EXPLORING THE EFFECTS OF FRONTALIZATION AND DATA SYNTHESIS
ON FACE RECOGNITION

A Dissertation

Submitted to the Graduate School
of the University of Notre Dame
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

by

Sandipan Banerjee

Patrick J. Flynn, Co-Director

Kevin W. Bowyer, Co-Director

Graduate Program in Computer Science and Engineering
Notre Dame, Indiana
July 2019
EXPLORING THE EFFECTS OF FRONTALIZATION AND DATA SYNTHESIS ON FACE RECOGNITION

Abstract

by

Sandipan Banerjee

Automatic face recognition performance has improved remarkably in the last decade. Much of this success can be attributed to the development of deep learning techniques like convolutional neural networks (CNNs). But the training process of CNNs requires a large amount of clean and correctly labelled data. In the first part of this work, we try to find the ideal orientation (facial pose, shape, context) of this data for training and testing such CNNs. If a CNN is intended to work with non-frontal face images, should this training data be diverse in terms of facial poses, or should face images be frontalized as a pre-processing step? To answer these questions we evaluate a set of popular facial landmarking and pose frontalization algorithms to understand their effect on facial recognition performance. We also introduce a new landmarking and frontalization scheme that operates over a single image without the need for a subject-specific 3D model, and perform a comparative analysis between the new scheme and other methods in the literature.

Secondly, we analyze the usefulness of synthetic images in improving the face recognition pipeline while taking into account its practicality from a computation stand-point. In this regard, we propose a novel face synthesis method for augmentation of existing face image datasets. An augmented dataset reduces overfitting, which in turn, can enhance the face representation capability of a CNN. Our method, start-
ing off with actual face images from an existing dataset, can generate a large number of synthetic images of real and synthetic identities, without the identity-labeling and privacy complications that come from downloading images from the web. Additionally, we develop a multi-scale generative adversarial network (GAN) model to hallucinate realistic context (forehead, hair, neck, clothes) and background pixels automatically from a single input face mask, without any user supervision. Our model is composed of a cascaded network of GAN blocks, each tasked with hallucination of missing pixels at a particular resolution while guiding the synthesis process of the next GAN block. Multiple experiments are performed to assess the realism of our synthetic face images and validate their effectiveness as supplemental data for training CNNs, and as distractors to test the robustness of trained model snapshots.
To Ma, Baba, Chhoto Mama, Sreya, and Ranjan Sir.
CONTENTS

FIGURES . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . vi

TABLES . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . xiv

ACKNOWLEDGMENTS . . . . . . . . . . . . . . . . . . . . . . . . . . . . . xv

CHAPTER 1: INTRODUCTION . . . . . . . . . . . . . . . . . . . . . . . . . 1
  1.1 Motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
  1.2 Exploring Face Image Frontalization (see Chapter 3 for details) . . 4
  1.3 Synthesizing Face Images to Improve Recognition (see Chapters 4, 5
       and 6 for details) . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
  1.4 Outline . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

CHAPTER 2: RELATED WORK . . . . . . . . . . . . . . . . . . . . . . . . 12
  2.1 Face Recognition . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
  2.2 Facial Landmarking . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
  2.3 Face Image Frontalization . . . . . . . . . . . . . . . . . . . . . . . . 14
  2.4 Face Image Synthesis . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
  2.5 Facial Inpainting . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
  2.6 Face Swapping . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
  2.7 Data Augmentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20

CHAPTER 3: EXPLORING THE EFFECTS OF FRONTALIZATION ON
FACE RECOGNITION . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
  3.1 Description of Chosen Landmarking and Frontalization Methods . . 24
      3.1.1 Landmarking . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
      3.1.2 Frontalization . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26
  3.2 Our Frontalization Method (OFM) . . . . . . . . . . . . . . . . . . . 27
      3.2.1 Face Detection, Landmarking and Model Fitting . . . . . . . . 27
      3.2.2 3D Transformation and Texture Mapping . . . . . . . . . . . . 28
      3.2.3 Image Correction and Postprocessing . . . . . . . . . . . . . . 29
  3.3 Face Recognition Pipeline . . . . . . . . . . . . . . . . . . . . . . . . 30
      3.3.1 Training Data: CASIA-WebFace . . . . . . . . . . . . . . . . . 31
      3.3.2 Pre-processing Methods . . . . . . . . . . . . . . . . . . . . . . 33
      3.3.3 CNN Architecture: VGG-FACE . . . . . . . . . . . . . . . . . . 33
3.3.4 Testing Datasets .......................................................... 35
3.3.5 Feature Extraction and Scoring ........................................... 36
3.4 Method Yield Rates .......................................................... 36
3.5 Experiments and Results ..................................................... 39
  3.5.1 Methodology ............................................................. 39
  3.5.2 Results of Recognition Experiments ................................. 40
3.6 Conclusion ................................................................. 55

CHAPTER 4: SYNTHESIS OF REALISTIC EXAMPLE FACE IMAGES .... 56
  4.1 Collection of the SREFI Donor Set ....................................... 58
  4.2 Our Synthesis Method - SREFI ........................................... 59
    4.2.1 Landmarking and Triangulation .................................... 59
    4.2.2 Selection of the Donor Pool ....................................... 60
    4.2.3 Attribute Based Reshaping of Facial Parts ....................... 62
    4.2.4 Triangle Replacement ................................................ 64
    4.2.5 Adjusting the Color Distribution of Replacement Triangles .. 64
    4.2.6 Blending in the Replaced Triangles ............................... 65
  4.3 Experiments and Results ................................................ 68
    4.3.1 Human Rater Study .................................................. 68
      4.3.1.1 Experiment 1 ................................................... 68
      4.3.1.2 Experiment 2 .................................................. 70
      4.3.1.3 Experiment 3 .................................................. 72
    4.3.2 Evaluating Uniqueness - Face Matching Experiments with VGG-
         FACE ................................................................. 76
    4.3.3 Evaluating Stability - Training VGG-FACE and Testing on the
         GBU Dataset .......................................................... 83
  4.4 Discussion .............................................................. 86

CHAPTER 5: FAST RENDERING OF SYNTHETIC 3D FACIAL MASKS .... 88
  5.1 The Proposed Method .................................................... 88
    5.1.1 Synthetic Texture Generation ...................................... 88
    5.1.2 3D Face Mask Construction ....................................... 94
  5.2 Experiments and Results ................................................ 99
    5.2.1 Experiment 1: Effectiveness in Data Augmentation for CNN
         Training .............................................................. 99
    5.2.2 Experiment 2: Effectiveness as Generator of Synthetic Distrac-
         tors For Model Testing ................................................. 105
    5.2.3 Experiment 3: Comparison with Popular GANs .................... 109
    5.2.4 Experiment 4: Impact of Facial Shape on Subject Identity ...... 110
    5.2.5 Experiment 5: Impact of Context and Background on Face
         Recognition Performance ............................................. 114
  5.3 Discussion ............................................................. 117
FIGURES

1.1 Images of the same subject with five different facial yaw angles, taken from the CMU-MultiPIE face database [54]. The facial yaw ranges from 0 degree to 90 degrees in both directions. ............... 2

1.2 Examples of different pre-processing on a sample image: (a) original image from the CASIA-WebFace dataset [175]; (b) 2D aligned – no frontalization, (c) Zhu and Ramanan [191] & Hassner et al. [60], (d) Kazemi and Sullivan [83] & Hassner et al., (e) CMR & our frontalization method (OFM), (f) CMR & Hassner et al. [60], (g) Zhu and Ramanan [191] & OFM, and (h) Kazemi and Sullivan [83] & OFM. The left and right images are frontalized asymmetrically and symmetrically respectively for (c), (d), (e), (f), (g) and (h). Note how different the results look for each approach. .................. 4

1.3 While traditional augmentation methods have explored face image cropping, translation and scaling (left column), we can augment a dataset by creating synthetic face images of existing identities (middle column) and new synthetic identities altogether (right column). This augments both the depth and breadth of a dataset. ............... 6

1.4 Varying shape and pose: each row represents three face images of the same synthetic identity at yaw values -30, 0 and +30. The face images in each top row belong to a synthetic identity while the corresponding bottom row has face images of a real identity for visual comparison. 7

1.5 Hallucinating facial context and background - (a) original face images from LFW [70] (2D aligned), (b) corresponding face masks (input), and (c) the hallucinated output generated by our cascaded network of GANs trained on [129]. All images are 128×128 in size. ............... 8

2.1 Illustration of a typical face recognition system. ............... 13

2.2 Examples of facial landmarking, key-points in yellow, reproduced from [83]. ............... 15

2.3 Examples of face frontalization: the top row consists of original images and their corresponding frontalized versions can be found in the bottom row. ............... 16
2.4 Two examples of face synthesis from [82] - both the face images are synthetically generated using a GAN [53, 81, 82], of non-existent identities, from random noise vectors.

2.5 Some examples of facial inpainting from [97] where (a) original image, (b) masked input, (c) inpainted output.

2.6 An example of face swapping using the DeepFake software [1], where the face in the original image (LEFT) is swapped with that of Nicholas Cage (RIGHT) while retaining the visual attributes.

2.7 Example of augmentation by reposing - (a) original face image from [175], (b) same face image reposed using the method proposed in [60] for augmenting the dataset. Images taken from [107].

3.1 Visualization of multiple regressors fitting the feature $x_t \in \mathbb{R}^{d \times 1}$ vs. shape update $\Delta S_t = R_t(I, S_t)$ curve.

3.2 Frontalization correspondences for the method of Hassner et al. [60]. a) Input face in non-frontal pose. b) Face matched to template pose via landmarks. c) Back-projection of template from input pose to frontal. d) Back-projection of pixel values to frontalized template, including artifacts from self-occlusion.

3.3 Overview of the proposed frontalization procedure. The procedure first detects the facial area and a number of facial landmarks in the input image (a). It then aligns a generic 3D model with the input face (b) and calculates a 3D transform that maps the aligned 3D model back to frontal pose (c). Based on the known 2D-to-3D point correspondences, a synthetic frontal view of the input face is generated (d) and post-processed to generate the final results of the frontalization (e).

3.4 The face recognition pipeline: the training phase is on the left and the testing phase is on the right of the image.


3.6 Randomly selected frames from - (a) 2D aligned tripod mounted, (b) 2D aligned handheld, (c) frontialized tripod mounted, (d), frontialized handheld PaSC videos. Columns (1) - (4) represent different subjects in the PaSC dataset. Note the difference in resolution, illumination, occlusion and facial pose in the different frames.

3.7 Frontalization success (expressed as yield rate) of the six methods over different pose angles in the CMU Multi-PIE dataset [54].
3.8 Recognition performance on the full set of handheld PaSC videos (1st pipeline). Pre-processing both the training and testing data with KS & our frontalization method (OFM) outperforms all other methods. Interestingly, the wide gap between the bottom two curves suggests that training with non pre-processed images actually hampered the face representation capability of the network (dotted curve).

3.9 Verification performance on full handheld PaSC videos (1st pipeline). The trends from Fig. 3.8 transfer to the ROC as well.

3.10 Recognition performance on the full set of stationary (controlled) PaSC videos (1st pipeline). The overall performance of all methods went up from Fig. 5 in the main text, especially for 2D alignment (dashed curve).

3.11 Verification performance on the full set of stationary (controlled) PaSC videos (1st pipeline).

3.12 Recognition performance on the common handheld PaSC videos (2nd pipeline). KS & OFM slightly exceeded the best performance reported in Fig. 3.8. The 2D alignment (dashed) curve made a big jump from Fig. 3.8 suggesting the 2D alignment bin from Table 3.1 had more difficult frames than other methods due to its higher yield.

3.13 Verification performance on the common handheld PaSC videos (2nd pipeline). Although KS & OFM slightly outperformed other methods at FAR = 0.01, a simple 2D alignment step beat other methods for higher FARs.

3.14 Recognition performance on the common set of stationary (controlled) PaSC videos (2nd pipeline). The 2D alignment network (dashed curve) outperformed all the frontalization methods in this experiment.

3.15 Verification performance on the common set of stationary (controlled) PaSC videos (2nd pipeline). The trends from Fig. 3.14 in the supplementary text followed in this experiment as well.

3.16 Recognition performance on the full set of handheld PaSC videos (1st pipeline), where PaSC was frontalized but CW was either frontalized or 2D aligned. The 2D alignment network (dashed curves) in most cases was outperformed by its corresponding frontalized network.

3.17 Verification performance on the full set of handheld PaSC videos (1st pipeline), where PaSC was frontalized but CW was either frontalized or 2D aligned. The trends from Fig. 3.16 in the supplementary text followed in this experiment as well.

3.18 Recognition performance on the full set of stationary (controlled) PaSC videos (1st pipeline), where PaSC was frontalized but CW was either frontalized or 2D aligned.
3.19 Verification performance on the full set of stationary (controlled) PaSC videos (1st pipeline), where PaSC was frontalized but CW was either frontalized or 2D aligned.

4.1 While traditional augmentation methods have explored face image cropping, translation and scaling (left column), we can augment a dataset by creating synthetic face images of existing identities (middle column) and new synthetic identities altogether (right column). This augments both the depth and breadth of a dataset.

4.2 Example set of six images from the donor set. A male and female of each ethnic group present in the donor set are shown.

4.3 Workflow of the SREFI method on a high level where the base face image (a) is landmarked and triangulated (b) using the Dlib [86] software, and then the triangles are re-distributed using the centroids (c). Similar face images from the donor pool are then used to replace the base face triangles (d) to synthesize a rough version of the new texture (e), which are further made seamless using multi-scale blending (f).

4.4 Example of adjusting the facial triangles - (a) the initial triangle set, (b) the new triangles after relocating them using the centroids of the initial set (a). The different facial regions like the eyes, nose and mouth are further separated by this adjustment.

4.5 Selecting the donor pool for a base image (middle). The face images connected with heavier edges are proximal to the base face in appearance compared to the ones connected with lighter edges. The $N$ proximal faces are represented as the faces within the circular region around the base face.

4.6 Example of the color readjustment process - (a) without color redistribution, (b) re-distribution in HSV color space, (c) re-distribution in RGB color space.

4.7 Example of a synthetic face blended using (a) Laplacian pyramids [30] and (b) the Poisson blending method [125]. Notice the overlapping eye, nose and mouth regions.

4.8 Sample face images used in the three sessions of the human study respectively - (a) TOP: two real face images, BOTTOM: two synthetic face images; (b) TOP: synthetic true match pair, BOTTOM: synthetic non-match pair; (c) TOP: real and synthetic face images of an existing identity, BOTTOM: two synthetic face images of the same real identity.

4.9 Real images which generated the most inaccurate responses across the 20 participants.

4.10 Synthetic images which generated the most inaccurate responses across the 20 participants.
4.11 Face images which generated the most ‘Cannot decide’ responses across the 20 participants. The male face image is actually real and 10/20 participants chose ‘Cannot decide’ for it. The female face image is actually synthetic and 8/20 participants chose ‘Cannot decide’ for it.

4.12 Male true match pair which generated the most inaccurate responses across the 20 participants (synthetic).

4.13 Female true match pair which generated the most inaccurate responses across the 20 participants (synthetic).

4.14 Male non-match pair which generated the most inaccurate responses across the 20 participants (synthetic).

4.15 Female non-match pair which generated the most inaccurate responses across the 20 participants (synthetic).

4.16 Male pair (actually ‘non-match’) which generated most ‘Cannot decide’ responses (10/20) across the 20 participants (synthetic).

4.17 Female pair (actually ‘non-match’) which generated most ‘Cannot decide’ responses (10/20) across the 20 participants (synthetic).

4.18 Average matching accuracy; standard deviations indicated with error bars, of session 3 participants for each pair type - real vs real (RvR), synthetic vs real (SvR) and synthetic vs synthetic (SvS).

4.19 Male pair which generated most inaccurate responses across the 20 participants. Both the face images are synthetic.

4.20 Female image pair which generated the most inaccurate responses across the 20 participants. One face image is real (left) while the other is synthetic (right).

4.21 Female pair which generated 2nd most inaccurate responses across the 20 participants. Both the face images are real.

4.22 Male image pair (both real) which generated the most ‘Cannot decide’ responses (10/20) across the 20 participants.

4.23 Female image pair (both real) which generated the most ‘Cannot decide’ responses (10/20) across the 20 participants.

4.24 ROC curves for the four matching experiments performed with pre-trained VGG-FACE and cosine similarity.

4.25 A cropped and masked (right) version of the same synthetic face image.

4.26 Two images of a subject present in both the SREFI donor set (left) and the GBU dataset (right).

4.27 ROC curves for the matching experiments performed by training VGG-FACE with the two augmented datasets generated from our method and a subset of the CASIA-WebFace dataset.
5.1 The SREFI pipeline. (a) The input base face $I_b$ is triangulated using landmark points. (b) Its donor pool $D$ is constructed using hypercolumn feature maps, represented by colored bars. (c) The synthetic texture $I_s$ is created using images from $D$. (d) $I_s$ is subjected to quality based filtering. (e) Best fitting 3D models are obtained using proximity in landmark and deep feature (represented by blue bars). (f) Dense triangular mesh is generated from corresponding 3D models. (g) Multi-pose and shape renderings of $I_s$.

5.2 ROC curves demonstrating how adding donors ($N$) affects the verification performance, i.e., subject identity. Note the ROC plateauing at $N = 4$.

5.3 Sample face images generated using SREFI for three synthetic Female-Asian subjects. The 512×512 synthetic face textures (2D) are shown in the leftmost column (a), with the corresponding 800×600 3D renderings presented to its right (b). The artifact in the top row at facial yaw of -90 degrees is due to faulty landmarking.

5.4 Sample face images generated using SREFI for three synthetic Female-Caucasian subjects.

5.5 Sample face images generated using SREFI for three synthetic Male-Asian subjects.

5.6 Sample face images generated using SREFI for three synthetic Male-Caucasian subjects.

5.7 Sample results generated by (a) SREFI (synthetic identity), and (b) Masi et al.’s method \cite{108} (real identity). Each row of a sub-figure shows the same facial texture (identity) rendered using a 3D face model at different facial poses. Notice how different the same facial texture looks when rendered with different 3D models, i.e., variable shape. An analysis of how this variability in facial shape affects recognition performance can be found in the next chapter.

5.8 Results on the FG-Net dataset \cite{120}. (a) Identification accuracy with variable real and synthetic distractor gallery size, (b) verification performance with 10,000 real and synthetic distractors, using pre-trained VGG-FACE \cite{146, 121}, ResFace-101 \cite{63, 107} and ResNet-50 \cite{63, 31} models.

5.9 Verification performance on the FG-Net dataset \cite{120} with 10,000 synthetic distractors generated using (a) SREFI and Masi’s method, and (b) SREFI and the pre-trained ProGAN model, using pre-trained VGG-FACE \cite{146, 121}, ResFace-101 \cite{63, 107} and ResNet-50 \cite{63, 31} models.

5.10 Comparison with GAN models based on visual quality.
5.11 Verification performance with varying facial shape at different facial yaw values with synthetic images generated using: (a) SREFI [15], and (b) Masi’s method [108].

5.12 (a) Synthetic face images (synthetic identities) generated by SREFI [15] with context and background hallucinated automatically using [12], (b) synthetic face images (real identities) generated by Masi et al.’s method [108] preserving the background.

5.13 Verification performance of the ResNet-50 model [63, 31] on the IJB-B [161] dataset, when fine-tuned with training data with and without the presence of context and background pixels. As the curves suggest, presence of context and background benefits network training.

6.1 Our multi-scale cascaded network pipeline. Starting from the lowest resolution block (8×8), we proceed higher up through a set of GAN blocks in a single pass (left to right in the figure). Except the last block, the output of each block is upscaled 2x and fed as input to the next block. To preserve fine facial details at each resolution, we add the mask image at each resolution before feeding the input. The final 128×128 output, with hallucinated context and background pixels, is generated by block_128. More details about the architecture of block_128 is provided in Figure 6.2.

6.2 block_128 architecture. The encoder is composed of five residual blocks while the decoder upsamples the encoded feature using five pixel shuffling blocks. The solid curved arrows between layers represent skip connections. During training the generator learns to hallucinate the original full face image $I^{GT}$ from the face mask $I^M$ via reconstruction, identity preserving, perceptual and adversarial losses. We replace pixels in the face mask of $G(I^M)$ with original pixels from $I^M$ to preserve fine details.

6.3 Pipeline of our progressively growing (ProGAN) network. We train the lowest resolution block for 50 epochs, then introduce additional layers for the next resolution block and resume training. This network growing continues till block_128. During testing, we only use the trained block_128.

6.4 Sample results from LFW [70] (128×128 in size), generated using GenFace [97], DeepFillv1 [173], SymmFCNet [96], EdgeConnect [116], and our cascaded and ProGAN [81] models. Note the variation in gender, pose, age, expression and lighting in the input images.

6.5 Top row - synthetic images generated using DeepFake [1] where the face mask (rectangle) is from ‘George_W_Bush’ but the context and background are from real face images of ‘Colin_Powell’ (from LFW [70]). Bottom row - synthesized context and background, using our trained cascaded model, for some images of the subject ‘George_W_Bush’.
6.6 Ablation studies - hallucination results of our multi-scale GAN model and its variants. ................................................................. 140
6.7 Sample synthesis results from LFW [70] at different levels of training - (a) the original face image (cropped), (b) masked face input, hallucination results after (c) 10 epochs, (d) 20 epochs, (e) 30 epochs, (f) 40 epochs, and (g) 50 epochs of training. ........................................ 142
6.8 Background replacement process - (a) hallucinated face image (b) the detected foreground mask using a combination of gradient map and the segmentation network from [188, 187, 166], and (c) background pixels replaced with Laplacian blending [30]. ........................................ 143
6.9 Additional qualitative results generated by our ProGAN and cascaded models. The first three rows are samples from the LFW [70] dataset, while the last three rows are taken from the IJB-B [161] dataset. All images are 128×128 in size. ................................................................. 144
6.10 Some problematic cases - missing pixels for the microphone occluding subject’s chin (left), no matching temples generated for the eyeglasses (middle), and hairstyle of wrong gender (right). ....................... 146

7.1 A high-level overview of the current SREFV system. It takes a synthetic donor face and an exemplar video as input data and generates three videos with the donor face animated in 3D. ........................................ 149
7.2 An example illustrating the division of the face image into different facial regions. ................................................................. 150
7.3 Different facial expressions generated using FaceGen [2]. We used a larger set of similar face images to train our expression encoders. . . . 151
7.4 Processing pipeline of the SREFV system. Elements highlighted with a red border depict input and output data. ............................. 152
TABLES

1.1 FACE DATASET STATISTICS [107]  

3.1 YIELD OF EACH PRE-PROCESSING METHOD  

4.1 SREFI DONOR SET DISTRIBUTION  

4.2 TAR AT FAR = 0.01 WITH CROPPED AND MASKED FACE IMAGES  

5.1 DISTRIBUTION OF THE GALLERY AND SYNTHETIC DATASETS GENERATED USING SREFI [13, 15]  

5.2 EFFECTIVENESS OF SREFI AS A DATA AugMENTATION MODULE TESTED USING CASIA-WEBFACE [175] DATASET AND RESNET-50 [63]  

5.3 EFFECTIVENESS OF MASI ET AL.'S METHOD [108] AS A DATA AUGMENTATION MODULE TESTED USING CASIA-WEBFACE [175] AND RESNET-50 [63]  

5.4 EFFECTIVENESS OF SREFI [13, 15] AS A DATA AUGMENTATION MODULE TESTED USING CASIA-WEBFACE [175] AND RESNET-50 [63]  

5.5 COMPARISON WITH GAN MODELS BASED ON REALISM, TRAINING AND SYNTHESIS TIME  

5.6 DISTRIBUTION OF THE TRAINING DATASETS CREATED FOR EXPERIMENT 5  

6.1 QUANTITATIVE RESULTS ON LFW [70]  

6.2 DISTRIBUTION AND PERFORMANCE OF TRAINING DATASETS WITH AND WITHOUT AUGMENTATION  

6.3 BLOCK_8 ARCHITECTURE (INPUT SIZE IS 8×8×3)  

6.4 BLOCK_16 ARCHITECTURE (INPUT SIZE IS 16×16×3)  

6.5 BLOCK_32 ARCHITECTURE (INPUT SIZE IS 32×32×3)  

6.6 BLOCK_64 ARCHITECTURE (INPUT SIZE IS 64×64×3)  

6.7 BLOCK_128 ARCHITECTURE (INPUT SIZE IS 128×128×3)  

6.8 ABLATION STUDIES USING LFW [70]  

xiv
ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisers Dr. Patrick Flynn and Dr. Kevin Bowyer for their continuous guidance and encouragement, this PhD would not have been possible without their support. Dr. Flynn gave me the opportunity to join the CVRL in my third year and work on the face synthesis project, which makes up the bulk of this dissertation. Dr. Bowyer taught me one of the most important aspects of research - how to ask the right questions. I am extremely grateful to Dr. Walter Scheirer as well, whose office door was always open for impromptu meetings, research related discussions and board game sessions.

