to fit the new landmarks with a 3D TPS warp resulting in a new face model (red).

A.2. We know that the corresponding 3D landmark must lie somewhere on this ray as they are the only points in the 3D space that project to the correct 2D point. In order to change the 3D model as little as possible, we select a new 3D point for each landmark by selecting the closest point on the corresponding ray as shown in Appendix A.2. Once these new 3D points have been selected, another TPS warp can be estimated in the fashion shown in Appendix B and used to generate a new model that fits the projected landmarks. After this warping, the landmark points on the model will project to exactly the regressed 2D landmarks, recovering the mapping between the 3D model and the 2D image. These resulting 3D models recover the 3D shape of the subject more accurately since they are based on accurate landmarks in the image. Fig. 3.6 shows a few examples of the recovered 3D shape rotated to the viewpoint of the image. It can be noted that each model is actually a different shape and not just the original mean shape. In addition, with the knowledge of the camera center and the 3D structure of the face, each landmark can be marked as visible of self-occluded by checking if the ray between the camera center and the 3D point intersects any other part of the 3D structure.

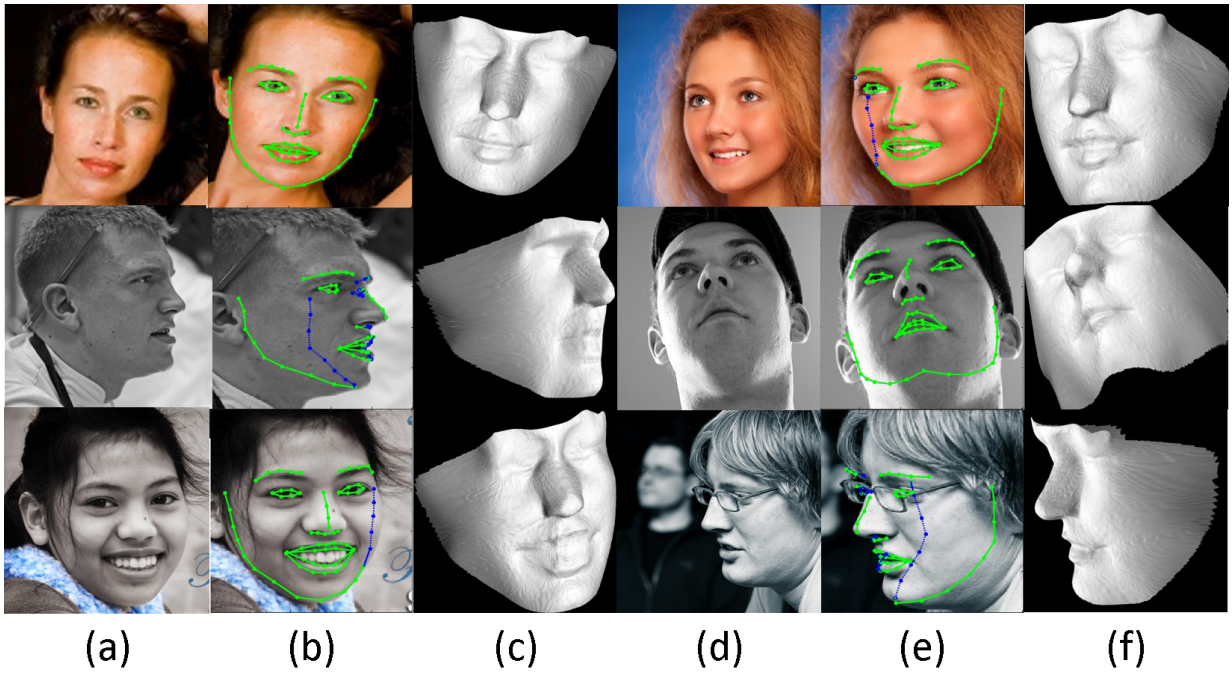

Figure 3.6: (a & d): Images in the wild from the AFLW dataset. (b & e): 3D landmarks (green: visible, blue: occluded) estimated from input image. (c & f): 3D model generated from input image. **Best viewed in color.**

Once the full 3D model is created from the input image, it can be rendered from any desired viewpoint as seen in Fig. 3.7. This kind of synthesis could potentially be used to augment face recognition training datasets so that models trained on the data would see the same subjects at every pose and the pose distribution of the data would be uniform. However, since the faces start to appear artificial at the extreme poses, mostly due to the fact that this particular model does not include things like the ears and neck, this could lead to poor performance in the recognition models. Additionally, this would require that a very controlled frontal image be present in the database to ensure there are no missing regions that have to be rendered at the more extreme poses. For these reasons, we focus on using these 3D models to correct a face back to a frontal viewpoint instead in Chapter 4.

Figure 3.7: A 3D model created from a frontal image of subject 1 from the MPIE dataset and rendered at many different viewpoints. The pitch ranges from -30° to 30° in 10° increments and the yaw ranges from -90° to 90° in 15° increments.

3.5 Facial Landmarking Experiments

Qualitatively, we can see that this landmarking method is quite capable and gives us a very good estimate of the non-visible landmarks as seen in Figs. 3.6 and 3.8. Some ablation experiments have been performed in order to evaluate the importance of the architecture of the feature extraction network and the importance of the landmark regression step. Unfortunately, there is no dataset that contains ground truth 3D models in real world images so the only way to evaluate the alignment method is to evaluate it on a sparse set of landmark points. However, most 2D landmarking datasets suffer from the problem of semantically inconsistent landmark points as shown in Fig. 1.4. Therefore, we have to evaluate only on the visible landmarks or on synthetic 3D landmarks.

Figure 3.8: (a & d): Images in the wild from the AFLW dataset with 3D landmarks (green: visible, blue: occluded) estimated from input image. Note how even under occlusions, large poses, and expressions, the landmarks are still quite reasonable.. **Best viewed in color.**

3.5.1 Datasets

300W-LP: The 300W-LP [41] dataset contains 122,450 synthetically generated views of faces from the AFW [26], LFPW [73], HELEN [74], and IBUG [75] datasets as seen in Fig. 2.1. These images not only contain rotated faces but also attempt to move the background

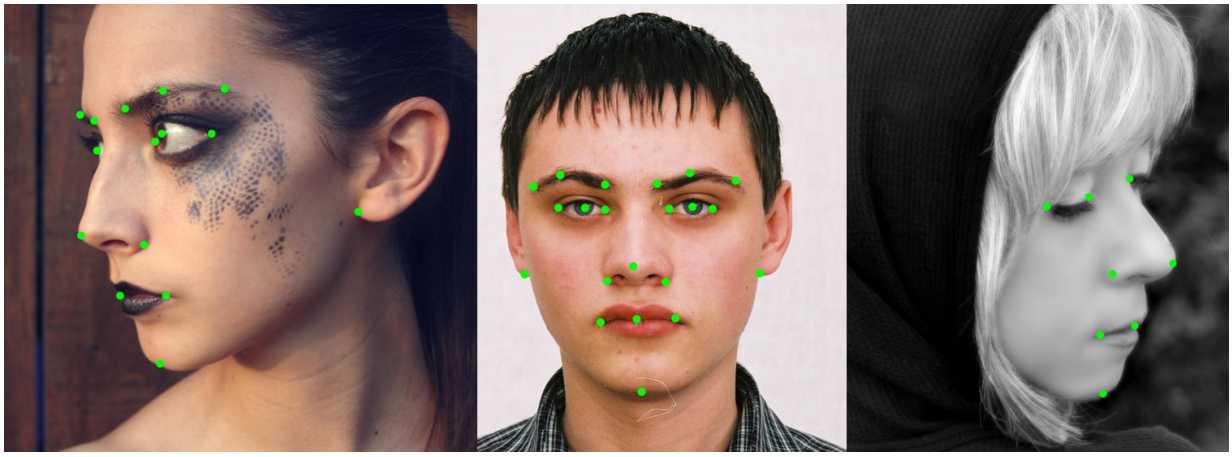

Figure 3.9: Examples of the AFLW ground truth landmarks. Note that only some landmarks are given on any particular image as non-visible landmarks were not clicked.

in a convincing fashion, making it a very useful dataset for training 3D approaches to work on real world images. This dataset is used to train the alignment network on the euclidean loss of both the 3D coordinates of the model's vertexes and the 2D projections of the model. The regression component is trained on the sparse set of 68 landmarks as defined in the MPIE [19, 20] dataset shown in Fig. 3.3.

AFLW: The Annotated Facial Landmarks in the Wild (AFLW) dataset [76] is a relatively large dataset for evaluating facial alignment on wild images. It contains approximately 25,000 faces annotated with 21 landmarks with visibility labels. Landmarks that are not visible in the image are not given but are not used in evaluation of the accuracy on that image. As a result, this dataset allows for both 2D landmarking and 3D landmarking methods to be evaluated against each other. A few examples of these images can be seen in Fig. 3.9. The dataset provides pose estimates so results are grouped into three different pose ranges, $[0°, 30°]$, $(30°, 60°]$, and $(60°, 90°]$. Due to the inconsistency in the bounding boxes in the AFLW dataset, we adopt the use of a face detector first to normalize the scale of the faces. The Multiple Scale Faster Region-based CNN approach [52] has shown good results and at a fast speed. We use the recent extension to this work, the Contextual Multi-Scale Region-based CNN (CMS-RCNN) approach [10] to perform the face detec-

Figure 3.10: The fake good alignment problem where clearly wrong 2D alignment (right) gives low error on the visible landmarks (left).

tion in any experiment where face detection is needed. The CMS-RCNN approach detects 98.8% (13,865), 95.9% (5,710), and 86.5% (3,830) of the faces in the $[0°, 30°]$, $(30°, 60°]$, and $(60°, 90°]$ pose ranges respectively.

AFLW2000-3D: Zhu *et al.* [41] accurately pointed out how merely evaluating an alignment scheme on the visible landmarks in a dataset can result in artificially low errors. Fig. 3.10 shows how a clearly wrong a 2D landmarking can be but still achieve low error on the visible landmarks in the AFLW dataset. This is known as the fake good alignment problem where the fact that there is a landmark visibility given allows for misleading results. Therefore, a true evaluation of any 3D alignment method must also evaluate alignment on the non-visible landmarks as well. The AFLW2000-3D dataset contains the first 2000 images of the AFLW dataset but with all 68 points defined by the scheme in the CMU MPIE dataset [19, 20]. These points were found by aligning the Basel Face Model to the images. This allows for the non-visible landmarks to have a ground truth location as seen in Fig. 3.11. While this is a synthetic dataset, meaning the true location of the non-visible landmarks is not known, it is the best one can do when dealing with real images. As these images are from the AFLW dataset, they are also grouped into the same

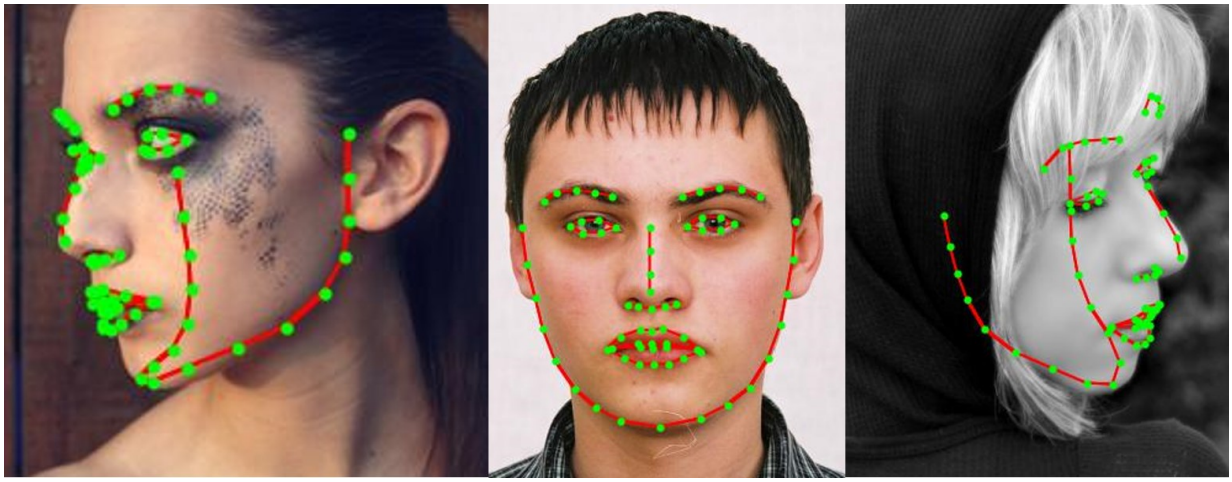

Figure 3.11: Examples of the AFLW2000-3D ground truth landmarks. Every image has all 68 landmarks given allowing for an evaluation of a 3D landmarking method.

Table 3.1: NME for both the AlexNet (AN) and VGG-16 (VGG) models. (LR: landmark regression)

	AFLW Dataset (21 pts)						
	$[0, 30]$	$(30, 60]$	$(60, 90]$	mean	std		FPS
AN	4.88	5.55	7.10	5.84	1.14		≈ 213
AN+LR	4.00	4.48	5.89	4.79	0.98		≈ 166
VGG	4.15	4.64	5.96	4.92	0.94		≈ 56
VGG+LR	3.46	3.78	4.77	4.00	0.69		≈ 47

pose ranges.

3.5.2 Ablation Experiments

To investigate the effect of each component in our network, we conduct two ablation studies. All the models in these experiments are trained on the same 300W_LP dataset and tested on the detected images in AFLW. We first test the effect of the different pre-trained models. We fine-tune our network from the AlexNet and VGG-16 models pre-trained on the ImageNet dataset and evaluate the landmark accuracy before the regression step. The VGG-16 model outperforms the AlexNet model in all three pose ranges on the AFLW detected set as shown in Table 3.1. This seems to indicate that a good base model is

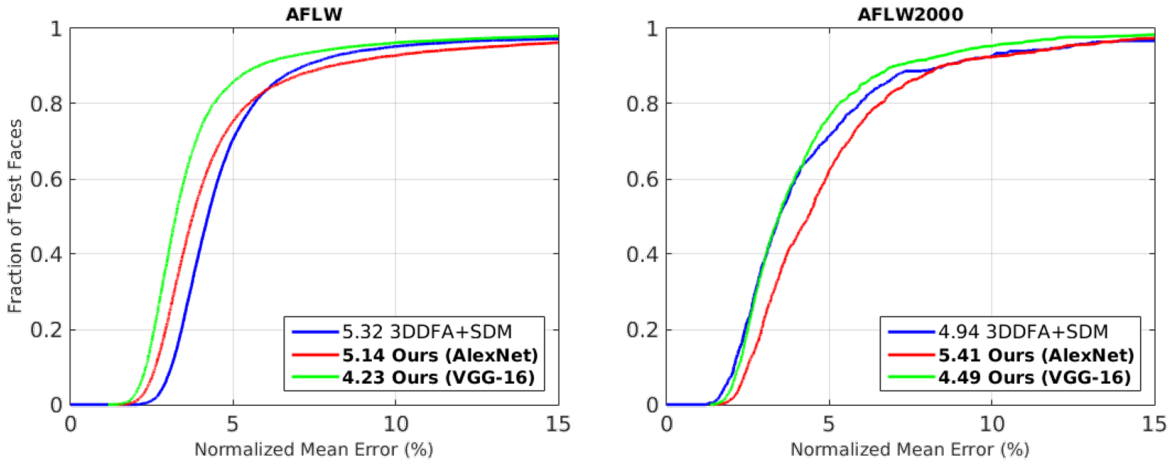

Figure 3.12: Cumulative Error Distribution (CED) curves for both the AlexNet (red) and VGG-16 (green) architectures on both the AFLW (left) and AFLW2000-3D (right) dataset. To balance the distributions, we randomly sample 13,209 faces from AFLW and 915 faces from AFLW2000-3D, split evenly among the 3 categories, and compute the CED curve. This is done 10 times and the average of the resulting CED curves are reported. The mean NME% for each architecture from Table 3.3 is also reported in the legend. The CED curve for 3DDFA+SDM was obtained from the authors of [41].

important for the parameter estimation portion of the network. Second, we evaluate the effect of landmark regression stage. The Normalized Mean Error (NME) is computed by averaging the Euclidean error of the regressed and projected landmarks and normalizing it by the square root of the bounding box size (h x w) provided in the dataset. Table 3.1 shows that the landmark regression step greatly helps to improve the accuracy.

3.5.3 Comparison Experiments

AFLW: Since the CMS-RCNN approach may only detect the easier to landmark faces, we use the provided bounding box anytime the face is not detected by the detector. Due to the inconsistency between the two bounding box schemes, faces are not always normalized properly. However, we feel this is the only way to get a fair comparison to other methods without artificially making the dataset easier by only evaluating on detected faces. We compare against baseline methods used by [41] on the same dataset, namely Cascaded Deformable Shape Models (CDM) [77], Robust Cascaded Pose Regression (RCPR) [78],

Explicit Shape Regression (ESR) [79], SDM [27] and 3DDFA [41]. All methods except for CDM were retrained on the 300W-LP dataset. The NME in these experiments is computed by averaging the error of the visible landmarks and normalizing it by the square root of the bounding box size (h x w) provided in the dataset. Table 3.3 clearly shows that our model using the VGG-16 architecture has achieved better accuracy in all pose ranges, especially the $(60°, 90°]$ category, and has achieved a smaller standard deviation in the error. This means that not only are the landmarks more accurate, they are more consistent than the other methods.

AFLW2000-3D: The baseline methods were evaluated on the AFLW2000-3D dataset using the bounding box of the 68 landmarks. In order to perform a fair comparison, we retrained our models using the same bounding box on the training data since defining any set of landmarks on the 300W-LP dataset is trivial due to the 3D models. The NME is again computed using the bounding box size. Here we see that though 3DDFA+SDM performs well, the VGG-16 architecture of our model still performs the best in both the $[0°, 30°]$ and $(60°, 90°]$ ranges. While the VGG-16 model is only second best in the $(30°, 60°]$ range, it is not a significant difference and the improvement in $(60°, 90°]$ means that, once again, our method not only generates more accurate, but more consistent landmarks, even in a 3D sense. Cumulative Error Distribution (CED) curves are reported for both architectures on both datasets in Fig. 3.12.

3.6 Pose Estimation Experiments

While this method was not created specifically to estimate the pose of the face in each image, it is a natural byproduct of the way it was designed. By estimating the camera projection matrix, $\mathbf{M}$, and the camera center as a result, we can find the pitch and yaw angles between the center of the 3D model and the camera as shown in Fig. 3.13. We do not estimate the roll angle in this fashion as it can be normalized out by the eye coordinates if

Table 3.2: The NME of face alignment results on AFLW. The best two numbers in each category are shown in bold.

	AFLW Dataset (21 pts)				
Method	$[0, 30]$	$(30, 60]$	$(60, 90]$	mean	std
CDM	8.15	13.02	16.17	12.44	4.04
RCPR	5.43	6.58	11.53	7.85	3.24
ESR	5.66	7.12	11.94	8.24	3.29
SDM	4.75	5.55	9.34	6.55	2.45
3DDFA	5.00	5.06	6.74	5.60	0.99
3DDFA+SDM	4.75	4.83	**6.38**	5.32	**0.92**
Ours (AlexNet)	**4.11**	**4.69**	6.61	**5.14**	1.31
Ours (VGG-16)	**3.55**	**3.92**	**5.21**	**4.23**	**0.87**

Table 3.3: The NME of face alignment results on AFLW2000-3D. The best two numbers in each category are shown in bold.

