MASTER

3D Face Reconstruction Using Deep Learning

Jawahar, P.

Award date:
2020

Link to publication
Abstract

3D Face reconstruction is a long-standing problem in the computer vision domain. Usually, they are solved by learning a mapping of multiple images of the same person across different angles to the 3D geometry. In this thesis, we try to achieve the 3D reconstruction using a single image. 3D Face reconstruction from a single image is one of the challenging and classical problems in the research community. Some traditional approaches have been proposed before to solve this challenge. In this thesis we found a hybrid approach to do 3D face Reconstruction using a single image. We extract 2D landmarks from a single image and from there we apply deep learning techniques to do 3D face reconstruction. As a post processing step, Kabsch algorithm is used before the evaluation of the constructed 3D image.
Preface

The thesis is part of my curriculum in Data Science Master’s. It is a documentation of my work done as an intern in Philips from August to December 2019. It presents the research aspects of my work in 3D face reconstruction. For the guidance and help, I would like to express my gratitude and appreciation to Alessio Gallucci, Dmitry Znamenskiy and Erik Schuijers for their unconditional help. I would also like to thank Dr. Renata Medeiros de Carvalho for her invaluable advice, guidance and feedback at every step. It was a fun filled and true learning experience doing my internship, reading numerous papers, implementing methods and writing my thesis.
# Contents

## Contents

List of Figures ix

List of Tables xi

1 Introduction 1
   1.1 Problem Description ............................................. 1
   1.2 Research Questions ............................................. 2
   1.3 Thesis Summary ................................................. 2

2 Background 3
   2.1 3D Data Representations ......................................... 3
   2.2 Artificial Neural Networks ....................................... 6
   2.3 Multilayer perceptrons .......................................... 6
   2.4 Backpropagation ................................................ 7
   2.5 Activation Functions .......................................... 8
   2.6 Batch Normalization ........................................... 9

3 Relevant Work 11
   3.1 Model Based methods - 3D Morphable Models ...................... 11
   3.2 Deep learning approaches ..................................... 12

4 Methodology 15
   4.1 Dataset ........................................................ 15
   4.2 Dataset Preparation ........................................... 16
   4.3 Hardware description .......................................... 17
   4.4 Libraries Used ................................................ 17
   4.5 Training, Validation and Test Data Split ....................... 18
   4.6 Dataset Preprocessing ........................................ 18
   4.7 Model Architecture and Training ................................ 19
   4.8 Fine Tuning the Deep Learning Model .......................... 20
   4.9 Evaluation metric ............................................. 20
   4.10 Post Processing ................................................ 20

5 Results and Discussions 23
   5.1 Results for different types of Model .......................... 23
   5.2 Summary Results ............................................. 28

6 Conclusions 29
   6.1 Future Work .................................................. 29

Bibliography 31
## List of Figures

2.1 pixels vs voxels. Image adapted from [30] .................................................. 3  
2.2 Example of Numerical Point cloud data. .......................................................... 4  
2.3 Example of Point cloud. Image adapted from [24] ............................................ 5  
2.4 Example of 3D Mesh. Image adapted from [13] ............................................... 5  
2.5 Functioning of a neuron. [32] ............................................................................ 6  
2.6 Multilayer perceptron. ....................................................................................... 7  
2.7 Sigmoid Function. ............................................................................................. 8  
2.8 tanh Function. ................................................................................................... 8  
2.9 Relu Function. .................................................................................................. 9  
2.10 Gradient descent on normalized versus unnormalized level curves. The descent path to the optimum is more decreased in the normalized case ........................................... 9  
4.1 landmarks on Image ......................................................................................... 16  
4.2 3D point cloud data in mathematical form ...................................................... 16  
4.3 Dataframe containing all details ......................................................................... 17  
4.4 Model Architecture .......................................................................................... 19  
4.5 Kabsch Algorithm example. Image adapted from [1] ...................................... 21  
4.6 Two face images not aligned. .......................................................................... 21  
5.1 Training Loss vs Validation loss for Head with Landmarks as input .............. 24  
5.2 Original output .................................................................................................. 24  
5.3 Predicted Output ................................................................................................. 24  
5.4 Training Loss vs Validation loss for Head with Landmarks and AGWH as input 25  
5.5 Original output .................................................................................................. 25  
5.6 Predicted Output ................................................................................................. 25  
5.7 Original output .................................................................................................. 26  
5.8 Predicted Output ................................................................................................. 26  
5.9 Training Loss vs Validation loss for Low Resolution Head with Landmarks as input 26  
5.10 Training Loss vs Validation loss for Face with Landmarks as input ............. 27  
5.11 Original output .................................................................................................. 27  
5.12 Predicted Output ................................................................................................. 27
List of Tables

4.1 Dataset Landmarks - Input to the neural networks ........................................ 15
4.2 GPU Configurations ....................................................................................... 17
4.3 Details about libraries used in the thesis ......................................................... 18
4.4 Columnwise Data Standardisation .................................................................... 19
5.1 Post Processing Procrustes Output Table Comparison ..................................... 28
Chapter 1

Introduction

Royal Philips is a global medical technology company focused on improving the health of individuals and facilitating better outcomes from healthy living and prevention to diagnosis, rehabilitation and home care throughout the health spectrum. Philips leverages advanced technology and deep clinical and consumer insights to deliver integrated solutions. The organization has its headquarters in the Netherlands and is a pioneer in diagnostic imaging, image-guided therapy, patient management and health informatics, as well as consumer health and home care.