I also thank my committee members, Dr. Domingo Mery and Dr. Chaoli Wang, for taking the time to review and improve the dissertation. I have greatly benefited from the classes they have taught at Notre Dame: Dr. Mery’s Computer Vision course was my formal introduction to the field, while the face rendering method proposed in Chapter 5 is a culmination of the concepts learned in Dr. Wang’s Computer Graphics class. I also thank Dr. Chris Sweet, Dr. James Sweet, and Dr. Marya Lieberman for taking me under their wings in my second year and letting me work on the PAD project, which ultimately led to my first conference publication.

I have learned a lot from my CVRL lab mates, and count myself lucky to have worked with my co-authors John, Joel, Aparna, Brandon and Daniel. The support from folks at the Notre Dame CRC, especially Dr. Dodi Heryadi and Dr. Scott Hampton, has also been stellar. They have rescued my computation jobs countless times! I am also grateful to the wonderful staff members of the CSE department, particularly Joyce who has been patient with my queries and need for help.
I have been really fortunate to find amazing friends during the different stages of my life, and Notre Dame is no exception. Nabarun and I experienced together the initial struggles and surprises of living abroad, and we survived! Sarbani di has become the big sister I never had. The Ivies welcomed me into their family and included me in their plans just so that I do not feel lonely. And the generous Jean of J.W. Chens’ made sure I was well-fed whenever I was too busy to cook.

I would also like to thank Chhoto Mama, Moni Pishi, Indraroop (IGD), Sourjya, Sourav (Bapi), Tapishnu and Sujoy da for helping me with the GRE, grad school applications and life in general. I am forever indebted to Subhadeep for his friendship and being part of countless memories that we will cherish for the rest of our lives. nWo 4 life, brother!

It takes a great deal of patience and endurance to make an 8th grader, who did not know how many internal angles a triangle has, fall in love with maths. Fortunately, Ranjan Sir believed in me enough to make that effort. Thank you Sir for your guidance throughout the years, I don’t know where I would be without you.

I am eternally grateful to my fiancee Sreya, who I met at Notre Dame, for showering me with love and support whenever I felt lost, and keeping me sane through this whole journey. I look forward to many, many more years of us cooking gourmet dinners and watching The Office together. She is truly the best thing that has ever happened to me.

My final words of gratitude are reserved for my parents, who have worked hard all their lives to provide me a better future and sacrificed all their wishes for my dreams. I thank them for giving me the freedom to make my own decisions at an early age and never imposing their beliefs on me. This PhD and all my achievements, all of it belongs to them.
CHAPTER 1

INTRODUCTION

1.1 Motivation

Face recognition has been an open problem in biometrics and computer vision research for decades \[127\]. Recently, the advent of deep learning \[94\] methods such as convolutional neural networks (CNNs) has allowed face recognition performance on hard datasets to improve significantly. For instance, Google FaceNet \[143\], a CNN based method, has achieved over 99\% verification accuracy on the LFW dataset \[70\], which was once considered a very challenging dataset for face recognition algorithms \[93\]. Because CNNs possess the ability to automatically learn complex representations of face data, they systematically outperform methods based on hand-crafted features. Since these representations are learned from the data itself, it is often assumed that we must provide CNNs clean, pre-processed data for training. However, it is still uncertain if both the training and testing data should be a mix of different facial poses or just frontal (yaw, pitch and roll all close to zero) in nature. The variation in facial yaw in photos of the same person can be seen in Figure 1.1. The popular public face image datasets \[85, 121, 70, 175\] frequently contain unconstrained non-frontal face images. While researchers have utilized face image frontalization as a pre-processing step before training \[60, 136\], a thorough evaluation of the effects of frontalization on face recognition is lacking. Such a study would be crucial in determining if CNNs can automatically learn robust representations invariant of facial pose, or whether training with frontalized face images yields better results? We seek to answer these questions.
Figure 1.1. Images of the same subject with five different facial yaw angles, taken from the CMU-MultiPIE face database [54]. The facial yaw ranges from 0 degree to 90 degrees in both directions.

Face image datasets are typically compiled by researchers manually downloading thousands or million photos from the web. The VGGFace2 [31] (3.3M images), the VGG-FACE (2.6M images) [121], the CASIA-WebFace [175] (500K images), and the CelebA [103] (202K images) datasets are the most notable among these. However, industrial giants like Facebook [150] and especially Google [143] have private face image datasets with over 200M images at their disposal. Given the tremendous resource overhead for compiling such a dataset by selectively downloading images from the internet, it is difficult to amass such large datasets. Recent results motivating the need for even larger datasets involve data augmentation for CNN training, where multiple views of an existing image are generated. It is common practice to augment any dataset before using it to train a CNN. Augmentation is done to reduce the chance of the CNN over-fitting on the training data. Typically, researchers use simple methods like image translation, cropping, rescaling and PCA whitening to augment the training data [91], as illustrated in the left column of Figure 1.3. Despite these augmentation methods, CNNs still tend to over-fit the training data, if each training class does not contain considerable number of samples [37]. Except for the recently released VGGFace2 dataset [31], publicly available face datasets generally contain a large disparity in the number of images per subject. This can cause deep networks to overfit, as often there are a few subjects with a large number of images each, and
TABLE 1.1

FACE DATASET STATISTICS

<table>
<thead>
<tr>
<th>Dataset</th>
<th>No. of Images (#Img)</th>
<th>No. of Subjects (#ID)</th>
<th>#Img/#ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google</td>
<td>200M</td>
<td>8M</td>
<td>25</td>
</tr>
<tr>
<td>Facebook</td>
<td>4.4M</td>
<td>4,030</td>
<td>1K</td>
</tr>
<tr>
<td>VGG-FACE</td>
<td>2.6M</td>
<td>2,622</td>
<td>1K</td>
</tr>
<tr>
<td>MegaFace</td>
<td>1.02M</td>
<td>690,572</td>
<td>1.5</td>
</tr>
<tr>
<td>CASIA-WebFace</td>
<td>494,414</td>
<td>10,575</td>
<td>46</td>
</tr>
<tr>
<td>UMDFaces</td>
<td>367,888</td>
<td>8,277</td>
<td>44</td>
</tr>
<tr>
<td>CelebA</td>
<td>202,599</td>
<td>10,177</td>
<td>19</td>
</tr>
<tr>
<td>VGGFace2</td>
<td>3.31M</td>
<td>9,131</td>
<td>362</td>
</tr>
</tbody>
</table>

a large number of subjects with few images each, as can be seen in Table 1.1. This is termed the ‘long-tail’ problem in [175, 107]. One result of this is that the CNN may fail to learn a feature representation that adequately represents the subjects with few images. Consequently, a CNN trained on such an unbalanced dataset might not be as effective as a CNN trained on a well-balanced dataset [143]. One of our goals is to develop a face synthesis method to artificially augment such an unbalanced dataset and gauge the effect of training with such artificial augmentation on the recognition performance of a CNN.
1.2 Exploring Face Image Frontalization (see Chapter 3 for details)

To better understand the impact of artificial frontalization of non-frontal face images, we perform several face recognition and verification experiments. For this study, we use the CASIA-WebFace (CW) \[175\] and PaSC video \[140\] datasets for CNN training and testing respectively. Two frontalization techniques are chosen for our training and testing evaluation: the well-established method proposed by Hassner et al. (H) \[60\], and our own newly proposed frontalization method (OFM). Furthermore, to evaluate the effect of facial landmarking on the frontalization process, we use three landmarking techniques: Zhu and Ramanan (ZR) \[191\], Kazemi and Sullivan (KS) \[83\], and our own technique - a Cascade Mixture of Regressors (CMR). Different frontalization results using various combinations of these methods can be seen in Fig. 1.2.

We use the popular VGG-FACE \[121\] as our base architecture for training net-
works using different pre-processing strategies. We extract face representations from individual video frames in PaSC using a network trained with a particular pre-processing strategy. These features are used for verification and recognition purposes by applying a cosine similarity score-based face matching procedure.

As a set of baselines, we use: 1) a simple 2D alignment that corrects for in-plane rotation, 2) no pre-processing at all, and 3) a snapshot of the VGG-FACE model pre-trained on the 2D aligned VGG-FACE dataset. This is used to evaluate how much the additional training on CW improved the face representation capability of the CNN model. The effect of each data augmentation is manifested in the performance of each subsequent CNN model. The focus of our study is to evaluate the effect of frontalization on CNN-based face recognition instead of achieving near state-of-the-art results on PaSC. Therefore, we chose not to use any elaborate detection algorithm or scoring scheme like those used by most of the PaSC 2016 Challenge participants.

In summary, our contributions in this area are as follows:

- We evaluate popular facial landmarking and frontalization methods and quantify their effect on performance in video-based face recognition tasks using a CNN.
- We determine whether symmetric or asymmetric frontal reconstruction works best for face recognition.
- We investigate how frontalization failure rates varies with pose, using the CMU-MultiPIE dataset.
- We propose a new, effective facial landmarking and frontalization technique for comparison with the other methods. Experiments show our landmarker and frontalizer both to be separately compatible with other frontalization and landmarking methods respectively.
- Our frontalization method outperforms other methods in the study, when both the training and testing data is frontalized.
Figure 1.3. While traditional augmentation methods have explored face image cropping, translation and scaling (left column), we can augment a dataset by creating synthetic face images of existing identities (middle column) and new synthetic identities altogether (right column). This augments both the depth and breadth of a dataset.

1.3 Synthesizing Face Images to Improve Recognition (see Chapters 4, 5 and 6 for details)

To tackle the issue of unbalanced datasets in face recognition research, i.e. the long-tail problem [175, 107], there is a need to be able to create face image datasets that: (1) contain an arbitrarily large number of identities (Figure 1.3), (2) have a balanced number of images per identity, (3) have a variety of facial pose and shape to capture the diversity across video frames and age respectively (Figure 1.4), (4) have multiple sets of context (hair, clothes) and background to account for real-world variations (Figure 1.5), and (5) do not run into potential issues of invasion of privacy.
Figure 1.4. Varying shape and pose: each row represents three face images of the same synthetic identity at yaw values -30, 0, and +30. The face images in each top row belong to a synthetic identity while the corresponding bottom row has face images of a real identity for visual comparison.

We solve these problems with a novel set of systems for automatically generating synthetic frontal face images, of a real identity or a synthetic identity not mapping to any real person. When face images are generated for synthetic identities rather than any real person, privacy issues are avoided. Datasets can be generated to have a balanced number of images per identity, avoiding the long-tail problem \cite{107}. Also, the computation and communication burden of downloading from the web is avoided. To add variation in facial pose and shape, we further render 3D face masks using the synthetic facial textures and a set of “best-fitting” 3D models. Finally, we propose a multi-scale cascaded network of GANs \cite{53}, that automatically hallucinates context and background pixels for the aforementioned synthetic 3D face masks. Combining these three functions, our approach can augment an existing dataset by increasing the number of face images per real identity (more intra-class variation), and by increasing the total number of (real + synthetic) identities (more inter-class variation). This is a significant improvement over traditional face image augmentation methods \cite{107, 108, 52}. 

7
Our facial texture generation approach named SREFI (Synthesis of Realistic Example Face Images) \cite{13, 15}, constructs synthetic face images from a donor set of face images of real identities. It requires few gallery images (a few thousand), minimal training (of an SVM \cite{41}) and little execution time (a few seconds) to generate high quality synthetic face images with different pose and shape of real or synthetic identities. SREFI begins by pooling real, frontal gallery face images into groups that are similar in appearance and combines facial components from these gallery images to generate realistic synthetic textures. Compatible 3D face shape models, obtained by performing a best-fitting test, are then used to render the synthetic texture in 3D with different pose and shape, where shape is a function of facial landmarks \cite{107}. Our methodology remains the same even for different target resolutions (100×100 to 800×600). Some of the faces in Figure \ref{fig1.4} belong to synthetic identities generated using our method.
For context and background generation, our GAN model takes as input a face image and its masked version, 128×128 in size, and downsamples both to their 64×64, 32×32, 16×16, and 8×8 versions during training. The training starts at the lowest GAN block (block.8), where it learns to reconstruct the 8×8 full face image from the corresponding 8×8 masked input. The output of this network is then upscaled 2x using a pixel shuffling block [145] and passed to the next GAN block (block.16). Thus instead of masked black pixels, block.16 receives a 16×16 input with roughly hallucinated context and background pixels, guiding it towards the direction of correct reconstruction. Its 16×16 output is then upscaled and sent to block.32 and so on. At each block, we independently learn to hallucinate context and background pixels through reconstruction loss, adversarial loss provided by a discriminator, perceptual loss from [181] and an identity preserving loss using the pre-trained VGG-FACE model [121]. During testing, we only use the trained generator and pixel shuffling blocks to hallucinate the final 128×128 full face image from an input face mask.

We perform multiple experiments to assess the effectiveness of the different components of SREFI. These experiments show our technique to be effective as a data augmentation module for training CNNs, resulting in a higher test performance on the IJB-B [161] verification protocol. The SREFI generated synthetic face images can also be used as synthetic distractor sets, like the MegaFace dataset [85, 117], to test the robustness of trained CNN models against false positives. Additionally, the SREFI generated images report higher scores on several quality metrics when compared with popular GAN models [132, 21, 81, 97, 173, 116, 96] and the DeepFake [1] face swapping application.

Our main contributions in the face synthesis domain are listed below:

- We propose SREFI, a technique to generate a large number of natural-looking synthetic face images of both existing and non-existing identities.
• Unlike deep learning based face synthesis models \cite{81,82}, SREFI requires only a few minutes of training and a few thousand gallery images and no GPU to generate a large set of synthetic face images.

• We release the Notre Dame Synthetic Face Image Dataset\textsuperscript{1} containing 2M face images of 12K synthetic identities.

• To gauge the quality of SREFI generated synthetic face images, we perform a set of quantitative benchmark experiments that can be used by researchers to assess face synthesis quality in the future.

• To gauge the effect facial shape and context has on the subject identity, we conduct multiple verification experiments using Masi \textit{et al.}'s method \cite{108} and the CASIA-WebFace \cite{175} and IJB-B \cite{161} datasets.

• We propose a cascaded network of GAN models that can automatically hallucinate context and background pixels from a given face mask without any user supervision. Trained with a few thousand images, our model can synthesize full face images from different datasets with a wide variety in gender, ethnicity, facial pose, expression and lighting.

• When compared with recently proposed facial inpainting models \cite{97,173,96,116} and the DeepFake face swapping software, our model generates photorealistic results that produce higher quality scores (identity preservation, realism and visual quality) compared to the other algorithms on LFW \cite{70}.

• When compared with the ProGAN training regime from \cite{81}, our end-to-end cascaded pipeline is shown to benefit the hallucination process and generate sharper results.

• We evaluate the potential application of our model as a generator of supplemental training data, and show the ResNet-50 model \cite{63} to produce a boost in test performance when such an augmentation is introduced.

1.4 Outline

The remainder of this document is organized as follows. Chapter 2 reviews the previous work done in the different domains of research related to this work - face recognition, facial landmarking, face image frontalization, face image synthesis, facial inpainting and data augmentation. In Chapter 3, we describe each of the existing facial landmarking and frontalization techniques used in our frontalization study.

\textsuperscript{1}https://cvrl.nd.edu/projects/data/
Then we present our novel landmarking and frontalization methods in detail. This is followed by a description of our training and testing pipeline with the CNN and discussion of the generated experimental results.

In Chapters 4 and 5, the different steps of our face synthesis technique are detailed - 2D facial texture generation and 3D face mask rendering respectively. We also explain the different experiments conducted with the SREFI generated synthetic images to evaluate their realism, and their effectiveness as supplemental training data and distractors, while comparing them with other popular face synthesis methods [107, 108, 132, 24, 81]. Experiments quantifying the effect of facial shape on the subject identity and the impact of context on recognition performance are also presented in Chapter 5. In Chapter 6, we present the details of our cascaded network of GANs for context and background hallucination, along with experiments comparing it with facial inpainting models [97, 173, 116, 96] and DeepFake [1].

Chapter 7 concludes the dissertation, outlining the current state of our video generation project along with preliminary results.
CHAPTER 2

RELATED WORK

Previous work relevant to our research can be categorized into seven broad groups as listed below:

2.1 Face Recognition

Face recognition (FR) has been explored by researchers for decades. Such recognition systems are typically trained with a set of face images spread across numerous subjects, from which the systems learn features to represent these faces. These features are compared to score the quality of a match, and scores are assessed to judge the accuracy of a FR system. A higher score for true matches results in a higher true accept rate (TAR), ideally accompanied by a low threshold of false positive or the false accept rate (FAR). The trained system is then used to recognize new face images, as illustrated in Figure 2.1. In its infancy, researchers used handcrafted features and descriptors for representing faces [127]. More recently, state-of-the-art deep learning [94] based methods for face recognition have achieved near-perfect recognition scores on the once-challenging LFW dataset [70]. Massive face image datasets, both private [150, 143] and public [121, 31, 18, 57, 175, 103], have been accumulated to train intricately developed CNN models [146, 148, 45, 63, 163, 68]. However, LFW with its 13,000 images is an under-representation of the diverse population of faces that a human encounters. The effect of this wide gap in data size on the recognition performance of the above methods has been documented in [85]. Most of the
Figure 2.1. Illustration of a typical face recognition system.
recognition methods which produced near-perfect results on LFW (13k images) performed poorly in verification tasks on the MegaFace dataset (over 1M images) [85]. Researchers have now shifted their attention to the more challenging problem of face recognition from videos. The YouTube Faces (YTF) dataset [162], IJB-A [89], IJB-B [161], and PaSC [141] exemplify both unconstrained and controlled video settings. Researchers have used multi-pose based CNN models to recognize face images with multiple yaw values during testing [6, 106] or normalized facial pose [45, 44] in an attempt to stabilize the data or used artificial pose synthesis as a means for training data augmentation [107, 108]. The ultimate goal is to construct a CNN model which is robust to massive sets of distractor face images [85, 117].

2.2 Facial Landmarking

Facial landmarking pertains to the automatic localization of different key-points of a face image, as shown in Figure 2.2. It is an essential part of frontalization, as the landmarks define the facial co-ordinates. Over the past decade, a number of landmarking techniques that rely on handcrafted features have been developed [83, 136]. Algorithms have provided landmark sets between 7 and 194 points in size. Recently, landmarkers have begun to conform to a 68-point standard to improve comparative analysis between algorithms and across different landmarking challenges and datasets [191, 83, 144]. More recently, methods leveraging deep learning have been proposed for face detection [69, 78] and landmark estimation [164, 165, 61, 74] at extreme yaw angles from relatively lower resolution face images, both in 2D and 3D.

2.3 Face Image Frontalization

Frontalization techniques are used to generate frontal versions of non-frontal face images, as depicted in Figure 2.3. Once the facial landmarks are detected on a
Figure 2.2. Examples of facial landmarking, key-points in yellow, reproduced from [83].
non-frontal face, frontalization can be performed using one of two approaches. The first approach utilizes unique 3D models for each face in the gallery, these models are either inferred statistically \[59, 77, 10, 15\], collected at acquisition time \[27\], or generic \[60, 14\]. Once the image is mapped to a 3D model, matching can be performed by either reposing the model to match the pose of the query image and generating a pose compatible image or the query image can be frontalized \[182\]. These methods have been utilized in influential recognition studies \[150\]. The second approach uses statistical models to infer a frontal view of the face by minimizing off-pose faces to their lowest rank reconstruction \[136\]. Additionally, methods have been explored for inferring frontal faces using deep learning \[177\]. A modern approach of face frontalization is to use generative adversarial networks (GANs) \[53\] for this task, which can learn to hallucinate the facial pixels at a given pose by learning the transformation directly from the data \[152, 20, 71, 183\].
2.4 Face Image Synthesis

One of the earliest face synthesis methods [102] used a combination of neighborhood patches from a set of images for hallucinating new faces. More structured methods have been formulated since then, like stitching similar faces or their parts together [26, 113, 13], or swapping face masks [118, 84] onto different background images for face image synthesis and inpainting. Others have implemented a model-based learning approach for different applications like swapping of aligned faces, expression flow across images of the same person and hallucination of new faces [189, 171, 112, 99, 139]. An inverse rendering approach for synthesizing a 3D structure of the face was proposed in [8], and more recently a one-shot version of the same was described in [176]. The method proposed in [113], which divides the source face into four different parts and replaces each part from a gallery image before blending, is thematically closer to our approach. The use of a 3D head model to repose or
frontalize a given face image in order to generate synthetic views and their impact on face recognition have been investigated in [107, 71, 178, 151, 74, 60, 14, 15]. GANs [53] have made tremendous progress in this domain, with different models being used to generate synthetic face images from noise vectors [132, 24, 158, 184, 81, 82], or existing faces with different pose [152, 20, 185], facial attributes [21, 64, 34, 115], age [51, 9], and expression [100]. However, they require a lot of training data and the synthetic face images are relatively low resolution and do not appear very realistic. Other deep feature based techniques have been explored by researchers for reconstructing a face image [38, 156, 19, 56, 67, 39], however they do not always generate new subjects. A more detailed bibliography of face synthesis research can be found in [104]. A pair of synthetic face images generated using NVIDIA’s recently proposed style based generator [82] can be seen in Figure 2.4.

2.5 Facial Inpainting

Image inpainting started with [23] transferring low-level features to small unknown regions from visible pixels. In [111, 75], this idea is used to reconstruct facial parts in missing regions using a positive, local linear representation. A simple inpainting scheme was proposed in [75], which uses features like ethnicity, pose and expression to fill missing facial regions. GANs have also been used for image completion, e.g., in [72, 97], a generator is used to hallucinate masked pixels, with discriminators and parser networks refining the results. In [174, 173, 116], information from the available data, surrounding image features, and edge structures are used for inpainting. Facial symmetry is directly enforced in [96] to improve global consistency. In [130, 79], the inpainting process is guided by a rough sketch provided by the user. All these methods work well with small targeted masks [186], located on or near the face region, but perform poorly when a large masked area is presented [156]. Some sample inpainting results from [97] can be seen in Figure 2.5.
Figure 2.5. Some examples of facial inpainting from [97] where - (a) original image, (b) masked input, (c) inpainted output.

Figure 2.6. An example of face swapping using the DeepFake software [II], where the face in the original image (LEFT) is swapped with that of Nicholas Cage (RIGHT) while retaining the visual attributes.
2.6 Face Swapping

The first face swapping pipeline was proposed in [26], where a face is de-identified by blending together facial parts from other images. Many methods have modified this idea of recombining facial parts to generate synthetic images for de-identification or data augmentation [113, 13, 84]. In [118], a 3D morphable model based shape estimation procedure is used to segment the source face and fit it to the target image prior to blending. Instead of using facial textures, the method in [115] uses latent variables from a deep network for face swapping. A style transfer [50] based face swapping approach was proposed in [90]; but it requires the network to be trained on only one source subject at a time. DeepFake [11] is another recent method for face swapping, and employs an autoencoder trained to reconstruct tight face crops of a subject from its warped versions. This trained autoencoder is then used to hallucinate the source subject from different target face images (an example is presented in Figure 2.6). However, it works with one subject at a time and requires the target images to be highly constrained in visual attributes, making it impractical for many real world applications.