	AFLW 2000-3D Dataset (68 pts)				
Method	$[0, 30]$	$(30, 60]$	$(60, 90]$	mean	std
RCPR	4.26	5.96	13.18	7.80	4.74
ESR	4.60	6.70	12.67	7.99	4.19
SDM	3.67	4.94	9.76	6.12	3.21
3DDFA	3.78	4.54	7.93	5.42	2.21
3DDFA+SDM	**3.43**	**4.24**	**7.17**	**4.94**	1.97
Ours (AlexNet)	3.71	5.33	7.19	5.41	**1.74**
Ours (VGG-16)	**3.15**	**4.33**	**5.98**	**4.49**	**1.42**

needed. Since the goal of this 3D alignment and modeling is to help create a pose invariant face recognition system, it is important to see how accurate the pose estimation portion of this model is. To this end, we run a few simple experiments to verify that the pose estimate is reasonable. We do not need to have an extremely accurate pose estimator as we just need a rough idea of the pose of the face to select which half to keep in recognition experiments. As the VGG architecture gave the most accurate landmarking results, this is the model we use for all face recognition experiments. Therefore, we only use this model in the pose estimation experiments.

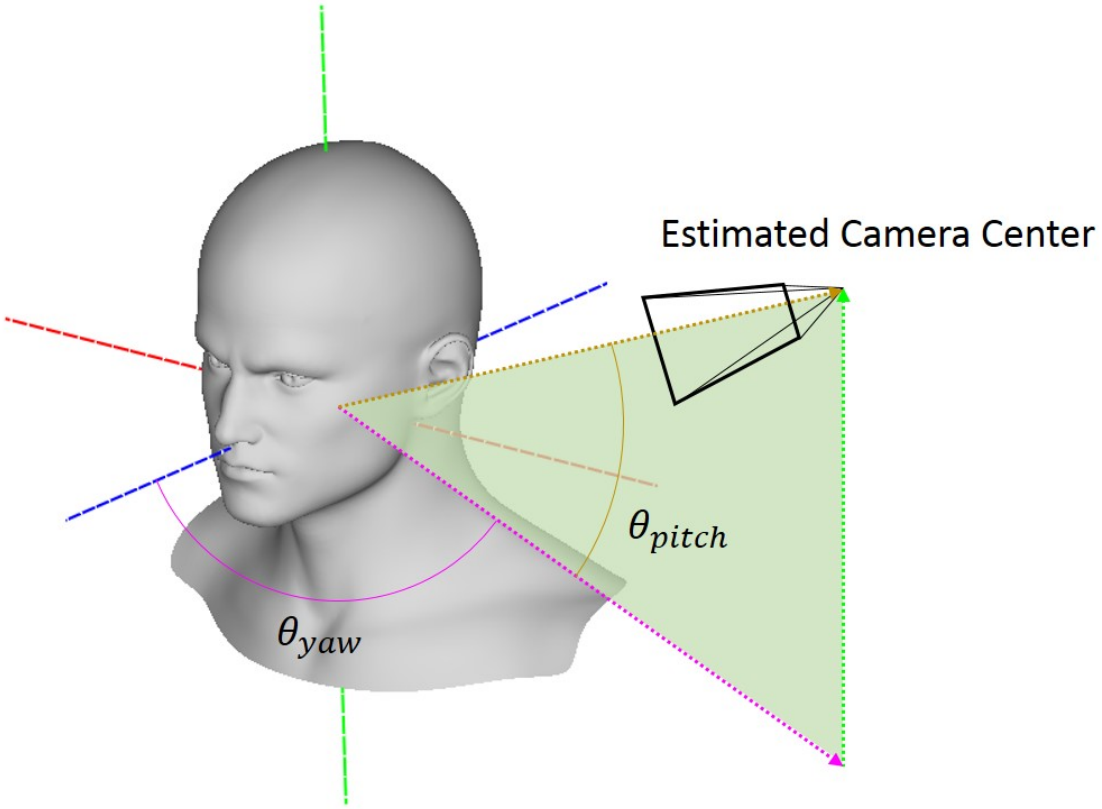

Figure 3.13: An illustration of the pitch and yaw angles in relation to the center of the 3D model and the center of the camera.

3.6.1 Datasets

MPIE: The MPIE [19, 20] dataset contains faces from 337 subjects over 4 sessions captured from $-90°$ to $90°$ in $15°$ increments. The illumination also varies in the same fashion as the pose and several expressions are captured as well. In total, there are 754,198 images in this dataset. For our analysis, we separate the data into three subsets that are used in both the pose estimation experiments here and the face recognition experiments found in Chapter 4. The first (MPIE-1) contains all the subjects from Session 1 (249 in total) at all poses, under neutral illumination and expression. This set is created to see the effect of pure pose variation on the pose estimate. However, pose variation will rarely happen on its own so we also use two other subsets, following a similar protocol to Peng *et al.* [67].

Figure 3.14: One subject from the FacePix dataset at $\{-90°, -60°, -30°, 0°, 30°, 60°, 90°\}$. The dataset contains images at all 1° increments between these extremes.

In this work, only the last 108 subjects are used in the test set as Peng *et al.* use the first 229 subjects for training their models. The second (MPIE-2) contains the last 108 subjects from the dataset over all 4 sessions under all pose variations and expressions. The third set (MPIE-3) contains the same 108 subjects under all pose, illumination, and expression variations. In the first two sets, the face detection is run as part of the landmarking and pose estimation. However, with MPIE-3, the harsh illumination changes causes the face detector to fail often enough that it may alter the statistics of the results. Therefore, to evaluate only the pose estimation portion of the system on illumination, we transfer the face detections from the neutral illumination images at each pose and expression to the others. The CMS-RCNN detector detects all of these faces without a problem. With all three of these sets, we aim to show that our method gives a decent pose estimate over many variations.

FacePix: The CUbiC FacePix dataset [80, 81] is another highly controlled pose estimation dataset. This dataset contains images of 30 subjects with images taken from $-90°$ to $90°$ in 1° increments. These images are very clean and controlled images with no change in illumination or expression as can be seen in Fig. 3.14.

Pointing'04: The Pointing'04 dataset [82] is the last pose estimation dataset we use to evaluate our model. This dataset contains both pitch and yaw variation simultaneously as seen in Fig. 3.15. The subjects were asked to look at markers at specified positions on the wall before each image capture. Unfortunately, this means the ground truth pose estimate is not as well controlled as the other datasets as people do not move their heads in the same way. Additionally, the extreme angles of $\pm 75°$ and $\pm 90°$

have the problem that it is very difficult for people to move their heads at these angles. There are 13 possible yaw angles from $-90°$ to $+90°$ at $15°$ intervals and 9 pitch angles $\{-90°, -60°, -30°, -15°, 0°, 15°, 30°, 60° + 90°\}$. In this dataset, however, not all possible combinations of yaw and pitch exist giving a total of 93 poses per subject. Each image for each subject was captured twice, once in each of two sessions.

3.6.2 Results

All three MPIE sets were run through the pose estimation and the mean absolute error (MAE) for each pose was computed over all the images in the set. The resulting MAEs are shown in Table 3.4. The resulting pose estimates seem to be quite accurate all the way out to $\pm 75°$. There is a significant increase in the error at the extremes of $\pm 90°$ but the error is still within reason. It seems the network is biased away from those extreme yaw angles. This is, perhaps, because the data only goes out to $\pm 90°$ so it has seen no data further past those angles whereas for any other angle, it has seen examples of faces at smaller and larger angles. Perhaps synthesizing images at even larger angles would help alleviate some of this error but synthesizing faces at $\pm 90°$ already creates unrealistic looking images and going further would only exacerbate the issue. Either way, it seems that the pose estimate is not dramatically affected by changes in expression as rows 1 and 2 show similar results. However, it does seem that illumination plays a much larger role in the pose estimation accuracy, especially at the more extreme poses. This is most likely due to the fact that expression changes only affect a small portion of the face whereas illumination changes affect the entire image. Since the pose estimate is computed from the camera projection parameters and those are estimated based on more global features, it makes sense that pose estimation would be more robust to expression than illumination. For the FacePix dataset, we had to group the very dense images into pose groupings to be able to compare it to the other datasets. We assigned every image to the one of the angles from the same

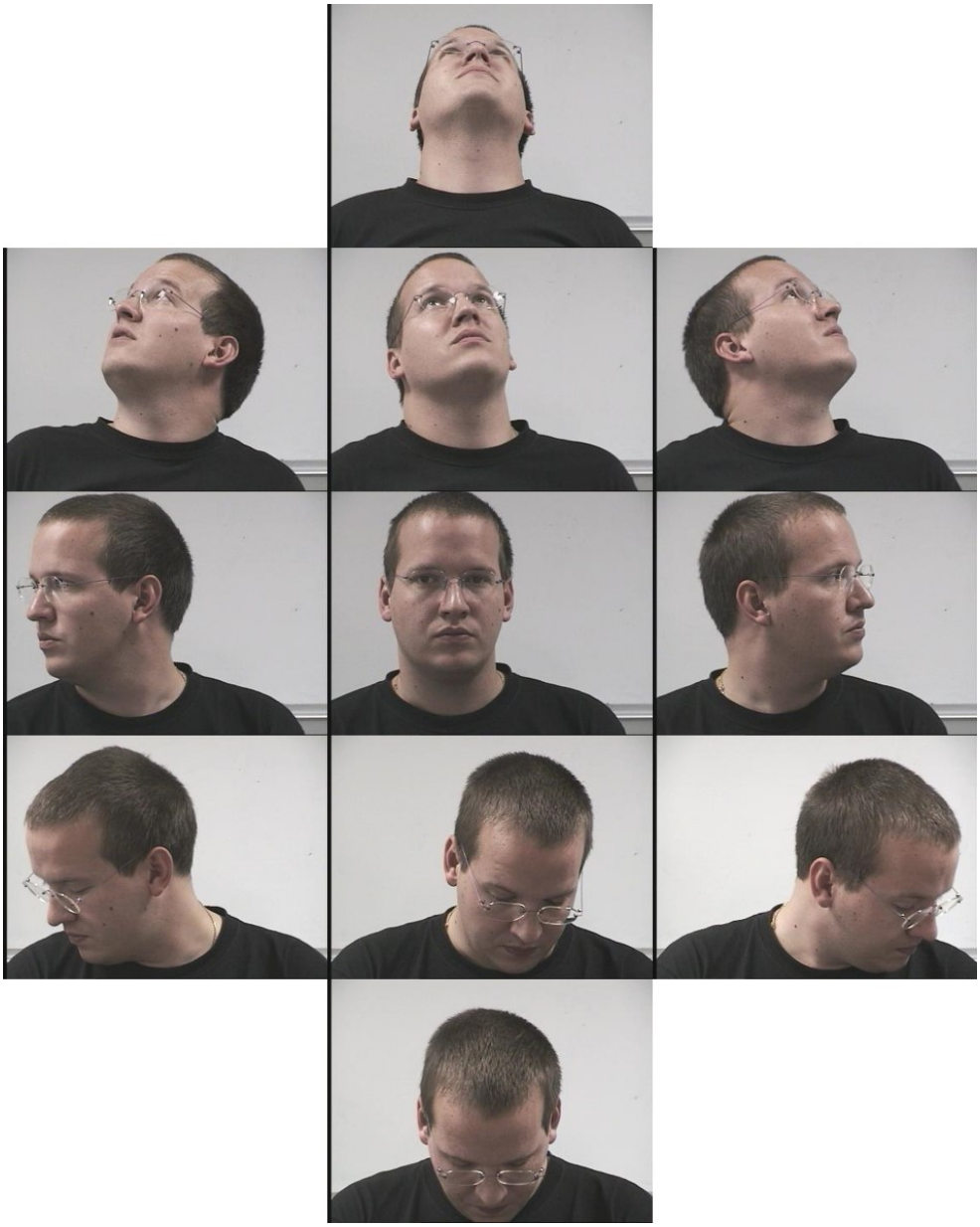

Figure 3.15: One subject from the Pointing'04 dataset. There are simultaneous changes in pitch and yaw for each subject but the face position is not well controlled. Additionally, not all combinations of pitch and yaw exist in the data. For instance, $\pm 90^\circ$ pitch variation only occurs with 0° yaw.

set that appear in the MPIE and Pointing'04 datasets. We assigned each image to the yaw angle it was closest to but used th actual angle to compute the error. The results show a similar trend to the other datasets though the errors are lower at the extremes indicating there is a smaller range of poses at which the error is large.

Table 3.4: MAE for pose estimation on various datasets

Dataset	-90°	-75°	-60°	-45°	-30°	-15°	0°	15°	30°	45°	60°	75°	90°
MPIE-1	15.8	6.8	4.1	3.4	2.8	3.0	2.5	2.6	2.9	4.1	3.9	3.6	9.4
MPIE-2	18.6	7.4	4.8	3.8	3.0	3.4	2.9	3.0	4.5	6.2	5.5	4.2	10.9
MPIE-3	22.8	9.6	7.0	5.3	3.5	3.4	3.2	3.4	4.8	6.9	7.7	10.2	17.5
FacePix	11.4	4.6	4.5	4.2	3.6	3.1	2.9	3.5	3.7	5.5	5.9	4.2	6.1
Pntg'04	22.0	13.3	8.4	6.0	5.9	5.3	4.1	4.7	5.4	6.7	7.4	13.5	18.9

On the Pointing'04 dataset, we see that when we look only at the yaw variation, we see a similar trend to the other datasets though larger errors, especially at the extremes. This can be explained by a mismatch between the pose in the image and the ground truth pose provided as can be seen in Fig. 3.16. Since the Pointing'04 dataset also includes yaw variation, we compute the error on the pitch and yaw estimate at all provided angles. As shown in Fig. 3.17, the yaw estimates tend to be decently accurate even as the pitch changes within reason. However, the pitch estimate seems to be much more sensitive to the combination of angles. It seems to only be able to give a decent pitch estimate on near frontal faces and only out to about $\pm 30°$. This is most likely due to the fact that the training data used, the 300W-LP dataset, only has synthetic yaw variation. Therefore any pitch variation seen is only that naturally found in the source images. As these source images were from older landmarking datasets, they do not contain extreme pitch variation since more focus has always been on yaw variation. This does expose the fact that, perhaps, more data is need to make this landmarking and modeling robust to the extreme pitch and yaw combinations. However, these are unlikely to occur in real world face recognition scenarios as this would mean the subject is directly under or above the camera.

3.7 Running Speed

Since facial alignment is usually a preprocessing step to other, more complex algorithms, it is important that any alignment method be able to run quickly. This is especially important

Figure 3.16: The $\pm 90^\circ$ pose images from the Pointing'04 dataset (top) and the $\pm 90^\circ$ pose images from the FacePix dataset. Notice how in the FacePix dataset, where the camera moved to a specified $\pm 90^\circ$ mark, the other eye and eyebrow completely disappear while in the Pointing'04 dataset, they are still visible. This indicates the ground truth pose given is not entirely accurate.

with the current trend towards larger and larger datasets being used in applications such as face recognition. In order to evaluate the speed of our method, we evaluate the models on a random subset of 1200 faces from the AFLW subset split evenly into the $[0^\circ, 30^\circ]$, $(30^\circ, 60^\circ]$, and $(60^\circ, 90^\circ]$ pose ranges. The models were evaluated on a 3.40 GHz Intel Core

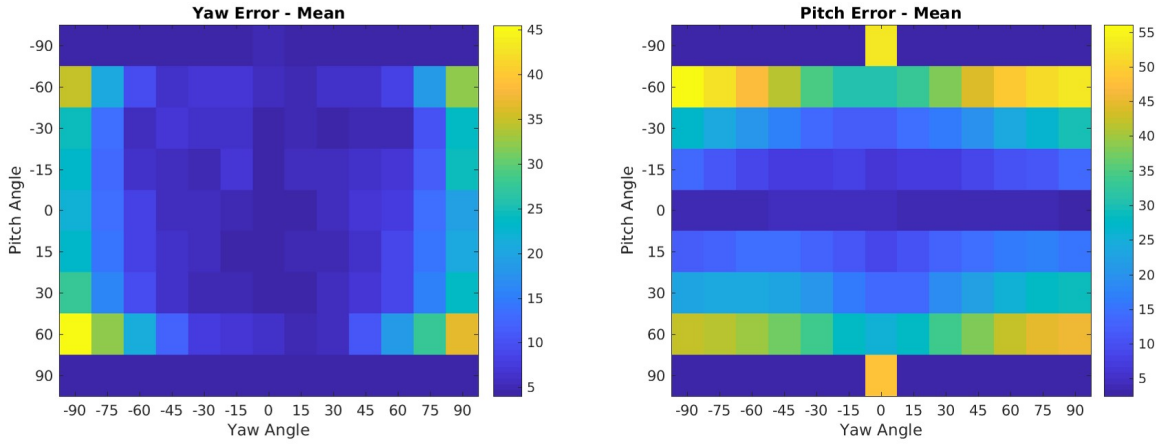

Figure 3.17: The errors in yaw (left) and pitch (right) at each yaw and pitch combination in the Pointing'04 dataset. Note that there is no data for ±90° pitch except at 0° yaw, hence the lack of results in the top and bottom rows.

i7-6700 CPU and an NVIDIA GeForce GTX TITAN X GPU. Our AlexNet trained model takes a total of 7.064 seconds to landmark the 1200 faces for an average of 0.0059 seconds per image or approximately 170 faces per second. The deeper and more accurate VGG-16 model landmarks the 1200 faces in 22.765 seconds for an average of 0.0190 seconds or approximately 52 faces per second.

Chapter 4

Pose Invariant Face Recognition

In previous methods of accounting for pose in a face recognition scenario, such as those outlined in [65] and [68], a set of synthesized face images were used in either training or testing the network. However, the underlying problem of a pose varying signal still existed. When synthesizing faces from different angles, the hope was that training a method on all angles of the same subject would allow for better accuracy. However, this did not actually normalize out the pose of the face or take advantage of the explicit knowledge of the 3D structure of the face. Instead, everything was treated as a 2D structure and the model was left to try and determine how best to use these inputs. With some knowledge of the 3D structure of the face, even the estimated structure from the technique outlined in Chapter 3, the face can be rendered from a frontal viewpoint, thus removing the pose as a factor in training and testing.