Plastic surgery [28] is a branch of medicine that involves rebuilding, reconstructing or changing the human body. It can be divided into two main categories: reconstructive surgery and cosmetic surgery. Reconstructive surgery includes craniofacial surgery, hand surgery, microsurgery, and the treatment of burns. While reconstructive surgery aims to reconstruct a part of the body or improve its functioning, cosmetic (or aesthetic) surgery aims at improving the appearance of it. Both of these techniques are used throughout the world.

Under Philips wide range of products, there is a specific group known as healthcare solutions, which aims to improve people’s health quality. They are researching potential ways to do 3D face reconstruction from a single photograph for medical equipments. For such methods reconstructive facial surgery turns out to be a potential application.

Reconstruction of the facial model and surgical simulation are essential in today’s medicine for plastic surgery. Before actual surgery, both can help surgeons design appropriate repair plans and procedures. So pre-surgery planning is very important for plastic surgery. It helps the patient and doctor to have better understanding of surgery outcome. In the case of reconstructive surgery, a portion of face has to be reconstructed after an accident or burn. This lost portion can be reconstructed from just 2D facial pictures.

Our thesis aims to create 3D face models from just single 2D picture. To address this aim, deep learning algorithms are experimented to solve this challenge.

In recent years deep learning has shown good success on solving 2D tasks like image classification, object detection and image segmentation. Not an exception, deep learning has shown great progress for 3D data, i.e., geometric deep learning [16] which deals with 3D data structures. In this thesis we show how deep learning can be applied for 3D data, i.e., 3D reconstruction of faces.

1.1 Problem Description

To provide a solution for pre-surgery planning we need an algorithm which can automatically convert a 2D facial image to 3D face image. Since deep learning is the state of the art technology
for many 2D image related problems, we would like to explore deep learning techniques in 3D domain specifically in constructing 3D faces. Technically the problem we have is multi output regression problem. For this problem deep learning has shown good results [21].

1.2 Research Questions

From the problem statement stated in the above section, the following research questions have been formulated with the aim of having them addressed:

- How should input data be represented to fit the deep learning model?
- How does model performance vary with preprocessing of the dataset?
- What is the lowest evaluation metric loss deep learning model can achieve when reconstructing 3D face or head?
- Which method can be used for the output to correspond to realistic human face or head?

1.3 Thesis Summary

Chapter 2 explains about the concepts of 3D data types and concepts of deep learning.

Chapter 3 provides an overview of related work done in the area of 3D reconstruction.

Chapter 4 describes the steps involved in the implementation approaches. It starts with dataset preparation, the architecture used and the evaluation metrics to evaluate the model.

Chapter 5 discusses the results obtained from the implementation approaches. It discusses about four different types of the model built along with the baseline approach.

Chapter 6 concludes the work based on the discussions posed in Chapter 5.
Chapter 2

Background

This chapter gives a comprehensive overview of the concepts behind the different methods and models. It explains about different 3D data type representations and methods used to reconstruct 3D data. It starts with types of 3D data available and then algorithms present to reconstruct them.

2.1 3D Data Representations

The main types of 3D data are:

- Voxels
- RGB-D
- Point Clouds
- Mesh

Voxels

Voxels [4] are similar to pixels in an image. An image is represented as a 2D array of pixels. Similarly, we can represent 3D image as a 3D array of voxels which is nothing but stacked cubes as shown in the Figure 2.1

![Figure 2.1: pixels vs voxels. Image adapted from [30]](image)

The disadvantage of Voxel-based representation is that it is not always effective as it represents both occupied and unoccupied sections of the scene, creating huge unnecessary demand for
computer storage. Voxel-based representation is therefore not suitable for high-resolution representation of information [5]. As advantage, it can be directly fed into the convolutional neural network. It is commonly used for representation of terrain in games and simulations.

**RGB-D**

RGB-D format [4] is becoming quite popular due to RGB-D sensors such as Microsoft Kinect. It captures the depth information of the object along with the RGB values. These sensors were primarily developed by the gaming industry to allow full body console control, they were extensively used for navigation in the field of robotics, accelerating the design of algorithms for semantic segmentation. Both RGB-D and voxels are said to be in regular form, i.e., it has regular grids.

**Point Clouds**

A point cloud [4] is the raw data product of a 3D scan by laser. It is formed by a set of data points in a virtual three-dimensional space that represents the surface of the object being scanned. This format can be converted into any other higher-level representation such as Polygon mesh. They are represented as a set of vertices where each vertex or row or point has x, y, z coordinates. So point clouds are points which are represented as an array of x, y, z coordinates. An example of point cloud data is shown in Figure 2.2. Point clouds can be converted to triangle mesh models by joining three vertices in a triangular fashion. An example of point cloud image shown in Figure 2.3 where points in 3D space are not connected.