2.7 Data Augmentation

Dataset augmentation has been known to help the training of CNN models as it reduces overfitting on the training data [32]. However, some popular augmentation methods at present are simply over-sampling or mirroring the training images. For re-identification purposes, another mode of augmentation proposed was to change the image background [110]. Hassner proposed an augmentation method for faces [59, 60] which generates frontal versions of the face images to reduce recognition errors. This idea was extended in [107, 108] by augmenting the CASIA-WebFace dataset [175], synthesizing new views of existing identities with multiple pose, shape and
expressions. This introduces more intra-class variation to the training data, making the augmented dataset deeper i.e., more images per subject. This augmented dataset, when used for training CNNs, results in a higher test performance [107]. An example of augmentation by reposing a face image is shown in Figure 2.7. Augmenting the non-facial attributes, like context (forehead, hair, clothes) and background, while keeping the facial identity intact has also been proven to benefit network training [12]. Researchers have also experimented with making training datasets ‘wider’ [17] [15], i.e., adding more images of external identities (real or synthetic). Such an augmentation step during training increases its inter-class variance and consequently improves network performance [17] [15].

A major difference between our face synthesis method and the other aforementioned methods is that it can generate multiple synthetic faces of a new non-existent identity as well as new images of an existing identity. Additionally, it can render the same synthetic identity with multiple facial shape and pose to increase the intra-class variation of an existing training set. Our method requires only a few thousand images and a few minutes to train, and does not necessarily need a GPU. For context and
background generation, existing models require - (1) the full face image to work, or
(2) apply appearance constraints for identity preservation, or (3) human annotation
to ensure high quality of the results. Unlike these works, our GAN model treats this
problem along the same lines as image colorization [180, 92] and directly hallucinates
the missing pixels taking cues from the input data without any involvement from the
user.
CHAPTER 3

EXPLORING THE EFFECTS OF FRONTALIZATION ON FACE RECOGNITION

This Chapter is based on the work described in [14], which was published in IEEE WACV 2018. The proposed landmarking and frontalization method was developed by Dr. Janez Krizaj and Dr. Vitomir Struc at the University of Ljubljana. It has been refined in collaboration with Joel Brogan and several other researchers at the University of Notre Dame.

Face image frontalization has been used as a pre-processing technique by computer vision researchers for years [60, 150]. It was used to effectively reduce the variability in appearance in face images of the same subject by producing better alignment and allowing easier comparison between face images. Consequently, frontalization is considered to aid in the face recognition process. However, different frontalization methods do not produce the same result while reposing the same face image. Moreover, the success of the frontalization step depends on the quality of the landmark points, as can be seen in Figure 1.2. A study of the effectiveness of the different combinations of popular landmarking and frontalization techniques, previously absent from the literature, is presented in this chapter. Additionally, we have proposed a novel method for facial landmarking and frontalization which can generate a frontalized version of the face from a single image. This method outperforms other techniques when both the training and testing data are frontalized.
3.1 Description of Chosen Landmarking and Frontalization Methods

Here we present brief descriptions of the facial landmarking and frontalization techniques used in this paper.

3.1.1 Landmarking

**Zhu and Ramanan (ZR) [191]:** The ZR method allows for simultaneous face detection, landmarking, and pose detection, accommodating up to 175 degrees of facial yaw. ZR uses a mixture of trees approach, similar to that of phylogenetic inference. The algorithm proposed in [88] is used to optimize the tree structure with maximum likelihood calculations based on training priors. Due to the algorithm performing localization and landmarking concurrently, it is relatively slow.

Figure 3.1. Visualization of multiple regressors fitting the feature $x_t \in \mathbb{R}^{d \times 1}$ vs. shape update $\Delta S_t = R_t(I, S_t)$ curve
Figure 3.2. Frontalization correspondences for the method of Hassner et al. [60]. a) Input face in non-frontal pose. b) Face matched to template pose via landmarks. c) Back-projection of template from input pose to frontal. d) Back-projection of pixel values to frontalized template, including artifacts from self-occlusion

Kazemi and Sullivan (KS) [83]: KS uses a cascade of multiple regressors to estimate landmark points on the face using only a small, sparse subset of pixel intensities from the image. This unique sub-sampling renders it extremely fast, while maintaining a high level of accuracy. This landmarker is popular due to its ease of use and availability — it is implemented in the widely used Dlib library [86].

Cascade Mixture of Regressors (CMR): We introduce the CMR landmarking model as one that builds on recent nonlinear regression methods. The CMR model simultaneously estimates the location of $N$ fiducial points $(x_i, y_i)^\top$ in a facial image $I$ through a series of $T$ regression steps, similar to [83, 11, 36, 101, 134, 153, 167, 168, 170, 190]. Starting with an initial shape estimate $S_0 = [x_1, y_1, \ldots, x_N, y_N]^\top \in \mathbb{R}^{2N \times 1}$, the following iterative scheme updates the face shape:

$$ S_{t+1} = S_t + \Delta S_t, \text{ for } t = 0, \ldots, T, $$

(3.1)
The $t$-th shape update $\Delta S_t = R_t(I, S_t)$ is predicted using the regression function $R_t$ defined as a mixture of $C$ linear regressors, similar to [191][154]:

$$\Delta S_t = \sum_{i=1}^{C} \psi_{i,t}(x_t)(W_{i,t}^T x'_t),$$

(3.2)

where $x_t \in R^{d \times 1}$ is a feature vector extracted from $I$ from landmark locations $S_t$, $x'_t = [x_t, 1] \in R^{(d+1) \times 1}$, $W_{i,t}^T \in R^{2N \times (d+1)}$ denotes the regression matrix of the $i$-th (local) regressor of mixture $t$, and $\psi_{i,t}(x_t)$ represents a membership function that clusters features to regressors, as depicted on the top brackets of Fig. 3.1. Memberships are trained using a bottom-up Gaussian Mixture Model (GMM) with Expectation-Maximization (EM) to create a predefined number of fuzzy clusters $C$, as described in [7]. Regression matrices are subsequently computed for each cluster in $C$ using a least-squares approach, using HoG features extracted from 300-W dataset [137].

This method strikes a balance between accuracy and speed, utilizing simultaneous updating like in [83] for fast performance, while delivering more accurate updates using a mixture-based landmarking scheme like in [191].

3.1.2 Frontalization

**Hassner et al. (H) [60]**: This method allows 2D face images to be frontalized without any prior 3D knowledge. We chose to analyze this method due to its prominence in the facial biometrics community, and because an open source implementation of the algorithm exists. Using a set of reference 3D facial landmark points determined by a 3D template, the 2D facial landmarks detected in an input image are projected into the 3D space. A 3D camera homography is then estimated between them. Back-projection is subsequently applied to map pixel intensities from the original face onto the canonical, frontal template. Optional soft symmetry can be applied by replacing areas of the face that are self-occluded with corresponding patches from
Figure 3.3. Overview of the proposed frontalization procedure. The procedure first detects the facial area and a number of facial landmarks in the input image (a). It then aligns a generic 3D model with the input face (b) and calculates a 3D transform that maps the aligned 3D model back to frontal pose (c). Based on the known 2D-to-3D point correspondences, a synthetic frontal view of the input face is generated (d) and post-processed to generate the final results of the frontalization (e).

the other side. Due to the global projection of this method, incorrect landmarking can stretch and distort the frontalized face, causing loss of high-frequency features used for matching. Figure 3.2 shows the general mapping procedure for projecting non-frontal faces onto a 3D model.

3.2 Our Frontalization Method (OFM)

In this section, we present our proposed frontalization procedure, which is capable of synthesizing a frontalized face image from a single input image with arbitrary facial orientation without requiring a subject-specific 3D model.

3.2.1 Face Detection, Landmarking and Model Fitting

Our proposed frontalization procedure starts (see Fig. 3.3 (a)) by detecting the facial region in the input image $I_0$ using the Viola-Jones face detector [157]. Using the CMR method, we detect $N = 68$ facial landmark points, i.e., $S_z = [x_1, y_1, \ldots, x_N, y_N]^T \in$
The landmarks can be used to determine the pose and orientation of the processed face. We crop the facial area, $I_c$, based on the detected landmarks and use it as the basis for frontalization.

To transform the face in the input image to a frontal pose, we require a depth estimate for each of the pixels in the cropped facial area. To this end, we use a generic 3D face model and fit it to the cropped image $I_c$. Our model is a frontal depth image $I_r$ from the FRGC dataset [127] manually annotated with the same 68 landmarks as detected by the CMR procedure. We fit the 3D model to the cropped image through a piece-wise warping procedure guided by the Delaunay triangulation of the annotated landmarks. Since the annotated landmarks reside in a 3D space, i.e., $S_r = [x_1, y_1, z_1, \ldots, x_N, y_N, z_N]^\top \in R^{3N \times 1}$, we use the 2D coordinates in the XY-plane for the triangulation. The fitting procedure then aligns the generic 3D model with the shape of the cropped image and provides the depth information needed for the 3D transformation of the input face to a frontal pose (see Fig. 3.3 (b)). The depth information generated by the warping procedure represents only a rough estimate of the true values, but as we show later, is sufficient to produce visually convincing frontalization results.

3.2.2 3D Transformation and Texture Mapping

After the fitting process, we use the landmarks $S_a \in R^{3N \times 1}$ corresponding to the aligned 3D model $I_a$ and the landmarks $S_r \in R^{3N \times 1}$ of the generic 3D face model to estimate a 3D transformation, $T \in R^{4 \times 4}$, that maps the fitted model $I_a$ back to frontal pose (Fig. 3.3 (c)). We use Horn’s quaternion based method [66] to calculate the necessary scaling, rotation and translation to align the 3D points in $S_a$ and $S_r$ and construct the transformation matrix $T$. Any given point of the aligned 3D model $P = [X, Y, Z, 1]^\top$ can then be mapped to a new point in 3D space based on
the following expression:

\[ P' = TP, \]

(3.3)

where \( P' = [X', Y', Z', 1]^\top \) represents a point of the frontalized 3D model \( I_f \) (see Fig. 3.3 (d)).

The cropped image \( I_c \) and the aligned model \( I_a \) are defined over the same XY-grid. The known 2D-to-3D point correspondences can, therefore, be exploited to map the texture from the arbitrarily posed image \( I_c \) to its frontalized form \( I_t \). Values missing from \( I_t \) after the mapping are filled in by interpolation. The results of the presented procedure are shown in Fig. 3.3 (d). Here, images in the upper row illustrate the transformation of the 3D models in accordance with \( T \), while the lower row depicts the corresponding texture mapping. The mapped texture image \( I_t \) represents an initial frontal view of the input face, but is distorted in some areas. We correct for these distortions with the postprocessing steps described in the next section.

3.2.3 Image Correction and Postprocessing

Similar to the method of [60], our approach utilizes a generic 3D face model to generate frontalized face images. Unlike [60], we adapt our model in accordance with the shape of the input face to ensure a better fit. Triangulation is performed on the input face landmark coordinates. Each triangle is then mapped back to the generic 3D face model, and an affine transform is calculated per-triangle. Because the piecewise alignment is performed with a warping procedure, minor distortions are introduced into the shape of the aligned 3D model, which lead to artifacts in the mapped texture image \( I_t \). Additional artifacts are also introduced by the interpolation procedure needed to compensate for the obscured or occluded areas in the input images caused by in-plane rotations and self-occlusions.

We correct for the outlined issues by analyzing the frontalized 3D model \( I_f \).
Since Eq. \((3.3)\) defines a mapping from \(I_a\) to \(I_f\), the frontalized 3D model \(I_f\) is not necessarily defined over a rectangular grid, but in general represents a point cloud with areas of different point density. We identify obscured pixels in \(I_a\) based on point densities. If the density for a given pixel falls below a particular threshold, we mirror the corresponding pixel from the other side of the face to form a more symmetric face.

The effect of the presented image correction procedure is illustrated in Fig. 3.3 (e). The image, marked as \(I_m\), contains white patches that were identified as being occluded in \(I_a\), while \(I_n\) represents the corrected image with pixels mirrored from one side of the face to the other (examine the difference in the appearance of the nostrils between \(I_t\) and \(I_n\)). In the final step of our frontalization procedure we map the image \(I_n\) to a predefined mean shape, similar to AAMs \([40]\). This mapping ensures a uniform crop as well as unified eye and mouth locations among different probe images. Consequently, distortions induced by the 3D shape fitting (via warping) and frontalization procedures are corrected and all facial features are properly aligned as all faces are mapped to the same shape (mesh). This is not the case with other frontalization techniques \([60, 59]\), as they simply ensure frontal pose but not necessarily alignment of all facial parts. This mapping generates the final frontalized output \(I_1\) of our procedure and is shown in the last image of Fig. 3.3 (e).

The code for our landmarking and frontalization method (OFM) can be accessed online on Github\(^1\).

3.3 Face Recognition Pipeline

In this section, we provide details about our face recognition pipeline. This includes details on the training and testing data, pre-processing methods, network architecture, and scoring protocol. A schematic representation of our pipeline can be found online on Github\(^1\).

\(^1\)https://github.com/joelb92/ND_Frontalization_Project/
Figure 3.4. The face recognition pipeline: the training phase is on the left and the testing phase is on the right of the image.

3.3.1 Training Data: CASIA-WebFace

The CASIA-WebFace dataset (CW) \cite{9} contains 494,414 well-labeled face images of 10,575 subjects, with 46 face images per subject on average. The dataset contains face images of varying gender, age, ethnicity and pose, and was originally released for training CNNs. In comparison, MegaFace \cite{2} and VGG-FACE \cite{42} contain over a million face images, but have significantly more labeling errors \cite{9}. For this reason, coupled with what was feasible to process with available GPU hardware, we ultimately chose a reduced subset of CASIA-WebFace, containing 303,481 face images of 7,577 subjects, as our training dataset. The exact list of CW face images used in our experiments can be found in \url{https://github.com/joelb92/ND_Frontalization_Project/Release/CW_Subset.txt}. Sample images from the dataset can be seen in Fig. 3.4.
Figure 3.5. Difference in frontalization techniques. Left-to-right: a set of faces with different degrees of yaw. Top-to-bottom: a) original input face image. b) H (asymmetric). c) H (symmetric). d) Our method (asymmetric). e) Our method (hard symmetric).
3.3.2 Pre-processing Methods

The pre-processing schemes used in our experiments were comprised of different combinations of landmarkers and frontalizers described in Sections 3.1 and 3.2: 1) ZR \[191\] & H \[60\], 2) KS \[83\] & H \[60\], 3) CMR & OFM, 4) CMR & H \[60\], 5) ZR \[191\] & OFM, and 6) KS \[83\] & OFM. Output of the frontalization techniques on sample images can be seen in Fig. 3.5.

In addition, we compared these methods to three baseline approaches: 1) Training VGG-FACE with only 2D aligned CW images, rotated using eye-centers, \textit{i.e.}, no frontalization (Figure 1.2b). The aligned faces were masked, to be consistent with the frontalization results. The eye-centers and mask contours were obtained using the KS \[83\] landmarker available with Dlib \[86\]. 2) Training VGG-FACE with original CW images, \textit{i.e.}, no pre-processing. 3) A snapshot of the original VGG-FACE model, pre-trained on 2.6 million 2D aligned face images from the VGG-FACE dataset \[121\], as a comparison against a prevalent CNN model.

3.3.3 CNN Architecture: VGG-FACE

We chose the VGG-FACE architecture \[146, 121\] because it generates verification results comparable to Google FaceNet \[143\] on LFW \[70\] while requiring a fraction of its training data. Additionally, the model performs reasonably well on popular face recognition benchmarks \[126\]. Lastly, a snapshot of this model, pre-trained with 2.6 million face images, is present in the Caffe \[76\] model zoo\[^2\]. We used this pre-trained model to fine-tune connection weights in our training experiments for faster convergence.

\[^2\]https://github.com/BVLC/caffe/wiki/Model-Zoo
Figure 3.6. Randomly selected frames from - (a) 2D aligned tripod mounted, (b) 2D aligned handheld, (c) frontalized tripod mounted, (d), frontalized handheld PaSC videos. Columns (1) - (4) represent different subjects in the PaSC dataset. Note the difference in resolution, illumination, occlusion and facial pose in the different frames.
3.3.4 Testing Datasets

For completeness, we performed two types of frontalization tests to gain a more holistic understanding of the behavior of different frontalizer schemes. The first set of tests, which analyze the performance impact of different frontalization methods on facial recognition, utilized the PaSC Dataset [141]. The second set of tests were designed to analyze the yield rates and failure modes of frontalizers for different pose conditions. For these tests, we utilized the CMU MultiPIE dataset [54]

**PaSC** - The PaSC dataset [141] is a collection of videos acquired at the University of Notre Dame over seven weeks in the Spring semester of 2011. The human participants in each clip performed different pre-determined actions each week. The actions were captured using handheld and stationary cameras simultaneously. The dataset contains 1,401 videos from handheld cameras and 1,401 videos from a stationary camera. A small training set of 280 videos is also available with the dataset. A set of example video frames from PaSC [140] is shown in Figure 3.6.

While both YTF [162] and IJB-A [89] are well-established datasets, they are collections of video data from the Internet. On the other hand, PaSC consists of video sequences physically collected specifically for face recognition tasks. This type of controlled acquisition is ideal for our video-to-video matching-based evaluation.

**MultiPIE** - To evaluate the success rate of each landmarker and frontalizer combination at specific facial pose angles (yaw), we used the CMU Multi-PIE face database [54] which contains more than 750K images of 337 different people. We utilized the multipose partition of the dataset, containing 101,100 faces imaged under 15 view points with differing yaw angles and 19 illumination conditions, with a variety of facial expressions. For pose consistency, we excluded the set of view points that also induce pitch variation.
3.3.5 Feature Extraction and Scoring

We used networks trained on data pre-processed with each of the combinations mentioned above as feature extractors for PaSC video frames. Before the feature extraction step, the face region from each frame was extracted using the bounding box provided with the dataset. Bad detections were filtered by calculating the average local track trajectory coordinates to roughly estimate the locations of neighboring detections, and removing detections with coordinates outside a $2.5\sigma$ (standard deviation) distance range from their estimated location.

After pose correction, a 4,096 dimensional feature vector was extracted from the $fc7$ layer for every face image using each CNN model. Once feature vectors for all frames were collected, the accumulated feature-wise means at each dimension were calculated to generate a single representative vector for that video. This accumulated vector can be represented as $[f_1, f_2, f_3, ..., f_{4096}]$, such that

$$f_k = \frac{1}{N} \sum_{i=1}^{N} (v_k)_i \quad (3.4)$$

where $(v_k)_i$ is the $k$-th feature in frame $i$ of the video and $N$ is the total number of frames in that video.

Cosine similarity was then used to compute match scores between different accumulated feature vectors from two different videos. These scores were used for calculating the verification and identification accuracy rates of each CNN.

3.4 Method Yield Rates

Compared to simple 2D alignment, face frontalization often experiences higher failure rates with decreased operational ranges. For instance, a landmarker may have failed to detect the 68 points needed for frontalization due to extreme pose and terminate before the frontalization step. Conversely, a landmarker could have
Figure 3.7. Frontalization success (expressed as yield rate) of the six methods over different pose angles in the CMU Multi-PIE dataset [54].
TABLE 3.1

YIELD OF EACH PRE-PROCESSING METHOD

<table>
<thead>
<tr>
<th>Pre-processing method</th>
<th>CMR &amp; H</th>
<th>KS &amp; H</th>
<th>ZR &amp; H</th>
<th>CMR &amp; OFM</th>
<th>KS &amp; OFM</th>
<th>ZR &amp; OFM</th>
<th>2D alignment (not frontalized)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASIA images (yield)</td>
<td>252,294</td>
<td>255,571</td>
<td>261,951</td>
<td>252,222</td>
<td>266,269</td>
<td>254,381</td>
<td>268,455</td>
</tr>
<tr>
<td></td>
<td>(83.13%)</td>
<td>(84.22%)</td>
<td>(86.31%)</td>
<td>(83.11%)</td>
<td>(87.74%)</td>
<td>(83.82%)</td>
<td>(88.45%)</td>
</tr>
<tr>
<td>PaSC videos (yield)</td>
<td>2,691</td>
<td>2,510</td>
<td>2,497</td>
<td>2,604</td>
<td>2,476</td>
<td>2,508</td>
<td>2,726</td>
</tr>
<tr>
<td></td>
<td>(96.03%)</td>
<td>(89.57%)</td>
<td>(89.11%)</td>
<td>(92.93%)</td>
<td>(88.36%)</td>
<td>(89.51%)</td>
<td>(97.28%)</td>
</tr>
</tbody>
</table>

detected all needed points, but incorrectly localized just one or two, leading to an invalid 3D transform matrix in frontalization. These type of cascading failures lead many samples in CW and PaSC to fail in the landmarking or frontalization step due to extreme scale, pose (> 45 yaw), or occlusion. Hence each pre-processing method yields a unique subset of frontalizable images well below the total original number. The yield varies for each combination, as shown in Table 3.1.

To better understand the operational ranges of each scheme, we frontalized face images from the multi-view partition of the Multi-PIE dataset [54]. All six frontalization techniques (ZR & H, KS & H, CMR & OFM, CMR & H, ZR & OFM and KS & OFM) were tested for each pose in the dataset, including differing expressions and illumination. The pose angles tested were binned into subsets of 0, 15, 30, 40, 60, 70 and 90, along with respective negative angles, using the included labeling from [54]. Failures from landmarking steps or from frontalization steps were not differentiated. The results can be seen in Fig. 3.7.

In general, all methods experienced high failure rates on facial pose angles beyond 40. Methods using CMR for landmarking performed best in the 0 - 40 range. OFM caused slightly more failures than H [60] within a +/- 40 range, but had equal per-
formance on more extreme poses. KS [83] provided superior performance on extreme poses (ZR’s [191] profile landmarker was not used in this study, as we deliberately chose not to include pose estimation).

3.5 Experiments and Results

In this section we present details about our experiments and the subsequent results.

3.5.1 Methodology

To analyze the effect of facial frontalization on recognition performance, we trained the VGG-FACE network separately for each subset of training data pre-processed with a given method. For each method, we randomly partitioned 90% of the CW subset for training, with 10% for validation. A single NVIDIA Titan X GPU was used to run all of our training experiments using Caffe [76]. Network weights were initialized from a snapshot of VGG-FACE pre-trained on 2 million face images. We used Stochastic Gradient Descent [28] for CNN training. The set of hyperparameters for this method was selected using HyperOpt [22] and the same set was repeated across the different experiments to maintain consistency. The base learning rate was set to 0.01, which was multiplied by a factor of 0.1 (gamma) following a stepwise learning policy, with step size set to 50,000 training iterations. The training batch size was set to 64, with image resolution of 224×224. The snapshot at the 50th epoch was used for feature extraction in the testing phase.

For each frontalization method, we also kept two pre-processed versions of the same face: one without any symmetry (asymmetric), such as the left hand side of Fig. 1.2 (c), and the other with symmetry, where one vertical half is used for both sides of the face, as in the right hand side of Fig. 1.2 (c). The half to replicate was chosen automatically based on the quality of the facial landmark points.
For testing each trained network we set two different pipelines for video to video face matching on PaSC - 1) the full set of PaSC video frames was fed to each pre-processing method and only the successfully pre-processed frames were used to test the network trained on CW pre-processed with the same scheme, and 2) the intersection of all PaSC videos successfully pre-processed by all methods was used for testing. Since the yield of each method was different (see Table 3.1), the number of PaSC videos varied for each method in the 1st pipeline. In the 2nd pipeline, all the networks were tested on their congruent pre-processed versions of the same 2267 (out of 2802) PaSC videos.