4.1 Generating a Pose Invariant Face

As we saw in Fig. 1.6, merely training on biased datasets and testing on large pose variations can cause a large drop in accuracy. Unfortunately, in order to train on unbiased datasets, we either have to restrict ourselves to much smaller datasets, which causes prob-

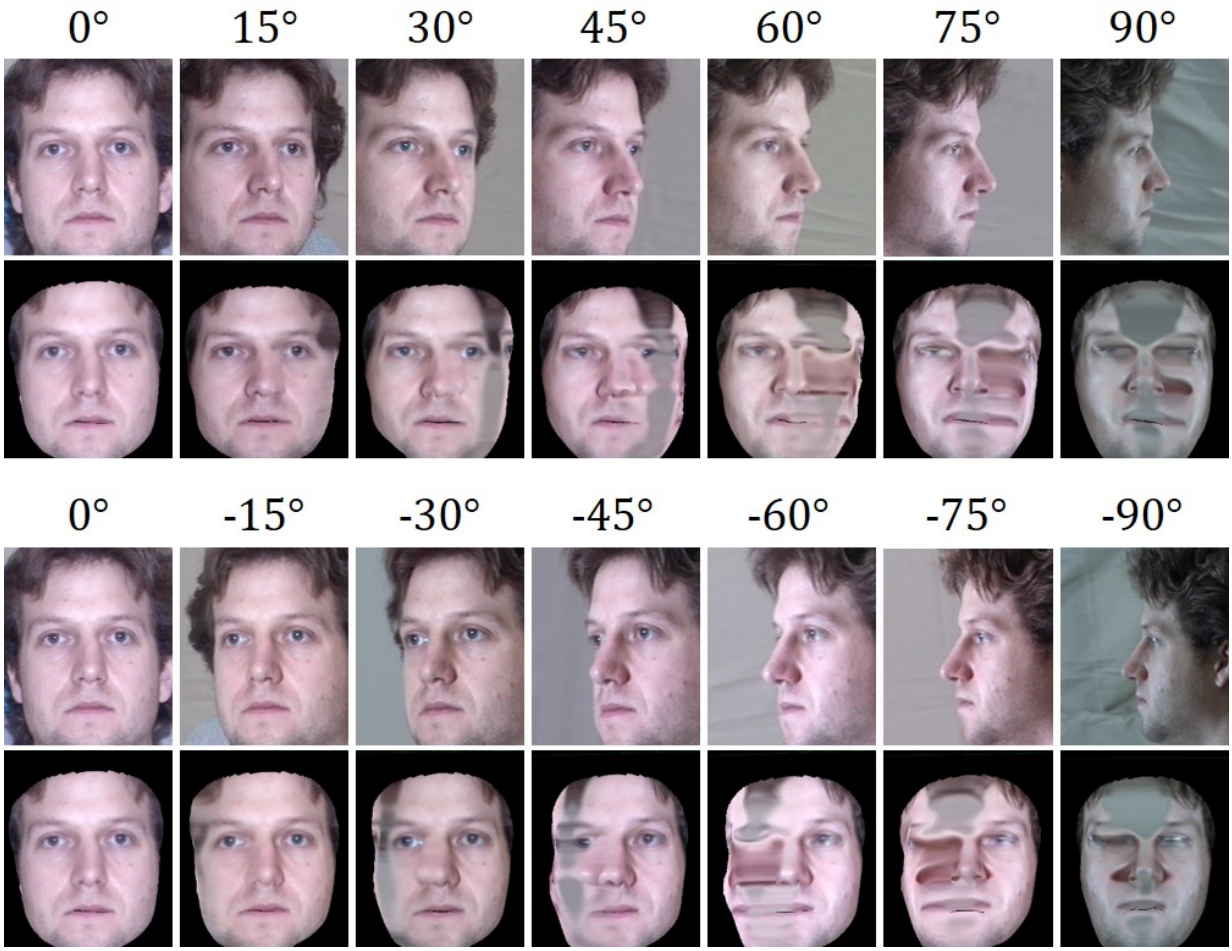

Figure 4.1: An example of how sampling the original image (1^{st} and 3^{rd} rows) from the location of the 2D projections of the 3D shape in each image leads to worse and worse artifacts (2^{nd} and 4^{th} rows) as the pose increases.

lems in training deep networks, or synthetically generate the missing data to fill in the rest of the pose distribution. Of course, however the synthetic data is generated, there will be some bias to the method and will, most likely, result in unrealistic images at extreme poses. Additionally, the large training datasets are not entirely frontal and therefore, the faces will have to be both synthesized at new poses to generate the larger angles and corrected to frontal to generate the smaller angles. When correcting faces to a more frontal angle, there will necessarily be parts of the face that are missing in the original image. If the texture for these regions is simply sampled from the image, the resulting faces show a lot of stretching and other artifacts as can be clearly seen in Fig. 4.1. In order to remove

such artifacts from the rendered images, it is necessary to determine which vertexes are occluded by the 3D structure of the face. Since we know the camera center, we can test if there is an intersection between the 3D model and the line segment between the camera center and any given vertex on the model. If the line segment intersects the 3D model somewhere other than at the vertex itself, that point would be occluded in the original image view. Appendix A contains details on two methods for determining self-occlusions which trade-off between accuracy on the self-occlusion labels and computation time. Once the self-occlusions are determined, they can be textured with any value, usually zero to make those regions black. Once these regions are removed, the resulting frontalized faces, shown in Fig. 4.2 look much more realistic but now have a black, missing region. These regions would normally have to be handled somehow, either by a preprocessing step to fill them in or by hoping the model can account for the missing regions in its feature extraction. However, when the poses are split into the positive and negative angles, it becomes clear that the missing regions only appear on one half of the image. In fact, if we compute the standard deviation of the images with the missing regions, seen in Fig. 4.3, it becomes very clear that a specific half of the face should be used at positive angles versus negative angles. In fact, these standard deviation images indicate that there is a larger variance in the forehead region than the rest of the face on the halves we want to keep. This seems to be because at $\pm 90^{\circ}$, large sections of the model are sampling from relatively small regions in the image. As there are no control points on the forehead and they are the furthest region from any defined landmarks, the TPS modeling has the hardest time in this region. As the images in Fig. 4.3 are generated from one subject, we would want whatever region we use to be as stable as possible. This would reduce the variation that the recognition component has to model in order to recognize the face. For that reason, we crop our frontalized faces to a smaller, "half-face", region that seems more consistent across all the poses, as seen in Fig. 4.4. Given that rendering tools are extremely fast nowadays and already take advantage of GPU processing, this pre-processing of the face

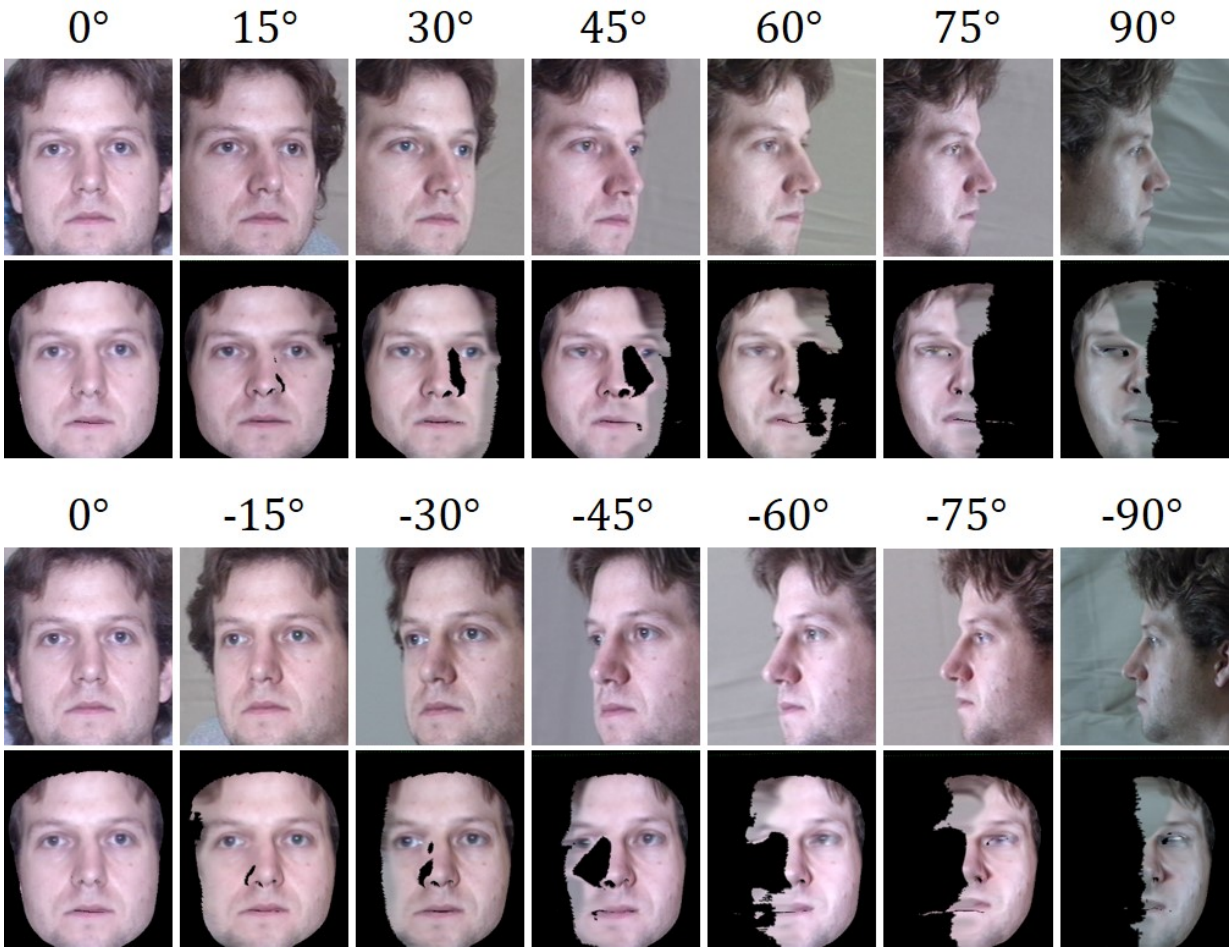

Figure 4.2: By removing the self-occluded regions from the frontalizations, the resulting faces look more realistic and are could be better for face recognition since these missing regions will at least be consistently black. However, this missing region can still be interpreted by the network as informative and it may learn to expect black in these regions.

takes almost no extra time over the landmarking and 3D modeling. This makes it a very promising way of normalizing the pose out of face images and passing a pose invariant input to the recognition systems. As we see in Chapter 4.3, these faces actually do provide a large boost in face recognition performance across large pose variations.

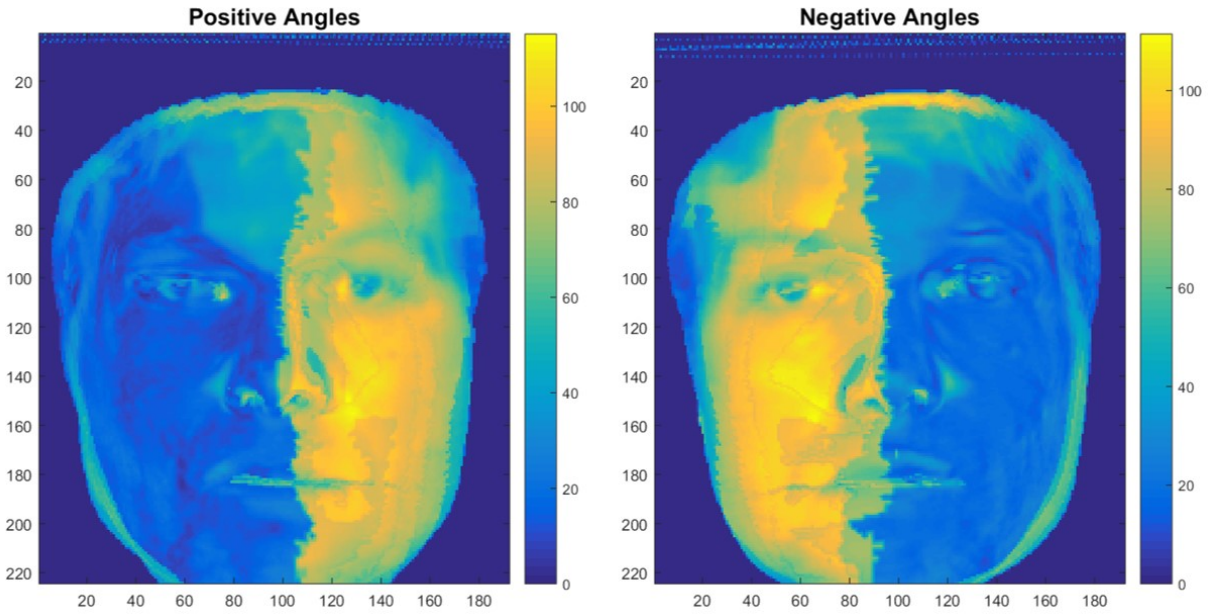

Figure 4.3: Standard deviation of the gray-scale intensities of the frontalized images from Fig. 4.2 with missing regions filled in black. These are separated by the sign of the pose angle. The clear differentiation between the halves of the face indicate that the left half of the image should be used on positive angles and the right half on negative angles.

4.2 Reconstructing the Missing Half

While these half-face images can be used directly as the input to a face recognition system, current face recognition architectures have not been tuned for such data. Instead, many different architectures have been tried on images of the whole face and tuned specifically for such images. As a result, especially if we are using pre-existing architectures, it may be important to reconstruct the whole face for these recognition architectures to learn properly. While there are many ways to reconstruct missing portions of signals, some of the most impressive results have been generated through the use of Generative Adversarial Networks (GANs). The work on GAN reconstructions in this section was done in collaboration with Felix Juefei-Xu as part of his dissertation work "Unconstrained Periocular Face Recognition: Dictionary Learning Meets Deep Learning and Beyond." We go into a brief background of GANs and the particular models we use for reconstruction here.

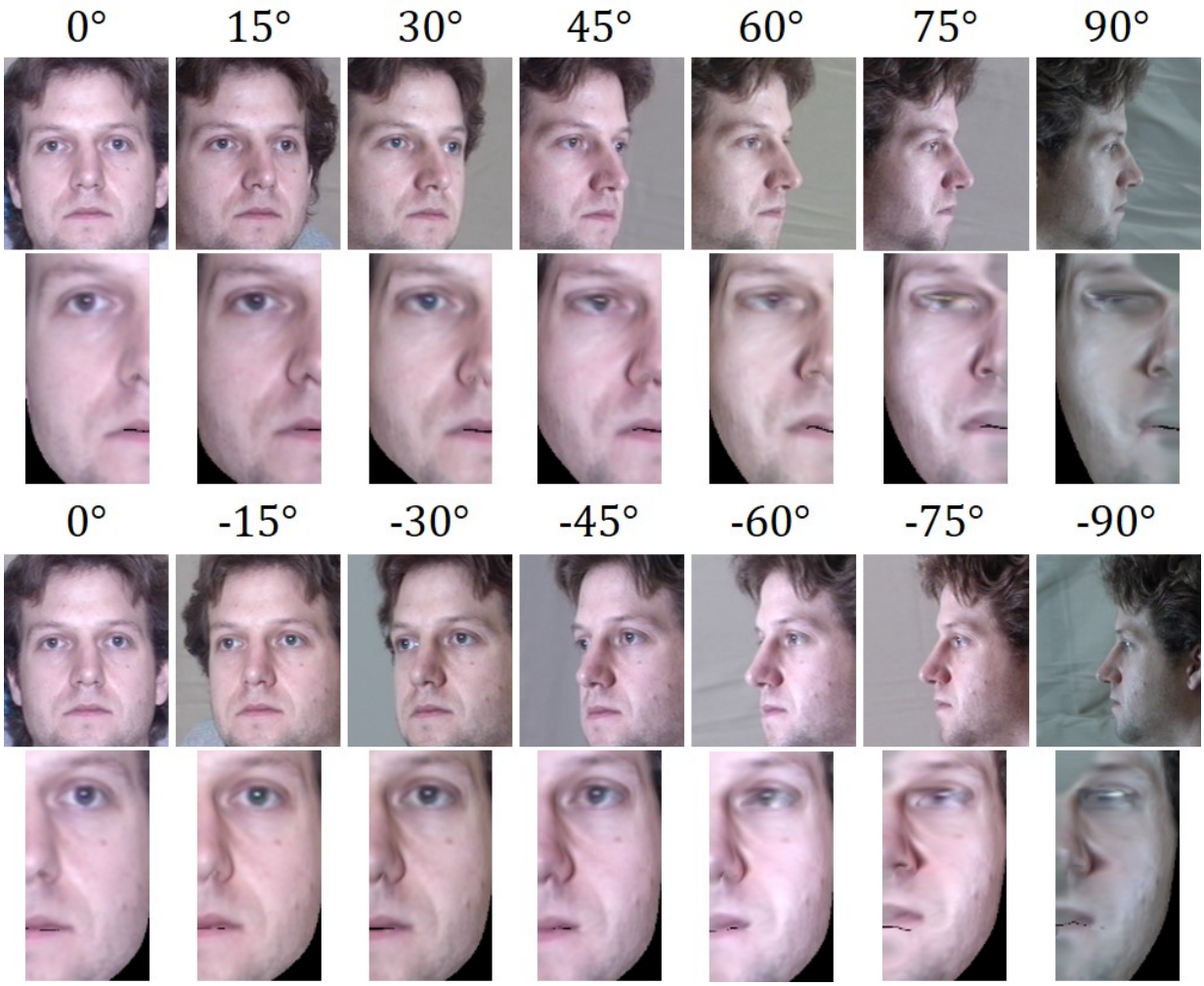

Figure 4.4: The half face frontalizations used in our face recognition experiments. These faces are much more consistent across pose than the original images or any of the other frontalizations in Figs. 4.1 and 4.2.

GANs, first introduced by Goodfellow *et al.* [83], are a class of deep networks that have been shown to generate impressive results in the image reconstruction problem. At its core, a GAN is made up of two distinct networks, a generator, $\mathcal{G}$, and a discriminator, $\mathcal{D}$. The goal of the generator is to take a random vector, $\mathbf{z}$, and generate a realistic sample from a desired distribution, thus encoding the distribution in some latent space. At the same time, the discriminator is trying to correctly identify whether an input sample was generated by $\mathcal{G}$ or is a real sample. These two networks play an adversarial game, hence the name, so that they will each force the other network to get better at its own task.

In the end, $\mathcal{G}$ should encode a very similar distribution to the true distribution from the training data and be able to generate very realistic samples from this distribution. $\mathcal{D}$, on the other hand, should be very good at distinguishing which samples come from the true data and which come from $\mathcal{G}$. Unfortunately, in actuality, achieving this is very difficult as if either $\mathcal{G}$ or $\mathcal{D}$ become too good at their task, the other network has no chance to update itself as the gradients vanish as the loss goes to zero.

While the GAN is not restricted to generating images, this has become one of the most common uses for these types of networks. The Deep Convolutional GAN (DCGAN) designed by Radford *et al.* [84] has become one of the staple architectures for training GANs on image data. The DCGAN uses fractional striding and upsampling to convert $\mathbf{z}$ into the desired image size. In addition to the DCGAN architecture, the Wasserstein GAN (WGAN), created by Arjovsky *et al.* [85], has shown a great improvement in ensuring a good match between the training distribution and the encoded latent space. They claim that traditional GANs are aiming to minimize the distance between the encoded distribution and the true distribution in a fashion that is not continuous in the parameters of the GAN. This can cause many problems in training deep networks of any kind. In fact, traditional GANs often suffered from vanishing gradients and mode collapse (where the generator abandons all but a single mode of the training distribution, effectively making it very good on one small region but bad everywhere else). By using a smooth Wasserstein distance metric to measure the distance between the distributions, WGANs seem to have addressed both of these issues. Additionally, the use of a Wasserstein distance has made the usual careful balancing of training $\mathcal{G}$ and $\mathcal{D}$ much easier as one can train $\mathcal{D}$ optimally for a given $\mathcal{G}$ and then train $\mathcal{G}$. We use both the DCGAN architecture, with 5 convolutional layers in both $\mathcal{G}$ and $\mathcal{D}$, and the WGAN loss to train our GAN model.

Practically, we want to be able to generate a whole face image from an input half-face and not from a random vector. In order to do this, we can attach an encoder, $\mathcal{E}$, to encode the half-face to the latent space vector, $\mathbf{z}$, before the generator. Together, $\mathcal{E}$ and $\mathcal{G}$ create

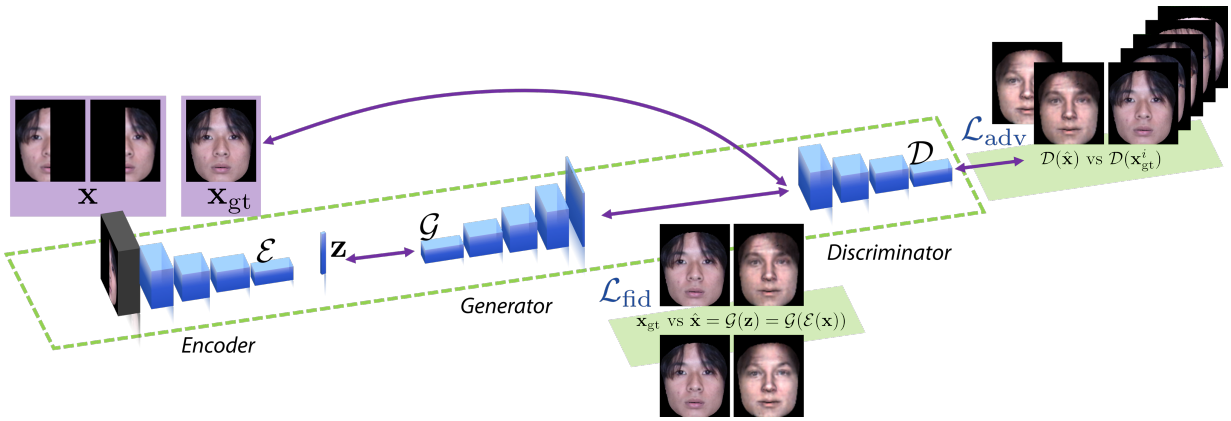

Figure 4.5: The GAN architecture used for reconstructing whole-face images from half-face images. The fidelity loss, $\mathcal{L}_{\text{fid}}$ is computed between the ground truth image, $\mathbf{x}_{\text{gt}}$, and the reconstruction, $\hat{\mathbf{x}}$ while the adversarial loss, $\mathcal{L}_{\text{adv}}$, is computed between the result of the discriminator on the reconstruction and the ground truth.

an autoencoder that will be used on any new half-face image. Our encoder uses the same architecture as $\mathcal{D}$. In addition to the WGAN loss, also known as the adversarial loss, $\mathcal{L}_{\text{adv}}$, we also impose a loss of the fidelity of the reconstructions, $\mathcal{L}_{\text{fid}}$. This fidelity loss is simply the $\mathcal{L}_1$ distance between the ground truth whole-face image, $\mathbf{x}_{\text{gt}}$, and the reconstruction, $\hat{\mathbf{x}}$ The full architecture used for the GAN is shown in Fig. 4.5.