![Figure 2.2: Example of Numerical Point cloud data.](image)
Mesh

A polygon mesh [4] is a collection of edges, faces and vertices that form a 3D polyhedron. The polygons constituting the model are usually triangles or quadrilaterals. This representation can produce a more realistic description with fewer data points, but when resolution increases, it can lead to high storage space. In short, mesh is nothing but triangles connecting the vertices. It is represented as an array of 3 vertices numbers where each vertex has x, y, z coordinates so that they form a triangle. An example mesh is shown in Figure 2.4. We will be using Mesh 3D format for our project as this is the most commonly used format nowadays in the computer graphics community due to its low space consumption compared to voxels.
2.2 Artificial Neural Networks

Artificial neural networks (ANNs) were originally developed as mathematical models of the information processing capabilities of biological brains [25, 29]. It is a computational model that is motivated by the way biological neural networks in the human brain process information. The basic unit of computation in the neural network is called neuron or node. A node gets an input, multiplies the input with some numbers called weights. Each node can get multiple inputs and all of them are multiplied by weight parameters. After multiplication, all of them are summed together to give a single number. After summation operation, they are passed into a nonlinear activation function that squashes the input to a certain range. A detailed explanation of activation functions is explained later. The functioning of a single neuron as explained above is shown in Figure 2.5.

![Figure 2.5: Functioning of a neuron. [32]](image)

In the previous years, numerous assortments of ANNs have showed up, with broadly differing properties. The significant differentiation is between ANNs whose connections form cycles, and those whose connections are acyclic. ANNs without cycles are known as feed forward neural networks. The most widely used form of feed forward neural network is multi layer perceptron (MLP) [29, 6]. ANNs with cycles are referred to as feedback, recursive, or recurrent neural networks.

2.3 Multilayer perceptrons

A multilayer perceptron [10] consists of a system of simple interconnected neurons or nodes, as shown in the Figure 2.6, which is a model representing a nonlinear mapping between an input vector and an output vector. It is composed of a chain of functions. To achieve non-linear transformations activation functions are used in the chain of functions. Each layer in an MLP consists of a set of nodes and every layer is fully connected to the next layer. The first layer is the input layer where the input data is fed. The input layer consists of one neuron per input value or column in the dataset. Layers after the input layer are called hidden layer and the last layer is the output layer. There exist no interconnections inside a layer whilst all neurons in a layer are linked to neurons in adjacent layers. There can be more than one hidden layer depending on the problem. Activation functions used in the last layer dictates the choice of the problem used for modeling. If it is a binary classification problem we use sigmoid functions. For Multi class classification problem we use softmax activation function and number of neurons will be one for each class. For regression problems, we don’t use any activation functions on the last layer.
Multilayer perceptrons can be trained in a supervised manner by providing the corresponding input and output pairs. The correct input and output pairs are repeatedly provided for training and the weight parameters are adjusted until the correct mapping takes place. The difference between the predicted and actual output is known as the error signal. The training uses this error signal to assess the degree to what extent weight parameters should be changed to minimize the perceptron’s overall error. Multilayer perceptron may generalize to new unseen data when trained with appropriate training data.

2.4 Backpropagation

Backpropagation is the common training algorithm used in multi-layer perceptron. The training algorithm for backpropagation [29] uses the gradient descent procedure to try to locate the absolute (or global) minimum of the surface of the error. Backpropagation has been demonstrated in many applications to perform adequately; the majority of the applications discussed in [29] used backpropagation to train the multilayer perceptrons.

The online training based backpropagation algorithm is summarised below based on the implementation in the book [6]:

1. initialise network weights
2. present an input vector from training data into the network
3. propagate the input vector through the network to obtain an output
4. calculate an error signal by comparing the actual output to the desired output
5. error signal is propagated back through the network
6. alter weights such that overall error gets minimised
7. until overall error is satisfactory, repeat steps 2-7 with next input vector.

In this online training based backpropagation algorithm, the network weights are adapted after each pattern has been presented. The alternative is known as batch training, where the summed error for all patterns is used to update the weights.
2.5 Activation Functions

The commonly used activation functions are sigmoid, tanh, and rectified linear units \[22\]. The sigmoid function has an S-shaped graph as shown in Figure 2.7 and squashes the input into a range between 0 and 1. In particular, large negative numbers become 0 and large positive numbers become 1. This function is differentiable and monotonic but its derivative is not monotonic. Its mathematical form is defined by

\[
\sigma(x) = \frac{1}{1 + e^{x}}
\]  

(2.1)

where,

\(x\) = The dot product of inputs and weights of neural network of a single neuron

![Figure 2.7: Sigmoid Function.](image)

The hyperbolic tangent activation function is symmetric bipolar activation function as shown in Figure 2.8 and it squashes the input into the range between -1 and 1. This function is differentiable and monotonic but its derivative is not monotonic. Its mathematical form is defined by

\[
tanh(x) = 2\sigma(2x) - 1
\]

(2.2)

where,

\(x\) = The dot product of inputs and weights of neural network of a single neuron

![Figure 2.8: tanh Function.](image)

The rectified linear unit is simply a threshold at zero as shown in Figure 2.9 and it is famous in recent years because it greatly accelerates the convergence of stochastic gradient descent when compared to sigmoid and tanh. Both the function and its derivative are monotonic. Its mathematical form is defined by

\[
f(x) = max(0, x)
\]

(2.3)
where, 
\[ x = \text{The dot product of inputs and weights of neural network of a single neuron} \]

\[
R(z) = \max(0, z)
\]

Figure 2.9: Relu Function.