3.5.2 Results of Recognition Experiments

Results on Full Set of PaSC Videos. For each pipeline, we computed verification performance with a ROC curve, as well as the rank-based recognition performance, i.e., identification using a CMC curve. These performance measures are pertinent in analyzing the behavior of each frontalization scheme. For the 1st pipeline i.e. full handheld PaSC video data, the identification and verification performance of the different CNN models can be seen in Fig. 3.8 and 3.9 respectively. We only show the replication mode (symmetric or asymmetric) which performed the best for each frontalization method.

For the full set of stationary PaSC [141] video data, the identification and verification performance of the different CNN models can be seen in Fig. 3.10 and 3.11 respectively. We only show the replication mode which performed the best for each frontalization method.

Pre-processing both CASIA-WebFace (CW) [175] and PaSC using KS [83] and our frontalization method (OFM) produced the best results with VGG-FACE. The

\[3\] During processing, a slightly larger set was obtained from this method due to an error causing frontalization on images with no detected face.
Figure 3.8. Recognition performance on the full set of handheld PaSC videos (1st pipeline). Pre-processing both the training and testing data with KS & our frontalization method (OFM) outperforms all other methods. Interestingly, the wide gap between the bottom two curves suggests that training with non pre-processed images actually hampered the face representation capability of the network (dotted curve).
Figure 3.9. Verification performance on full handheld PaSC videos (1st pipeline). The trends from Fig. 3.8 transfer to the ROC as well.
Figure 3.10. Recognition performance on the full set of stationary (controlled) PaSC videos (1st pipeline). The overall performance of all methods went up from Fig. 5 in the main text, especially for 2D alignment (dashed curve).
Figure 3.11. Verification performance on the full set of stationary (controlled) PaSC videos (1st pipeline).

rank-1 accuracy improved overall when the data was frontalized (using any method) compared to a simple 2D-alignment pre-processing strategy. Moreover, OFM outperformed H [60] for the different landmarker combinations. The reason for this has been discussed in Section 7.2 of the main text.

Pre-processing both CW and PaSC using the KS [83] landmarker coupled with OFM produced the best results with VGG-FACE. The rank-1 accuracy improved overall when the data was frontalized (using any method) compared to just 2D-alignment. OFM outperformed H [60] in almost all cases, i.e., using different land-
markers. We attribute this to the local adaptation of our 3D model described in Section 3.2.3 in contrast to H [60] which can distort faces (see Section 3.1.2). This preserves higher-frequency features as a result, which can be seen in Fig. 1.2d and 1.2h.

Results on Common Set of PaSC Videos. To further investigate these findings, we leveled the playing field, using a subset of PaSC testing videos successfully pre-processed by all methods in the 2nd pipeline. A total of 1070 handheld videos were used for these experiments. The performance results of this experiment can be seen in Fig. 3.12 and 3.13. Even with equal datasets, KS [83] and OFM outperform other methods. The increased performance of the 2D alignment network suggests that its higher yield in the previous experiment provided more difficult frames to match, and subsequently hindered performance.

A curious observation we made was that training the network with 2D-aligned face images (diverse in facial pose) negatively affected recognition performance when PaSC was frontalized, regardless of the pre-processing method used for frontalization. This suggests that performing frontalization at testing time may not benefit performance on pre-trained networks. Instead, training and testing must be pre-processed under consistent methods to realize any performance benefit.

The performance of the same networks when tested on the common set of stationary PaSC videos i.e., 2nd pipeline, can be seen in Fig. 3.14 and 3.15. A total of 1197 controlled videos on PaSC, out of the full set of 1401 controlled videos, were successfully frontalized by all methods and used for these experiments.

Interestingly, the network trained with 2D aligned CW [175] performed best among all networks in this experiment. This suggests that for video to video face matching task with the same number of good quality (controlled) videos, a 2D alignment pre-processing step might be the best and cheaper option when compared to frontalization.
Figure 3.12. Recognition performance on the common handheld PaSC videos (2nd pipeline). KS & OFM slightly exceeded the best performance reported in Fig. 3.8. The 2D alignment (dashed) curve made a big jump from Fig. 3.8 suggesting the 2D alignment bin from Table 3.1 had more difficult frames than other methods due to its higher yield.
Figure 3.13. Verification performance on the common handheld PaSC videos (2nd pipeline). Although KS [83] & OFM slightly outperformed other methods at FAR = 0.01, a simple 2D alignment step beat other methods for higher FARs.
Figure 3.14. Recognition performance on the common set of stationary (controlled) PaSC videos (2nd pipeline). The 2D alignment network (dashed curve) outperformed all the frontalization methods in this experiment.
Figure 3.15. Verification performance on the common set of stationary (controlled) PaSC videos (2nd pipeline). The trends from Fig. 3.14 in the supplementary text followed in this experiment as well.
Another recurring trend that we noticed is that recognition performance is slightly improved when the face images are reconstructed asymmetrically rather than symmetrically. This is validated by the fact that only the symmetric version of CMR & H [60] outperformed its asymmetric counterpart among the six frontalization schemes. While symmetrically reconstructing faces can provide a more visually appealing result, important data still present in the occluded side of an off-pose face can be destroyed by such operations. By superimposing portions of the non-occluded face regions to fill in gaps on the occluded side, artifacts are inevitably introduced onto the reconstructed face. We suspect these artifacts to be detrimental to the feature learning of a CNN, and consequently its recognition performance suffers.

**Cross Domain Experiments.** To evaluate the effect of frontalization only on the testing side, we tested the network trained on 2D aligned CW with the PaSC subsets frontalized using the different methods. We compared the performance of this network on each version of frontalized PaSC with that of the network trained with CW frontalized with the same scheme. The results of these experiments on handheld and stationary PaSC videos can be found in Fig. 3.16, 3.17 and Fig. 3.18, 3.19 respectively. We found the pre-trained VGG-FACE to always perform consistently (25 - 30% rank-1 accuracy) for any mode of training frontalization, while the network trained with the original CW images (no pre-processing) to always perform poorly (rank-1 accuracy < 5%). That is why they are not presented in these plots.

Surprisingly, we found the network trained on 2D-aligned CW to be outperformed in most cases by the network trained on its frontalized version when tested on PaSC frontalized with the same scheme. This suggests that performing frontalization at testing time does not benefit performance on pre-trained networks. Instead, training and testing must be pre-processed under consistent methods to realize any performance benefit. We attribute this result to the domain mismatch between training (2D alignment) and testing (frontalization) data. The network, when trained with
Figure 3.16. Recognition performance on the full set of handheld PaSC videos (1st pipeline), where PaSC was frontalized but CW was either frontalized or 2D aligned. The 2D alignment network (dashed curves) in most cases was outperformed by its corresponding frontalized network.
Figure 3.17. Verification performance on the full set of handheld PaSC videos (1st pipeline), where PaSC was frontalized but CW was either frontalized or 2D aligned. The trends from Fig. 3.16 in the supplementary text followed in this experiment as well.
2D aligned face images, is exposed to a variety of facial poses, including frontal or near-frontal images. However, the facial features of a face image are systematically displaced from their expected location in the trained 2D-aligned model during frontalization, as clearly evident from Fig. 1.b and 1.d in the main text. This suggests that the CNN model learned using 2D-aligned faces is not flexible enough to accommodate for this difference in facial feature locations.
Figure 3.19. Verification performance on the full set of stationary (controlled) PaSC videos (1st pipeline), where PaSC was frontalized but CW was either frontalized or 2D aligned.
3.6 Conclusion

Several conclusions can be drawn from our experiments and used to inform future face recognition experiments:

- Frontalization is a complex pre-processing step, meaning it can come at a cost. Due to the large number of failure modes it introduces, there can be significant loss of data, *i.e.*, lower yield, specifically with images containing extreme pose or occlusion. Additionally, frontalization can prove to be computationally expensive, meaning the performance benefit frontalization can provide must be weighed against the needed increase in computational resources.

- Our proposed method, which dynamically adapts local areas of the 3D reference model to the given input face, provides better performance improvements than that of Hassner et al. [60] for PaSC video recognition.

- Both the training and testing data must be pre-processed under consistent methods (*i.e.* same frontalization steps) to realize any performance benefit out of frontalization.

- While symmetrically reconstructed frontalized faces may yield more visually appealing results, asymmetrical frontalization provides slightly superior performance for face recognition.

- Training a CNN with millions of face images makes it relatively agnostic (in terms of recognition performance) to the pre-processing method used on testing data, *e.g.*, the VGG-16 model [146] pre-trained on the 2.6M face images from the VGG-Face dataset [121] performs consistently across different experiments.

From these observations, we can conclude that the usefulness of frontalization to pre-process test set faces can be dependent on the facial recognition system used. Depending on how the recognition system in question was trained, and the failure threshold set, as noted in Section 3.5.2 a simple 2D-alignment might be more productive in some cases. Therefore, face frontalization should be taken with a grain of salt, as it may not always provide superior results.
CHAPTER 4

SYNTHESIS OF REALISTIC EXAMPLE FACE IMAGES

This Chapter is based on the work described in [13], which was published in IEEE IJCB 2017. The proposed face synthesis method was originally developed in collaboration with John S. Bernhard while he was a graduate student at the University of Notre Dame.

The need for artificially augmenting a dataset, before using it to train a CNN, comes from the network overfitting on the training data. The more variations of a sample one can include in the training set, the higher the chances of avoiding overfitting. Researchers have generally used simple transformation methods like rotation, scaling, cropping, mirroring and translation for augmenting their datasets. The idea of augmenting an existing face dataset by actually changing the visual appearance of real face images was first proposed in [107]. Their augmentation techniques include - 1) reposing the face images into different pose angles using the frontalization method in [60], 2) reshaping the face into 10 different variations using 3D models from the Basel 3D face set [123] and 3) changing the orientation of the mouth (open and closed). Using these methods, the CASIA-WebFace dataset was augmented from 500K to 2.4M images for 10,575 subjects. Experiments showed that training a CNN (VGG-19 [146]) with the augmented dataset did indeed improve face recognition accuracy, compared to using only the original dataset.

Although this augmentation method solves the problem of generating large number of samples per subject, it is still limited by the number of samples available in the original dataset. An augmentation technique capable of generating large number
Figure 4.1. While traditional augmentation methods have explored face image cropping, translation and scaling (left column), we can augment a dataset by creating synthetic face images of existing identities (middle column) and new synthetic identities altogether (right column). This augments both the depth and breadth of a dataset.

of samples while arbitrarily expanding the number of subjects of a dataset would definitely be an improvement. We solve these problems with a system for automatically generating synthetic face images, of either a known real identity or of a synthetic identity not corresponding to any real person. When face images are generated for synthetic identities rather than any real person, privacy issues are avoided. Combining these two modes, our approach can augment an existing dataset by increasing the number of face images per real identity, and by increasing the total number of (real + synthetic) identities, as can be seen in Figure 4.1.
4.1 Collection of the SREFI Donor Set

The donor set for SREFI could be based on any dataset of real face images of a sufficient number of different persons. We used a dataset of donor face images created from an existing publicly-available dataset [129], in which multiple frontal images of each subject were acquired in different sessions. Subjects varied in gender, age and ethnicity. We used cropped, aligned and resized (to 512x512) versions of the images for our experiments (Figure 4.2). For the sake of generating realistic synthetic face images, the donor set was subdivided by race and gender and images with very thick facial hair or glasses were removed. Table 4.1 shows the image count and subject breakdown of the donor set.

Figure 4.2. Example set of six images from the donor set. A male and female of each ethnic group present in the donor set are shown.
### TABLE 4.1

SREFI DONOR SET DISTRIBUTION

<table>
<thead>
<tr>
<th>Ethnicity</th>
<th>Male images (subjects)</th>
<th>Female images (subjects)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caucasian</td>
<td>7108 (678)</td>
<td>5510 (600)</td>
</tr>
<tr>
<td>Asian</td>
<td>1903 (100)</td>
<td>1286 (74)</td>
</tr>
<tr>
<td>African American</td>
<td>67 (9)</td>
<td>65 (10)</td>
</tr>
</tbody>
</table>

4.2 Our Synthesis Method - SREFI

The steps in our synthesis method, the workflow of which is depicted in Figure 4.3, are described below.

#### 4.2.1 Landmarking and Triangulation

The base face was initially divided into triangular regions using a subset of the 68 facial landmark points acquired using the method from [83] (Fig 4.3.a). Since these landmark points occur at important locations of the face, such as the mouth, inaccuracies in detecting them would lead to artifacts (e.g., multiple mouths) appearing in the synthesized faces. To address this, we developed a triangulation that moved important facial features away from triangle corners, resulting in more visually stable regions. Using the landmark points and our initial triangulation of the face (Figure 4.4.a), we obtained centroids using the three vertices of a triangle. We created a new adjusted triangulation of the face by joining a centroid with the adjoining centroids of the triangles in its neighborhood (Fig 4.4.b). This allowed each region to be replaced in the target face using the region shape from the donors. Additionally, the outer part of the image was triangulated in order to allow the outer shape of the
Figure 4.3. Workflow of the SREFI method on a high level where the base face image (a) is landmarked and triangulated (b) using the Dlib \[86\] software, and then the triangles are re-distributed using the centroids (c). Similar face images from the donor pool are then used to replace the base face triangles (d) to synthesize a rough version of the new texture (e), which are further made seamless using multi-scale blending (f).

face to be modified. Our triangulation method is different from the one proposed in \[113\], as they used pre-defined points from the barycentric averages of landmark points extracted using the method from \[191\], to create masks of only three main facial regions.

4.2.2 Selection of the Donor Pool

We constructed the donor pool by selecting face images to the base face in a lower-dimensional feature space. For this purpose, we extracted the 4096-dimensional feature vector for each face image in our donor set (Table 4.1) from the fc7 layer of a snapshot of the VGG-FACE CNN model \[121\] pre-trained on 2.6M face images in the VGG-FACE dataset \[121\]\footnote{Available here: \url{https://github.com/BVLC/caffe/wiki/Model-Zoo}}. We computed the mean feature vector for each subject of the dataset by averaging the feature vectors for each image of that subject.

The similarity score between two images was calculated using these features.
Figure 4.4. Example of adjusting the facial triangles - (a) the initial triangle set, (b) the new triangles after relocating them using the centroids of the initial set (a). The different facial regions like the eyes, nose and mouth are further separated by this adjustment.
distance function can be used for scoring: we chose cosine similarity as it has been extensively used to match VGG-FACE features by researchers [140]. The similarity score $S(v_1, v_2)$ between feature vectors $v_1$ and $v_2$ was calculated as:-

$$S(v_1, v_2) = \frac{v_1 \cdot v_2}{\|v_1\|_2 \|v_2\|_2} \quad (4.1)$$

where $S(v_1, v_2)$ is the similarity score between two vectors $v_1 = [f_1, f_2, f_3, ..., f_{4096}]$ and $v_2 = [f_1, f_2, f_3, ..., f_{4096}]$. If $v_1$ and $v_2$ are perfectly alike (0 angle), then $S(v_1, v_2)$ is 1. In our case, a score closer to 1 indicated proximal subjects in the feature space.

For each subject, we stored the face images of its $N$ such proximal subjects as the potential selection pool, as illustrated in Figure 4.5. To generate synthetic images of existing subjects, i.e. expanding a real identity, we simply created the donor pool with all the real face images of that subject.

4.2.3 Attribute Based Reshaping of Facial Parts

Once the donor pool was selected, we implemented a re-shaping step before the triangle replacement. Due to variations in facial structure, positioning regions from the donor on their relative position in the base face could give the resulting synthetic face a distinctly non-real appearance (overly stretched or compressed). To address this we reshape the synthetic face using natural face shape ratios.

Previously, researchers have used ratios based on the height of the full head or hand marked forehead points in relation to perceived beauty of a face [142]. As we did not have facial landmarks of the forehead or head top points, we instead obtained our own distribution of face shape ratios from the donor set. For each ethnic group of a specific gender, we constructed a rank ordered list of the ratio of length of the facial regions like the eyes, nose and the mouth from the subjects in the donor pool. Then we computed the inter-quartile range (IQR) from this list for each group.
Figure 4.5. Selecting the donor pool for a base image (middle). The face images connected with heavier edges are proximal to the base face in appearance compared to the ones connected with lighter edges. The $N$ proximal faces are represented as the faces within the circular region around the base face.
When synthesizing a new face for that group, the eyes, nose, and mouth regions were positioned on the target face so as to adhere as closely as possible to this IQR value. Using these face ratio ranges, the vertical positioning of the parts of the face became much more natural in appearance.

4.2.4 Triangle Replacement

After the facial reshaping, we replaced those regions with corresponding triangles from face images in the base face’s donor pool. To preserve the visual uniformity of the vital facial regions, such as the mouth, nose and the eyes, we designated all triangles in that region to come from the same donor. Without this restriction, the mouth, nose and eyes appear as an amalgamation of multiple donors. This was not a concern for the rest of the face, however, so the triangles in the cheek and jaw area were chosen separately from the donor pool. Therefore, the number of donor face images used for synthesis, $C_{\text{donor}}$, can be tuned by the user depending on the faces available in the donor pool and the degree of distinctiveness sought in the synthetic face. A smaller value of $C_{\text{donor}}$ ensures the uniformity of each facial part, while a higher value makes the synthetic face appear more distinctive. For our experiments and donor pool, we found the synthetic faces looked more realistic when $C_{\text{donor}}$ was set between 7 (for smaller donor pool size) and 10.

4.2.5 Adjusting the Color Distribution of Replacement Triangles

After the donor triangles were selected, their color distributions were individually adjusted to be closer to that of the base face triangles to deal with intensity changes across the face due to lighting. This was done by simply shifting the color distribution of the donor triangle to have the same mean as the base triangle that it replaced. At first, the difference in mean intensities between the base and the corresponding donor triangles was computed for each color channel and this value was used as an
offset to the intensities of the corresponding channels in the donor triangle.

Without this color readjustment, the synthetic face can appear splotchy in places, but with this color adjustment step, these patches are no longer noticeable. Adjustment in the HSV color space tends to leave a pinkish tint to some faces and does not perform well for darker skin tones. For this reason SREFI currently uses color adjustment in RGB color space, as shown in Figure 4.6.

4.2.6 Blending in the Replaced Triangles

Placing the triangles together on reshaped facial parts does make the synthetic face look different from the base face, but it seems unnatural as the transition of intensity across the donor triangles may not be smooth. This un-blended synthetic face can be seen in Fig 4.3e. To make the synthetic face more natural, the last step of our method blends the triangles together using Laplacian pyramids [30].

The blending process starts with the base face, a mask to specify the position of the triangle on the face to be replaced, and the donor face, after reshaping and color adjustment. Then, Gaussian pyramids are created for each color channel of these
three images. The images at each level of the pyramid were scaled down by a factor of 4 from the level below it by applying a Gaussian blur using a $5 \times 5$ kernel. Each pyramid was built with 4 such levels. We created two Laplacian pyramids from the base face and donor face Gaussian pyramids using the intensity difference between the image at a level of the Gaussian pyramid and the expanded version (by a factor of 4) of the image at the level immediately above it. The image at the top-most level of the Gaussian pyramid was stored as it is at the top-most level of the corresponding Laplacian pyramid.

A third Laplacian pyramid was generated for the blended face using the triangular mask values as a switch. For each level of the Gaussian pyramid of the mask, we added pixel values from that level of the base Laplacian pyramid if the mask value was 1, or from that level of the donor pyramid if the mask value was 0, to the third Laplacian pyramid. This was done as follows:

$$
\begin{align*}
    p_1 &= (g_{mask})_i \ast (l_{base})_i \\
    p_2 &= (1 - g_{mask})_i \ast (l_{donor})_i
\end{align*}
$$

where $(g_{mask})_i$ is the image at the $i$-th level of the mask Gaussian pyramid and $(l_{base})_i$ and $(l_{donor})_i$ are the images at the $i$-th level of the base and donor Laplacian pyramids respectively. The image at the $i$-th level of the new Laplacian pyramid for the blended face, $(l_{blend})_i$, was generated by simply adding the two images, $p1$ and $p2$, together.

To integrate the blended images of different resolutions in the new Laplacian pyramid, we collapsed them together from top-to-bottom. This was done by adding the expanded version of the image at the $i$-th level $expand((l_{blend})_i)$ and the image at the $(i-1)$-th level $i.e.$ $(l_{blend})_{i-1}$. The blended image was further normalized with pixel intensities less than 0 changed to 0 and those greater than 255 changed to
Figure 4.7. Example of a synthetic face blended using (a) Laplacian pyramids [30] and (b) the Poisson blending method [125]. Notice the overlapping eye, nose and mouth regions.
255. We merged the collapsed images for the three color channels together to get the final blended image. The blended image looks quite natural as shown in Fig 4.3f. Interestingly, this method achieves better regional blending for human faces than the popular Poisson blending (seamless blending) [123], which tries to incorporate pixels from both the source and destination in the blended region, generating faces with overlapping second nose or mouth (Figure 4.7).

4.3 Experiments and Results

The experiments described in this section are meant to assess the realism, uniqueness and stability of the synthetic faces.

4.3.1 Human Rater Study

One way that we used to assess the realism of the synthetic faces was a human rater study. The experiment had 20 raters who were novices at the task, and had not previously seen any of our synthetic faces. Each rater participated in three different experiments. In one they rated the realism of a face image, and in two other experiments they rated whether two face images came from the same subject. The experiments used the PsychoPy psycho-physics framework to present stimuli and record the responses [124].

4.3.1.1 Experiment 1

Human subjects participated in this study under a human subjects protocol approved by the University’s IRB. The participants were shown either a real or a synthetic face image, and asked to rate whether it is a real face, using a three-valued Likert scale [98]. We selected 100 real and 100 synthetic frontally posed, 2D aligned face images (512 × 512 in size), 50 male and 50 female each, which belonged to 3 ethnic groups (Caucasian, Asian and African American). Some example images can
be seen in Figure 4.8a. Before starting the experiments, raters were shown two practice trials - a real face image labeled as real and a synthetic face image labeled as synthetic. Each participant was shown the 200 images in a random order with each image being shown for two seconds (as in [160, 119]) and asked the question - “Is this face image real?” The rater had to respond by pressing a key for “Yes”, “No” and “Cannot decide”.

Although the face image disappeared after the two seconds, the next frame (image) appeared only after the participant had registered their response. The participants performed well in this experiment, marking the face images correctly 92% of the time on average. The scores also suggest that the raters had less difficulty in detecting synthetic faces in female subjects than male subjects. Furthermore, we found the participants to be slightly more inclined towards marking a real face synthetic than a synthetic face real.

To better understand which real (synthetic) face images look synthetic (real)
to the participants, we selected the face images generating the most inaccurate responses from the participants, as can be seen in Figures 4.9 and 4.10 respectively. Interestingly, these faces are not limited to a specific ethnicity. The face images, both male and female, for which the participants chose the ‘Cannot decide’ option most frequently can be seen in Figure 4.11.

4.3.1.2 Experiment 2

The goal of this experiment was to evaluate how reliably a pair of images of the same (different) synthetic identity are rated by a human observer as being of the same (different) person. For this experiment, 200 pairs of frontally posed and 2D aligned synthetic face images (512 × 512 in size) were generated. For the authentic-pair trials, one pair of images was generated for each of 50 synthetic male identities and 50 synthetic female identities. For the impostor-pair trials, 50 different-identity male image pairs and 50 different-identity female image pairs were generated. All
Figure 4.10. Synthetic images which generated the most inaccurate responses across the 20 participants.