In order to train the GAN reconstructions, we have to have both an input half-face image as well as a ground truth whole-face image that the model will use as a comparison. We can generate the half-face image for a face at any pose as shown before. However, in order to get the whole-face image, we have to restrict our training data to frontal images. Additionally, we will need a large scale database of frontal images with many unique subjects in order to train the GAN effectively. This fairly severely limits our choices of training datasets as many of the publicly available datasets do not guarantee a frontal image for any given subject. Even though we saw in Fig. 1.5 that there was a large bias towards near-frontal images in many datasets, we can also see in Fig. 4.2 that even moving out $\pm 30^\circ$ causes missing regions to appear on the frontalizations. In order to ensure that every subject in the training set has a fully frontal image, we use the mugshot data from the Pinellas County Sheriff's Office (PCSO) dataset. This dataset contains approximately

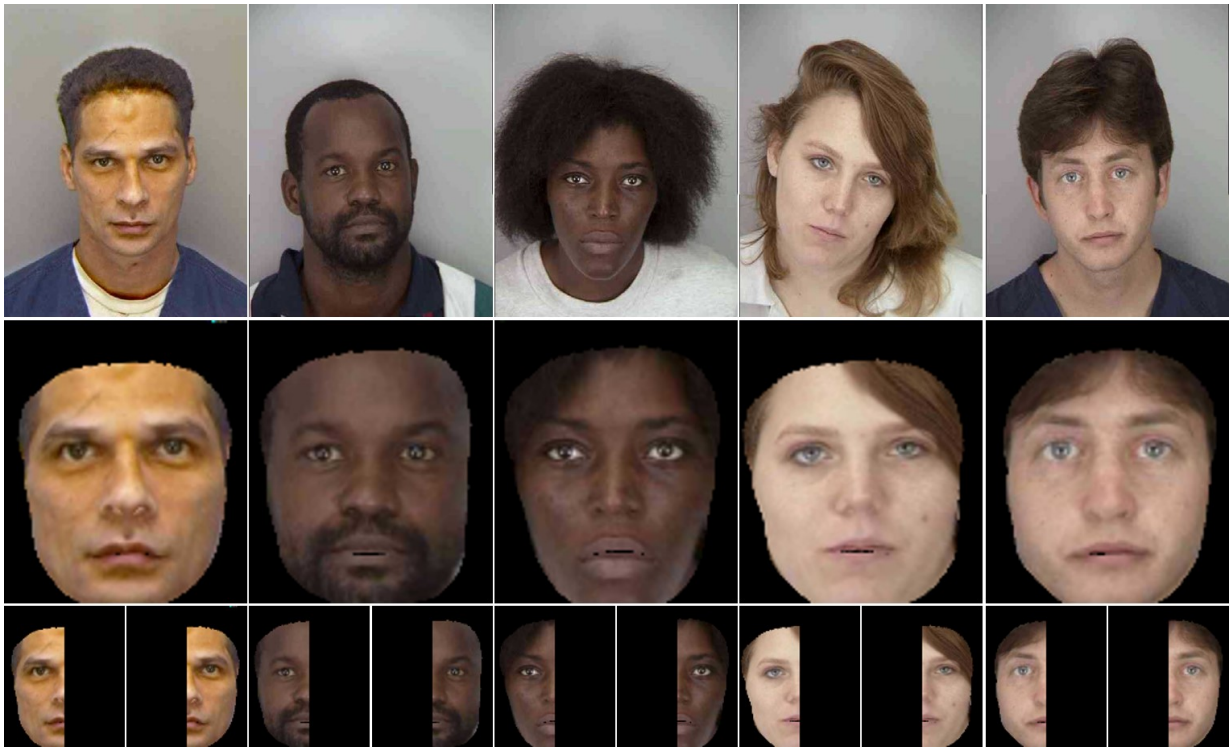

Figure 4.6: The original PCSO images (top row), the whole face rendered from a 0° viewpoint (middle row), and both half-faces (bottom row). The half-faces are used as the input data to a GAN that is trained to reconstruct the whole face.

1.5 million frontal mugshot images of over 400,000 subjects under neutral illumination and expression. These images can easily be processed by our 3D modeling pipeline and both the full face and both half-faces can be generated from each input image as seen in Fig. 4.6. These half-faces and the corresponding whole faces can be passed as the training data to our GAN model. When we take the GAN model trained on the PCSO data and run it on some subjects that were held out of the training set, we see in Fig. 4.7 that the reconstructions do look reasonable. Of course, these results are somewhat biased as the input images come from the same dataset as the training data and are all frontal images to start with. When we look at the reconstructions from using the pose images of the first subject in the MPIE dataset in Fig. 4.8, we can see that the resulting reconstruction looks considerably worse. However, until the pose reaches ±90°, it does seem like the reconstructions are similar to each other. This means that these reconstructions could still

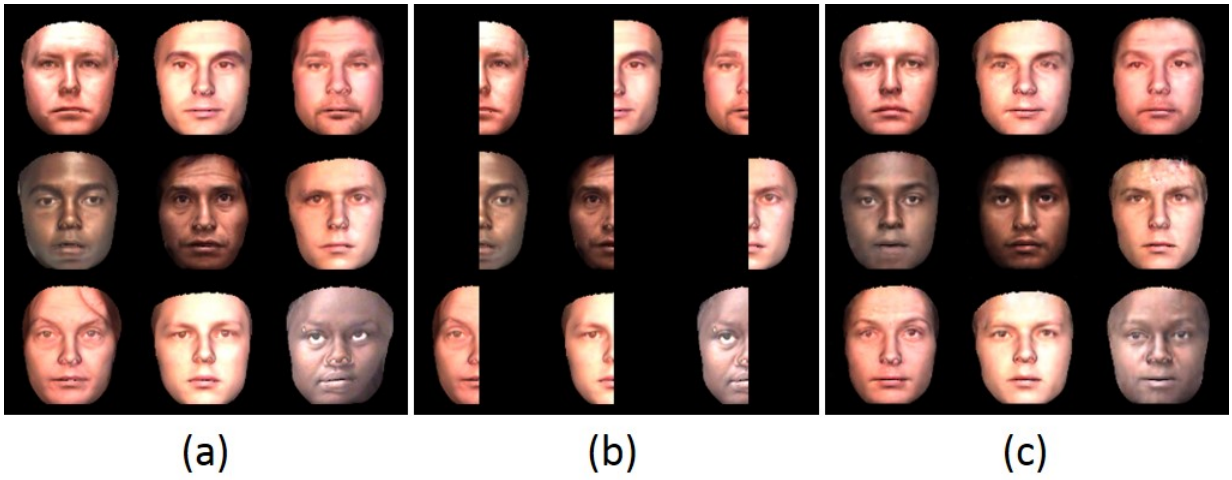

Figure 4.7: (a) The original whole faces extracted from PCSO subjects not used in training, (b) the half faces used as inputs to the GAN (c) the reconstructions created by the GAN. While there are some changes to the images, the reconstructed faces do seem like reasonable reconstructions given the half face and could potentially be used for face recognition.

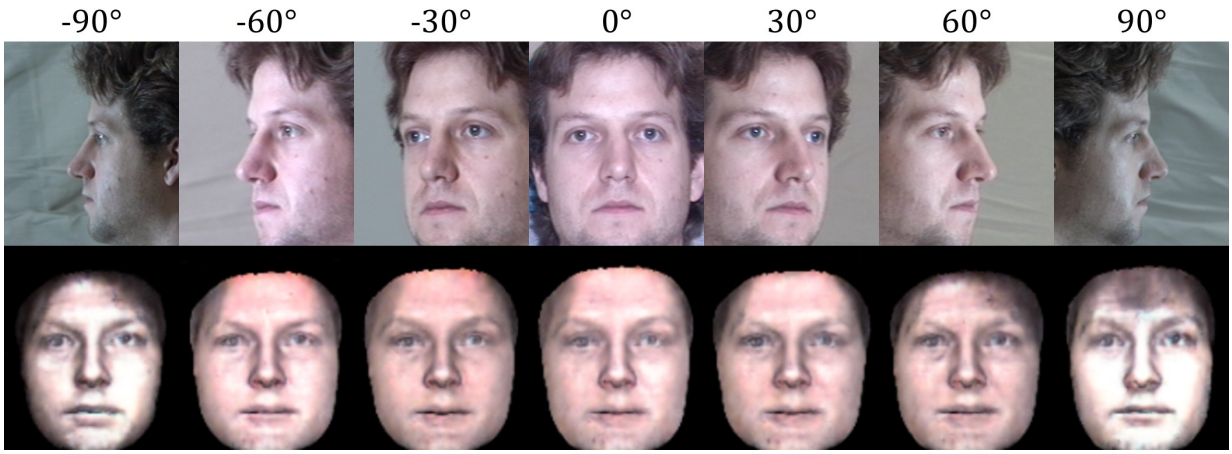

Figure 4.8: The original images from the MPIE dataset (top row) and the resulting reconstructions from the GAN processing the extracted half face (bottom row). The resulting reconstructions do not look like the original subject unfortunately. However, until the pose reaches the extremes, it does seem that the reconstructions look somewhat similar.

be useful as even if the appearance changes from the input image to the reconstruction, as long as the change is consistent, it is possible a face recognition system could use these images. It does mean that all images, even the gallery, have to be passed through the GAN to try and ensure whatever changes occur during reconstruction happen in the enrollment stage as well.

4.3 Pose Invariant Face Recognition Experiments

In order to evaluate these frontalization methods, we run several face recognition experiments on the MPIE and the Celebrities in Frontal-Profile (CFP) [86] datasets. We choose these datasets because they control the pose variation in the images. As we saw in Fig. 1.5, many of the large scale datasets used in academic research are heavily biased towards a frontal viewpoint and performing well on these datasets is not indicative of the overall performance on the pose problem. The results of the experiments on the MPIE and CFP datasets are detailed in Sections 4.3.2 through 4.3.5. As far as the actual model is concerned, we are interested in seeing the effect of the various frontalization methods on the recognition performance so we stick with a single network architecture for all the methods described previously in this chapter.

4.3.1 Implementation Details

Architecture: As Chapter 2.2 elaborated upon, there are many architectures and loss functions to choose from when training a deep network. One of the most popular architecture for face recognition right now is the ResNet architecture, in which the network is made up of residual blocks with skip connections in each block. We adopt a ResNet architecture used by others [1, 21] as a baseline model. Specifically, we use the FaceResNet-28 layer model because it has performed well on many of the pose-biased datasets, such as IJB-A and CS3, in the past but has not been extensively evaluated for pose tolerance. The architecture itself is shown in Fig. 4.9.

Training Data: We use the CASIA WebFace dataset as the training data in all of our experiments. The CASIA dataset contains 494,414 images of 10,575 subjects scraped from the internet. As we saw in Fig. 1.5, the dataset is heavily biased towards near frontal faces. This has not been a problem in the past due to the same bias being present in the testing

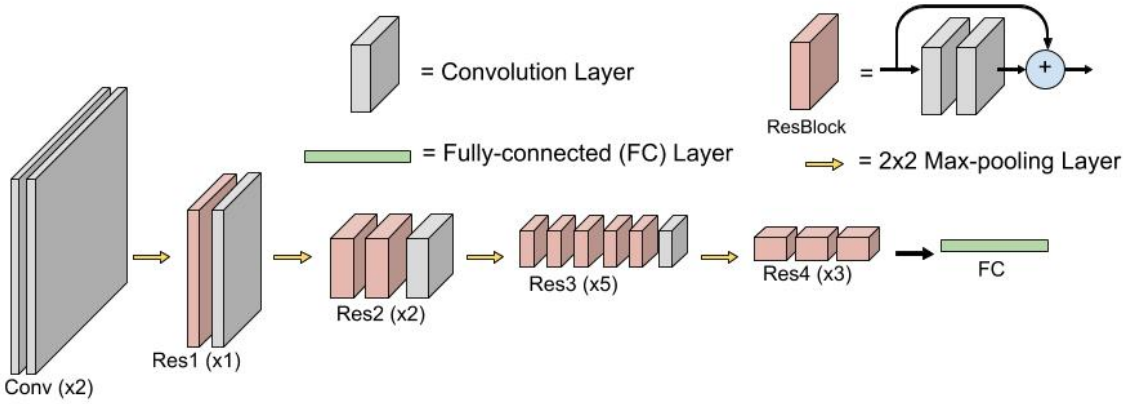

Figure 4.9: The FaceResNet-28 layer architecture used in our face recognition experiments. Each residual block is made up of two convolutional layers followed by the skip connection.

datasets as well. As a result, the CASIA dataset has become a standard benchmark training dataset for face recognition. This makes it a perfect dataset for our experiments using our frontalization methods as they should reduce both the training and testing datasets to the same pose distribution. An example of the images from the CASIA dataset can be seen in Fig. 4.10. We run the face detector used in Chapter 3 to detect as many of the faces as possible resulting in a dataset of 491,507 images of the 10,575 subjects. As this is the training set, we do not do anything with the missing detections, unlike in the testing scenarios described below. All our face recognition models are trained on this data using 99% of the data as the training set and 1% of the data as a validation set. In any result tables, the model trained with the half-face data is denoted as (HF), the model trained with the whole-face data is denoted as (WF), and the model trained with the GAN reconstructions without any pose synthesis in training the GAN is denoted as (GAN).

Hyper-parameters: When training our networks, we start with an initial learning rate of 0.1 with the learning rate dropping by an order of magnitude every 15 epochs. The architectures are trained from scratch on the CASIA data. Due to the numerous max pooling layers in the FaceResNet-28 layer architecture, we have to ensure the half-face

Figure 4.10: Images of a single subject from the CASIA WebFace dataset. The original images cropped to a fixed size (top row) are used for training the baseline whole-face model while the half-faces (middle row) and the GAN reconstructions (bottom row) are used to train their respective models.

images are of a large enough size so that the feature maps do not vanish in the deeper layers. As a result, we use a 143×154 size for the whole face images and a 143×77 size for half-face images. For the actual evaluation on the testing data, the model that was learned at the end of the epoch with the highest validation accuracy is used. For the MPIE experiments, we use the traditional SoftMax loss function for training the networks since there is a high degree of control over the variations in the data. We want to see exactly what types of variations the half-face and GAN frontalizations can handle without the help of more complex losses.

Testing Datasets: The CMU MPIE dataset [19, 20] is arguably the most well-controlled

face recognition dataset of its kind. It contains images of 337 subjects under different but controlled poses, illuminations, and expressions. The yaw angle varies from $-90°$ to $90°$ in $15°$ increments. The near perfect control of pose, expression and illumination in the dataset makes it the ideal choice for our controlled studies. We select 3 subsets of this dataset for experiments on the effect of pose alone, pose and expression, and all three modes of variation simultaneously.

The Celebrities Frontal-Profile (CFP) dataset was recently presented as a framework to evaluate frontal to profile matching in the presence of more real-word variations such as illumination, color jitter, expression, etc. [86]. The dataset has 500 subjects with a frontal to frontal matching protocol as well as a frontal to profile one. Each protocol defines 7,000 pairs of matches with 3,500 match pairs and 3,500 non-match pairs with 10 splits of 350 match pairs and 350 non-match pairs. For each split, the other 9 are used to pick a threshold and the average accuracy over all 10 splits is reported. The dataset is challenging since only a single frontal enrollment is allowed before matching to near profile faces.

4.3.2 MPIE - Pose Variation

Baseline Experiment: The goal for our first experiment is to evaluate the effect of our frontalization approaches on purely pose varying faces. As illumination and expression will also play a part in the recognition performance, we need to separate the effect of pose from these to determine if the frontalization approaches are helping face recognition performance on pose. As a baseline experiment, we restrict our dataset to only have a single frontal enrollment image that must be matched against images at all other poses. We use the $0°$, neutral expression images from Session 1 as a gallery for our experiments with a single frontal image as enrollment. We use all the non-frontal, neutral illumination and expression images from Session 1 as a set of probe images. There are a total of 249 subjects in Session 1 with one image per pose. Since a frontal image is used as the gallery, no matter

which pose in in the probe set, the corresponding half of the face can be sampled and used as the gallery image or for GAN reconstruction. As a result, for the half-face method, we compare the left half of the gallery faces to the left half faces generated in the probe set and the right half of the gallery faces to the right half faces generated in the probe set. For the GAN method, the half-face from the probe image and the corresponding half-face in the gallery are used to reconstruct the full face image for both and then matched. The whole-face and half-face images used in this experiment for a single subject can be seen in Fig. 4.4 while the GAN reconstructions can be seen in Fig. 4.8. This setup leads to a total of 744,012 match pairs being evaluated with 62,001 match pairs per pose.

Baseline Results: We showcase the ROC plots for this experiment in Fig. 4.11. We find that the half-face frontalization provides a significant boost in performance across pose over the baseline whole-face model. More importantly, as the pose increases, the performance improvement increases significantly showcasing that this frontalization approach allows for very robust face recognition at large angles with a single frontal enrollment. Specifically, the SoftMax model VR at an FAR of 10^{-3} for $\pm 75^{\circ}$ increases from 0.28 to 0.86 and for $\pm 90^{\circ}$ it increases from 0.06 to 0.54. While there is still a drop in performance as the pose increases, the drop is much slower than the same architecture using the original images. The large drop from $\pm 75^{\circ}$ to $\pm 90^{\circ}$ is concerning but could be explained by two factors. The first is that the 3D modeling is not as accurate at the extremes of $\pm 90^{\circ}$, which we have seen to some extent with the pose estimation experiments in Chapter 3.6 and with the resulting half-face images as seen in Fig. 4.4. The second is that, even though there is supposed to be the same illumination across all the poses in this experiment, the MPIE dataset itself has a large change in illumination at the $\pm 90^{\circ}$ mark, also seen in Fig. 4.4.

The GAN reconstruction, on the other hand, perform worse at every pose than the baseline model. This is, most likely, due to the fact that the reconstructions are introducing a good amount of noise into the data, rather than creating easier to recognize images.

These curves would seem to indicate that the current GAN reconstruction method is not a promising avenue to continue down. However, we still run the evaluation on all the other sets of the MPIE set in case the GAN somehow is more stable across other variations than the half-face or whole-face models.