### 2.6 Batch Normalization

While training a neural network, the weights of the layers change, which also changes the distribution of the inputs. This makes the training slower, especially for very deep networks. This problem, known as internal covariate shift, can be reduced by a common technique called batch normalization (BN) [18]. The basic idea is that you normalize the inputs to a layer with a zero mean and unit standard deviation. This makes each layer in the network learn faster and independently of the other layers, illustrated in Figure 2.10.

Figure 2.10: Gradient descent on normalized versus unnormalized level curves. The descent path to the optimum is more decreased in the normalized case.
Chapter 3

Relevant Work

This section discusses some of the relevant work done in 3D face reconstruction. The first section explains traditional methods which are model based methods. Second section is about deep learning approaches that we are more interested in.

3.1 Model Based methods - 3D Morphable Models

Morphable models [9] are powerful 3D statistical models of a human face. In the original formulation morphable model was shown to be capable of inferring a full 3D facial surface from a single image of a person. It is constructed by performing some kind of dimensionality reduction typically PCA [26] on a set of 3D meshes which are placed in dense correspondence. It is also known to be a generative model consisting of a linear 3D shape and appearance model plus an imaging model, which maps the 3D surface onto an image. The 3D shape and appearance are modeled by taking linear combinations of a training set of example faces.

Dense correspondence means for example, if the vertex with index i in one mesh corresponds to the nose tip it is required that the vertex with the same index in every mesh correspond to the nose tip too. Meshes satisfying the above properties are said to be in dense correspondence with one another. While this correspondence problem is easy to state, it is challenging to solve accurately and robustly between highly variable facial meshes [1].

In [7], Vetter and Blanz observed that both the geometric structure and the texture of human faces can be approximated as a linear combination of vectors. For constructing this linear model, also known as the 3D Morphable Model (3DMM), they scanned a few hundred subjects, found a dense registration between them, and applied a principal component analysis on the corresponding scans. One of the advantages of using the 3DMM is that the solution space is constrained to represent only likely solutions. In the original paper, you have to manually align the 3D mesh to the 2D image and then optimize for the shape and texture parameters. While the original paper assumes manual initialization, an automatic reconstruction process [14] is proposed by more recent efforts. Nevertheless, automated initialization pipelines do not generate the same reconstruction quality when only one image is used, as noted in [27].

In the original formulation, Blanz and Vetter solved the dense correspondence problem by representing each facial mesh in a cylindrical UV map, flattening each 3D surface down into a 2D space. This reduced establishing correspondence to a well-understood image registration problem, which was solved with a regularized form of optical flow [8].

In general, Morphable face model of Blanz is derived from a dataset of 200 coloured 3D scans of faces. Individual faces are combined to a single morphable model by computing dense point
to point correspondence to reference face. A modified optic flow algorithm establishes 3D correspondence automatically. The morphable model combines 3D shape and texture information of all example faces into one vector space of faces. We can form an arbitrary linear combination of examples and generate continuous transitions. To reconstruct the 3D face first we manually align the average face to the target image roughly estimating the position, size, orientation and illumination. Then a fully automated algorithm finds the best reconstruction of the face within the morphable model. 3D shape and texture are optimized along with size, orientation and colour contrast. The output is high resolution 3D mesh of the face. It is an estimate 3D shape and surface colours based on the single image. After the reconstruction of a face, a whole range of facial variations can be applied. We can simulate weight gain and weight loss. Illumination conditions can be changed and pose can be varied to some extent.

In Large scale 3D morphable models paper [8] their main contributions are firstly they trained on 10000 faces with varied ethnic backgrounds. The existing state of the art model like Basel face model is built only using 200 faces and they are less diverse.

3.2 Deep learning approaches

Most earlier methods are involved in predicting the 3DMM parameters. They used images as input and sometimes landmarks of the images too and directly regressed the 3DMM parameters. From the parameters, a 3D model is obtained. The above process is much complicated and to simplify the above pipeline Aron Jackson on his work [19] used CNN to directly predict the voxels instead of parameters which simplified the process very much. He turned the regression problem to binary volume segmentation problem because he says regressing directly the vertices output could cause difficulty in learning.

Yao Feng et al. [15] simplified the framework and they claim to have the fastest running model since their model is simple. They converted the 3D data to position maps which becomes now a UV position map. They are nothing but UV coordinates which is an unwrapped 3D model to 2D texture space. Then they used images and UV maps to train the model using resnet architecture as base. From this procedure they were able to learn the UV maps and construct them fastly. They converted again from UV map to 3D space. They claim to have the fastest model and low error rate in 3D face reconstruction.

Tal Hassner et al. [31] used CNN to directly estimate the 3DMM shape parameters from single photo. Since CNN need huge data to train they used multiple images of same person. By training this way improved the performance of the model to capture discriminative features of same person.

Some methods like Densereg [17] and Learning Dense Facial Correspondences [34] uses CNNs to learn the dense correspondence between the input image and the 3D model, and then calculates the 3DMM parameters with predicted dense constraints.