Figure 4.11. Face images which generated the most ‘Cannot decide’ responses across the 20 participants. The male face image is actually real and 10/20 participants chose ‘Cannot decide’ for it. The female face image is actually synthetic and 8/20 participants chose ‘Cannot decide’ for it.
different-identity pairs were of the same ethnicity. Before the actual trials, raters were shown two practice trials, one with a pair of images from the same synthetic identity, labeled as such, and a second pair from two different identities, labeled as such. For each image rater, the 200 image pairs were shown in a random order for two seconds each. For each pair, the image rater responded to the question - “Are these two images of the same person?” by pressing a key for “Yes”, “No” or “Cannot decide”. This is similar in theme to the study described in [133]. The human ratings from this experiment, with 82% average accuracy, suggested participants found it relatively harder to correctly match synthetic face images than the simpler detection task. Additionally, we found the participants to be slightly more inclined towards marking a true match pair a non-match than vice versa.

To better understand which true match (non-match) face image pairs look dissimilar (similar) to the participants, we selected the image pairs generating the most inaccurate responses from the participants, as can be seen in Figures 4.12, 4.13, 4.14 and 4.15. Interestingly, many of these faces are limited to the ‘Caucasian’ and ‘Asian’ ethnic groups - the ethnic groups with the two highest donor image counts. The face image pairs, both male and female, for which the participants chose the ‘Cannot decide’ option most frequently can be seen in Figures 4.16 and 4.17 respectively.

4.3.1.3 Experiment 3

The goal of this experiment was to evaluate the degree to which synthetic images of a real identity are interchangeable with real images of the same identity in a matching context. Three sets of 100 face image pairs were used in this experiment: one pair of real images of each of 100 real identities, one (real image, synthetic image) pair of each of 100 real identities, and one synthetic image pair of each of 100 identities. In all three cases, the 100 pairs were split evenly between male and female. The real image pairs came from different day acquisitions. Each rater was shown the 300 image
Figure 4.12. Male true match pair which generated the most inaccurate responses across the 20 participants (synthetic).

Figure 4.13. Female true match pair which generated the most inaccurate responses across the 20 participants (synthetic).
Figure 4.14. Male non-match pair which generated the most inaccurate responses across the 20 participants (synthetic).

Figure 4.15. Female non-match pair which generated the most inaccurate responses across the 20 participants (synthetic).
Figure 4.16. Male pair (actually ‘non-match’) which generated most ‘Cannot decide’ responses (10/20) across the 20 participants (synthetic).

Figure 4.17. Female pair (actually ‘non-match’) which generated most ‘Cannot decide’ responses (10/20) across the 20 participants (synthetic).
pairs in a random order. For each trial, the image rater answered the question - “Are these two images of the same person?” by pressing a key for “Yes”, “No” or “Cannot decide”. Before starting the experiments, the image raters were shown three practice trials, one corresponding to each of the three conditions. The participants did better in this experiment, with the average accuracy of 90%. However, they tended to make incorrect decisions more frequently for female face pairs compared to male face pairs.

We also calculated the average matching accuracy of the 20 participants in matching a pair of real images (RvR), a real image to a synthetic one (SvR) and a pair of synthetic images (SvS) for the same subject for male and female subjects separately, as shown in Figure 4.18. The error bars suggest that there is no discernible difference in performance between the three pair types. This means we can interchange a real face image with a synthetic face image of the same real identity for face pair matching without any significant drop in recognition performance.

To better understand which face image pairs looked dissimilar to the participants, we selected the image pairs generating the most inaccurate responses from the participants, as can be seen in Figures 4.19 (male), 4.20 (female), and 4.21 (female). Interestingly, both the face images in Fig 4.21 are real. The face image pairs, both male and female, for which the participants chose the ‘Cannot decide’ option most frequently can be seen in Figures 4.22 and 4.23.

4.3.2 Evaluating Uniqueness - Face Matching Experiments with VGG-FACE

To evaluate the uniqueness of synthetic face images and identities generated using our method, we performed face matching experiments using VGG-FACE [121] pre-trained on 2.6M face images. We prepared two augmented versions of the SREFI donor set (Table 4.1), which had 15,939 real face images of 1471 real subjects (Real ID). The first augmented dataset contained 15,939 synthetic face images of the same 1471 real subjects as in the SREFI donor set (i.e. we artificially expanded each
Figure 4.18. Average matching accuracy, standard deviations indicated with error bars, of session 3 participants for each pair type - real vs real (RvR), synthetic vs real (SvR) and synthetic vs synthetic (SvS).
Figure 4.19. Male pair which generated most inaccurate responses across the 20 participants. Both the face images are synthetic.

Figure 4.20. Female image pair which generated the most inaccurate responses across the 20 participants. One face image is real (left) while the other is synthetic (right).
Figure 4.21. Female pair which generated 2nd most inaccurate responses across the 20 participants. Both the face images are real.

Figure 4.22. Male image pair (both real) which generated the most ‘Cannot decide’ responses (10/20) across the 20 participants.
identity (Expand ID)). The second augmented set comprised of 31,878 synthetic images of 2942 synthetic identities generated by selectively recombining elements from the SREFI donor pool (Synth ID).

The images of the three datasets (Real ID, Expand ID and Synth ID) were supplied to the pre-trained VGG-FACE. The 4096-D vector from its fc7 layer was stored for each image as its feature representation. We performed four independent matching experiments, using cosine similarity as our scoring scheme, with these feature representations - 1) Real ID with Real ID, 2) Synth ID with Synth ID, 3) Expand ID with Expand ID, and 4) Expand ID with Real ID. The ROC curves generated for the experiments can be seen in Fig 4.24. Although the accuracy while matching only real subjects is the highest, the augmented dataset generated from the real subjects matched well with each other and the real dataset as well.

The similarity in verification performance for all the three datasets suggests that the synthetic images can be used to supplement existing face image datasets by not
Figure 4.24. ROC curves for the four matching experiments performed with pre-trained VGG-FACE and cosine similarity.
only increasing the number of images per subject but also generating different face
images of entirely new subjects without any significant loss in recognition perfor-
mance. Therefore, this augmentation process can aid researchers looking to augment
face datasets modest in size.

To get an idea of how the background of the face region influences the face match-
ing experiments, we performed the same experiment with the same VGG-FACE
model and the same face images, but masked the face region out this time. An
example masked out face has been shown in Figure 4.25.

The recognition accuracy dropped in general, except for real ID vs real ID, after
masking out the faces when compared to just cropped faces, as can be seen in Table
4.2. Specific true accept rates (TAR) at a fixed false accept rate (FAR) of 0.01 have
been listed in that table. We suspect the drop in performance is due to two reasons
- 1) the background of the face images in our dataset do contribute to the overall
feature representation process of a CNN; the black pixels mess up low level detectors
TABLE 4.2

TAR AT FAR = 0.01 WITH CROPPED AND MASKED FACE IMAGES

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Cropped Face</th>
<th>Masked Face</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real ID vs. Real ID</td>
<td>0.990</td>
<td>0.991</td>
</tr>
<tr>
<td>Expand ID vs. Expand ID</td>
<td>0.954</td>
<td>0.924</td>
</tr>
<tr>
<td>Expand ID vs. Real ID</td>
<td>0.964</td>
<td>0.937</td>
</tr>
<tr>
<td>Synth ID vs. Synth ID</td>
<td>0.931</td>
<td>0.915</td>
</tr>
</tbody>
</table>

and the pooling layers spread the contamination around., and 2) since the VGG-FACE dataset [121] consists of only cropped face images, the VGG-FACE CNN has learned to use background for face matching as well.

4.3.3 Evaluating Stability - Training VGG-FACE and Testing on the GBU Dataset

To test the stability of the synthetic faces generated using our method for CNN training, we prepared two augmented datasets from the original SREFI donor set (Table 4.1). We trained networks with the VGG-FACE architecture from scratch on these datasets and used the trained CNNs to match extremely challenging face image pairs from the ‘Ugly’ partition of the “Good, Bad and Ugly” dataset (GBU) [128]. Since some subjects were present in both the SREFI donor set and GBU, we removed the face images of these common subjects from the SREFI donor set. Example images of a common subject can be seen in Figure 4.26. This reduced the donor pool to 10,692 face images from 1296 real subjects.

The first augmented dataset had 10,692 real images of 1296 real subjects, 10,692 synthetic images of the same 1296 subjects and 84,636 synthetic images of 10,440
Figure 4.26. Two images of a subject present in both the SREFI donor set (left) and the GBU dataset (right).

synthetic subjects generated from the SREFI donor set. The second augmented dataset contained 10,692 real and 21,384 synthetic images of the 1296 original subjects and 176,098 synthetic face images from 21,538 synthetic subjects. Therefore, the two augmented datasets contained 106,020 and 208,174 face images respectively. To gauge the effectiveness of these augmented datasets, we prepared a third dataset by selecting 260,882 (real) face images from the CASIA-WebFace (CW) dataset. All the images in these three datasets were aligned about their eye-centers extracted using the method in, cropped and resized to 224 × 224. We trained the VGGFACE model from scratch with these three datasets independently for 50 epochs, using Caffe. To maintain consistency, we used the same set of hyper-parameters, learning function (SGD) and the same NVIDIA Titan X GPU for all three training sessions.

Once training for all three sessions terminated, we used the saved snapshots to extract 4096-D feature representations from the fc7 layer of the CNN for each face
image from every pair in the ‘Ugly’ partition of GBU. We normalized the feature vectors between a range of 0 and 1 using linear min-max normalization. After normalization, we computed matching scores, using cosine similarity, between feature vectors of the query and target face images in the GBU ‘Ugly’ partition. The ROC curves generated from the three matching experiments can be seen in Fig 4.27.

Interestingly, the networks trained with the two augmented datasets significantly outperformed the snapshot trained with the face images from CW. This might be due to the fact that face images in CW vary in facial pose and illumination while the
original and synthetic images in our augmented datasets and the GBU face images were frontally posed. Moreover, the CNN trained with the larger augmented dataset (208k) performed better than the CNN trained with the smaller dataset (106k). This suggests that increasing the number of synthetic face images and using the augmented dataset for CNN training might further boost the network’s performance in recognition experiments.

4.4 Discussion

In this chapter, we propose a novel method for generating natural looking synthetic facial textures (2D images) for real and synthetic identities. This method can benefit face biometric research by: (a) replacing the manual collection of face image datasets which require abundant time and resource, and (b) augmenting existing face image datasets to create larger and deeper supersets for training CNN models and improve their face representation capability.

To test the fidelity of synthetic face images and identities generated with our method, we performed a human evaluation study where human raters: (1) correctly detected real and synthetic face images once shown an example, (2) correctly matched a pair of face images from the same person (real or synthetic). To test the uniqueness of these synthetic faces, we used the pre-trained VGG-FACE model’s feature representations. The ROC curves generated for matching a synthetic image pair or a real and a synthetic image pair was found to be proximal to the ROC curve for matching a pair of real face images. Finally, we trained the VGG-FACE model from scratch with two augmented datasets (208k images and 106k images) generated using our method and a subset of the CASIA-WebFace dataset (260k images) independently. We found the CNN trained with the larger dataset to outperform the CNNs trained on the smaller dataset and the CASIA-WebFace subset while matching face image pairs from the ‘Ugly’ partition in the GBU dataset. This suggests
the synthetic face images and identities generated using our method are stable for CNN training and can boost its face recognition performance.
Chapter 5

Fast Rendering of Synthetic 3D Facial Masks

This Chapter is based on the work described in [15], which was published in IEEE WACV 2019, and a manuscript currently under review for the IEEE T-IP [16]. The synthetic texture generation procedure described here is an improved version of that explained in the last chapter.

5.1 The Proposed Method

On a high level, SREFI starts with a real frontal face image (the base face), whose facial region is triangulated using landmark points. We replace the triangles of this face image with corresponding triangles from other real face images (donor images), similar in appearance, and blend the triangles together to obtain a synthetic face texture and filter by visual quality. To render this texture in 3D, at different yaw values, we use best-fitting 3D models using texture and shape parameters. A high level overview is shown in Figure 5.1.

5.1.1 Synthetic Texture Generation

In this section we describe our texture generation pipeline. We use a subset of the public dataset in [129], comprised of 15,807 face images of 1,352 identities, to synthesize these textures. The distribution of images and demographic diversity in this subset can be seen in Table 5.1. All the images were aligned about their eye centers and resized to 512×512 beforehand as a pre-processing step. Our texture
Figure 5.1. The SREFI pipeline. (a) The input base face $I_b$ is triangulated using landmark points. (b) Its donor pool $D$ is constructed using hypercolumn feature maps, represented by colored bars. (c) The synthetic texture $I_s$ is created using images from $D$. (d) $I_s$ is subjected to quality based filtering. (e) Best fitting 3D models are obtained using proximity in landmark and deep feature (represented by blue bars). (f) Dense triangular mesh is generated from corresponding 3D models. (g) Multi-pose and shape renderings of $I_s$. 
Finding the Number of Donors Necessary for Anonymization. The method in [13], described in Chapter 4 of this dissertation, replaces facial regions with seven to ten donors to create a synthetic texture based on gallery size. However, this *ad hoc* technique may lead to natural looking texture when the gallery is large; for a smaller gallery, the generated texture can look distinctly non-uniform as we transition from one larger face part to another. Therefore, we regularize the donor selection process by determining the best number of donors, \( N \).

We choose a random set of 1,545 face images of 100 identities from our gallery
to gauge the optimal $N$. The facial region of each image was triangulated using Delaunay’s method from landmark points extracted using Dlib [86]. We shifted these triangles using their centroids to separate facial regions like the eyes, nose, mouth from each other [13]. We replace triangles with one donor assigned per region, to anonymize the original face image, i.e., change its identity. We vary donor size $N$ from 0 (original image) to 5 (swapping regions with 5 donors) and use normalized $fc7$ layer features from the pre-trained VGG-FACE network [121] to match the set of synthesized faces with the original set (using cosine similarity). We use the True Accept Rate (TAR) at False Accept Rate (FAR) of 0.01 as our performance metric. For $N = 0$, the face images match each other very well with TAR over 0.96 but at $N = 4$ the TAR drops precipitously to only 0.12 (Fig. 5.2). Therefore, replacement with 4 donors can anonymize the original face image and result in a new synthetic identity.

**Pooling Proximal Faces: Hypercolumns.** Given a base face image, we construct its donor pool $D$ with face images of potential donor identities. We choose these identities based on proximity in feature space from the base face. For a gallery face image, we extract its hypercolumn [58] descriptor using $conv$-$[1_2,2_2,3_3,4_3,5_3]$ feature maps from a pre-trained VGG-FACE network [121]. We use hypercolumn features, instead of the high level features of the $fc7$ layer [13], as they capture information at different spatial contexts. However, the hypercolumn feature maps extracted consist of 434 dimensions for each pixel of an image. Hence, for a $512 \times 512$ gallery image $I$, a $512 \times 512 \times 434$ hypercolumn vector $V_I$ is obtained. To reduce computation time and feature redundancy, we sample a $68 \times 434$ feature map of $V_I$ from the 68 landmark points (pixels) of the face, obtained using Dlib [86]. A mean vector $S_I$ for each identity is obtained by computing the average of $V_I$ for all its images. We calculate
the distance between any two identities as follows:

\[
d(I_1, I_2) = \sum_{i=1}^{68} \sum_{j=1}^{434} |(S_{I_1})_{i,j} - (S_{I_2})_{i,j}|,
\]

where \( I_1 \) and \( I_2 \) are two gallery identities, \( S_{I_1} \) and \( S_{I_2} \) are their mean sampled hypercolumn feature maps and \( d(I_1, I_2) \) is the distance between them. For the base face, we cluster identities with the lowest \( d \) values and construct its donor pool \( D \) with their face images. If creating synthetic texture of a real identity, \( D \) is simply composed of all images of that identity in the gallery. For each gallery identity, \( D \) is constructed and stored as an offline step.

**Stitching Triangles Together.** Once the donor pool \( D \) is constructed, we randomly select four donors to replace triangulated regions of the base face with a specific donor only assigned to a particular region. Prior to replacement, each donor triangle is randomly reshaped with parameters within the inter-quartile range of permissible shapes biologically possible for a given gender and race [13] (as in Chapter 4). Each triangle is then color adjusted by shifting the mean RGB values of its pixels to be same as that of the base face triangle it replaces. The donor triangle is then overlaid on the base face and blended in using Laplacian pyramids [30]. This approach blends pixels from the donor image \( I_d \) for the triangle mask \( M \) at different resolutions of the base face \( I_b \) situated at different levels of the Laplacian pyramid as:

\[
I_s = M \circ I_b + (1 - M) \circ I_d,
\]

where \( \circ \) denotes element wise product and \( I_s \) is the blended image. The output synthetic face image is generated by collapsing the pyramid top-down and blending all triangles one by one from all four donors.
Quality Estimation. Even after constructing the donor pool $D$ with identities similar in appearance to the base face, we found many of the generated samples to look unnatural. Most of the unnatural images had either asymmetric facial shape or subtle visible blending seams. To discard these unnatural looking synthetic textures, we tried two approaches: i) using the discriminator of a DCGAN [132] model trained with our gallery data [129] to label our synthetic samples as ‘real’ (natural) or ‘fake’ (unnatural), and ii) train a linear SVM model with ‘good’ and ‘bad’ samples as labelled by human raters. We took the DCGAN implementation from https://github.com/carpedm20/DCGAN-tensorflow and trained the model with the 15,807 images from our gallery [129]. We trained the model for 200 epochs and then used this trained discriminator model to label synthetic samples generated by our method as real or fake. However, it labelled all samples as real. We found the quality of the samples generated by the generator of the DCGAN model to be vastly inferior compared to our method, which consequently pushed the discriminator to approve our samples in all cases.

As an alternative to using the DCGAN discriminator, we implemented an SVM [41] based filter. We generated an initial batch of synthetic textures which were rated as ‘good’ or ‘bad’ by human raters based on visual quality. A simple binary-class SVM (linear kernel, $c = 0.8$) was then trained with deep features ($fc7$ layer of pre-trained VGG-FACE [121]) of 2,059 such rated synthetic images, 1,319 of which were marked ‘bad’ and the rest as ‘good’. The SVM was intentionally biased towards the ‘bad’ side to reduce false negatives. Since generating a new texture is computationally cheap, the SVM can afford to mis-classify some ‘good’ images. This trained SVM is then used as the filtering mechanism in our pipeline. We found the trained SVM to discard about 46% of the overall images we generated, while approving a total of 99,762 textures (frontal face images) for 12,338 synthetic identities from the 15,807 gallery images, as shown in Table 5.1.
TABLE 5.1

DISTRIBUTION OF THE GALLERY AND SYNTHETIC DATASETS
GENERATED USING SREFI [13, 15]

<table>
<thead>
<tr>
<th>Ethnicity</th>
<th>Gallery Images (identities)</th>
<th>Synthetic Textures (identities)</th>
<th>Synthetic 3D Images (identities)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Female: 5,510 (600)</td>
<td>Female: 35,127 (4,527)</td>
<td>Female: 730,540 (4,527)</td>
</tr>
<tr>
<td>Asian</td>
<td>Male: 1,903 (100)</td>
<td>Male: 9,206 (820)</td>
<td>Male: 188,116 (820)</td>
</tr>
<tr>
<td></td>
<td>Female: 1,286 (74)</td>
<td>Female: 8,261 (593)</td>
<td>Female: 169,825 (593)</td>
</tr>
</tbody>
</table>

5.1.2 3D Face Mask Construction

Here we describe our 3D rendering pipeline to produce multi-pose views of the synthetic textures generated (described above in Sec. 5.1.1). Instead of using a generic 3D model [60, 107] or a 3D morphable model (3DMM) [151], we find the ‘best’ fitting 3D model for a synthetic texture from a large set of 3D models to render the texture in 3D. This produces higher quality and more distinct visual results compared to using a generic 3D model and is much faster computationally (two to four seconds to generate a face mask), than 3DMM-based rendering.

3D Models. We use a set of 3D head images acquired using a Konica-Minolta ‘Vivid 910’ 3D scanner [48, 129] as models for face shape. The set consists of over 14,000 different 3D face images (640x480 point clouds) accompanied by their corresponding
registered color images (2D scans). A majority of the scans are of a near-frontal face with neutral expression (mouth closed). Since our texture synthesis pipeline generates frontal face images with neutral countenance, we discard scans with non-frontal pose or an open mouth. We extract the yaw ($\theta$) of each 2D scan using the CNN model from [61] and remove scans with $\|\theta\| > 10$. A two-pass filtering, first using the pre-trained AFFACT network [55] and then manual inspection, is implemented to remove scans with the subject’s mouth open. Consequently, we end up with 8,462 near-frontal, neutral 3D face models, grouped by gender (Male/Female) and race (Asian/Caucasian) using the metadata available.

**Finding the ‘Best’ Fitting 3D Model.** To find the best-fitting 3D model for a synthetic texture $I_s$, we first align each 2D scan (with same race and gender as $I_s$) using Dlib [86]. Then the aligned scans are resized to 512x512, same as $I_s$. We extract a vector $V_l$ for $I_s$ and each 2D scan comprising of its 68 landmark points $p \in R^2$ detected using Dlib [86]. We also extract the 4,096 dimensional feature vector $V_f$ by feeding $I_s$ and each 2D scan to the pre-trained VGG-FACE network [121]. Hence, $V_l$ and $V_f$ together quantify the overall shape and textural appearance of each face image. To find the best fitting 3D models for $I_s$ we minimize the following objective function:

$$
\text{dist}(I_s, B) = w_1 * |V_l(I_s) - V_l(B)| + \\
w_2 * |V_f(I_s) - V_f(B)|,
$$

(5.3)

where $B$ is a 2D scan of the same gender and race as $I_s$ and $\text{dist}(I_s, B)$ captures the dissimilarity in their visual appearance [38]. Since $B$ is of the same gender and race as $I_s$, we set $w_1 = 10$ and $w_2 = 1$ to focus more on facial shape than textural appearance. We assert the 3D model corresponding to the 2D scan ($B$) which minimizes $\text{dist}(I_s, B)$ as the best fitting model for $I_s$. 

95
Figure 5.3. Sample face images generated using SREFI for three synthetic Female-Asian subjects. The 512×512 synthetic face textures (2D) are shown in the leftmost column (a), with the corresponding 800×600 3D renderings presented to its right (b). The artifact in the top row at facial yaw of -90 degrees is due to faulty landmarking.

Rendering in 3D. We render $I_s$ with its three best-fitting 3D models, i.e., which produce the three minimal $dist$ values, using OpenGL [3]. Since the 3D scanner simultaneously acquires 2D and 3D scans, there is a direct correspondence between their landmark points $p_i \in R^2$ and $P_i \in R^3$ respectively. So, the $(x,y)$ coordinates for $p_i$, detected using Dlib [86], are used to retrieve the $(X,Y,Z)$ co-ordinates for $P_i$. For any scanner mis-registration (no valid $P_i$ for a $p_i$), we interpolate $P_i$ using its valid neighboring points. Since $P_i$ belongs to the same 3D plane as its immediate neighbors, we compute its $(X,Y)$ using $\Delta x \rightarrow \Delta X$ and $\Delta y \rightarrow \Delta Y$ correspondences and solve the plane equation with $(X,Y)$ to retrieve its $Z$ coordinate.

We create a mesh of the face mask by triangulating the landmarks $P_i$ of the 3D model (point cloud). This mesh (107 triangles) is further refined by calculating the centroid of each triangle, adding it as a new vertex, and re-triangulating with the new
Figure 5.4. Sample face images generated using SREFI for three synthetic Female-Caucasian subjects.