Comparison Experiment: We also want to compare how our method performs against previous work on pose invariant face recognition. While there are many works in this area, few have tried to separate out pose as the only factor in their experiments. However, Kan *et al.* [87] have done just that in their work using Multi-view Deep Networks (MvDN) for pose invariant face recognition. In their experimental setup, they train specifically on the MPIE dataset by splitting the data into the training set, which consists of images of the first 229 subjects, and the testing set, which consists of images of the last 108 subjects. All images in these sets are under neutral illumination and expression but there may be more than one image per subject due to some subjects appearing in multiple sessions of data capture. In a similar fashion to our baseline experiment, the 0° images are used as a gallery and the results are reported at each pose separately. For this experiment, the Rank-1 recognition rate is reported instead of the ROC as this is what was provided by the other methods. However, as there may be multiple gallery images for a particular subject, if any of the gallery images are the Rank-1 match, it is considered to be a correct match for Rank-1 recognition rates. Unlike the MvDN approach, we do not train specifically on MPIE as we do not wish to accidentally learn any dataset bias that may be present. Even so, when we compare our results to those reported by Kan *et al.* [87] in Table 4.1, we can see that the half-face model is able to outperform all of the other methods. Of course, some of this is due to the fact that we are using the FaceResNet-28 layer architecture as even the whole face model is able to achieve very high accuracy out to ±60°. However, the whole-face model shows the same large drop as we move to ±75° and especially at ±90°. At these poses, the other methods do not suffer the same drop and are more accurate than the whole-face model. Still, the half-face model shows a large improvement over the

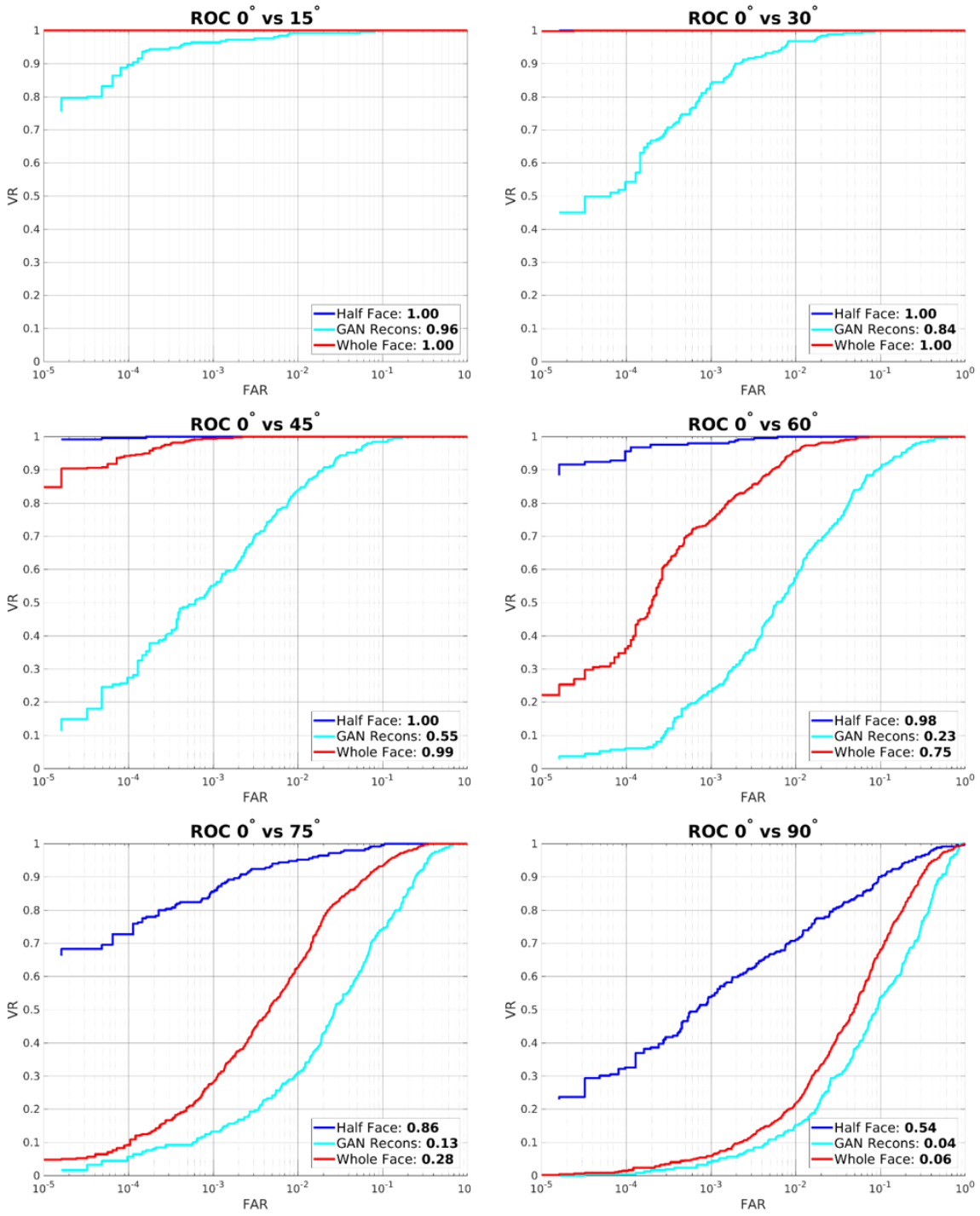

Figure 4.11: ROC curves for Exp 1 on MPIE. All matches are single gallery enrollment versus an off-angle probe image with only pose variation (neutral illumination and neutral expression). The VR at an FAR of 10^{-3} is shown in the legend. The half-face frontalization provides significant improvement over the whole-face and GAN models, especially as the pose approaches full profile.

Table 4.1: Rank-1 recognition accuracy on the MPIE dataset with only pose variation. The top result at each pose is shown in bold. WF, HF, and GAN denote our whole-face, half-face, and GAN reconstruction models respectively. The highest value in each column is shown in bold.

Method	15°	30°	45°	60°	75°	90°	Avg.
MvDA[88]	**1**	0.991	0.897	0.864	0.714	0.559	0.837
GMA[89]	**1**	**1**	0.906	0.859	0.718	0.573	0.843
MvDN[87]	**1**	0.991	0.930	0.911	0.798	0.709	0.890
WF	**1**	**1**	**1**	0.962	0.784	0.244	0.832
GAN	0.991	0.948	0.873	0.620	0.376	0.188	0.666
HF	**1**	**1**	**1**	**1**	**0.991**	**0.761**	**0.959**

Method	-15°	-30°	-45°	-60°	-75°	-90°	Avg.
MvDA[88]	**1**	0.967	0.920	0.845	0.723	0.568	0.837
GMA[89]	**1**	**1**	0.901	0.845	0.732	0.526	0.834
MvDN[87]	**1**	0.991	0.911	0.883	0.822	0.704	0.885
WF	**1**	**1**	**1**	0.981	0.808	0.319	0.851
GAN	0.986	0.911	0.798	0.577	0.357	0.188	0.636
HF	**1**	**1**	**1**	**1**	**0.991**	**0.892**	**0.981**

other methods on this experiment, showing that the half-face frontalization does provide a benefit past that of the FaceResNet28-layer architecture. The model trained on the GAN reconstructions shows the same trend as in the previous experiment and consistently under-performs compared to all the other methods.

4.3.3 MPIE - Pose and Expression Variation

While the increase in performance across pose is very encouraging for these frontalization approaches, we know that pose variation rarely happens in a vacuum. Most of the time, pose variation occurs with illumination or expression variation. In the work by Peng *et al.* [67], in which they develop a method which disentangles pose from the feature, they present very impressive results on a subset of MPIE containing pose and expression variation which are state-of-the-art. In this protocol, like the one used by Kan *et al.* [87], the last 108 subjects are used for testing. Unlike the previous protocol, only the illumination is controlled and both the pose and expression are allowed to vary as seen

Figure 4.12: Images of a single subject from the MPIE dataset with both pose and expression variation used in the experiment detailed in Chapter 4.3.3. The first three rows show the gallery images for the whole-face, half-face, and GAN reconstruction models respectively. The second three rows show a selection of the probe images in the same order.

in Fig. 4.12. The gallery consists of a random selection of 2 frontal images per subject, meaning that not every expression of each subject will be seen in the gallery. The probe set consists of all expressions at the non-frontal poses. The results are averaged over 10 such gallery selections and averaged over both the positive and negative angles.

Results: We find that in Table 4.2, our half-face frontalization approach outperforms [67] in Rank-1 recognition accuracy by a significant amount at all poses except ±90°. For instance, Peng *et al.* [67] achieves 79.9% accuracy at ±75° whereas the half-face model achieves 84.9% accuracy. Note that SS in Table 4.2 for [67] signifies single source training, which is the same as our half-face models (all trained on CASIA-WebFace). SS-FT for [67] signifies that the models were fine-tuned on MPIE. We only compare against these models as the others developed by Peng *et al.* [67] were trained on much more data, making it difficult to determine whether the increase in data or the change in their network architecture is responsible for the improvement in accuracy. Only the SS-FT model outperforms the half-face frontalization at 90°. The MPIE dataset 90° images do differ significantly in illumination, even at the neutral illumination setting, as can be seen when comparing the first and last columns of Fig. 4.4. We believe that fine-tuning on the MPIE dataset would allow for models to account for this change but would not lead to generalizability. In fact, in their work itself, it can be seen in their Table 5 that when they train on the 300W-LP dataset instead of the MPIE dataset, their performance at the 90° mark drops over 25%. This is an indication that their models trained on any amount of MPIE data may be over-fitting to the MPIE bias itself. Interestingly, our half-face models significantly outperform [67] *without* any fine-tuning whatsoever at all other poses and has closed much of the gap between the models with and without fine-tuning. However, we can see that the performance has dropped significantly once expression has been added into the dataset. Still, the resulting performance has shown that the half-face frontalization has provided a benefit to recognition over using the original data as the half-face model is still much better overall than the whole-face model. We can see that the whole-face model performs better at the near-frontal poses of ±15° and ±30° but not by a large amount. This does suggest that it may be better to have two recognition models, one for near frontal images and one for non-frontal images. To see how such a model would perform, we add another evaluation to the table, WF+HF Fusion, that fuses the two models by using the

Table 4.2: Rank-1 recognition accuracy on the MPIE dataset with pose and expression variation. The top result at each pose is shown in bold.

Method	15°	30°	45°	60°	75°	90°	Avg.
Peng *et al.* SS[67]	0.908	0.899	0.864	0.778	0.487	0.207	0.690
Peng *et al.* SS-FT[67]	0.941	0.936	0.919	0.883	0.799	**0.681**	0.860
WF	**0.965**	**0.953**	0.928	0.828	0.584	0.216	0.746
GAN	0.722	0.663	0.573	0.424	0.272	0.138	0.465
HF	0.953	0.944	**0.931**	**0.907**	**0.849**	0.604	0.865
WF+HF Fusion	**0.965**	**0.953**	**0.931**	**0.907**	**0.849**	0.604	**0.868**

whole-face model on the ±15° and ±30° ranges and the half-face model on the rest. As can be seen, this two model method performs the best overall and will be evaluated on the following MPIE experiments as well.

4.3.4 MPIE - Pose, Illumination, and Expression Variation

For our final experiment, we define a new protocol on MPIE and present the results of our methods on it. This protocol utilizes all three variations available in MPIE *i.e.* Pose, Illumination and Expression. We define all frontal images with all neutral illumination and expression for the last 108 subjects as the gallery. This results in 213 images in the gallery. We include all variations of the same set of subjects with off-angle pose as the probe set. This results in 148,560 images in the probe set(12,380 per pose). For each probe image, we match against the entire gallery, measuring the average rank-1 accuracy. This is a particularly difficult experiment as some of the illuminations are extremely harsh,especially at some of the more extreme angles, as can be seen in Fig. 4.13.

Results: Table 4.3 presents the results of this experiment. As to be expected, the performance starts to drop when other modes of variation, such as illumination and expression, all start to occur simultaneously. However, even under these combinations, we can see that the trend of the half-face models outperforming the whole-face models at extreme angles continues to hold and the GAN models continue to perform poorly. We do observe the

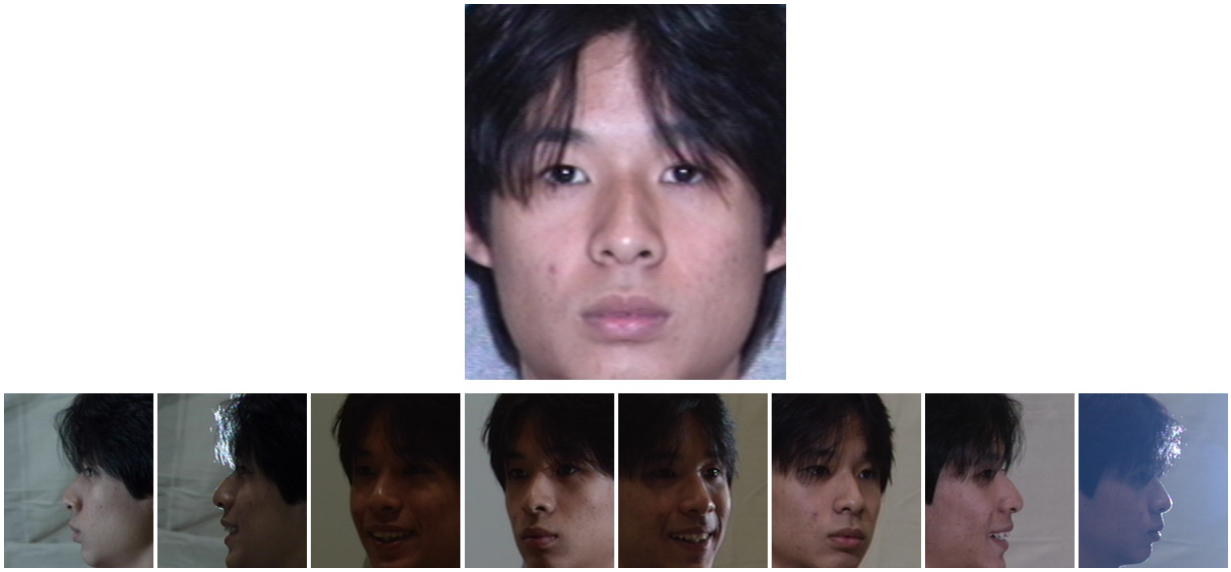

Figure 4.13: Images of a single subject from the MPIE dataset with all variations used in the experiment detailed in Chapter 4.3.4. The image on top is the only one used in the gallery while the images below are a small, random sampling of the probe images. Note how in some images, the face is almost completely dark or washed out due to the lighting, making recognition very difficult.

same effect as in the previous experiment in that the whole-face model actually performs better at the near frontal poses but quickly drops off in accuracy as the pose increases and that the joint model performs best overall. All of these experiments on the MPIE data suggest that there is a benefit to using such a normalization of the faces, even when pose is not the only variation present in the data. This does seem to suggest that there must also be methods in place to deal with illumination and expression explicitly in order to be able to perform well in these difficult scenarios. We discuss some possible methods of handling this in Chapter 5. All of these previous experiments have shown that there is merit to these frontalization methods but are still being performed on controlled, laboratory data. To that end, we also test the half-face models on the CFP dataset to get an idea of how this performs in the real world.

Table 4.3: Rank-1 recognition accuracy on the MPIE dataset with pose, illumination, and expression variation. The top result at each pose is shown in bold.

Method	15°	30°	45°	60°	75°	90°	Avg.
WF	**0.972**	**0.947**	0.879	0.719	0.479	0.188	0.697
GAN	0.429	0.385	0.324	0.243	0.164	0.104	0.275
HF	0.950	0.929	**0.886**	**0.804**	**0.669**	**0.443**	0.780
WF+HF Fusion	**0.972**	**0.947**	**0.886**	**0.804**	**0.669**	**0.443**	**0.786**

4.3.5 Experiments on CFP

We evaluate the CFP dataset with our half-face frontalization and compare to other approaches on the same dataset. Unfortunately, unlike the MPIE dataset, the CFP dataset has been pre-cropped using another face detector by the authors of the dataset. The resulting cropped faces remove portions of the face that are seen in the training of the 3D modeling leading to poor half-face frontalizations, as seen in Fig. 4.14, on a non-negligible portion of the dataset. We requested the original, full images from the authors of the dataset but they did not keep these images so we ran our method as is on the CFP dataset. For the Frontal-Frontal protocol, we match the left and right halves of the face and average the scores. For the Frontal-Profile protocol, we flip the face and match to both halves of the enrollment and average the scores. We do this for the CFP dataset because the "in-the-wild" nature of this dataset means that other variations, such as occlusions, may appear on one side of the face and not the other.

Results As can be seen in Table 4.4, our half face model outperforms the whole-face model slightly on the Frontal-Profile protocol while still following the trend of showing a small drop in performance on the Frontal-Frontal protocol.

Both of these model perform fairly similarly to other methods trained on the same data on the Frontal-Frontal protocol. However, on the Frontal-Profile protocol, the half-face method is capable of giving some improvement over Sengupta *et al.* [86] but does not perform as well as any other method. However, this drop is reasonable given the difference

Figure 4.14: A few examples from the CFP dataset showing that the faces are pre-cropped fairly tightly, removing necessary parts, such as the ears or neck, for the 3D modeling. This results in poor half-face frontalizations and negatively impacts the recognition accuracy.

Table 4.4: Accuracy on the CFP dataset. The Training column indicates how many images were in the training set. Sengupta *et al.* [86], TPE [90], and the WF and HF models were all trained on the CASIA dataset.

Method	Frontal-Frontal	Frontal-Profile	Training Size
Human	96.24	94.57	-
Sengupta *et al.* [86]	96.40	84.91	500K
TPE [90]	96.93	89.17	500K
DR-GAN [66]	97.84	93.41	1.1M
Peng *et al.* [67]	98.67	93.76	1.1M
WF	96.89	86.80	500K
HF	96.07	87.30	500K

in the size of the training datasets used between the models and the fact that the images are pre-cropped, making it impossible to run our full system as intended on this dataset.

Still, our method has been able to achieve a decent performance on this real world dataset and shows promise for future research.

Chapter 5

Concluding Remarks and Future Work

While the problem of pose invariant face recognition has not yet been solved, this work presents a possible step down that path. By using the half-face frontalizations described in Chapter 4, we were able to achieve some very large improvements in recognition performance at large pose variations. The key to this whole work was understanding how the physical world is interpreted by a camera and how best to integrate that knowledge into the deep networks that have shown so much promise in the recent years. This integration of explicit domain knowledge into the deep networks is what lets us perform at higher levels than before. While it may be easier, and therefore tempting, to just trust the deep networks to learn everything given enough data, we have shown that imposing the restrictions from prior knowledge can greatly benefit face recognition. This was immediately noticeable in both the 3D modeling which made use of both the TPS modeling of a face and the explicit knowledge of camera projection to model how a face actually appears in 2D due to 3D transformations as well as in the case of the half-face frontalization, where the deep networks were never shown pose varying regions of the face. Possibly of greater importance, however, is the realization that current large scale datasets do not necessarily

model the distribution of real-world scenarios and must be carefully examined before being used naively. Even if many other works are comparing on the currently available datasets and performing extremely well, this does not necessarily mean that they actually perform well in deployment scenarios.

5.1 Future Work

Of course, the work outlined here is only the first step in the direction of truly unconstrained face recognition. As we saw in the successive experiments on the MPIE dataset, as more modes of variation are introduced, the performance continues to drop. While the techniques we created allow for pose to be fairly well handled, these successive drops in performance are likely to still prevent such a system from being very usable in real unconstrained settings. Here we outline a few possible future research directions to try and account for these problems.