Chen et al. [23] in their work explored that they could learn 3D point clouds from 2D images. They built a generator that learns the object image using 2D CNN and generates the object image from different viewpoints. So the input is a single RGB image and the output of the generator is multiple 2D images from predetermined viewpoints. Next step they fused the different viewpoint images to point cloud. They evaluated the model using the generated viewpoint images to ground truth point cloud rendered images.

Christopher et al. [12] in their work deal with voxels because surfaces are non euclidean and convolutions do not work with them. To work with them the trick is to do voxelisations. So, instead of working on the surface, they work with 3D volumes. The model has encoder like VGG to encode the image, in the middle, they have a recurrent model and then decoder. The main
idea is to leverage the power of LSTM to retain previous observations and incrementally refine the output reconstruction as more observations become available. The loss they use is voxelwise cross entropy. So the goal is to know if each voxel is filled or not. The limitations of this model are less resolution. The model set of images as sequences because of the recurrent layer.

Nanyang Wang et al. [33] proposed an end-to-end architecture of deep learning that generates a 3D structure from a single colour image in a triangular mesh. Limited by the design of the deep neural network, previous methods generally represent a 3D structure in volume or point cloud, and converting them to the more ready-to-use mesh model is more important. In contrast to existing methods, Nanyang’s network represents 3D mesh in a graph-based convolutional neural network and produces correct geometry by gradually deforming an ellipsoid using perceptual features extracted from the input image. The drawback of this method is it is trained for 3D objects and it has not been experimented for 3D face models.
Chapter 4

Methodology

This chapter demonstrates an understanding of the dataset that has been worked on and the steps that have been adopted and undergone to answer the research questions stated in Chapter 1.

4.1 Dataset

The dataset used for this project was provided by Philips and was used for the experimental study to make a product for pre-surgery planning in reconstructive plastic surgery. 3463 participants took part in this study. 3D point cloud data was recorded for each participant with 8 different poses. Therefore, this dataset consists of images, point cloud data for those images and faces also known as triangles which are used to connect the point clouds in a triangular fashion.

Philips has its landmark detection algorithm which is used to get the landmarks from the images. There are, in total, 65 landmarks identified by the Philips algorithm for each image. We use these landmarks as features in terms of machine learning language as inputs to our deep learning model. An example of landmarks on the image is shown in Figure 4.1. The white points on the image are landmarks.

Table 4.1 shows an example dataset of landmarks. These landmarks are obtained after dataset preparation which are explained further on the upcoming section. We detect 65 \((x,y)\) landmark points for each image. So when we flatten the landmarks it has 130 points. i.e., \(x_1,y_2,x_3,y_4,...x_{129},y_{130}\).

Each row in table 4.1 corresponds to flattened landmarks for one image. i.e., landmarks for one person for one particular pose. There are approximately 14000 rows or samples.

<table>
<thead>
<tr>
<th>s.no</th>
<th>(x_1)</th>
<th>(y_2)</th>
<th>(x_3)</th>
<th>(y_4)</th>
<th>...</th>
<th>...</th>
<th>(x_{129})</th>
<th>(y_{130})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>427.09</td>
<td>226.49</td>
<td>302.53</td>
<td>255.16</td>
<td>...</td>
<td>...</td>
<td>287.35</td>
<td>410.19</td>
</tr>
<tr>
<td>2</td>
<td>469.21</td>
<td>287.73</td>
<td>469.72</td>
<td>314.95</td>
<td>...</td>
<td>...</td>
<td>379.96</td>
<td>453.11</td>
</tr>
<tr>
<td>3</td>
<td>431.88</td>
<td>236.19</td>
<td>429.86</td>
<td>266.66</td>
<td>...</td>
<td>...</td>
<td>300.66</td>
<td>338.20</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>13999</td>
<td>409.18</td>
<td>290.39</td>
<td>408.35</td>
<td>316.21</td>
<td>...</td>
<td>...</td>
<td>249.85</td>
<td>387.92</td>
</tr>
<tr>
<td>14000</td>
<td>424.53</td>
<td>246.77</td>
<td>423.47</td>
<td>273.56</td>
<td>...</td>
<td>...</td>
<td>326.11</td>
<td>300.74</td>
</tr>
</tbody>
</table>

Table 4.1: Dataset Landmarks - Input to the neural networks
4.2 Dataset Preparation