Figure 5.5. Sample face images generated using SREFI for three synthetic Male-Asian subjects.
points. This re-triangulation step is performed twice to generate a denser mesh of the same facial mask (973 triangles). Using the same \( p_i \rightarrow P_i \) correspondences, we map the synthetic texture \( I_s \) on this mesh and render the synthetic face mask \( M_s \) in 3D. As we do not possess the correspondence between the forehead and background points of a face scan and 3D model, \( M_s \) is generated with a black background. For each 3D model, we render \( M_s \) with \( \theta = [0, \pm 30, \pm 60, \pm 90] \). Since we map the same texture on different 3D models, the overall appearance of the face masks remains the same with variations in its shape. So we assign to \( M_s \) the same label as \( I_s \) and generate 21 (7 poses \( \times \) 3 models) new views of the synthetic identity. A final set of 2,060,992 face masks is obtained for the 12,338 synthetic identities via this rendering scheme (Table 5.1). Sample 2D face images of such synthetic identities, with their 3D renderings, can be seen in Figs. 5.3, 5.4, 5.5 and 5.6 respectively. The complete dataset, along with the 3D head models, can be accessed by clicking on the Notre Dame Synthetic Face Dataset link on this page: https://cvrl.nd.edu/projects/data/.

Figure 5.6. Sample face images generated using SREFI for three synthetic Male-Caucasian subjects.
5.2 Experiments and Results

We perform multiple quantitative benchmark experiments to evaluate the effectiveness of our synthetic face images as training data supplement for CNNs and as distractors in face verification, comparing with results from the face synthesis method in [107, 108] and popular GAN models [132, 24, 81]. We also evaluate the effect of varying facial shape and background on the performance of face recognition experiments.

5.2.1 Experiment 1: Effectiveness in Data Augmentation for CNN Training

In this experiment, we aim to answer the following questions:

1. Can our synthetic face images be used to augment an existing face dataset for CNN training? (similar to [13, 107]).

2. Is a synthetic face image nearly as effective (i.e., does it lead to the same level of accuracy) in training a CNN compared to a real face image?

3. Can a CNN be trained effectively on only a set of synthetic face images?

4. How does our synthetic face images compare to the synthetic images generated by Masi et al.’s method [107, 108]?

To answer these questions, we prepare seven different training datasets using real face images, masked with a black background, from the CASIA-WebFace (CW) dataset [175] and randomly drawn face images from our synthetic 3D dataset (Table 5.1). We mask the context (forehead, hair, neck, etc.) and background pixels in the face images from CW to maintain consistency with the synthetic images generated by our method. The distribution in the datasets can be seen in Table 5.2. We fine-tune the ResNet-50 network [63], pre-trained on the VGGFace2 dataset [31], with these seven datasets in seven separate training sessions using Caffe [76]. For each dataset

1Specifically, the ‘ResNet-50-256D’ model, available here: https://github.com/ox-vgg/vgg-face2
### TABLE 5.2

**EFFECTIVENESS OF SREFI AS A DATA AUGMENTATION MODULE TESTED USING CASIA-WEBFACE [175] DATASET AND RESNET-50**

<table>
<thead>
<tr>
<th>Training Data</th>
<th>CW [175] Images (real identities)</th>
<th>Synthetic SREFI images (synthetic identities)</th>
<th>IJB-B [161] Performance (TAR@FAR = 0.01)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset 1</td>
<td>494,414 (10,575)</td>
<td>0</td>
<td>0.942</td>
</tr>
<tr>
<td>Dataset 2</td>
<td>329,609 (7,050)</td>
<td>164,807 (3,525)</td>
<td>0.941</td>
</tr>
<tr>
<td>Dataset 3</td>
<td>164,807 (3,525)</td>
<td>329,625 (7,050)</td>
<td>0.945</td>
</tr>
<tr>
<td>Dataset 4</td>
<td>329,609 (7,050)</td>
<td>329,634 (3,525)</td>
<td>0.946</td>
</tr>
<tr>
<td>Dataset 5</td>
<td>329,609 (7,050)</td>
<td>329,625 (7,050)</td>
<td>0.949</td>
</tr>
<tr>
<td><strong>Dataset 6</strong></td>
<td><strong>494,414 (10,575)</strong></td>
<td><strong>494,414 (10,575)</strong></td>
<td><strong>0.953</strong></td>
</tr>
<tr>
<td>Dataset 7</td>
<td>0</td>
<td>494,414 (10,575)</td>
<td>0.864</td>
</tr>
</tbody>
</table>
in Table 5.2, 90% of the data is used for training and the rest for validation, with each image resized to $224 \times 224$ prior to training. We use a polynomial decay policy \cite{28} for training each network with the same batch size $= 16$, base learning rate $= 0.001$, gamma $= 0.96$, momentum $= 0.009$ and a step size of 32,000 training iterations. We stop training a network when its validation loss plateaus across a training epoch.

For testing each trained network, we use the IJB-B verification protocol \cite{161} as our performance metric. Each still image or video frame from a template is first aligned about its eye center, with the face region masked out using its landmarks \cite{86}. We feed the masked images to each of the seven trained networks and extract its 256-dimensional $\text{ feat_extract}$ layer descriptor. We generate an average feature vector for a template using $\text{video}$ and $\text{media pooling}$ operations, described in \cite{108}, and match a pair of templates using a Pearson correlation co-efficient metric ($\rho$) between their feature vectors as:

$$\rho = \frac{\text{Cov}(F_i, F_j)}{\sigma_{F_i} \sigma_{F_j}}$$  \hspace{1cm} (5.4)

where $\text{Cov}$ denotes covariance, $F_i$ and $F_j$ are the pooled feature vectors of the $i$-th and $j$-th templates respectively. In an ideal situation pooled features of two templates of the same identity should match perfectly, i.e., $\rho$ should be close to 1.

The verification performance of each network can be found in Table 5.2. Results show that supplementing an existing dataset with our synthetic face images invariably improves CNN performance (Dataset 4, 5 and 6). However, a synthetic image does not appear to have the same value as a real image (Dataset 7). This can be attributed to the uniformity in lighting and expression in our synthetic face images while face images of both CW \cite{175} and IJB-B \cite{161} have plenty of variation in these areas. Also, the network trained with Dataset 5 (14,100 identities) slightly outperforms the network trained with Dataset 4 (10,575 identities) although they have the same number of images. This suggests wider training datasets (i.e. more identities)
Figure 5.7. Sample results generated by (a) SREFI (synthetic identity), and (b) Masi et al.’s method [108] (real identity). Each row of a sub-figure shows the same facial texture (identity) rendered using a 3D face model at different facial poses. Notice how different the same facial texture looks when rendered with different 3D models, i.e., variable shape. An analysis of how this variability in facial shape affects recognition performance can be found in the next chapter.

are indeed beneficial to CNN performance [17].

Comparison with Masi et al.’s method [108].

To compare the effectiveness of our method as a generator of supplemental training data with that of Masi et al.’s method, we perform another set of experiments. Their method warps a given face image using a static set of 10 3D face models from the Basel 3D model dataset [123] at pre-determined yaw values [0, -22, -40, -55, -75]. Therefore, for a given face image of a real identity, it can generate 50 (10×5) new views varying in facial shape and pose. However, it cannot generate new synthetic identities and therefore can only augment the depth (images per subject) of the dataset, but not the width (total number of subjects), as demonstrated in [107]. To generate the supplemental data using Masi et al.’s method, we use the real face images from our gallery set (see Table 5.1). It contains a total of 15,807 face images (512×512 in size) of 1,452 real subjects. Using

2Since GAN models like ProGAN [51] do not generate synthetic face images with identity labels, we could not compare them with SREFI for this experiment.
TABLE 5.3


<table>
<thead>
<tr>
<th>Training Data</th>
<th>CW [175] Images (real identities)</th>
<th>Synthetic Masi images (synthetic identities)</th>
<th>IJB-B [161] Performance (TAR@FAR = 0.01)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset 1</td>
<td>111,276 (1,452)</td>
<td>0</td>
<td>0.925</td>
</tr>
<tr>
<td>Dataset 2</td>
<td>74,184 (968)</td>
<td>37,092 (484)</td>
<td>0.929</td>
</tr>
<tr>
<td>Dataset 3</td>
<td>37,092 (484)</td>
<td>74,184 (968)</td>
<td>0.926</td>
</tr>
<tr>
<td>Dataset 4</td>
<td>74,184 (968)</td>
<td>74,184 (484)</td>
<td>0.932</td>
</tr>
<tr>
<td>Dataset 5</td>
<td>74,184 (968)</td>
<td>74,184 (968)</td>
<td>0.935</td>
</tr>
<tr>
<td>Dataset 6</td>
<td>111,276 (1,452)</td>
<td>111,276 (1,452)</td>
<td>0.939</td>
</tr>
<tr>
<td>Dataset 7</td>
<td>0</td>
<td>111,276 (1,452)</td>
<td>0.857</td>
</tr>
</tbody>
</table>

Masi et al.'s method we generate a dataset, referred to as Masi dataset henceforth, of 790,350 synthetic views of the same 1,452 subjects varying facial shape and pose. To maintain consistency with our synthetic images, we generate these synthetic views without any background (i.e., masked faces), as shown in Fig. 5.7b.

Since Masi et al.'s method cannot generate more subjects, we re-sample our original training data from the CASIA-WebFace [175] dataset to keep the number of subjects the same with the Masi dataset. Specifically, we randomly select a total of 111,276 face images of 1,452 subjects from the head, i.e., half of the dataset containing the subjects with the most number of images, of CW's distribution. For the supplemental data generated by Masi et al.'s method, we select 111,276 images from the 1,452 real subjects from the Masi dataset. Similarly, we sample 111,276 images
from 1,452 synthetic subjects generated by our method. We do not check for consistency in the pose distribution in each of these datasets. As before, we sub-divide the original CW data, and the two sets of synthetic data separately into seven different datasets, shown in Tables 5.3 and 5.4 respectively. We fine-tune the ResNet-50 [63] model with each dataset, while keeping the hyper-parameters the same and test the trained models on the IJB-B [161] verification protocol as before.

Tables 5.3 and 5.4 show that both of the face synthesis methods are effective as data augmentation modules. We do not observe any significant difference between the two methods performance-wise. Furthermore, if the number of images and subjects is kept constant while the network is training, the test performance remains approximately the same. This suggests the visual quality of the supplemental data
in face recognition experiments is not as important as the size of the overall training dataset; this is consistent with recent findings in other domains [147]. Although the two synthetic image sets perform similarly when added as supplemental data to existing datasets (Datasets 2 - 6 in Tables [5.3] and [5.4]), Masi et al.’s method can capture more variation in facial expression and therefore outperforms SREFI when used by itself for training (Dataset 7 in Tables [5.3] and [5.4]). However, SREFI can generate additional synthetic identities to augment the training data width, unlike Masi’s method, and get a higher performance by training the same network with more synthetic data (Dataset 6 in Tables [5.2] and [5.3]).

5.2.2 Experiment 2: Effectiveness as Generator of Synthetic Distractors For Model Testing

In this experiment, we seek to answer these questions:

1. Can our synthetic face images be used as distractors to influence the recognition accuracy of a trained CNN? (Similar to [85, 117].)

2. How does the size of this distractor set affect CNN performance?

3. Is a synthetic distractor set as effective as a real distractor set (containing real face images, like MegaFace [85]) of the same size?

4. Are the synthetic face images generated by our method as good or even better than those generated using the synthesis method from [108] and the ProGAN model [81]?

For our experiments, the FG-Net dataset [120], containing 982 images of 82 identities, is used as the probe set. We introduce two kinds of distractor galleries: $D_S$ containing synthetic face images from Table 5.1 and $D_M$ with randomly drawn real face images from the MegaFace dataset [85]. For a given probe identity with $M$ photos, we add each photo to the gallery and use each of the other $(M - 1)$ photos as probes (similar to the MegaFace protocol [85]). We repeat this process for all the $N$ identities in the probe set.
Figure 5.8. Results on the FG-Net dataset [120]. (a) Identification accuracy with variable real and synthetic distractor gallery size, (b) verification performance with 10,000 real and synthetic distractors, using pre-trained VGG-FACE [146, 121], ResFace-101 [63, 107] and ResNet-50 [63, 31] models.

We use the Caffe [76] implementation of three different network models as independent feature extractors for this experiment: (a) the 4096-D $fc7$ layer descriptor from the VGG-16 model [146], pre-trained on the VGG-FACE [121] dataset, (b) the 256-D $feat\_extract$ layer descriptor from the ResNet-50 model, pre-trained on the VGGFace2 dataset [31], and (c) the 2048-D $pool5$ layer descriptor from ResFace-101 [107], a version of ResNet-101 [63] trained on CASIA-WebFace [175] following the augmentation described in [107]. It is to be noted that we do not repose the test images, as done in [107], when using ResFace-101. The metric for comparison is $L_2$ distance, similar to [85]. Since the distractor gallery exclusively generates a huge set of non-match scores, a low value of such scores hampers the performance of the network. We perform two different experiments: (1) a rank-based identification experiment (CMC) with $D_S$ and $D_M$ containing $[10^1, 10^2, 10^3, 10^4, 10^5]$ distractors, (2)

\footnote{ResFace-101 is available from: \url{https://talhassner.github.io/home/publication/2016_ECCV_1}}
a verification experiment (ROC) with $D_S$ and $D_M$ containing $10^4$ distractors. The results are shown in Figs. 5.8a and 5.8b respectively. A lower curve suggests more false positives, i.e., more effective distractors.

As depicted in the figure, our synthetic face images can be effectively used as distractors in face recognition experiments. But they are not individually as effective as the MegaFace images in their role as distractors, as evident from the gap between the solid and dashed curves. This can be attributed to the uniform black background and neutral expression of our face images compared to the wide background variation in the MegaFace images. However, since our face synthesis process demands fewer resources compared to downloading data (face images) from the web or acquiring new face images in a collection effort, we can always increase the synthetic distractor set size by adding in more 3D models or gallery base faces. The CMC curves (Fig. 5.8a) also suggest that the VGG-FACE network (blue curve) is less robust to distractors than ResNet-50 and ResFace-101.

Comparison with Masi et al.’s method and the ProGAN model.

To compare the effectiveness of SREFI with Masi’s method, we randomly sample 10,000 synthetic views of real identities from the Masi Dataset, as described in Section 5.2.1. We use the same gallery dataset to train the Tensorflow implementation of the ProGAN model. We use the same hyper-parameters as used by the authors and train the model for 32,000 iterations, from $4 \times 4$ to $128 \times 128$ resolutions. Once training finishes, the snapshot of the model is used to generate 10,000 synthetic face images. The face images generated by ProGAN do not contain identity labels, but that does not matter as the distractors are solely used to generate non-match scores.

After preparing the Masi and ProGAN based synthetic distractor sets, we perform

---

Available from: https://github.com/tkarras/progressive_growing_of_gans
Figure 5.9. Verification performance on the FG-Net dataset \cite{120} with 10,000 synthetic distractors generated using (a) SREFI and Masi’s method, and (b) SREFI and the pre-trained ProGAN model, using pre-trained VGG-FACE \cite{146, 121}, ResFace-101 \cite{63, 107} and ResNet-50 \cite{63, 31} models.

the same verification experiment as before using features from the same three pre-trained models, and compare the results with 10,000 distractor set generated using SREFI. The ROC curves are shown in Figures 5.9a and 5.9b respectively. In both cases, the SREFI distractors far outperform the other distractor set by generating a lower curve for all the network models. This suggests that the SREFI distractors generate deep features that more closer to the real FG-Net \cite{120} images, compared to the synthetic images generated by Masi et al.’s method or ProGAN \cite{81}, and therefore they generate lower curves. Interestingly, the ResNet-50 model \cite{63, 31} generates the highest ROC curve (green solid line in Fig. 5.9b) among all when images generated by the ProGAN model \cite{81} are used as distractors. This suggests the pre-trained ResNet-50 model is more efficient in detecting GAN generated synthetic faces.
5.2.3 Experiment 3: Comparison with Popular GANs

To compare with SREFI, we choose a Tensorflow implementation of three popular GAN models for face image synthesis: DCGAN [132], BEGAN [24] and the recently released ProGAN [81]. To level the playing field resolution wise, we resize the 15,807 real face images from Table 5.1 to 128×128 and train each GAN model with those images for 200 epochs using a single NVIDIA Titan XP GPU. For the ProGAN [81] training session, we used a lower batch size (16) compared to DCGAN [132] and BEGAN [24] (64) to accommodate for its much higher memory requirements. Each trained model is then used to generate synthetic face images, 128×128 in size. We also generate 128×128 synthetic face images, using our pipeline, from the same 15,807 face images (used as gallery). We compare the training and synthesis time required by all four methods. To assess the realism of the synthetic images generated by each method, we use the Fréchet Inception Distance (FID) metric proposed in [65]. FID uses activations from the Inception-v3 [149] network to compare the statistics of the images.

![Image of comparisons between DCGAN, BEGAN, PGGAN, and the proposed method for face image synthesis.]

Figure 5.10. Comparison with GAN models based on visual quality.

---

5Available from: [https://github.com/carpedm20/DCGAN-tensorflow](https://github.com/carpedm20/DCGAN-tensorflow)

6Available from: [https://github.com/carpedm20/BEGAN-tensorflow](https://github.com/carpedm20/BEGAN-tensorflow)
generated dataset to the real one. A lower value of FID suggests generated samples
to be more realistic, and signifies a better model. The results can be seen in Table
5.5, along with a visual comparison of the corresponding synthesis results (Figure
5.10).

As expected, the GAN models take a considerable amount of training time, es-
pecially the ProGAN framework [81] due to its multi-phase training regime. SREFI,
in comparison, only uses pre-trained network models, and therefore requires no GPU
resources. The only training component of our method is to train the linear SVM for
quality estimation of the synthetic 2D textures, which requires only a few seconds.
The most expensive steps of our pipeline are the blending and the 3D triangula-
tion, which can vary depending on the target image resolution. The low FID score
generated by our 2D synthetic images also suggests them to be of a higher visual
quality compared to the GANs. However, a disadvantage of our current method is
that it cannot render multi-pose face images with forehead and background in 3D
due to lack of 2D → 3D correspondence between the 2D synthetic textures (with
background) and 3D models. As a result, vital information pertaining to the context
and background of the face is lost, which can enhance its realism, as evident from
the high FID score generated by our 3D face masks. The presence of context and
background in face images has also been proven to benefit network training [17]. To
hallucinate the context and background of the SREFI generated synthetic faces, we
use the trained cascaded network of GANs from [12], as detailed in Section 5.2.5.

5.2.4 Experiment 4: Impact of Facial Shape on Subject Identity

In this experiment, we seek to answer these questions:

1. How much of an influence does the facial shape have on a subject’s identity?
   More specifically, if different 3D models are used to render the same facial
texture, how much does the appearance change?

2. Is the appearance change, if any, consistent at different facial poses?
### Table 5.5

**Comparison with GAN Models Based on Realism, Training, and Synthesis Time**

<table>
<thead>
<tr>
<th>Synthesis Method</th>
<th>GPU</th>
<th>Training Time (hours)</th>
<th>Generation Time (seconds)</th>
<th>FID</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCGAN [132]</td>
<td>Yes</td>
<td>9.8</td>
<td>0.58</td>
<td>78.11</td>
</tr>
<tr>
<td>BEGAN [24]</td>
<td>Yes</td>
<td>11.6</td>
<td>0.47</td>
<td>41.07</td>
</tr>
<tr>
<td>ProGAN [81]</td>
<td>Yes</td>
<td>60.9</td>
<td>0.35</td>
<td>38.43</td>
</tr>
<tr>
<td>SREFI (2D texture)</td>
<td>No</td>
<td>0.004</td>
<td>1.36</td>
<td>33.35</td>
</tr>
<tr>
<td>SREFI (3D mask)</td>
<td>No</td>
<td>0</td>
<td>2.85</td>
<td>181.12</td>
</tr>
</tbody>
</table>
3. Is the appearance change, if any, noticeable with different rendering methods like SREFI and Masi et al.’s method [108]?

A popular technique for augmenting training datasets is to render face images with different 3D models at different facial yaw values. It is argued that although the facial shape changes, the facial texture remains intact, and therefore the synthetic images are assigned the same identity label (an example can be seen in Fig. 5.7). These synthetic views infuse more variation facial shape and pose, adding to the depth of the training dataset, and makes the network more robust to fluctuations in test images, as shown in [107, 108, 13, 15] and Sec. 5.2.1 of this paper. However, the amount of variation in appearance this introduces is not analyzed, nor is how different the face looks overall even with the facial texture intact. To answer these questions, we randomly sample 1,000 synthetic identities from the Notre Dame Synthetic Face Dataset, with each facial texture (identity) rendered with three different 3D models, at yaw values [0, -30, -60, -90]. Similarly, we sample 1,000 real identities from the Masi Dataset, with each facial texture (identity) rendered with three randomly chosen 3D models (of the 10 available), at yaw values [0, -22, -55, -75]. The yaw values are picked for both the sets to be as consistent as possible to each other.

For a particular facial yaw value in each dataset, we perform a verification experiment with face images, i.e., facial textures rendered with: (1) the same 3D model (consistent shape) and (2) different 3D models (variable shape). We use the 256-dimensional feat_extract layer descriptor of the ResNet-50 model pre-trained on the VGGFace2 dataset [31] as image features, and the Pearson correlation coefficient to compute match score between features. Since each facial texture, real (Masi et al.’s method) or synthetic (SREFI), is rendered with three different 3D models, we perform the experiment separately for each 3D model. The verification scores, resulting from this experiment, can be seen in Figures 5.11a and 5.11b.

As can be seen, the verification performance drops rapidly as the facial pose moves...
Figure 5.11. Verification performance with varying facial shape at different facial yaw values with synthetic images generated using: (a) SREFI [15], and (b) Masi’s method [108].

away from 0. This effect is much more prominent in the SREFI generated synthetic images, compared to the ones generated by Masi et al.’s method. The reason behind this is the normalization along the jaw-line that is present in Masi et al.’s method, which makes the variation in shape not so noticeable. SREFI, on the other hand, keeps the 3D model intact and therefore the variation in facial shape becomes more noticeable at higher yaws (as apparent from Fig. 5.7). It is interesting to note that for SREFI, the three 3D models chosen for rendering are the most compatible with the facial texture (using Eq. 5.3), and therefore are similar to each other to a certain degree. Even in that case, the variation in appearance between two different renderings is quite large as evident from Fig. 5.7a and the plots in Fig. 5.11a. Such a variation in facial shape, when used for training a network, is definitely beneficial to its feature learning as exhibited by [107, 108, 13, 15] and Tables 5.2, 5.3, and 5.4 of this paper. However, the trained networks may also be more vulnerable to presentation attacks, i.e., distractors, with different people wearing the same facial texture mask to generate false matches [25].
5.2.5 Experiment 5: Impact of Context and Background on Face Recognition Performance

In this experiment, we seek to answer these questions:

1. Does the presence of context (forehead, hair, neck, clothes) and background in the training data for CNNs generate a higher test performance, when compared with masked face images?

2. In a similar vein, does the presence of context and background in the supplemental training data, generated by SREFI or Masi et al.’s method [108], generate a higher test score?

For this experiment, we take the same hybrid datasets we described in Experiment 1, combining 111,276 real images from CW and 111,276 synthetic images from either Masi et al.’s method or SREFI. We label them as Dataset 1 and 2 respectively in Table 5.6. Since Masi’s method has the functionality to preserve the context and background while rendering a face image, we prepare another hybrid dataset (Dataset
3 in Table 5.6) where the context and background of the same synthetic images are preserved. The SREFI method however does not render context and background pixels at non-frontal poses due to missing 2D → 3D correspondences for context and background pixels. To hallucinate the context and background pixels for the SREFI generated images, we use the trained snapshot of the multi-scale GAN model from [12]. This model is composed of a cascaded network of GAN blocks, each tasked with hallucination of missing pixels at a particular resolution while guiding the synthesis process of the next GAN block. It can generate the missing pixels automatically, by taking cues from the features of the provided face mask, without requiring any human supervision. We generate context and background pixels for the 111,276 face images of the 1,452 synthetic identities, and prepare a hybrid dataset with images from CW (Dataset 4 in Table 5.6). Sample synthetic face images with context and background can be found in Fig. 5.12.