- Perhaps the first step in trying to make this system more accurate should be to retrain the 3D modeling to ensure it fits well on the $\pm 90^\circ$ range. As we could see in Fig. 4.2, a large portion of the face ends up being textured from the background. This could be because the model was trained on purely synthetic data and the profile faces are not very realistic or because there is not enough of a restriction on the camera projection matrix and the model ends up putting more of an emphasis on the TPS warping to try and handle profile images. To try and fix the camera projection restrictions, a weak perspective projection matrix could be used instead of the full camera projection matrix or the model could use a predefined set of intrinsic parameters and only adjust the rotation and translation of the camera. Another possibility is that the model does not fit the forehead very well because there are no control points on the forehead to guide the TPS warping of the model. By adding a few control points on the forehead, the model may be more tightly fitting the profile faces. Both of these approaches

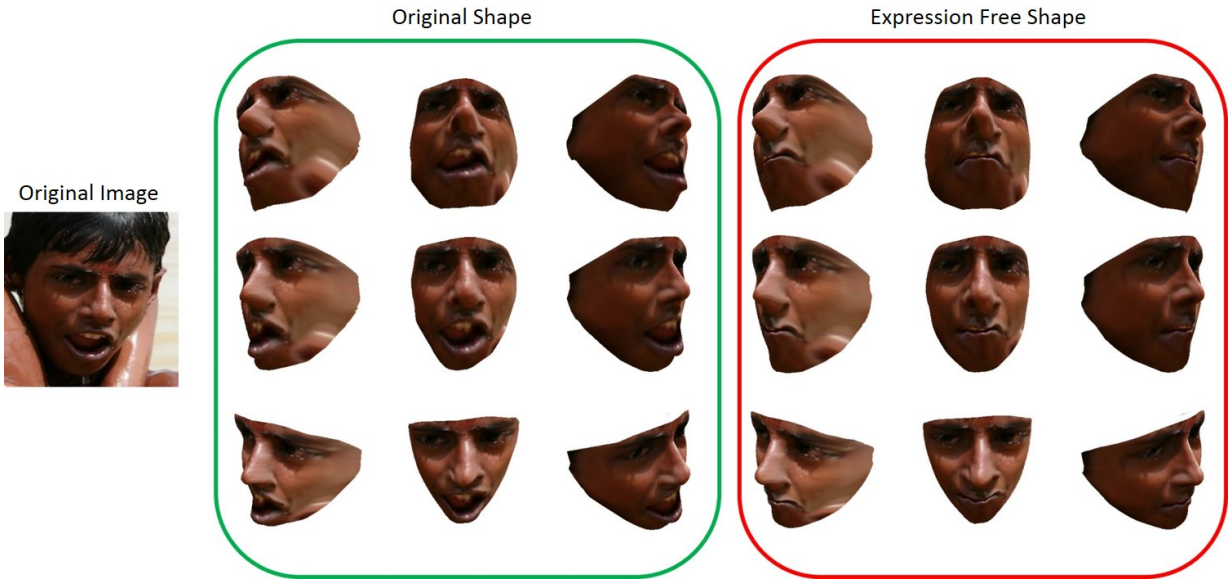

Figure 5.1: After using a TPS to warp a face model to close the mouth, the resulting 3D shape

could help improve the profile face modeling and improve recognition performance at the profile angles.

- Expressions could potentially be handled in a very similar way to the pose invariant face matching. Although facial expressions do change the texture of the face, by exposing more teeth when smiling for example, the portions of the face visible during a neutral expression are all still visible. In that send, normalizing out expression becomes a problem of sampling the correct regions from the face in a similar fashion to the half-face generation. Essentially, one would need to ensure the 3D model was capable of fitting on expressions well and then apply the sampled texture to an expression free model. This expression free model could be generated through the use of some basis method, like the 3DMMs used by others, or by a TPS warp of the expressive areas, such as the mouth, to a fixed location as shown in Fig. 5.1.

- Illumination presents a tougher challenge to face recognition as illumination tends to affect larger areas of the face region than expressions. While we cannot easily adapt the 3D modeling approach to normalize out illumination, we can use it to

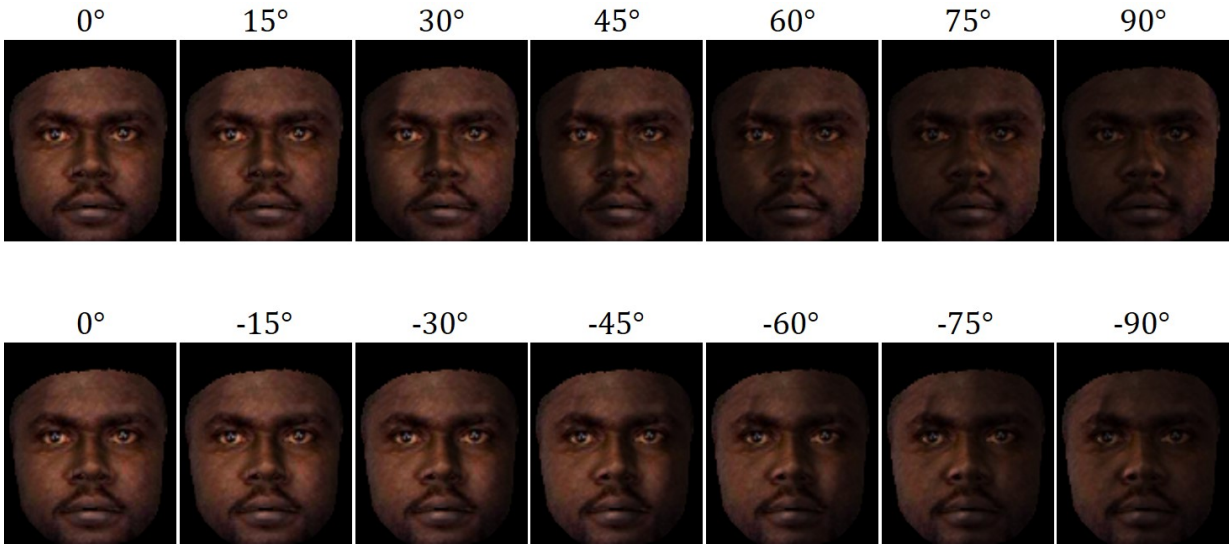

Figure 5.2: A single image from the PCSO dataset with the 3D model rendered under different illumination directions. The angle indicates where the point light source is located with respect to the face at a fixed distance. These could possibly be used to generate more training data to handle illumination problems.

synthesize new illuminations from input images with neutral illumination. This is a relatively simple task for any 3D rendering software as it is simply inserting a point source illumination into the scene and specifying the reflectance properties of the face. These properties can be shared across all faces and would only have to be found once. This means we can train networks by showing them the same subject at many different illuminations, as seen in Fig. 5.2, and hopefully the trained models will be more tolerant to illumination. Of course, this assumes the model fitting is relatively stable under illumination changes which has not been explicitly studied here.

- Another possible use for the 3D modeling method created is to perform face substitution for both privacy preserving purposes and entertainment value. By extracting the shape from a desired face and the texture from a face that we want to swap onto it, we can replace faces in the original image as seen in Fig. 5.3.
- All of the previous extensions to this work have focused on the 3D modeling aspect.

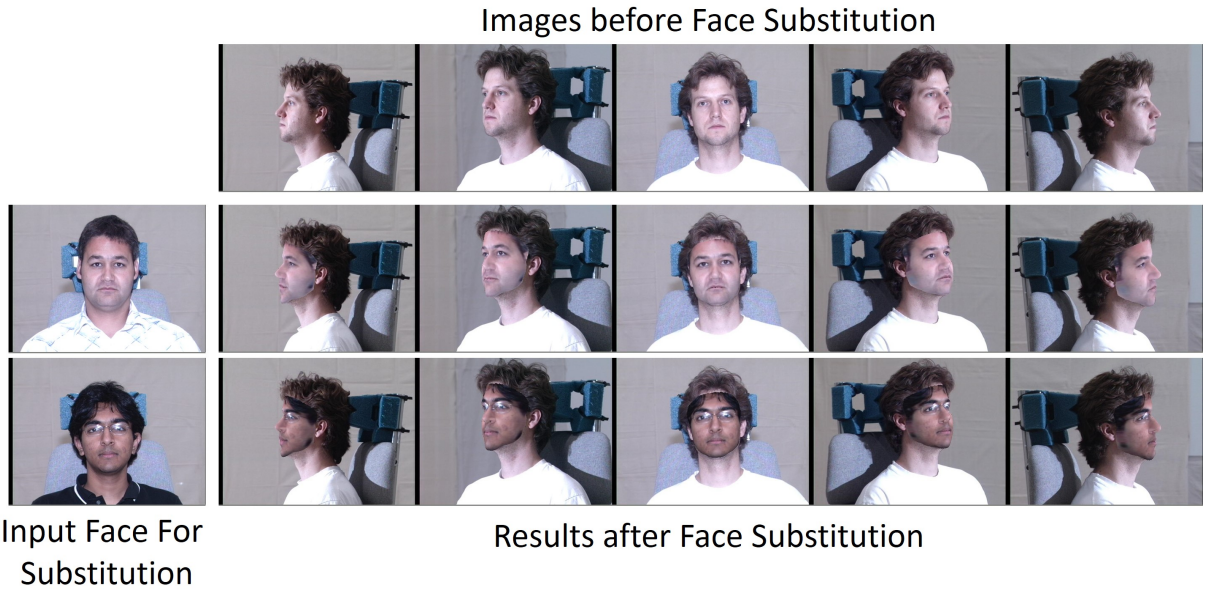

Figure 5.3: Two examples of face substitution from the MPIE dataset. The original images that faces will be inserted into are shown on the top row. The frontal images with the faces that will be inserted are shown in the leftmost column. The rest of the 2^{nd} and 3^{rd} rows show the results after face substitution has been performed.

However, there is still much research to be done on the training of the recognition models themselves. While everything we have done was intended to be agnostic of the architecture and loss functions used in training the networks, these still play an important roll in developing a state-of-the-art face recognition system and cannot be dismissed. Therefore, one avenue of future research is to determine what combination of architectures and losses give the best face recognition performance when using these methods for frontalizing the face.

Appendix A

Properties of Camera Geometry

This appendix is meant to give a brief overview of modeling camera projection and the properties used in this work. It is not intended to be a full tutorial on these topics. For full details on camera geometry and camera projection, the book "Multiple View Geometry in Computer Vision" by Richard Hartley and Andrew Zisserman is highly recommended.

A.1 Formulation of the Camera Projection Matrix

To understand the formulation of the camera projection matrix, we first model how an ideal pinhole camera images objects in the world. As shown in Fig. A.1, in an ideal pinhole camera, all rays of light pass through a single point and are then projected onto the image plane at the back of the camera. When this projection occurs, all depth information is lost as the resulting image is only 2D. With only one view, it is impossible to recover the depth information in general as any point along the projection ray could be the correct location of the imaged point. The projected image coordinates, (x_i, y_i), for any given point in the

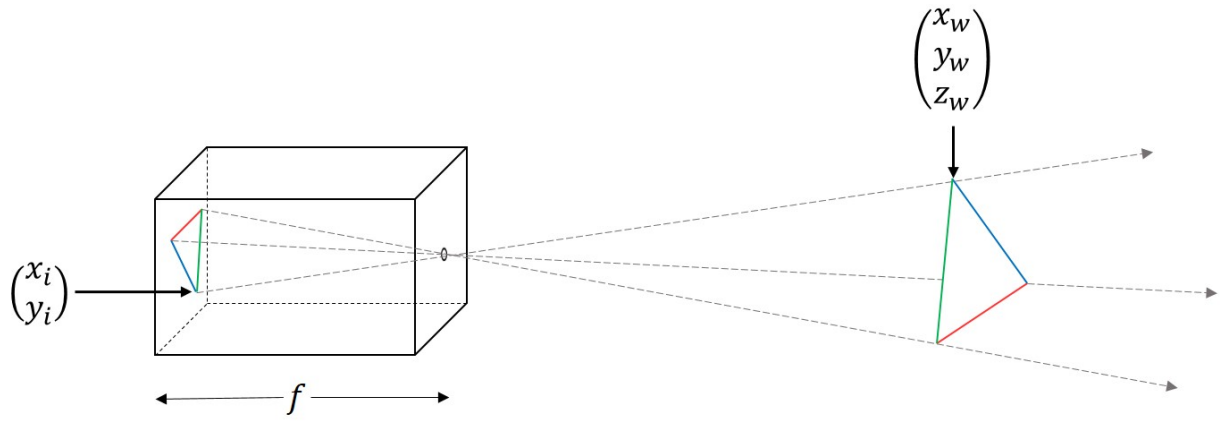

Figure A.1: The ideal pinhole camera model with a focal length, f. Points in the world coordinate system, represented by (x_w, y_w, z_w), project through a single point onto the image plane at (x_i, y_i), determined by Eqn. A.1. The depth information is lost through this projection. In fact, all points on a given ray project to the same point on the image, making it impossible to recover the depth information from a single view.

world, (x_w, y_w, z_w), through a camera with a focal length of f is given by the equation

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = -\frac{f}{z_w} \begin{bmatrix} x_w \\ y_w \end{bmatrix} \tag{A.1}$$

In order to account for this ambiguity of scale, we have to define our points in a homogeneous coordinate system in which points are considered equal if they only differ by a scale factor, c. This equality up to scale is represented by the symbol, $\widetilde{=}$.

$$\begin{bmatrix} x \\ y \\ c \end{bmatrix} \widetilde{=} \begin{bmatrix} x \backslash c \\ y \backslash c \\ 1 \end{bmatrix} = \begin{bmatrix} x\prime \\ y\prime \\ 1 \end{bmatrix} \tag{A.2}$$

Using this coordinate system, Eqn. A.1 can be rewritten as

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{A.3}$$

$$\mathbf{p}_i \cong \mathbf{C}\mathbf{p}_w$$

where $\mathbf{p}_i$ and $\mathbf{p}_w$ are the homogeneous representations of the image and world point respectively and $\mathbf{C}$ is the pinhole projection matrix. From here on, all points are assumed to be in a homogeneous representation. While this equation models the ideal pinhole camera projection, it assumes the world coordinate system and the camera coordinate system are aligned together. This is often not the case as the camera could be arbitrarily rotated and translated from the world frame of reference. In order to project a world point in the world coordinate system, $\mathbf{p}_w^w$, to the image, we have to apply the rotation and translation to get the point to the camera coordinate system, $\mathbf{p}_w^c$. This can be done by

$$\mathbf{p}_w^c = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{p}_w^w \tag{A.4}$$

If we assume the focal length of the pinhole camera is $f = 1$ and combine Eqns. A.1 and A.4, we can get a new relationship between $\mathbf{p}_i$ and $\mathbf{p}_w^w$ as

$$\begin{aligned} \mathbf{p}_i &\cong \mathbf{C}\mathbf{p}_w^c \\ &\cong \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{p}_w^w \\ &\cong \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \mathbf{p}_w^w \end{aligned} \tag{A.5}$$

This rotation and translation matrix is known as the extrinsic matrix since it does not model any of the camera parameters themselves, only how the camera and the world exist in relation to each other. Of course, the focal length will almost never be 1 and so we can left-multiply the equation by the 3×3 matrix containing the focal length parameter

$$\begin{aligned} \mathbf{p}_i &\cong \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \mathbf{p}_w^w \\ &\cong \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \mathbf{p}_w^w \end{aligned} \tag{A.6}$$

While this is fine for an ideal pinhole camera, real cameras do not behave this way. They have intrinsic properties that make each camera differ such as the squareness of their pixels or a skew between the horizontal and vertical directions that needs to be modeled as well. These are all pulled into the $\mathbf{K}$ matrix, which is known as the intrinsic matrix. In total, there are 5 parameters that are usually modeled by the intrinsic matrix. They are α_x and α_y, the scaling in the x and y directions on the image plane, s, the skewness of the axes, and p_x and p_y, the (x, y) location on the image of the principal point. The principal point is the intersection of the camera coordinate system's z-axis, also known as the optical axis, and the image plane. The full intrinsic matrix is represented as

$$\mathbf{K} = \begin{bmatrix} \alpha_x & s & p_x \\ 0 & \alpha_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{A.7}$$

The full camera projection matrix, $\mathbf{M}$, is the composition of the intrinsic and extrinsic matrices.

$$\mathbf{M} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \tag{A.8}$$

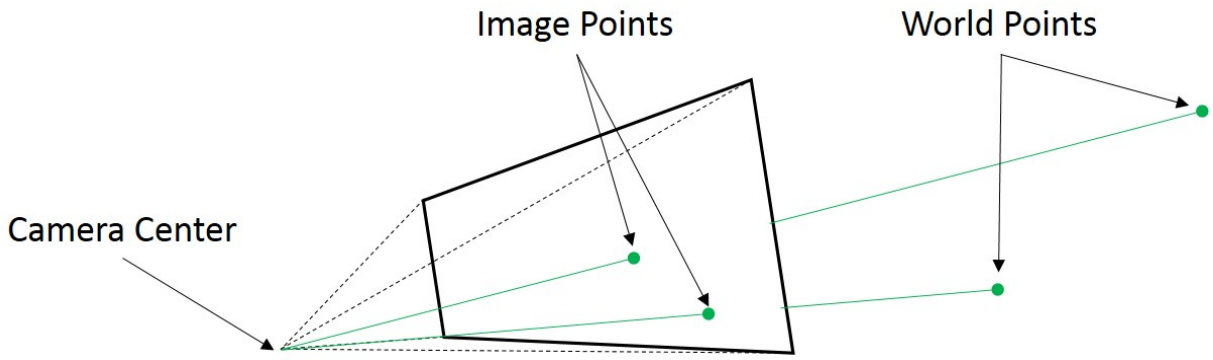

Figure A.2: An illustration of how 3D points project to the image plane along a ray joining them to the camera center. Every projection ray will meet at the camera center.