Initially, the dataset was given as .json and .mat files. The .mat files correspond to Matlab files and it contained the 3D point cloud data for each participant. 3D point cloud data means x, y, z positions in 3D space as explained in Chapter 2. Its dimension for high resolution image is \((48051, 3)\) where 48051 represents the number of vertices or number of rows and 3 represents the number of columns or x, y, z positions. An example 3D point cloud data which is mathematically read as a matrix is shown in the Figure 4.2.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>51.00599</td>
<td>-10.46734</td>
<td>-147.00601</td>
</tr>
<tr>
<td>1</td>
<td>55.086285</td>
<td>15.323897</td>
<td>-82.56736</td>
</tr>
<tr>
<td>2</td>
<td>51.309467</td>
<td>10.950463</td>
<td>-74.80814</td>
</tr>
<tr>
<td>3</td>
<td>52.636654</td>
<td>-11.837608</td>
<td>-80.44656</td>
</tr>
<tr>
<td>4</td>
<td>45.911484</td>
<td>-39.8466</td>
<td>-67.240395</td>
</tr>
<tr>
<td>5</td>
<td>31.058628</td>
<td>70.20614</td>
<td>-54.864605</td>
</tr>
<tr>
<td>6</td>
<td>6.677826</td>
<td>72.42662</td>
<td>-44.77671</td>
</tr>
<tr>
<td>7</td>
<td>13.838981</td>
<td>59.580124</td>
<td>-85.434298</td>
</tr>
<tr>
<td>8</td>
<td>10.825263</td>
<td>-0.41951567</td>
<td>-42.91552</td>
</tr>
<tr>
<td>9</td>
<td>26.825733</td>
<td>56.44901</td>
<td>-49.95837</td>
</tr>
<tr>
<td>10</td>
<td>5.680725</td>
<td>46.78623</td>
<td>-54.242006</td>
</tr>
<tr>
<td>11</td>
<td>-72.85482</td>
<td>-99.86224</td>
<td>-104.62377</td>
</tr>
<tr>
<td>12</td>
<td>-85.4805</td>
<td>-43.66121</td>
<td>-68.5963</td>
</tr>
<tr>
<td>13</td>
<td>-88.22289</td>
<td>4.220644</td>
<td>-71.02975</td>
</tr>
</tbody>
</table>

The .json file contained the participant id, pose number, landmarks for each pose, features (age, gender, weight, height), landmark confidence for each pose. From the .json file we prepared two scripts. One script which could extract the landmarks of a person whose landmark confidence was greater than 0.7 and put them in a separate CSV file for each image with the participant id name as filename. Another script which could create a data frame consisting of participant ID, pose number, landmarks and features. The data obtained from the two scripts helped to create a deep learning framework which could train the data in two methods by having two different data...
CHAPTER 4. METHODOLOGY

4.3 Hardware description

The model was trained in a high-performance node, using an NVIDIA Tesla K80 and P100 GPU. GPU configurations are explained in the table 4.2.

<table>
<thead>
<tr>
<th>multi processors on device</th>
<th>Tesla K80</th>
<th>Tesla P100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>13</td>
<td>56</td>
</tr>
<tr>
<td>global memory available on device</td>
<td>10 GB</td>
<td>16 GB</td>
</tr>
</tbody>
</table>

Table 4.2: GPU Configurations

4.4 Libraries Used

The main libraries used during the thesis are mentioned in table 4.3. With python as the choice of programming language, scipy and pandas were used for data preparation, PyTorch library was used to create deep learning models and meshlab was used for visualization. The implementation of the model uses PyTorch as the main library for building and training the networks. After installing the required libraries, the dataset preparation was carried out as explained in the previous section. After preparing the data, the data loader function was built on PyTorch to train the deep learning models. A detailed explanation of dataset preprocessing, model training and fine tuning of networks are explained in the upcoming sections.
4.5 Training, Validation and Test Data Split

The training/validation/test split-ratios used were 90 percent for the training set, 5 percent for validation set and the remaining 5 percent for the test set. We ensured all the data are randomly split. There are eight image poses for a single participant. We also ensured that a participant pose is split consistently across all the sets such that all participant pose images stay on a single set.

4.6 Dataset Preprocessing

This section describes the detailed strategy to answer the research questions raised in Chapter 1:

- How should input data be represented to fit the deep learning model?
- How do model performance vary with preprocessing of the dataset?

Standardisation is a technique that is often used as part of machine learning and deep learning data preparation. The goal of standardisation is to adjust numeric column values in the dataset to a common scale without distorting differences in the ranges of values. In deep learning, the model learns better and training becomes easy for the network if the values are standardized. Standardisation makes the data to have one standard deviation, following the formula:

\[
standardisation = \frac{x - \text{mean}}{\text{sd}}
\]  

where,

\(x = \) The input to the neural network. In our case landmarks of a image.

In this thesis, we experimented with different standardisation methods and found the best one that worked for the dataset. The different methods experimented in this thesis are described below:

- Training a neural network without standardisation. i.e., training with just raw landmarks data.
- Training a neural network with a constant average mean and average standard deviation of the whole dataset. In this, every landmark is subtracted and divided by a constant mean and a constant standard deviation value of the entire landmarks.
- Training a neural network with column-wise data standardisation. In this, after flattening the dimension of landmarks which was (65,2) dimension, we get 130 landmarks. We get stacked landmarks of x1, y2, x3, y4 .... x129, y130. We deduct each element from their respective column mean and divide them by their respective column standard deviation. Table 4.4 shows an example of this standardisation.

Out of these three methods of standardisation described above column-wise data standardisation gave better results for this dataset resulting in low RMSE loss value.
4.7 Model Architecture and Training

The goal of the model is to reconstruct a 3D face or head which is technically a multiple output or multiple target regression problem. For this multiple output regression problem, we explored how multi layer perceptrons (ANN) could solve this problem. In this section, it is specified how the problem is addressed, the required inputs and the correspondent outputs and how they are represented. Given an RGB image and its point cloud image as inputs, the algorithm provides the 3D shape of the corresponding face.