As described in the last chapter, we fine-tune the ResNet-50 [63] model separately in four different training sessions, using the four datasets from Table 5.6. We keep the hyper-parameters fixed at the values used in Experiment 1. Once training finishes, we use the 256-dimensional feature vector from the feat_extract layer of the trained model snapshots to perform the verification experiment using the IJB-B templates [161]. As before, we perform video and media pooling operations [108] before computing the match score between two templates using the Pearson correlation co-efficient. The resulting ROC curves are presented in Fig 5.13. As the curves clearly suggest, training the network with supplemental face images containing context and background pixels makes it more robust to visual changes during testing. This can be attributed to the additional information and variation the presence of context and background provides to the network during training, which in turn results in better test performance of the trained model. Moreover, these results are along the same lines of previous research that underpin the importance of context in face recognition by humans [42, 49] and
TABLE 5.6

DISTRIBUTION OF THE TRAINING DATASETS CREATED FOR EXPERIMENT 5

<table>
<thead>
<tr>
<th>Training Data</th>
<th>CW Images</th>
<th>Masi Images (wo/ context)</th>
<th>Masi Images (w/ context)</th>
<th>SREFI Images (wo/ context)</th>
<th>SREFI Images (w/ context)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset 1</td>
<td>111,276 (1,452 real identities)</td>
<td>111,276 (1,452 real identities)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Dataset 2</td>
<td>111,276 (1,452 real identities)</td>
<td>0</td>
<td>0</td>
<td>111,276 (1,452 synth. identities)</td>
<td>0</td>
</tr>
<tr>
<td>Dataset 3</td>
<td>111,276 (1,452 real identities)</td>
<td>0</td>
<td>111,276 (1,452 real identities)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Dataset 4</td>
<td>111,276 (1,452 real identities)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>111,276 (1,452 synth. identities)</td>
</tr>
</tbody>
</table>
5.3 Discussion

Our method needs minimal training and requires little data and time compared to popular GAN models [132, 24, 81] to generate natural looking high quality face images with different shape, pose and resolution. We formulate a set of quantitative benchmark experiments that can be potentially used to assess quality metrics of synthetic face images generated using different algorithms. These experiments showcase deep networks [126, 17].

Figure 5.13. Verification performance of the ResNet-50 model [63, 51] on the IJB-B [161] dataset, when fine-tuned with training data with and without the presence of context and background pixels. As the curves suggest, presence of context and background benefits network training.
the application of the SREFI method as: (1) a data augmentation module for CNN training, and (2) a generator for synthetic distractors. The SREFI generated synthetic images also prove to be more effective in these benchmarks compared to Masi et al.’s face synthesis method [107, 108] and the ProGAN model [81].

Additionally, we perform experiments to quantify: (1) the effect of facial shape on subject identity, and (2) the effect of the presence of facial context and background in the training data on network performance. We generate a dataset containing over 2 million face images of over 12,000 synthetic identities which can be used as supplemental data while training a CNN, or as a distractor set when performing verification experiments with CNNs. The dataset containing 2 million synthetic face images and 8,000 3D head models has been released, and can be downloaded by accessing the Notre Dame Synthetic Face Dataset link from this webpage: https://cvrl.nd.edu/projects/data/
CHAPTER 6

HALUCINATING CONTEXT AND BACKGROUND PIXELS FROM A FACE MASK USING MULTI-SCALE GANS

This chapter is based on the work described in [12], a manuscript currently under review, as of this writing.

When supplied with a face mask (i.e., limited data) the goal of our model is to automatically hallucinate realistic context and background pixels. While doing so the gender, ethnicity, pose, expression of the input subject should be preserved. While face swapping [90, 118, 115] and face editing [21, 64] algorithms have dealt with transferring the face and facial attributes from one identity to another, they require - (1) the full face image to work, and (2) similarity in visual appearance, and pose for identity preservation. Unlike previous work, we treat this problem along the same lines as image colorization [180, 92] and directly hallucinate the missing pixels taking cues from the input data without any involvement from the user.

6.1 Our Method

Since there can be many plausible hallucinations from a single face mask, we control this unconstrained problem using the training data. When provided with a face mask $I^M$ during training, our model tunes its weights $w$ such that its generated output $G(I^M)$ looks similar to the original face image $I^{GT}$. The weights are parameterized by $I^{GT}$ itself and after a few training epochs, the model learns to generate $G(I^M)$ closely identical to $I^{GT}$. During testing, this trained model requires only a
Figure 6.1. Our multi-scale cascaded network pipeline. Starting from the lowest resolution block (8×8), we proceed higher up through a set of GAN blocks in a single pass (left to right in the figure). Except the last block, the output of each block is upscaled 2x and fed as input to the next block. To preserve fine facial details at each resolution, we add the mask image at each resolution before feeding the input. The final 128×128 output, with hallucinated context and background pixels, is generated by block_128. More details about the architecture of block_128 is provided in Figure 6.2.

face mask ($I^M$), and not the full face image ($I^{GT}$), to hallucinate realistic context and background pixels from the learned representations.

6.1.1 Network Architecture

**Cascaded Network.** Inspired by [43, 155, 90], we implement a multi-scale architecture comprising of five GAN blocks to learn hallucination at multiple resolutions (8×8 to 128×128), as depicted in Figure 6.1. Each block contains an encoder-decoder pair working as the generator. The encoder at the highest resolution block ‘block_128’, as shown in Figure 6.2, takes the input and downsamples it through a set of strided convolution layers (stride = 2), except the first layer where we incorporate extra spa-
tial information using an atrous convolution layer \cite{179} with dilation rate of 2. Each of the next strided convolution layers is followed by a residual block \cite{63} to facilitate the learning process. The output of the encoder is fed to the decoder which is composed of five convolution and pixel shuffling blocks \cite{145} for upscaling the feature by two in each dimension.

We add skip connections \cite{135, 63, 71} between encoder and decoder layers with the same tensor shape to propagate finer details from the input. The final 3 channel output is obtained by passing the upsampled result through a convolution layer with \textit{tanh} activation \cite{132, 138}. Since the input and output of ‘block\_\text{\(N/2\)}’ is half in height and width compared to ‘block\_\text{\(N\)’}, each GAN block contains one fewer residual and pixel shuffling layers than its next GAN block. Except ‘block\_\text{\(128\)’}, the output of each block is upscaled 2x through a pixel shuffling layer and fed as input to the next block. Thus, instead of a face mask, the block receives a rough hallucination to guide it towards the right direction. For all blocks, we also replace pixels in the face mask region of \(G(I^M)\) with original pixels from \(I^M\), before loss computation, to keep finer details of the face intact and focus only on the task of context and background generation.

During training, we provide each block with a discriminator to guide the generated samples towards the distribution of the training data. We use the popular \textit{CASIA-Net} architecture from \cite{175} as the discriminator, after removing all max pooling and fully connected layers and adding batch normalization \cite{73} to all convolution layers except the first one. A leaky \textit{ReLU} \cite{109} activation (slope = 0.2) is used for all layers except the last one where the \textit{sigmoid} activation is adopted to extract a probability between 0 (fake) and 1 (real), as suggested by \cite{132}. Each layer is initialized using He’s initializer \cite{62, 81}. During testing, only the trained generator and pixel shuffling blocks are used to hallucinate the synthetic output, with resolution of 128×128.

**Progressively Growing Network (ProGAN).** Addressing the recently pro-
Figure 6.2. The encoder is composed of five residual blocks while the decoder upsamples the encoded feature using five pixel shuffling blocks. The solid curved arrows between layers represent skip connections. During training the generator learns to hallucinate the original full face image $I^{GT}$ from the face mask $I^{M}$ via reconstruction, identity preserving, perceptual and adversarial losses. We replace pixels in the face mask of $G(I^{M})$ with original pixels from $I^{M}$ to preserve fine details.
posed progressive growing of GANs to generate high quality samples [81, 33, 82], we also develop a ProGAN version of our model for comparison. Instead of the cascaded architecture where all the GAN blocks are trained in each iteration, we train the lowest resolution block $8 \times 8$ first with $8 \times 8$ face masks. After a few training epochs, we stop and load additional layers from block $16$ and start training again with $16 \times 16$ face masks. This process of progressively growing the network by stopping and resuming training is continued till we have a trained block $128$ model, as depicted in Figure 6.3. During testing, the trained block $128$ is used to hallucinate context and background pixels directly from previously unseen $128 \times 128$ face masks. To maintain consistency, the loss function, hyper parameters and training data are kept the same with our cascaded network.

6.1.2 Loss Function

For each block of our network we learn context and background hallucinations independently. So we assign a combination of different losses, described below, to make the synthesized output at each resolution both realistic and identity preserving. We represent the image height, width and training batch size as $H$, $W$ and $N$ respectively.

1. **Pixel loss ($L_{pixel}$):** To enforce consistency between the pixels in the ground truth $I^{GT}$ and hallucinated face images $G(I^M)$, we adopt a mean $l_1$ loss computed as:

$$L_{pixel} = \frac{1}{N \times H \times W} \sum_{n=1}^{N} \sum_{i=1}^{H} \sum_{j=1}^{W} |(I_n^{GT})_{ij} - (G(I_n^M))_{ij}|$$

(6.1)

where $H$ and $W$ increase as we move to higher blocks in our network, $8 \times 8 \rightarrow 16 \times 16$, $16 \times 16 \rightarrow 32 \times 32$, and so on. We use $l_1$ loss as it preserves high frequency signals better than $l_2$ in the normalized image thus generating sharper results.

2. **Perceptual loss ($L_{pc}$):** To make our hallucinations perceptually similar to
Figure 6.3. Pipeline of our progressively growing (ProGAN) network. We train the lowest resolution block for 50 epochs, then introduce additional layers for the next resolution block and resume training. This network growing continues till block_128. During testing, we only use the trained block_128.

Real face images, we add the **LPIPS** metric (ver. 0.0) from [181] to our loss function. This metric finds a dissimilarity score between a pair of images, derived from deep features with varying levels of supervision, and is shown to be more consistent with human perception than classic similarity metrics like PSNR and SSIM [159]. We use LPIPS as a regularizer to support $L_{\text{pixel}}$. It is computed as:

$$L_{pc} = \frac{1}{N} \sum_{n=1}^{N} LPIPS(G(I_n^M), I_n^{GT}) \quad (6.2)$$

where $LPIPS$ is the dissimilarity score generated by the AlexNet [91] model\(^1\) (in PyTorch [122]) provided by the authors. An $L_{pc}$ value of 0 suggests perfect similarity between $G(I^M)$ and $I^{GT}$. Since the code does not support low-res images, $L_{pc}$ is not applied on ‘block_8’ and ‘block_16’.

\(^1\)Available here: [https://github.com/richzhang/PerceptualSimilarity](https://github.com/richzhang/PerceptualSimilarity)
3. **Adversarial loss** ($L_{adv}$): To push our hallucinations towards the manifold of real face images, we introduce an adversarial loss. This is achieved by training a discriminator along with the generator (encoder-decoder) at each block of our network. We use a mean square error based LSGAN [105] for this work as it has been shown to be more stable than binary cross entropy [53]. The loss is calculated as:

$$L_{adv} = \frac{1}{N} \sum_{n=1}^{N} (D(G(I^M_n)) - c)^2 \quad (6.3)$$

where $D$ is the discriminator and $c$ is set to 1 as we want to fool $D$ into labeling the synthetic images as real.

4. **Identity loss** ($L_{id}$): To preserve essential features of the identity in the input face mask in the generated output, we use the pre-trained VGG-FACE [121] model to provide a supporting metric. We calculate the $l_2$ distance between the $fc7$ layer features between $I^{GT}$ and $G(I^M)$ and apply that as content loss similar to neural style transfer [50]. The closer this metric moves towards 0, the better the hallucination quality. The loss is calculated as:

$$L_{id} = \frac{1}{N \times \#F} \sum_{n=1}^{N} \sum_{i=1}^{\#F} (F(G(I^M_n))_i - F(I^{GT}_n)_i)^2 \quad (6.4)$$

where $F$ is the 4096-D feature vector from VGG-FACE [121].

5. **Total variation loss** ($L_{tv}$): Similar to [80, 71, 90], we add a total variation loss as a regularizer to suppress spike artifacts, calculated as:

$$L_{tv} = \sum_{i=1}^{H} \sum_{j=1}^{W} (G(I^M)_{i,j+1} - G(I^M)_{i,j})^2 + (G(I^M)_{i+1,j} - G(I^M)_{i,j})^2 \quad (6.5)$$

The final loss $L$ is computed as the weighted sum of the different losses:

$$L = L_{pixel} + \lambda_1 L_{pc} + \lambda_2 L_{adv} + \lambda_3 L_{id} + \lambda_4 L_{tv} \quad (6.6)$$
6.2 Experiments

**Training Data.** For training our model, we randomly sample 12,622 face images (7,761 male and 4,861 female) from the public dataset in [129]. These images were acquired specifically for recognition tasks, with variety of facial pose and neutral background. Image mirroring is then applied for data augmentation. To acquire the face masks, we first detect the face region using Dlib [86] and estimate its 68 facial keypoints with the pre-trained model from [29]. We remove images for which Dlib fails to detect a face. The eye centers are then used to align the faces and pixels outside the convex hull of the facial landmark points in the aligned image are masked. Both the aligned and masked versions are then resized using bilinear interpolation to 8×8×3, 16×16×3, 32×32×3, 64×64×3 and 128×128×3, with pixels normalized between [0,1], for training different network blocks.

**Training Details.** We train our model with the Adam optimizer [87] with generator and discriminator learning rates set as 10^{-4} and 2 × 10^{-4} respectively. For each block, we train its discriminator with separate real and synthesized mini-batches with label smoothing applied to the real mini-batch, as suggested by [132, 138]. Other hyper-parameters are set empirically as \( \lambda_1 = 1 \), \( \lambda_2 = 0.1 \), \( \lambda_3 = 10 \), \( \lambda_4 = 10^{-6} \). We train our model on the NVIDIA Titan Xp GPU, using Tensorflow [5] and Keras [35], with a batch size of 10, for a hard limit of 50 epochs, as we find validation loss to plateau around this stage. We use the trained generator and pixel shuffling blocks from this model for our experiments.

**Metrics for Quality Estimation.** To evaluate the effectiveness of our model in the task of context and background hallucination, and compare with other works, we use the following metrics:

1. **Mean Match Score:** We use the 256-dimensional penultimate layer descriptor from the ‘ResNet-50-256D’ model [63] (‘ResNet-50’ here on), pre-trained on
Figure 6.4. Sample results from LFW [70] (128×128 in size), generated using GenFace [97], DeepFillv1 [173], SymmFCNet [96], EdgeConnect [116], and our cascaded and ProGAN [81] models. Note the variation in gender, pose, age, expression and lighting in the input images.
VGGFace2\(^2\) as feature representation for an image for all our face recognition experiments. The deep features are extracted for each original image and the hallucinated output in the dataset. The mean match score \(\rho\) is calculated by averaging the Pearson correlation coefficient between each feature pair as:

\[
\rho = \frac{1}{N} \sum_{i=1}^{N} \frac{\text{Cov}((F_o)_i, (F_h)_i)}{\sigma_{(F_o)_i}\sigma_{(F_h)_i}}
\]  

(6.7)

where \(\text{Cov}\) denotes covariance, \(N\) is the number of images in the dataset, and \((F_o)_i\) and \((F_h)_i\) are the feature vectors of the i-th original and hallucinated images respectively. Ideally, we would like the hallucinated images to match well, but not perfectly, with the original images i.e., \(\rho\) should be a little less than 1. Such a value would suggest that our model retains vital facial features of the input identity while adding variations in its visual attributes. The more the source face is modified, the more the gap widens, as specified in [118].

(2) **Mean SSIM**: To evaluate the degree of degradation, or noise, in the hallucinated output, we compute the SSIM [159] value for each (original, synthetic) image pair in the dataset. A higher mean SSIM value suggests less noisy hallucinations and therefore a better model.

(3) **FID**: To evaluate the realism of the generated samples, we use the Frechet Inception Distance (FID) metric proposed in [65]. FID uses activations from the Inception-v3 [149] network to compare the statistics of the generated dataset to the real one. A lower value of FID suggests generated samples to be more realistic, and signifies a better model.

(4) **Mean Perceptual Error**: To evaluate the perceptual dissimilarity between the original and the hallucinated images, we use the PieAPP v0.1 metric using the pre-trained model from [131]. The metric calculates the level of distortion between a

---

\(^2\)Available here: [https://github.com/ox-vgg/vgg_face2](https://github.com/ox-vgg/vgg_face2)
<table>
<thead>
<tr>
<th>Model</th>
<th>Mean Match Score</th>
<th>Mean SSIM [159]</th>
<th>FID [65]</th>
<th>Mean Perceptual Error [131]</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenFace [97]</td>
<td>0.543</td>
<td>0.491</td>
<td>177.06</td>
<td>3.536</td>
</tr>
<tr>
<td>DeepFillv1 [174]</td>
<td>0.481</td>
<td>0.321</td>
<td>241.696</td>
<td>3.204</td>
</tr>
<tr>
<td>SymmFCNet [96]</td>
<td>0.457</td>
<td>0.333</td>
<td>207.117</td>
<td>2.434</td>
</tr>
<tr>
<td>EdgeConnect [116]</td>
<td>0.454</td>
<td>0.178</td>
<td>141.695</td>
<td>3.106</td>
</tr>
<tr>
<td>DeepFake</td>
<td>0.459</td>
<td>0.448</td>
<td>43.03</td>
<td>1.857</td>
</tr>
<tr>
<td>Ours (ProGAN)</td>
<td>0.668</td>
<td>0.466</td>
<td>103.71</td>
<td>2.255</td>
</tr>
<tr>
<td>Ours (Cascaded)</td>
<td>0.722</td>
<td>0.753</td>
<td>46.12</td>
<td>1.256</td>
</tr>
</tbody>
</table>
pair of images, using a network trained on human ratings. A lower mean perceptual error indicates less noise in the hallucinated output, therefore a better model.

6.2.1 Comparison with Facial Inpainting Models

To gauge how our model compares with algorithms for generating missing pixels, we make use of four popular facial inpainting models: GenFace \[97\], DeepFillv1 \[173\], SymmFCNet \[96\], and EdgeConnect \[116\]. We choose these models for our experiments, as - (1) they are open source with a pre-trained (on face images from CelebA \[103\]) models available for use, unlike \[72\], \[33\], \[130\], (2) they can work with 128×128 face images, unlike \[174\], and (3) require no any user annotation, unlike \[79\].

To compare the models, we generate hallucinations using face masks from LFW \[70\]. Since each model is trained with different binary masks of missing pixels, we provide the model a binary mask with every pixel outside the face labeled as ‘0’ instead of the actual masked face we feed to our trained model. Both qualitative and quantitative comparisons can be seen in Fig. 6.4 and Table 6.1 respectively. As shown in the table, our model (both versions) performs much better than the inpainting models for all metrics. These models aim to hallucinate the missing pixels, usually on or near the face region, using visual cues provided by facial pixels available in the image. Such cues are absent when whole of the context and background is masked, leading to noisy output. On the other hand, our model is specifically trained, and better suited for this task.

6.2.2 Comparison with DeepFake Face Swap

We compare our model against the extremely popular DeepFake face swapping application. The software essentially trains an autoencoder to learn transformations to change an input face crop (target) to another identity (source) while keeping
target visual attributes intact. Since this autoencoder learns transformations for one subject at a time, we train it using 64×64 tight face crops of ‘George_W_Bush’, the LFW[70] identity with the most images (530). The autoencoder is trained for 10K iterations using these 530 images, following which it can be used to hallucinate images of ‘George_W_Bush’ from face crops of other subjects and then blended onto the target images. The results of such a face swapping process can be seen in Figure 6.5 where we swap ‘George_W_Bush’ face images onto the context and background of ‘Colin_Powell’. We choose ‘Colin_Powell’ as the mean hypercolumn [58] descriptor of his images, using conv-[1,2,3,4,5] features from VGG-FACE [121], is proximal to that of ‘George_W_Bush’.

Although DeepFake produces plausible results (lower FID [65] in Table 6.1), it

\footnote{We use the implementation from the most popular repo: \url{https://github.com/deepfakes/faceswap}}
requires both the source and target subjects to have fairly similar skin tone, pose and expression. Without such tight constraints, artifacts at the boundary of the blending mask are present as can be seen in the top row of Figure 6.5 due to the difference in skin tone and absence of eyeglasses in the source identity. Our model, on the other hand, has no such constraints as it learns to hallucinate the full set of context and background pixels from the provided face mask itself. Also, our model achieves a higher mean match score than DeepFake suggesting that it preserves more discriminative features of the source in the hallucinated images while adding variations in appearance.

6.2.3 Comparison with our ProGAN Model

For the progressively growing (ProGAN [81]) version of our model, we set a training interval of 50 epochs after which we add new layers to the current block and resume training. Compared to the 96.53 hours required to train our cascaded network, our ProGAN model requires 66.24 hours to complete the full training at all scales, when trained on the same Titan Xp GPU system. The absence of multi-scale training, upscaling between blocks and depth concatenations during each iteration is the reason behind its lower training time. At the end of training, we feed 128×128 face masks to block_128 and get the hallucinated face images at the same resolution. We compare our cascaded and ProGAN models using masked face images from LFW [70]; the quantitative results are shown in Table 6.1 and few qualitative samples can be seen in Figure 6.4.

Although the ProGAN model hallucinates slightly sharper results than the cascaded model due to the absence of upscaling between GAN blocks, it suffers from blurry artifacts, especially in the hair. This can be attributed to the fact that we only use block_128 of the ProGAN model to synthesize the output directly at of 128×128 like the trained generator from a single resolution GAN. Since the hallucination pro-
cess in the cascaded network is guided at each resolution by the previous block, such artifacts are less frequent in its case. This might also be the reason for the difference in FID and perceptual error values between the two models in Table 6.1.

6.2.4 Effectiveness as Supplemental Training Data

To evaluate if our model can be used to augment existing face image datasets, we perform a recognition experiment using the CASIA-WebFace (CW) dataset [175]. CW contains 494,414 face images of 10,575 real identities collected from the web. We align, mask and resize all the face images from CW using the same pre-processing steps as our training data. These masked images are then fed to our trained cascaded model to hallucinate synthetic context and background pixels. Since the identity of the input face mask is preserved in our model (as shown by the Mean Match Score in Table 6.1), we label the hallucinated image as the same class as the original input from CW, similar to [107, 108, 15]. In this way, we generate 494,414 synthetic images, with hallucinated context and background, from 494,414 existing images of 10,575 real identities. We prepare two training sets from the images - 1) a dataset containing 494,414 real images from CW and no synthetic images (Dataset 1 from Table 6.2), and 2) a dataset containing 494,414 real images and 494,414 synthetic images of the same 10,575 subjects (Dataset 2 from Table 6.2).

We fine-tune the ResNet-50 [63] model with these datasets in two separate training sessions, where 90% of the data is used for training and the rest for validation. The networks are trained using the Caffe [76] framework, with a base learning rate = 0.001 and a polynomial decay policy where gamma = 0.96, momentum = 0.009, and step size = 32K training iterations. We set the batch size = 16, and train each network till its validation loss plateaus across an epoch. After training terminates, we save its snapshot for testing on the LFW dataset [70]. Each image is passed to the snapshot and its 256-D vector is extracted from the penultimate (feat_extract) layer.
TABLE 6.2

DISTRIBUTION AND PERFORMANCE OF TRAINING DATASETS WITH AND WITHOUT AUGMENTATION

<table>
<thead>
<tr>
<th>Training Data</th>
<th>CW [175] Images (Identities)</th>
<th>Hallucinated Images (Identities)</th>
<th>LFW [70] Performance (TPR@FPR = 0.01)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset 1</td>
<td>494,414 (10,575)</td>
<td>0</td>
<td>0.963</td>
</tr>
<tr>
<td>Dataset 2</td>
<td>494,414 (10,575)</td>
<td>494,414 (10,575)</td>
<td>0.971</td>
</tr>
</tbody>
</table>

We use these features to perform a verification experiment (all vs. all matching) with Pearson correlation for scoring, the results of which are presented in Table 6.2. As shown, the supplemental synthetic images introduce more intra-subject variation in context and background, which in turn slightly boosts the performance of the network during testing. Our trained model can therefore be used to augment existing face image datasets for training CNNs, especially to generate the diverse context and background pixels in synthetic face masks generated by [107, 15].