A.2 Properties of the Camera Projection Matrix

The camera projection matrix, $\mathbf{M}$, defines how the 3D space is projected onto the camera's image plane. Geometrically, $\mathbf{M}$ defines the image plane and a camera center through which all projective rays must pass. This means that the 2D projection of any 3D point can be found by drawing a line that connect the 3D point and the camera center and finding the intersection of that line with the image plane as seen in Fig. A.2. The coordinates for the camera center, $\mathbf{c}$, can be found from $\mathbf{M}$ directly. We know that for any given point in the world, $\mathbf{p}_w$, all points on the line joining it and $\mathbf{c}$ should project to the same image point, $\mathbf{p}_i$. The line joining $\mathbf{p}_w$ and $\mathbf{c}$ can be written as

$$\mathbf{p}_w(\lambda) = \lambda \mathbf{p}_w + (1 - \lambda)\mathbf{c} \tag{A.9}$$

The projection through $\mathbf{M}$ onto the image plane of any point on this line is

$$\begin{aligned} \mathbf{p}_i &\cong \mathbf{M}\mathbf{p}_w(\lambda) \\ &\cong \lambda \mathbf{M}\mathbf{p}_w + (1 - \lambda)\mathbf{M}\mathbf{c} \end{aligned} \tag{A.10}$$

All of these points must project to the same location that $\mathbf{p}_w$ projects or, in other words,

$$\mathbf{M}\mathbf{p}_w \cong \lambda\mathbf{M}\mathbf{p}_w + (1-\lambda)\mathbf{M}\mathbf{c} \tag{A.11}$$

Since this must be true regardless of λ and $\mathbf{M}\mathbf{p}_w \cong \lambda\mathbf{M}\mathbf{p}_w$ because of the equality only being defined up to scale,

$$\mathbf{M}\mathbf{c} = \mathbf{0} \tag{A.12}$$

or $\mathbf{c}$ must be in the null space of $\mathbf{M}$. Since $\mathbf{M}$ projects from a 3D space to a 2D space, the null space must only have one dimension and so $\mathbf{c}$ must be the null space. This can be found easily by first splitting $\mathbf{M}$ into the first 3x3 matrix and the last 3x1 column as

$$\mathbf{M} = [\mathbf{A}\ \mathbf{b}] \tag{A.13}$$

Then the homogeneous world coordinates of the camera center is

$$\mathbf{c} = \begin{bmatrix} -\mathbf{A}^{-1}\mathbf{b} \\ 1 \end{bmatrix} \tag{A.14}$$

With the camera center found, we can now back-project rays through image points to determine what possible set of 3D points could project to that image coordinate. This is needed for our 3D model refinement in Chapter 3.4. In order to find a projection ray that passes through an image point, $\mathbf{p}_i$, we need to know one point in the world that lies on this ray. Any point, $\mathbf{p}'_w$ that would lie on the ray must project back to $\mathbf{p}_i$ and therefore satisfy

$$\mathbf{p}_i \cong \mathbf{M}\mathbf{p}'_w \tag{A.15}$$

Since $\mathbf{M}$ is a 3×4 matrix with a one-dimensional null space, it must have a left pseudo-inverse

$$\mathbf{M}^{+} = \mathbf{M}^{T} \left(\mathbf{M}\mathbf{M}^{T}\right)^{-1} \tag{A.16}$$

From this we can get

$$\mathbf{p}'_{w} = \mathbf{M}^{+}\mathbf{p}_{i} \tag{A.17}$$

We can verify that this point projects back to the original image point by multiplying it by the projection matrix, $\mathbf{M}$, to get

$$\begin{aligned} \mathbf{M}\mathbf{p}'_{w} &= \mathbf{M}\mathbf{M}^{+}\mathbf{p}_{i} \\ &= \mathbf{p}_{i} \end{aligned} \tag{A.18}$$

With this point that we know lies on the projective ray, we can parameterize the points on the ray as

$$\mathbf{p}_{w}(\lambda) = \mathbf{M}^{+}\mathbf{p}_{i} + \lambda\mathbf{c} \tag{A.19}$$

since $\mathbf{c}$ lies on every projective ray. Since we look for the closest point on these rays to our 3D models for refining them, we can normalize out the scale factor on both homogeneous points once to find their true 3D coordinates. At that point, we just have a 3D line and are not dealing with projections anymore so we can ignore the extra homogeneous coordinate and just deal with 3D points. The line can then be expressed as

$$\mathbf{p}'_{w}(\lambda) = \mathbf{p}'_{l} + \lambda\mathbf{c}' \tag{A.20}$$

where $\mathbf{p}'_{w}$, $\mathbf{p}'_{l}$, and $\mathbf{c}'$ are the non-homogeneous 3D points for the output point on the line, a pre-selected point on the line that satisfies Eqn. A.19, and the camera center respectively. For any 3D point, $\mathbf{p}_{m}$, finding the closest point on the back-projection ray is equivalent to

finding the λ that minimizes

$$\|\mathbf{p}_m - \mathbf{p}'_w(\lambda)\|_2^2 \tag{A.21}$$

Minimizing this with respect to λ gives the solution as

$$\begin{aligned}
&\min_{\lambda} \ \|\mathbf{p}_m - \mathbf{p}'_w(\lambda)\|_2^2 \\
=&\min_{\lambda} \ (\mathbf{p}_m - \mathbf{p}'_w(\lambda))^T (\mathbf{p}_m - \mathbf{p}'_w(\lambda)) \\
=&\min_{\lambda} \ \mathbf{p}_m^T\mathbf{p}_m - 2\mathbf{p}_m^T\mathbf{p}'_w(\lambda) + \mathbf{p}'_w(\lambda)^T\mathbf{p}'_w(\lambda) \\
=&\min_{\lambda} \ \mathbf{p}_m^T\mathbf{p}_m - 2\mathbf{p}_m^T(\mathbf{p}'_l + \lambda\mathbf{c}') + (\mathbf{p}'_l + \lambda\mathbf{c}')^T(\mathbf{p}'_l + \lambda\mathbf{c}') \\
=&\min_{\lambda} \ \mathbf{p}_m^T\mathbf{p}_m - 2\mathbf{p}_m^T\mathbf{p}'_l + 2\lambda\mathbf{p}_m^T\mathbf{c}' + \mathbf{p}_l'^T\mathbf{p}'_l + 2\lambda\mathbf{p}_l'^T\mathbf{c}' + \lambda^2\mathbf{c}'^T\mathbf{c}' \\
\implies &\frac{d}{d\lambda} \ \mathbf{p}_m^T\mathbf{p}_m - 2\mathbf{p}_m^T\mathbf{p}'_l + 2\lambda\mathbf{p}_m^T\mathbf{c}' + \mathbf{p}_l'^T\mathbf{p}'_l + 2\lambda\mathbf{p}_l'^T\mathbf{c}' + \lambda^2\mathbf{c}'^T\mathbf{c}' = 0 \\
\implies &2\mathbf{p}_m^T\mathbf{c}' + 2\mathbf{p}_l'^T\mathbf{c}' + 2\lambda\mathbf{c}'^T\mathbf{c}' = 0 \\
\implies &\lambda = -\frac{\mathbf{p}_m^T\mathbf{c}' + \mathbf{p}_l'^T\mathbf{c}'}{\mathbf{c}'^T\mathbf{c}'}
\end{aligned} \tag{A.22}$$

A.3 Determining Self-Occluded Vertexes

With the camera matrix, we can determine which parts of a 3D model are occluded due to the geometry of the model and the viewpoint of the camera. The camera matrix tells us the camera center, $\mathbf{c}$ as described in Appendix A.2. Assuming any given 3D model has both the set of vertexes making up the model and the triangulation specifying which triple of vertexes make up surface triangle on the model, we can use this to find self-occluded vertexes. For any given vertex on the model, p_0, we know it is occluded if the line segment between it and the camera center intersects any triangle on the model, as shown in Fig. A.3. In order to determine if such an intersection occurs with a triangle made up of three vertexes, $\mathbf{v}_0$, $\mathbf{v}_1$, and $\mathbf{v}_2$, we first have to find the plane that the triangle lies on. Any point,

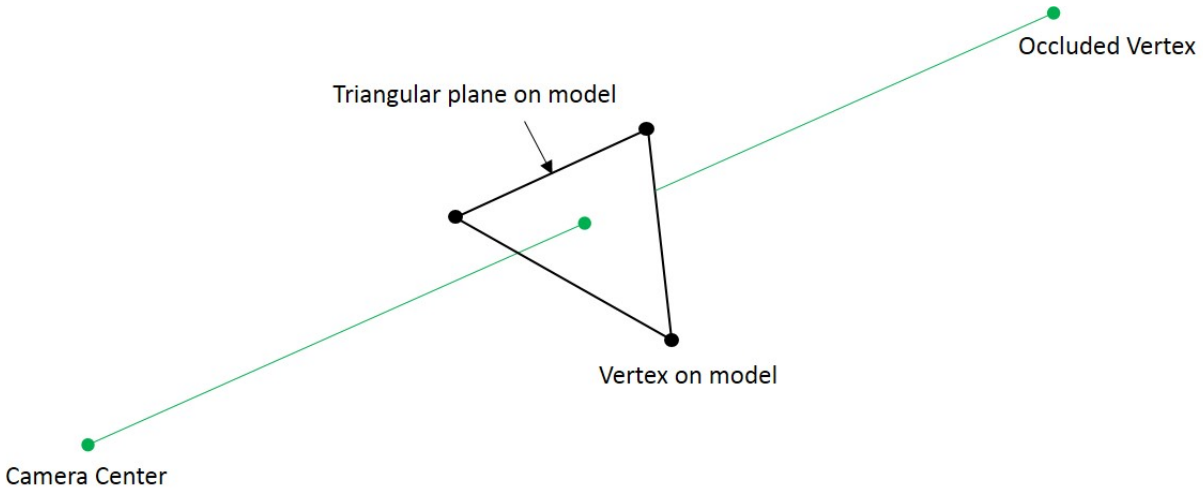

Figure A.3: An illustration of how a vertex on the model will be occluded if the line segment joining it and the camera center intersect another triangle on the model. This lien segment is the viewing line and must be clear of obstacles in order for the vertex to project onto the image plane.

$\mathbf{x}$, that lies on a plane must satisfy

$$\mathbf{n}^T(\mathbf{x} - \mathbf{p}) = 0 \tag{A.23}$$

where $\mathbf{n}$ is the vector normal to the plane and $\mathbf{p}$ is a point on the plane. Since the triangle lies on the plane we are interested in, any point on the triangle can be chosen as $\mathbf{p}$. To find $\mathbf{n}$, we can take the cross product of the two vectors made up by $\mathbf{a} = \mathbf{v}_1 - \mathbf{v}_0$ and $\mathbf{a} = \mathbf{v}_2 - \mathbf{v}_0$ as seen in Fig. A.4. We know that any point on the line segment, $\mathbf{p}_0 - \mathbf{c}$, can be expressed as

$$\mathbf{p}(\lambda) = \mathbf{p}_0 + \lambda(\mathbf{c} - \mathbf{p}_0),\ 0 \leq \lambda \leq 1 \tag{A.24}$$

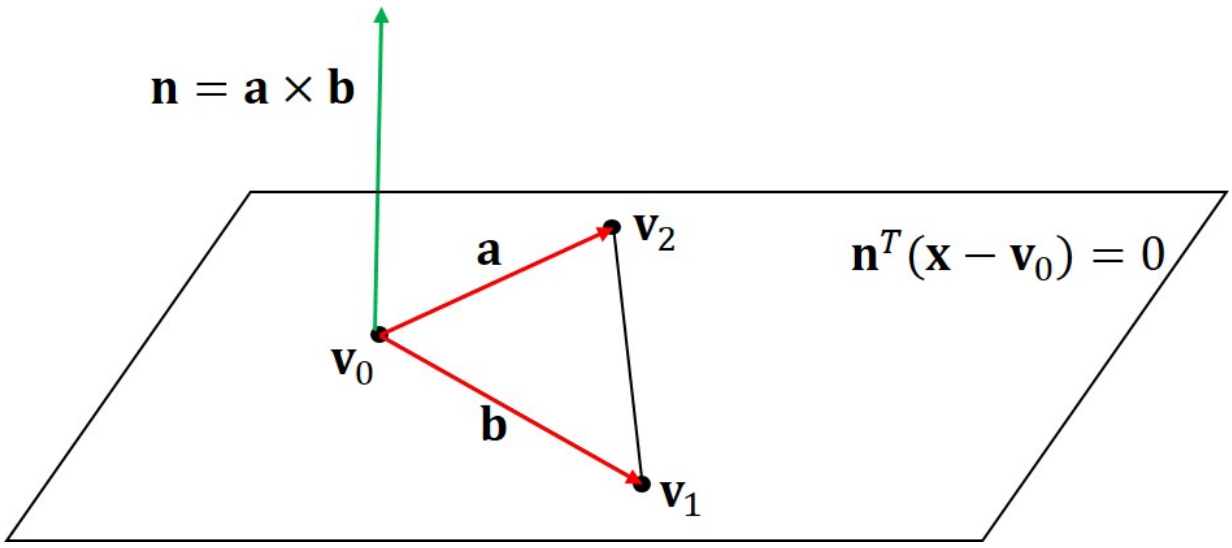

Figure A.4: The plane that the surface triangle made up by $\mathbf{v}_0$, $\mathbf{v}_1$, and $\mathbf{v}_2$.

We can find the intersection between the line segment and the plane by finding the point, $\mathbf{p}(\lambda)$, that satisfies the plane equation in Eqn. A.23. Therefore,

$$\begin{aligned}
\mathbf{n}^T(\mathbf{p}(\lambda) - \mathbf{v}_0) &= 0 \\
\mathbf{n}^T((\mathbf{p}_0 + \lambda(\mathbf{c} - \mathbf{p}_0)) - \mathbf{v}_0) &= 0 \\
\mathbf{n}^T\mathbf{p}_0 + \lambda\mathbf{n}^T(\mathbf{c} - \mathbf{p}_0) - \mathbf{n}^T\mathbf{v}_0 &= 0 \\
\mathbf{n}^T(\mathbf{p}_0 - \mathbf{v}_0) + \lambda\mathbf{n}^T(\mathbf{c} - \mathbf{p}_0) &= 0 \\
\lambda\mathbf{n}^T(\mathbf{c} - \mathbf{p}_0) &= \mathbf{n}^T(\mathbf{v}_0 - \mathbf{p}_0) \\
\lambda &= \frac{\mathbf{n}^T(\mathbf{v}_0 - \mathbf{p}_0)}{\mathbf{n}^T(\mathbf{c} - \mathbf{p}_0)}
\end{aligned} \tag{A.25}$$

If $0 \leq \lambda \leq 1$, then the line segment intersects the plane at $\mathbf{p}_i = \mathbf{p}(\lambda)$. However, we still need to check if the intersection lies within the triangle defined. To check this, we can first find the Barycentric coordinates of the point with respect to the triangle. To find these coordinates, we project the vector, $\mathbf{p}(\lambda) - \mathbf{v}_0$ onto the two sides of the triangle, $\mathbf{a}$ and $\mathbf{b}$. These projection coordinates, s and t respectively, are known as the Barycentric coordinates. If both of these values lie between 0 and 1 and add up to be less than 1, then

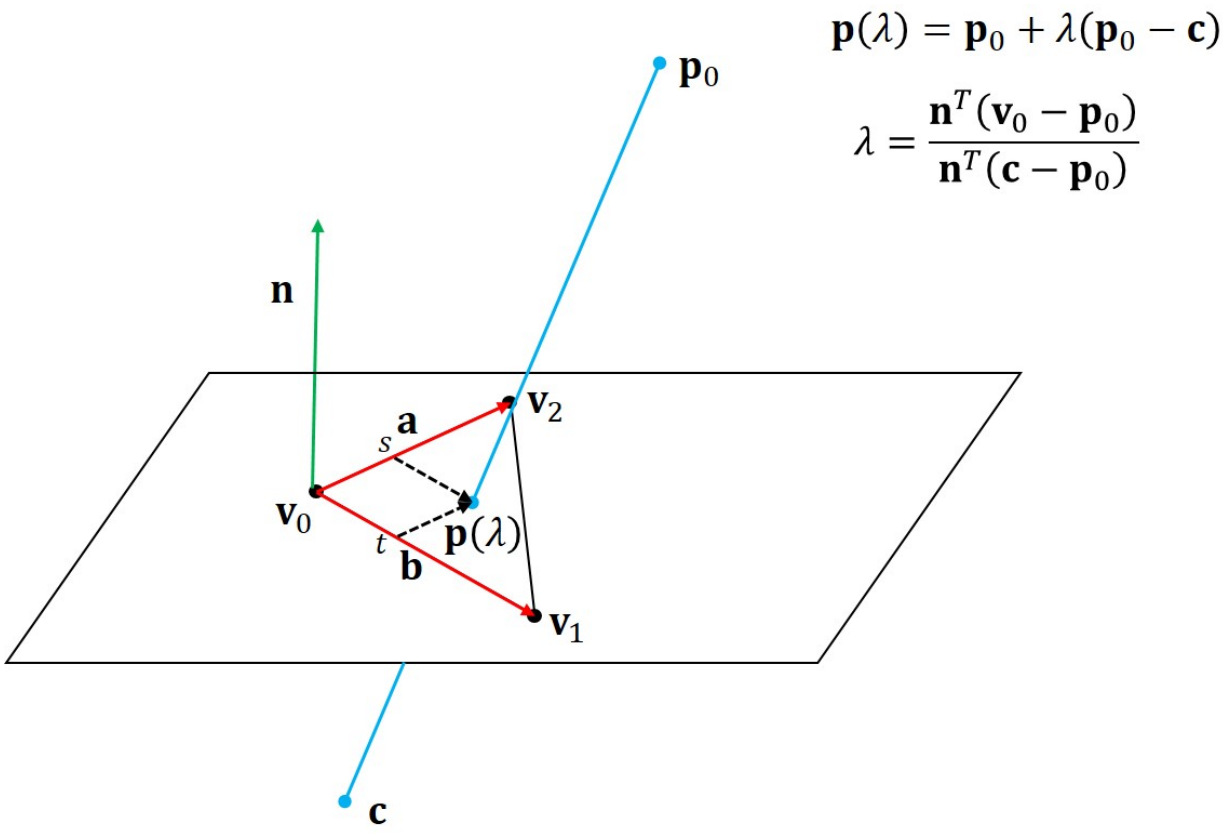

Figure A.5: The intersection of the line segment and the plane created by a surface triangle. When $0 \leq s \leq 1$, $0 \leq t \leq 1$, and $s + t \leq 1$, then the point $\mathbf{p}(\lambda)$ lies within the surface triangle and we know that the point $\mathbf{p}_0$ is occluded from the camera's point of view.

the point falls within the triangle. If this is the case, the point cannot be seen from the camera's point of view and can be marked as occluded by the object. However, to check each triangle in a model can be very time consuming as we have over 37k vertexes and 53k triangles in our 3D model of a face. Instead of checking every combination, we can try to make it more efficient by using the knowledge of the 2D projections of the vertexes. We know that any vertex that is occluded has to project to a very similar region to another vertex as our model is quite dense. Using this information, for a given vertex, we only need to check the intersection with triangles made up of vertexes that project to a similar location as shown in Fig. A.6.

Figure A.6: A single vertex projected onto the image plane is shown in green. To determine if this vertex is occluded by other parts of the 3D model, we only need to check for intersections with the triangles made up of vertexes in the yellow circle. Anything outside this would be too far away to occlude the point given the density of the model.

Appendix B

Thin Plate Splines

The original idea behind a Thin Plate Spline (TPS) warp was to find out how to minimally bend a thin, flat sheet of metal such that specified points on the sheet were at desired heights. These heights could be anywhere but were hard constraints. While many bends of the sheet could achieve such constraints, the more energy that was put into bending the sheet of metal, the more expensive the process would be and the weaker the metal would become. By minimizing the bending energy used, a smooth deformation of the sheet of metal could be found. TPS functions have been used since the late 1980s for image warping [71] as well. We can think of the $x-y$ coordinate system as the thin plane and the desired change in either the x or y coordinate, δ_x or δ_y, as the desired height as seen in Fig. B.1. In this way, two TPS functions could be created that, together, form a warping function that could be applied to any point in the plane and a new, warped point could be found. The advantage of these functions was that they guaranteed the control points would move to the desired location, the rest of the points would smoothly interpolate between them, and solving for the parameters of the TPS function had a closed form solution as long as you had a minimum of 4 points and the $(x, y, \delta_{(x/y)})$ points were not co-planar in 3D.