A landmark detection algorithm of Philips (not disclosed in this thesis) is applied to the RGB image to extract the landmarks of the face. With extracted landmarks, an artificial neural network is then used to train landmarks and point cloud data in a supervised fashion. From a single image, 65 landmark points are extracted as \( x, y \) positions. These are then flattened to get 130 points. The point cloud data for the corresponding image has 48051 vertices. Each vertex has \( x, y \) and \( z \) positions. These are flattened to get 144153 points. Hence with 130 neurons as input layer and 144153 neurons in the output layer, the neural network was trained. The hidden layer and the number of neurons in the hidden layer were set up by experimenting with different configurations. The model architecture is shown in Figure 4.4. Four different models were built and their results are shown in chapter 5.

![Figure 4.4: Model Architecture](image-url)

Table 4.4: Columnwise Data Standardisation

<table>
<thead>
<tr>
<th>( (x_1 - \mu_1)/\sigma_1 )</th>
<th>( (y_2 - \mu_2)/\sigma_2 )</th>
<th>( (x_3 - \mu_3)/\sigma_3 )</th>
<th>( (y_4 - \mu_4)/\sigma_4 )</th>
<th>...</th>
<th>( (x_{129} - \mu_{129})/\sigma_{129} )</th>
<th>( (y_{130} - \mu_{130})/\sigma_{130} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>427.09</td>
<td>226.49</td>
<td>302.53</td>
<td>255.16</td>
<td>...</td>
<td>287.35</td>
<td>410.19</td>
</tr>
<tr>
<td>469.21</td>
<td>287.73</td>
<td>469.72</td>
<td>314.95</td>
<td>...</td>
<td>379.96</td>
<td>453.11</td>
</tr>
<tr>
<td>431.88</td>
<td>236.19</td>
<td>429.86</td>
<td>266.66</td>
<td>...</td>
<td>300.66</td>
<td>338.20</td>
</tr>
<tr>
<td>409.18</td>
<td>290.39</td>
<td>408.35</td>
<td>316.21</td>
<td>...</td>
<td>249.85</td>
<td>387.92</td>
</tr>
<tr>
<td>424.53</td>
<td>246.77</td>
<td>423.47</td>
<td>273.56</td>
<td>...</td>
<td>326.11</td>
<td>360.74</td>
</tr>
<tr>
<td>( \mu_1 = 443.86 )</td>
<td>( \mu_2 = 294.04 )</td>
<td>( \mu_3 = 443.07 )</td>
<td>( \mu_4 = 322.38 )</td>
<td>...</td>
<td>( \mu_{129} = 316.67 )</td>
<td>( \mu_{130} = 405.84 )</td>
</tr>
<tr>
<td>( \sigma_1 = 46.05 )</td>
<td>( \sigma_2 = 45.08 )</td>
<td>( \sigma_3 = 46.23 )</td>
<td>( \sigma_4 = 44.40 )</td>
<td>...</td>
<td>( \sigma_{129} = 53.17 )</td>
<td>( \sigma_{130} = 55.83 )</td>
</tr>
</tbody>
</table>
4.8 Fine Tuning the Deep Learning Model

Bias / Variance Trade-Off

High bias means underfitting. It occurs due to the approximation of a complex model to a simple model. High variance means overfitting. Different training sets will result in different estimation. A good model should have low bias and low variance. The two key numbers to look at to understand bias and variance will be the training set and validation set error. When training set error is low and validation set error is high then it means we have overfitted the training data (overfitting). We have not generalized well. When the training set error is high and the validation set error is low then it means underfitting. When both training and validation set errors are high it means high bias and high variance. If we have high bias means we could try bigger network such as more hidden layers or more hidden units or could train it longer. If we have a high variance problem then we should get more data. If that is not possible we can use regularization methods or adding dropout to neural networks.

Initially, while experimenting with the neural network model, the model was suffering from a high bias problem. We increased the number of hidden neurons in the hidden layer to solve this problem.

Optimizers

We choose to use Adam optimizer [3] for training deep regression networks because that configuration performs better as stated in the paper “A Comprehensive Analysis Of Deep Regression” [21].

We also experimented with other optimization algorithms like SGD [2] but Adam tends to converge faster and yielded better results.

4.9 Evaluation metric

Current research on 3D face reconstruction has no standard benchmark datasets or standard evaluation protocols [11]. This makes very hard for the research community to compare results between different methods of 3D face reconstruction. In this work, we use the vertex wise root mean square error metric as our evaluation metric. It is the standard metric used in Philips to evaluate their models. It is defined as follows:

$$E_j = \sqrt{\frac{1}{N} \sum_{i=1}^{N} ||\hat{v}_{i,j} - v_{i,j}||^2}$$

(4.2)

where,

$N$ = The number of participant poses

$\hat{v}_{i,j}$ = The predicted vertex

$v_{i,j}$ = The original vertex

4.10 Post Processing

The Kabsch algorithm [20], named after Wolfgang Kabsch, is a method for calculating the optimal rotation matrix that minimizes the RMSE (root mean squared error) between two paired sets of points. It is useful in graphics, cheminformatics to compare molecular structures, and also bioinformatics for comparing protein structures.
For example, you have molecules A and B and you want to calculate the difference in structure between the two. When you measure the RMSE straight-forward, you may get a value that is too big as shown in Figure 4.5. The two molecules need to be recentered first and then rotated to each other to achieve the true minimum RMSE.