6.3 Detailed Model Architecture

In this section, we list the layers of each generator block of our model. For both the cascaded and progressively growing (ProGAN) [81] versions of our model, the architectures of the generator block remain the same. For the cascaded model however, we use a set of four pixel shuffling [145] blocks to upscale the hallucination of a block 2x before feeding it as input to the next generator block. The architecture of each upscaling pixel shuffling blocks remains the same. The detailed layers of ‘block_8’, ‘block_16’, ‘block_32’, ‘block_64’, and ‘block_128’ layers are listed in Tables
### TABLE 6.3

**BLOCK_8 ARCHITECTURE (INPUT SIZE IS 8×8×3)**

<table>
<thead>
<tr>
<th>Layer</th>
<th>Filter/Stride/Dilation</th>
<th># of filters</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv0</td>
<td>3×3/1/2</td>
<td>128</td>
</tr>
<tr>
<td>conv1</td>
<td>3×3/1/2</td>
<td>1,024</td>
</tr>
<tr>
<td>RB1</td>
<td>3×3/1/1</td>
<td>1,024</td>
</tr>
<tr>
<td>fc1</td>
<td>512</td>
<td>-</td>
</tr>
<tr>
<td>fc2</td>
<td>16,384</td>
<td>-</td>
</tr>
<tr>
<td>conv2</td>
<td>3×3/1/1</td>
<td>4×512</td>
</tr>
<tr>
<td>PS1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>conv3</td>
<td>5×5/1/1</td>
<td>3</td>
</tr>
</tbody>
</table>

6.3, 6.4, 6.5, 6.6 and 6.7 respectively. The convolution layers, residual blocks and pixel shuffling layers are indicated as ‘conv’, ‘RB’, and ‘PS’ respectively in the tables. For each of these layers in the generator, we used leaky ReLU with slope of 0.1 as the activation, except for the last ‘conv’ layer where a tanh activation is used [132, 138].

### 6.4 Ablation Studies

In this section, we analyze the effect of each component of our loss function on the overall quality of context and background synthesis. We present a comprehensive comparison that includes both qualitative results and quantitative experiments, using face images from the LFW dataset [70].

For this experiment, we prepare four variations of our multi-scale cascaded GAN model, while keeping the network architecture intact. We replace $l_1$ loss with $l_2$ loss as the metric for computing $L_{pixel}$ for one model. For the other three models, we
### TABLE 6.4

**BLOCK-16 ARCHITECTURE (INPUT SIZE IS 16×16×3)**

<table>
<thead>
<tr>
<th>Layer</th>
<th>Filter/Stride/Dilation</th>
<th># of filters</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv0</td>
<td>3x3/1/2</td>
<td>128</td>
</tr>
<tr>
<td>conv1</td>
<td>3x3/2/1</td>
<td>512</td>
</tr>
<tr>
<td>RB1</td>
<td>3x3/1/1</td>
<td>512</td>
</tr>
<tr>
<td>conv2</td>
<td>3x3/2/1</td>
<td>1,024</td>
</tr>
<tr>
<td>RB2</td>
<td>3x3/1/1</td>
<td>1,024</td>
</tr>
<tr>
<td>fc1</td>
<td>512</td>
<td>-</td>
</tr>
<tr>
<td>fc2</td>
<td>16,384</td>
<td>-</td>
</tr>
<tr>
<td>conv3</td>
<td>3x3/1/1</td>
<td>4*512</td>
</tr>
<tr>
<td>PS1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>conv4</td>
<td>3x3/1/1</td>
<td>4*256</td>
</tr>
<tr>
<td>PS2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>conv5</td>
<td>5x5/1/1</td>
<td>3</td>
</tr>
<tr>
<td>Layer</td>
<td>Filter/Stride/Dilation</td>
<td># of filters</td>
</tr>
<tr>
<td>-------</td>
<td>------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>conv0</td>
<td>3×3/1/2</td>
<td>128</td>
</tr>
<tr>
<td>conv1</td>
<td>3×3/2/1</td>
<td>256</td>
</tr>
<tr>
<td>RB1</td>
<td>3×3/1/1</td>
<td>256</td>
</tr>
<tr>
<td>conv2</td>
<td>3×3/2/1</td>
<td>512</td>
</tr>
<tr>
<td>RB2</td>
<td>3×3/1/1</td>
<td>512</td>
</tr>
<tr>
<td>conv3</td>
<td>3×3/2/1</td>
<td>1,024</td>
</tr>
<tr>
<td>RB3</td>
<td>3×3/1/1</td>
<td>1,024</td>
</tr>
<tr>
<td>fc1</td>
<td>512</td>
<td>-</td>
</tr>
<tr>
<td>fc2</td>
<td>16,384</td>
<td>-</td>
</tr>
<tr>
<td>conv3</td>
<td>3×3/1/1</td>
<td>4*512</td>
</tr>
<tr>
<td>PS1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>conv4</td>
<td>3×3/1/1</td>
<td>4*256</td>
</tr>
<tr>
<td>PS2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>conv5</td>
<td>3×3/1/1</td>
<td>4*128</td>
</tr>
<tr>
<td>PS3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>conv6</td>
<td>5×5/1/1</td>
<td>3</td>
</tr>
</tbody>
</table>
### TABLE 6.6

**BLOCK\_64 ARCHITECTURE (INPUT SIZE IS 64×64×3)**

<table>
<thead>
<tr>
<th>Layer</th>
<th>Filter/Stride/Dilation</th>
<th># of filters</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv0</td>
<td>3×3/1/2</td>
<td>128</td>
</tr>
<tr>
<td>conv1</td>
<td>3×3/2/1</td>
<td>128</td>
</tr>
<tr>
<td>RB1</td>
<td>3×3/1/1</td>
<td>128</td>
</tr>
<tr>
<td>conv2</td>
<td>3×3/2/1</td>
<td>256</td>
</tr>
<tr>
<td>RB2</td>
<td>3×3/1/1</td>
<td>256</td>
</tr>
<tr>
<td>conv3</td>
<td>3×3/2/1</td>
<td>512</td>
</tr>
<tr>
<td>RB3</td>
<td>3×3/1/1</td>
<td>512</td>
</tr>
<tr>
<td>conv4</td>
<td>3×3/2/1</td>
<td>1,024</td>
</tr>
<tr>
<td>RB4</td>
<td>3×3/1/1</td>
<td>1,024</td>
</tr>
<tr>
<td>fc1</td>
<td>512</td>
<td>-</td>
</tr>
<tr>
<td>fc2</td>
<td>16,384</td>
<td>-</td>
</tr>
<tr>
<td>conv3</td>
<td>3×3/1/1</td>
<td>4*512</td>
</tr>
<tr>
<td>PS1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>conv4</td>
<td>3×3/1/1</td>
<td>4*256</td>
</tr>
<tr>
<td>PS2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>conv5</td>
<td>3×3/1/1</td>
<td>4*128</td>
</tr>
<tr>
<td>PS3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>conv6</td>
<td>3×3/1/1</td>
<td>4*64</td>
</tr>
<tr>
<td>PS4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>conv7</td>
<td>5×5/1/1</td>
<td>3</td>
</tr>
</tbody>
</table>
### TABLE 6.7

**BLOCK_128 ARCHITECTURE (INPUT SIZE IS 128×128×3)**

<table>
<thead>
<tr>
<th>Layer</th>
<th>Filter/Stride/Dilation</th>
<th># of filters</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv0</td>
<td>3×3/1/2</td>
<td>128</td>
</tr>
<tr>
<td>conv1</td>
<td>3×3/2/1</td>
<td>64</td>
</tr>
<tr>
<td>RB1</td>
<td>3×3/1/1</td>
<td>64</td>
</tr>
<tr>
<td>RB2</td>
<td>3×3/1/1</td>
<td>128</td>
</tr>
<tr>
<td>conv2</td>
<td>3×3/2/1</td>
<td>128</td>
</tr>
<tr>
<td>RB3</td>
<td>3×3/1/1</td>
<td>256</td>
</tr>
<tr>
<td>RB4</td>
<td>3×3/1/1</td>
<td>128</td>
</tr>
<tr>
<td>conv3</td>
<td>3×3/2/1</td>
<td>256</td>
</tr>
<tr>
<td>RB5</td>
<td>3×3/1/1</td>
<td>256</td>
</tr>
<tr>
<td>conv4</td>
<td>3×3/2/1</td>
<td>256</td>
</tr>
<tr>
<td>conv5</td>
<td>3×3/2/1</td>
<td>512</td>
</tr>
<tr>
<td>RB6</td>
<td>3×3/1/1</td>
<td>512</td>
</tr>
<tr>
<td>conv6</td>
<td>3×3/2/1</td>
<td>1,024</td>
</tr>
<tr>
<td>RB7</td>
<td>3×3/1/1</td>
<td>1,024</td>
</tr>
<tr>
<td>fc1</td>
<td>512</td>
<td>-</td>
</tr>
<tr>
<td>fc2</td>
<td>16,384</td>
<td>-</td>
</tr>
<tr>
<td>conv3</td>
<td>3×3/1/1</td>
<td>4*512</td>
</tr>
<tr>
<td>PS1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>conv4</td>
<td>3×3/1/1</td>
<td>4*256</td>
</tr>
<tr>
<td>PS2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>conv5</td>
<td>3×3/1/1</td>
<td>4*128</td>
</tr>
<tr>
<td>PS3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>conv6</td>
<td>3×3/1/1</td>
<td>4*64</td>
</tr>
<tr>
<td>PS4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>conv7</td>
<td>3×3/1/1</td>
<td>4*64</td>
</tr>
<tr>
<td>PS5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>conv8</td>
<td>5×5/1/1</td>
<td>3</td>
</tr>
</tbody>
</table>
Figure 6.6. Ablation studies - hallucination results of our multi-scale GAN model and its variants.

remove one of the other three losses (i.e., $L_{adv}$, $L_{id}$, and $L_{pc}$) in each case. We keep the weight of the other loss components intact in each case. To analyze the role of the training regime, we compare each of these cascaded models with our ProGAN model keeping other factors constant. For this experiment, we use the same set of quality metrics as before - (1) mean match score with ResNet-50 [63], (2) mean SSIM [159], (3) FID [65], and (4) mean perceptual error [131] (description of each metric is available in Section 4 of main text). The quantitative results are presented in Table 6.8 along with visual results in Figure 6.6.

As expected, we find using $l_2$ loss for $L_{pixel}$ drastically deteriorates the quality of the hallucinated face images by producing blurrier results. Since the pixel intensities are normalized to [0, 1], $l_2$ loss suppresses high frequency signals, compared to $l_1$, due to its squaring operation. The absence of a discriminator (w/o $L_{adv}$) at a network block fails to push the results towards the distribution of real face images, consequently hampering the performance of the model. Although not as critical as
Table 6.8

Ablation Studies Using LFW [70]

<table>
<thead>
<tr>
<th>Model</th>
<th>Mean Match Score</th>
<th>Mean SSIM [159]</th>
<th>FID [65]</th>
<th>Mean Perceptual Error [131]</th>
</tr>
</thead>
<tbody>
<tr>
<td>(l_2) loss</td>
<td>0.520</td>
<td>0.413</td>
<td>166.76</td>
<td>2.489</td>
</tr>
<tr>
<td>w/o (L_{adv})</td>
<td>0.522</td>
<td>0.411</td>
<td>132.71</td>
<td>2.320</td>
</tr>
<tr>
<td>w/o (L_{id})</td>
<td>0.609</td>
<td>0.519</td>
<td>91.65</td>
<td>1.956</td>
</tr>
<tr>
<td>w/o (L_{pc})</td>
<td>0.624</td>
<td>0.528</td>
<td>101.44</td>
<td>2.046</td>
</tr>
<tr>
<td>Ours (ProGAN)</td>
<td>0.668</td>
<td>0.466</td>
<td>103.71</td>
<td>2.255</td>
</tr>
<tr>
<td>Ours (Cascaded)</td>
<td><strong>0.722</strong></td>
<td><strong>0.753</strong></td>
<td><strong>46.12</strong></td>
<td><strong>1.256</strong></td>
</tr>
</tbody>
</table>

\(L_{pixel}\) and \(L_{adv}\), the inclusion of both \(L_{id}\) and \(L_{pc}\) refine the hallucination result, as is apparent from both the example images and the quality scores. The impact of the training regime, comparing end-to-end cascaded training with progressive growing (ProGAN), has already been discussed in Section 4 of the main text.

6.5 Epoch by Epoch Learning

To understand how the context and background are learned by the model during training, we save snapshots of our cascaded GAN model at different levels of training - 10 epochs, 20 epochs, 30 epochs, 40 epochs and 50 epochs. Except the training iterations, all other parameters and hyper-parameters remain the same. These models are then used to generate context and background pixels on masked face images from LFW [70]. Hallucinations for three such images have been shown in Figure 6.7.

As is apparent from the Figure, the model learns to generate a rough set of hair and skin pixels in the first few training epochs, not focusing on the clothes or
background (10-20 epochs). Then it adds in pixels for the clothes and background, while further refining the overall skin and hair pixel quality (30-40 epochs). The validation loss stabilizes around the 50-th epoch (our hard termination point), and hence this snapshot has been used in our experiments. We also find the model to take a few extra iterations of refinement in hallucinating context and background for images with posed faces compared to those with frontal faces.

6.6 Changing the Background Pixels

To add more variety to our images, we add a post-processing step to further change the background pixels, while keeping the face and context pixels unchanged, using background images supplied by the user. We first locate the pixels outside the background (context + face mask) using the segmentation network from [188, 187, 166]. The pixels with the label ‘Person’ are kept inside the mask, which is further refined by a saliency map. This saliency map is computed using the gradient of each
Figure 6.8. Background replacement process - (a) hallucinated face image (b) the detected foreground mask using a combination of gradient map and the segmentation network from [188], [187], and [166], and (c) background pixels replaced with Laplacian blending [30].

pixel of the image and the outer contour detected as the salient edge. The union of the initial mask and the points inside this contour produces the final foreground mask. Alternatively, the foreground mask can also be generated using the image matting network provided in [169]. The new background image is then blended in with the help of this foreground mask using a Laplacian pyramid based blending [30], [13].

6.7 Additional Qualitative Results

In this section, we present additional qualitative results for visual perusal. Face images, varying in gender, ethnicity, age, pose, lighting and expression, are randomly selected from the LFW dataset [70] and IJB-B [161] video frames. Each image is then aligned about their eye centers using landmark points extracted from Dlib [86], face masked and resized to 128×128. Each image is then fed to the trained snapshots, used in our original experiments, of our cascaded and progressively growing models for context and background pixel synthesis. The results are shown in Figure 6.9.
Figure 6.9. Additional qualitative results generated by our ProGAN and cascaded models. The first three rows are samples from the LFW [70] dataset, while the last three rows are taken from the IJB-B [161] dataset. All images are 128×128 in size.
6.8 Model Limitations

As our model learns to hallucinate from the training data, we observe visual artifacts for face masks which vary drastically in appearance from it. For example, it fails to hallucinate missing pixels of occluding objects present in the face mask (like the microphone in leftmost image in Figure 6.10). This can be fixed by refining the input face mask to remove such occluding objects. In some cases our model mis-labels the gender of the face mask and generates the wrong hairstyle. Such an example can be seen Figure 6.10 (rightmost image), where the input male subject gets a female hairstyle. This issue can be resolved by either training two networks separately with male and female subjects or by adding a gender preserving loss (using [95]) to the loss function. Our model also fails to generate matching temples when the subject wears eyeglasses due to their absence in the training images (Figure 6.10 middle image). To tackle this issue, the training data can be augmented by adding eyeglasses to some images using [114, 64, 34].

6.9 Discussion

In this chapter, we propose a cascaded network of GAN blocks that can synthesize realistic context and background pixels given a masked face input, without requiring any user supervision. Instead of swapping a source face onto a target image or inpainting small number of missing facial pixels, our model directly hallucinates the entire set of context and background pixels, by learning their representation directly from the training data. Each GAN block learns to hallucinate the missing pixels at a particular resolution via a combination of different losses and guides the synthesis process of the next block.

While trained on only 12K face images acquired at a controlled setting, our model is effective in generating on challenging images from the LFW [70] dataset. When
Figure 6.10. Some problematic cases - missing pixels for the microphone occluding subject’s chin (left), no matching temples generated for the eyeglasses (middle), and hairstyle of wrong gender (right).
compared with popular facial inpainting models [97, 173] and face swapping methods (DeepFake), our model generates more identity-preserving (evaluated using deep features from ResNet-50 [63]) and realistic (evaluated using SSIM [159], FID [63], and perceptual error [131]) hallucinations. Our model can also be used to augment training data for CNNs by generating different hair and background of real subjects [175] or rendered synthetic face masks using [107, 15]. This can increase the intra-class variation in the training set, which in turn can make the CNN more robust to changes in hair and background along with variations in facial pose and shape. The generated face images can also be used as stock images by the media without any privacy concerns.
CHAPTER 7

FUTURE WORK AND CONCLUSION

This SREFV portion of this chapter is based on an ongoing project at Notre Dame. Dr. Daniel Moreira has contributed equally, if not more, to the project and many of the figures shown here was generated by him.

7.1 SREFV: Synthesis of Realistic Example Face Videos

The SREFI face generation modules described so far in this dissertation synthesizes face images of non-existent identities with varying facial texture, pose, shape, context and background. However, an important tune-able parameter missing from our technique is the capability to change the facial expression at different facial pose. Varying the expression adds to the facial diversity, like shape, pose and context, and can be used as an augmentation technique [107]. Researchers have used generative models to synthesize novel set of expressions for a given identity [172] [16], however the images are limited in terms of resolution due to network size and memory constraints. To tackle this problem, we use a combination of deep learning and computer graphics for changing the expression of the SREFI generated synthetic images. As an application, we can produce a large set of artificial facial videos that are coherent with other video frames of real people in terms of pose, expression and phoneme. These synthetic videos can be used to de-identify existing subjects to preserve privacy or as supplemental data for training networks tasked with the detection of DeepFake [1] based videos.
Figure 7.1. A high-level overview of the current SREFV system. It takes a synthetic donor face and an exemplar video as input data and generates three videos with the donor face animated in 3D.
Figure 7.1 presents a high-level overview of our current video generation module. The system is fed two input elements - (1) a synthetic face image (donor), and (2) an exemplar video. The system generates three output videos - (1) the masked synthetic face animated in 3D with pose and expressions corresponding to the real face, (2) the animated synthetic face warped into the original video background, and (3) the original background warped into the animated synthetic face.

The system starts with the detection of the faces using Dlib [86] and extraction of facial landmarks in the exemplar video frames using [74]. We estimate the expressions of the faces detected across the exemplar video frames using trained encoders for the different facial regions. More specifically, we divide the face detected in the video frame into three sub-images, corresponding to the eyes, the nose and the mouth, based on the extracted landmark keypoints (check Figure 7.2). The expressions present in these regions are encapsulated using different facial action units [47]. We prepare two encoder models based on the VGG-16 [146] and ResNet-50 [63] architectures. The networks are trained with curated data generated using FaceGen [2], a software...
for high-quality 3D head model generation, which provides a large pre-defined set of facial expressions, such as blinking, eyebrow raise, nose dilation, lip funneling, jaw drop, etc. Figure 7.3 depicts some examples of curated facial expression applied over the same synthetic identity. Once trained, these models can be used to find the action unit values in exemplar videos during testing.

To apply the estimated set of facial expressions to the synthetic face images, we use OpenGL [3] and a template 3D head model to create facial masks corresponding to the frames in the exemplar video. These masks are then rendered with the head models and then blended into the exemplar video frame. For the blending operation, we divide the face into a set of triangles using Delaunay’s method and use the Laplacian pyramids [30] blend the triangles one at a time, similar to [13, 15]. We use OpenCV [4] to properly warp and splice the 3D mask faces onto their respective exemplar video frames. These frames are then pasted together to generate video output 2, in Figure 7.4. To generate the frames of video output 3, we use warp and splice the frame backgrounds around the generated 3D masks, which are kept steady. An overview of our pipeline can be seen in Figure 7.4.
7.2 Conclusion

This dissertation is a compilation of published works [14, 13, 15] and works under review [12, 16]. Their combination represents an effort to understand the effect of frontalization and data augmentation on face recognition performance, while advancing the state-of-the-art in face synthesis research. This effort comprises of four distinct areas: (1) analyzing facial frontalization, (2) generation of synthetic frontal facial texture, (3) rendering synthetic 3D face masks with varying pose and shape, and (4) hallucinating synthetic context and background pixels for a synthetic face mask.

Since face image datasets [70, 128, 89, 161] used to test trained CNN models contain a variety of facial yaw values, researchers have experimented with the idea of normalizing all face images to a zero yaw value, i.e. frontalization, before testing. However, there are multiple facial frontalization techniques [60, 71] which produce dif-
ferent results depending on the quality of landmarking algorithm used \cite{191,83,154}. To compare with such existing frontalization modes, we develop our own baseline landmarking and frontalization techniques. Through our experiments, we realize that any mode of frontalization ultimately leads to loss of data and only results in a performance gain when both the training and testing sets are pre-processed in the same way. Interestingly, networks trained with millions of face images like VGG-FACE \cite{121} are relatively agnostic to the pre-processing method applied on the testing data.

Applying augmentation techniques on existing face image datasets, used for training CNNs, has been shown to improve the network performance during testing \cite{107,108,147}. However, existing methods \cite{107,108} only apply transformations to existing real identities in the dataset. We propose a simple face synthesis module, named SREFI, that can create an arbitrarily large number of face images of both real and synthetic identities. Our method, unlike popular GAN models \cite{132,24,81,82}, requires only a few minutes of training and a few thousand gallery images to generate natural-looking face images of varying resolution ($100 \times 100$ to $800 \times 600$). These images vary both in terms of facial texture, gender, ethnicity as well as facial pose and shape. Our synthetic images score high on realism tests performed with human evaluators, and can be used: (a) as supplemental training data to introduce more inter-class variance, and (b) as distractor sets to evaluate the robustness of trained models, similar to \cite{85,117}. Additionally, we release the Notre Dame Synthetic Face Dataset (available here: \url{https://cvrl.nd.edu/projects/data/}), which comprises of 2 million face images of 12,000 synthetic identities, generated using SREFI.

The presence of context (forehead, hair, neck, clothes) and background in face images have been shown to improve recognition performance both by humans \cite{42,49} and deep networks \cite{126,17}. Therefore, adding context and background pixels to synthetic face masks, generated using SREFI or other methods \cite{107}, can provide
more information to the network during training. In this regard, we propose a multiscale GAN model to hallucinate realistic context and background pixels automatically from a single input face mask, without any user supervision. Instead of swapping a source face onto a target image or inpainting small number of missing facial pixels, our model directly hallucinates the entire set of context and background pixels, by learning their representation directly from the training data. Our model is composed of a cascaded network of GAN blocks, each tasked with hallucination of missing pixels at a particular resolution while guiding the synthesis process of the next GAN block. The hallucinated full face image is made photo-realistic by using a combination of reconstruction, perceptual, adversarial and identity preserving losses at each block of the network. With a set of extensive experiments, we demonstrate our model to generate more real-looking synthetic face images when compared with popular inpainting algorithms \cite{97, 173, 116, 96} and the viral DeepFake \cite{1} face swapping application. Additionally, we show the potential usage of our model as a data augmentation module for training CNNs, by infusing more intra-class variance and improving its test performance.
BIBLIOGRAPHY


156


35. F. Chollet et al. Keras. [https://github.com/fchollet/keras](https://github.com/fchollet/keras), 2015.