The 2D TPS function for the change in x that gives the minimally bent plane that

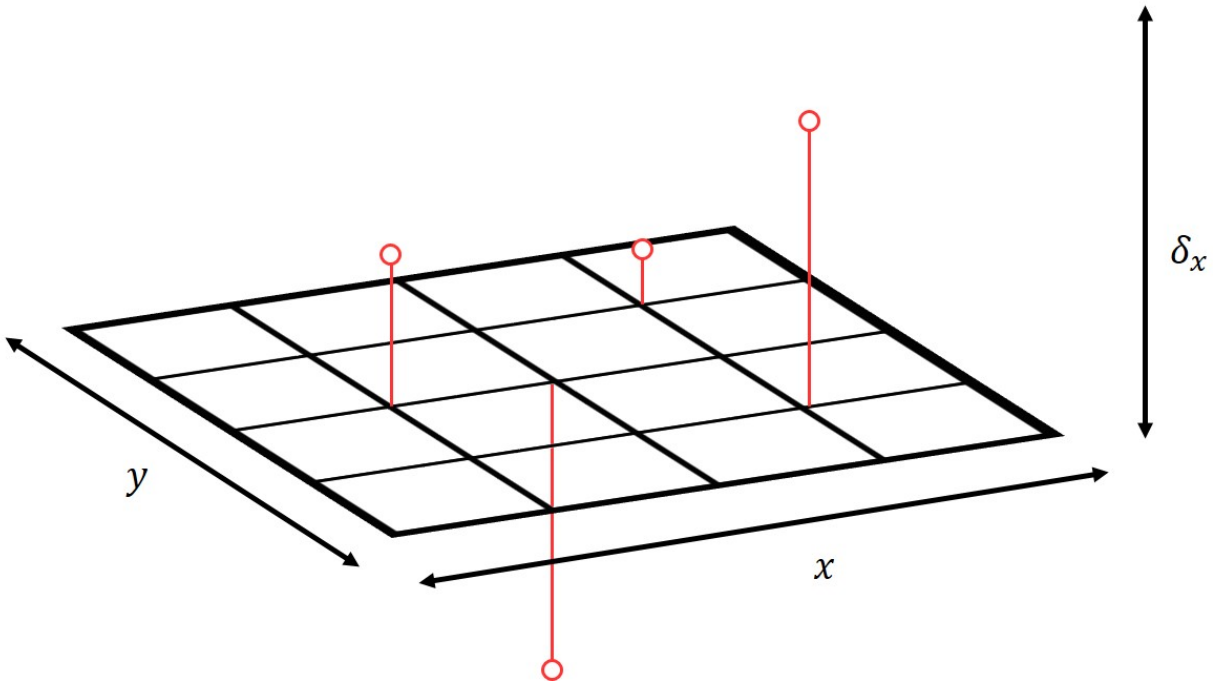

Figure B.1: The $x-y$ plane can be considered as a thin plane that we want to bend to different heights, representing the δ_x values. These are represented by the red dots in the image. The (x, y) coordinates where the red lines meet the plane are the input points to the TPS function for these constraints. Once the TPS function is found for this plane, the δ_x for the rest of the points on this plane can be found by passing those coordinates through the TPS function.

meets a set of n control points is

$$\begin{aligned} f_{\Delta_x}(x, y) &= b_1 + b_2 x + b_3 y + \sum_{i=1}^{n} w_{ix} U\left(\left|\,(x_i, y_i) - (x, y)\,\right|\right) \\ s.t. \sum_{i=1}^{n} w_{ix} &= 0, \ \sum_{i=1}^{n} w_{ix} x_i = 0, \ \sum_{i=1}^{n} w_{ix} y_i = 0 \end{aligned} \tag{B.1}$$

where b_1, b_2, b_3, and $w_i x$ are the parameters of the function, x and y are the input points, x_i and y_i are the coordinates of the i^{th} control point, and $\left|\,(x_i, y_i) - (x, y)\,\right|$ is the Euclidean distance between the i^{th} control point and the input point. The U function is defined as

$$U(r) = r^2 \log r \tag{B.2}$$

with $U(0) = 0$. When we break Eqn. B.1 down into the b terms and the summation using

the w terms, we can see that the equation has two parts. The first part, $b_1 + b_2x + b_3y$, is the equation of the closest unbent plane to the constraints. The remainder of the function are the terms that control the smooth deformation of the coordinate space. This is why we need points that are not co-planar as any set of co-planar points can be fit by the b terms alone and will result in a degenerate solution. In these cases, there is no need for a TPS function. To find the parameters of the TPS function given a set of constraints, we can rewrite Eqn. B.1 as

$$\begin{bmatrix} \mathbf{U} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} w_{1x} \\ w_{2x} \\ \vdots \\ w_{nx} \\ b_{1x} \\ b_{2x} \\ b_{3x} \end{bmatrix} = \begin{bmatrix} \delta_{x_1} \\ \delta_{x_2} \\ \vdots \\ \delta_{x_n} \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{B.3}$$

where

$$\mathbf{U} = \begin{bmatrix} U(r_{11}) & \cdots & U(r_{1n}) \\ \vdots & \ddots & \vdots \\ U(r_{n1}) & \cdots & U(r_{nn}) \end{bmatrix} \quad \mathbf{P} = \begin{bmatrix} 1 & x_1 & y_1 \\ \vdots & \vdots & \vdots \\ 1 & x_n & y_n \end{bmatrix} \tag{B.4}$$

and $r_{ij} = |\,(x_i, y_i) - (x_j, y_j)\,|$ or the distance between the i^{th} and j^{th} control points. From

this system of linear equation, we can solve for the parameters of the TPS function as

$$\begin{bmatrix} w_{1x} \\ w_{2x} \\ \vdots \\ w_{nx} \\ b_{1x} \\ b_{2x} \\ b_{3x} \end{bmatrix} = \begin{bmatrix} \mathbf{U} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{0} \end{bmatrix}^{-1} \begin{bmatrix} \delta_{x_1} \\ \delta_{x_2} \\ \vdots \\ \delta_{x_n} \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{B.5}$$

As long as we have at least 4 points that are not co-planar, this matrix inverse will exist and we can find our solution. Extending this to 3D is very simple and results in Eqn. 3.2. Solving for the parameters of the 3D TPS equation follows exactly the same steps except that 5 points which are not co-planar are required in 3D. Finding the TPS function for δ_x and δ_y is very simple, especially as the matrix that is inverted does not change between these two functions. Therefore, we only have to perform this inverse once. Additionally, this matrix only grows linearly in the number of control points used making computing the matrix inverse relatively simple for small numbers of control points. Once a TPS function is found for δ_x and δ_y, the transformed version of any point on the plane, (x, y), can be found as

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x + f_{\delta_x}(x, y) \\ x + f_{\delta_y}(x, y) \end{bmatrix} \tag{B.6}$$

To quickly evaluate this over all desired points, we can set up a similar equation to Eqn. B.3. In our new $\mathbf{U}$ matrix, r_{ij} is the distance between the i^{th} control point and the j^{th} input point. Similarly, the i^{th} row of the $\mathbf{P}$ matrix contains the i^{th} input point with a 1 preprended. Both of these matrices can be computed offline for a given input model like our generic 3D face model. Once these are created and the parameters have been found,

computing δ_x and δ_y is simply computing

$$\begin{bmatrix} \mathbf{U} & \mathbf{P} \end{bmatrix} \begin{bmatrix} w_{1x} & w_{1y} \\ w_{2x} & w_{2y} \\ \vdots & \vdots \\ w_{nx} & w_{ny} \\ b_{1x} & b_{1y} \\ b_{2x} & b_{2y} \\ b_{3x} & b_{3y} \end{bmatrix} = \begin{bmatrix} \delta_{x_1} & \delta_{y_1} \\ \delta_{x_2} & \delta_{y_2} \\ \vdots & \vdots \\ \delta_{x_n} & \delta_{y_n} \end{bmatrix} \tag{B.7}$$

Bibliography

[1] R. Ranjan, C. D. Castillo, and R. Chellappa, "L2-constrained softmax loss for discriminative face verification," *CoRR*, vol. abs/1703.09507, 2017. [Online]. Available: http://arxiv.org/abs/1703.09507 1, 2.2, 4.3.1

[2] L. Xiong, J. Karlekar, J. Zhao, J. Feng, S. Pranata, and S. Shen, "A good practice towards top performance of face recognition: Transferred deep feature fusion," *CoRR*, vol. abs/1704.00438, 2017. [Online]. Available: http://arxiv.org/abs/1704.00438 1

[3] J. Yang, P. Ren, D. Zhang, D. Chen, F. Wen, H. Li, and G. Hua, "Neural aggregation network for video face recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[4] I. Masi, A. T. an Trãn, T. Hassner, J. T. Leksut, and G. Medioni, "Do we really need to collect millions of faces for effective face recognition?" in *The European Conference on Computer Vision (ECCV)*, 2016. 1

[5] W. AbdAlmageed, Y. Wu, S. Rawls, S. Harel, T. Hassner, I. Masi, J. Choi, J. Lekust, J. Kim, P. Natarajan, R. Nevatia, and G. Medioni, "Face recognition using deep multi-pose representations," in *The IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2016. 1

[6] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," University of Massachusetts, Amherst, Tech. Rep. 07-49, 2007. 1, 1.1.3

[7] B. F. Klare, B. Klein, E. Taborsky, A. Blanton, J. Cheney, K. Allen, P. Grother, A. Mah, M. Burge, and A. K. Jain, "Pushing the frontiers of unconstrained face detection and recognition: IARPA Janus Benchmark A," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1, 1.1.3

[8] S. Yang, P. Luo, C. C. Loy, and X. Tang, "WIDER FACE: A face detection benchmark," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1.1.1

[9] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," in *The European Conference on Computer Vision (ECCV)*, 2016. 1.1.1

[10] C. Zhu, Y. Zheng, K. Luu, and M. Savvides, "CMS-RCNN: Contextual multi-scale region-based CNN for unconstrained face detection," in *Deep Learning for Biometrics*, B. Bhanu and A. Kumar, Eds. Springer International Publishing, 2017, pp. 57–79. 1.1.1, 2.2, 3.5.1

[11] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li, "S3FD: Single shot scale-invariant face detector," in *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 1.1.1

[12] M. Najibi, P. Samangouei, R. Chellappa, and L. S. Davis, "SSH: Single stage headless face detector," in *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 1.1.1

[13] P. Hu and D. Ramanan, "Finding tiny faces," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1.1.1

[14] S. Zafeiriou, G. Trigeorgis, G. Chrysos, J. Deng, and J. Shen, "The Menpo facial landmark localisation challenge: A step towards the solution," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017. 1.2, 1.1.2,

1.4

[15] 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," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1.2, 1.1.2, 1.4

[16] X. Zhu, Z. Lei, J. Yan, D. Yi, and S. Z. Li, "High-fidelity pose and expression normalization for face recognition in the wild," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1.1.2, 2.1, 2.2

[17] J. Heo and M. Savvides, "Gender and ethnicity specific generic elastic models from a single 2D image for novel 2D pose face synthesis and recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 12, pp. 2341–2350, Dec. 2012. 1.1.2, 2.1

[18] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning face representation from scratch," *CoRR*, vol. abs/1411.7923, 2014. [Online]. Available: http://arxiv.org/abs/1411.7923 1.1.3

[19] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker, "Multi-PIE," in *The IEEE International Conference on Automatic Face and Gesture Recognition*, 2008. 1.1.3, 3.5.1, 3.5.1, 3.6.1, 4.3.1

[20] ——, "Multi-PIE," *Image and Vision Computing*, vol. 28, no. 5, pp. 807–813, May 2010. [Online]. Available: http://dx.doi.org/10.1016/j.imavis.2009.08.002 1.1.3, 3.5.1, 3.5.1, 3.6.1, 4.3.1

[21] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *The European Conference on Computer Vision (ECCV)*, 2016. 1.1.3, 2.2, 4.3.1

[22] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active shape models-their

training and application," *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 38–59, 1995. 2.1

[23] T. F. Cootes, G. J. Edwards, C. J. Taylor *et al.*, "Active appearance models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 681–685, 2001. 2.1

[24] T. F. Cootes, G. Edwards, and C. Taylor, "Comparing active shape models with active appearance models," in *The British Machine Vision Conference*. BMVA Press, 1999, pp. 173–182. 2.1

[25] K. Seshadri and M. Savvides, "Robust modified active shape model for automatic facial landmark annotation of frontal faces," in *The IEEE International Conference on Biometrics: Theory, Applications, and Systems (BTAS)*, 2009. 2.1

[26] X. Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 2.1, 3.5.1

[27] X. Xiong and F. De la Torre, "Supervised descent method and its applications to face alignment," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 2.1, 3.5.3

[28] ——, "Global supervised descent method," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2.1

[29] Z. Zhang, P. Luo, C. C. Loy, and X. Tang, "Learning deep representation for face alignment with auxiliary attributes," *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol. 38, no. 5, pp. 918–930, May 2016. 2.1

[30] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, Oct 2016. 2.1

[31] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001. 2.1

[32] R. Ranjan, V. M. Patel, and R. Chellappa, "Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition," *CoRR*, vol. abs/1603.01249, 2016. [Online]. Available: http://arxiv.org/abs/1603.01249 2.1

[33] F. Liu, C. Shen, G. Lin, and I. Reid, "Learning depth from single monocular images using deep convolutional neural fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 2024–2039, Oct 2016. 2.1

[34] A. Bansal, B. Russell, and A. Gupta, "Marr revisited: 2D-3D alignment via surface normal prediction," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2.1

[35] A. Roy and S. Todorovic, "Monocular depth estimation using neural regression forest," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2.1

[36] F. Liu, C. Shen, and G. Lin, "Deep convolutional neural fields for depth estimation from a single image," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2.1

[37] B. Li, C. Shen, Y. Dai, A. van den Hengel, and M. He, "Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2.1

[38] U. Prabhu, J. Heo, and M. Savvides, "Unconstrained pose-invariant face recognition using 3D generic elastic models," *IEEE Transactions on Pattern Analysis and Machine*

Intelligence, vol. 33, no. 10, pp. 1952–1961, Oct 2011. 2.1

[39] T. Hassner, S. Harel, E. Paz, and R. Enbar, "Effective face frontalization in unconstrained images," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2.1, 2.2

[40] A. Jourabloo and X. Liu, "Pose-invariant 3D face alignment," in *The IEEE International Conference on Computer Vision (ICCV)*, 2015. 2.1, 3

[41] X. Zhu, Z. Lei, X. Liu, H. Shi, and S. Z. Li, "Face alignment across large poses: A 3D solution," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2.1, 2.1, 3, 3.5.1, 3.5.1, 3.12, 3.5.3

[42] A. Bulat and G. Tzimiropoulos, "How far are we from solving the 2D & 3D face alignment problem? (and a dataset of 230,000 3d facial landmarks)," in *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 2.1

[43] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *The European Conference on Computer Vision (ECCV)*, 2016. 2.1

[44] Y. Sun, X. Wang, and X. Tang, "Deep learning face representation from predicting 10,000 classes," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 2.2

[45] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," in *Advances in Neural Information Processing Systems (NIPS)*, 2014. 2.2

[46] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 2.2

[47] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*

(NIPS), 2012. 2.2

[48] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015. 2.2

[49] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *The International Conference on Learning Representations (ICLR)*, 2015. 2.2

[50] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NPIS)*, 2015. 2.2

[51] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *The European Conference on Computer Vision (ECCV)*, 2016. 2.2

[52] Y. Zheng, C. Zhu, K. Luu, C. Bhagavatula, T. H. N. Le, and M. Savvides, "Towards a deep learning framework for unconstrained face detection," in *The IEEE International Conference on Biometrics Theory, Applications and Systems (BTAS)*, 2016. 2.2, 3.5.1

[53] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *The International Conference on Machine Learning (ICML)*, 2015. 2.2

[54] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 664–676, April 2017. 2.2

[55] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *The British Machine Vision Conference*, 2015. 2.2

[56] Y. Sun, D. Liang, X. Wang, and X. Tang, "Deepid3: Face recognition with very deep neural networks," *CoRR*, vol. abs/1502.00873, 2015. [Online]. Available: http://arxiv.org/abs/1502.00873 2.2

[57] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *The IEEE International Conference on Computer Vision (ICCV)*, 2015. 2.2

[58] T. Y. Lin, A. RoyChowdhury, and S. Maji, "Bilinear CNN models for fine-grained visual recognition," in *The IEEE International Conference on Computer Vision (ICCV)*, 2015. 2.2

[59] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2.2

[60] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2.2

[61] L. Wolf, T. Hassner, and I. Maoz, "Face recognition in unconstrained videos with matched background similarity," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. 2.2

[62] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2.2

[63] I. Kemelmacher-Shlizerman, S. M. Seitz, D. Miller, and E. Brossard, "The megaface benchmark: 1 million faces for recognition at scale," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2.2

[64] Y. Zheng, D. K. Pal, and M. Savvides, "Ring loss: Convex feature normalization for

face recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2.2

[65] J. Yim, H. Jung, B. Yoo, C. Choi, D. Park, and J. Kim, "Rotating your face using multi-task deep neural network," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2.2, 4

[66] L. Tran, X. Yin, and X. Liu, "Disentangled representation learning GAN for pose-invariant face recognition," in *The IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017. 2.2, 4.4

[67] X. Peng, X. Yu, K. Sohn, D. N. Metaxas, and M. Chandraker, "Reconstruction-based disentanglement for pose-invariant face recognition," in *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 2.2, 3.6.1, 4.3.3, 4.3.3, 4.2, 4.4

[68] I. Masi, S. Rawls, G. Medioni, and P. Natarajan, "Pose-aware face recognition in the wild," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2.2, 4

[69] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter, "A 3D face model for pose and illumination invariant face recognition," in *The IEEE International Conference on Advanced Video and Signal based Surveillance for Security, Safety and Monitoring in Smart Environments*, 2009. 3

[70] J. Booth, A. Roussos, S. Zafeiriou, A. Ponniahy, and D. Dunaway, "A 3d morphable model learnt from 10,000 faces," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[71] F. L. Bookstein, "Principal warps: Thin-plate splines and the decomposition of deformations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 6, pp. 567–585, Jun. 1989. 3, B

[72] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial

transformer networks," *CoRR*, vol. abs/1506.02025, 2015. [Online]. Available: http://arxiv.org/abs/1506.02025 3.1, 3.1, 3.3

[73] P. N. Belhumeur, D. W. Jacobs, D. J. Kriegman, and N. Kumar, "Localizing parts of faces using a consensus of exemplars," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. 3.5.1

[74] V. Le, J. Brandt, Z. Lin, L. Bourdev, and T. S. Huang, "Interactive facial feature localization," in *The European Conference on Computer Vision (ECCV)*, 2012. 3.5.1

[75] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, "300 faces in-the-wild challenge: The first facial landmark localization challenge," in *The IEEE International Conference on Computer Vision (ICCV) Workshops*, 2013. 3.5.1

[76] 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 *The First IEEE International Workshop on Benchmarking Facial Image Analysis Technologies*, 2011. 3.5.1

[77] 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," in *The IEEE International Conference on Computer Vision (ICCV)*, 2013. 3.5.3

[78] X. P. Burgos-Artizzu, P. Perona, and P. Dollár, "Robust face landmark estimation under occlusion," in *Tthe IEEE International Conference on Computer Vision (ICCV)*, 2013. 3.5.3

[79] X. Cao, Y. Wei, F. Wen, and J. Sun, "Face alignment by explicit shape regression," *International Journal of Computer Vision*, vol. 107, no. 2, pp. 177–190, Apr. 2014. 3.5.3

[80] J. A. Black, M. Gargesha, K. Kahol, P. Kuchi, and S. Panchanathan, "Framework for performance evaluation of face recognition algorithms," in *Proceedings of SPIE - The*

International Society for Optical Engineering, vol. 4862, 2002. 3.6.1

[81] G. Little, S. Krishna, J. Black, and S. Panchanathan, "A methodology for evaluating robustness of face recognition algorithms with respect to variations in pose angle and illumination angle," in *The IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2005. 3.6.1

[82] N. Gourier, D. Hall, and J. L. Crowley, "Estimating face orientation from robust detection of salient facial features," in *The International Conference on Pattern Recognition (ICPR) International Workshop on Visual Observation of Deictic Gestures*, 2004. 3.6.1

[83] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems (NIPS)*, 2014. 4.2

[84] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *CoRR*, vol. abs/1511.06434, 2015. [Online]. Available: http://arxiv.org/abs/1511.06434 4.2

[85] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," *CoRR*, vol. abs/1701.07875, Jan. 2017. [Online]. Available: https://arxiv.org/abs/1701.07875 4.2

[86] S. Sengupta, J. Chen, C. Castillo, V. M. Patel, R. Chellappa, and D. W. Jacobs, "Frontal to profile face verification in the wild," in *The IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2016. 4.3, 4.3.1, 4.3.5, 4.4

[87] M. Kan, S. Shan, and X. Chen, "Multi-view deep network for cross-view classification," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 4.3.2, 4.1, 4.3.3

[88] M. Kan, S. Shan, H. Zhang, S. Lao, and X. Chen, "Multi-view discriminant analysis," in *The European Conference on Computer Vision (ECCV)*, 2012. 4.1

[89] A. Sharma, A. Kumar, H. Daume, and D. W. Jacobs, "Generalized multiview analysis: A discriminative latent space," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 4.1

[90] S. Sankaranarayanan, A. Alavi, C. D. Castillo, and R. Chellappa, "Triplet probabilistic embedding for face verification and clustering," in *The IEEE International Conference on Biometrics Theory, Applications and Systems (BTAS)*, 2016. 4.4