![Figure 4.5: Kabsch Algorithm example. Image adapted from [1]](image)

In our 3D face reconstruction, the output of deep learning model and true output will not be aligned in the same order. An example of this non-alignment is shown in the Figure 4.6. For perfect alignment, the two images should be placed perfectly on top of the other. The RMSE loss value will be high when the face is not aligned. To align them we use the Kabsch algorithm and then measure the RMSE metric which gives the true RMSE loss value which will be low.

![Figure 4.6: Two face images not aligned.](image)
Chapter 5

Results and Discussions

In this chapter, the performance of different architectures of the ANN on Philips datasets is shown. In this thesis, there are four types of models built, such as: head with only landmarks as input; head with landmarks and age, gender, weight, height (AGWH) features as input; low-resolution head with landmarks as input; face with landmarks as input. The evaluation metric used in the thesis to evaluate the performance of the models in reconstructing a 3D face or head was RMSE.

5.1 Results for different types of Model

Besides the baseline model, the four different types of models built are:

- Head with Landmarks as input
- Head with Landmarks and AGWH as input
- Low Resolution Head with Landmarks as input
- Face with Landmarks as input

When starting on a new application it is almost impossible to correctly guess the hyperparameter choices in the first attempt. So, in practice, applied deep learning is a highly iterative process where you experiment with the number of hidden layers, hidden units, learning rates, activation functions etc. Each model operated with a learning rate of 0.001, and the Adam optimizer was used.

Baseline Model

We have a Regression problem in our situation. So, a central measure like the mean is commonly used as a baseline model. The average head is measured across the entire head dataset. This average head is used as the output of prediction and the loss function is calculated between this average head and the original head. The RMSE for the average head is approximately 6.14.

Head with Landmarks as input

In this experiment, ANN was trained with 130 landmarks as inputs and 144153 vertices as outputs. The best hyperparameter choices after experimenting were one single hidden layer with 500 hidden neurons. The training and validation loss obtained after 200 epochs are 11.6 and 12 respectively. The resulting chart of training and validation loss is shown in Figure 5.1. In the graph, the x-axis represents the number of epochs and the y-axis represents the RMSE loss value. We can see that as number of epochs increases the RMSE loss value decreases. One epoch means when the entire
CHAPTER 5. RESULTS AND DISCUSSIONS

dataset is passed forward and backward through the neural network. After a certain number of epochs, in this case around the 112th epoch, both the training loss and validation loss start to saturate. After saturation of these curves, training can be stopped. Seeing these curves we can understand if the model is trained properly without overfitting or underfitting. The concept of overfitting and underfitting is explained clearly in chapter 4. If both the curves saturate without diverging then means the model is trained properly. The RMSE loss value measured on the test set is 12.3268. These 11.6, 12, 12.32 RMSE values are measured without alignment with the original input. After alignment, the loss value is expected to reduce. For alignment, the Kabsch Procrustes algorithm is used. So, after applying the Kabsch algorithm as a post-processing step the loss value measured on training, validation and test set are 3.9091, 3.9479, 4.0892 respectively. An example prediction of this method is shown in Figures 5.2 and 5.3. This method is 33.55 percent better than the baseline model.

![Training / Validation Loss](image)

Figure 5.1: Training Loss vs Validation loss for Head with Landmarks as input

![Figure 5.2: Original output](image)

![Figure 5.3: Predicted Output](image)
Head with Landmarks and AGWH as input

For this experiment, ANN was trained with age, gender, weight and height (AGWH) along with 130 landmarks as inputs and 144153 vertices as outputs. Hence, the model was trained with a total of 134 neurons in the input layer, 144153 neurons in the output layer and 500 hidden neurons in the hidden layer. The training and validation loss obtained after 200 epochs are 9.2 and 10.5 respectively. The resulting chart of training and validation loss is shown in Figure 5.4. The RMSE loss value measured on the test set is 10.5421. All of these losses are measured without alignment with the original output mesh. In order to align the outputs, we used the Kabsch Procrustes algorithm as a post-processing step. The loss values measured on training, validation and a test set after post-processing are 3.4551, 3.5863, 3.6229 respectively. An example prediction of this method is shown in Figures 5.5 and 5.6. Comparing with the test loss values with the previous method (head with landmarks as input), this method is 11.27 percent better.

![Figure 5.4: Training Loss vs Validation loss for Head with Landmarks and AGWH as input](image)

![Figure 5.5: Original output](image)

![Figure 5.6: Predicted Output](image)
CHAPTER 5. RESULTS AND DISCUSSIONS

Low Resolution Head with Landmarks as input

During this experiment, ANN was trained with 130 landmarks as inputs and 7503 vertices as outputs. The output point clouds were converted to a low-resolution image. Hence, the vertices were reduced from 144153 to 7503 vertices. The model was trained with a total of 130 neurons in the input layer, 7503 neurons in the output layer and 500 hidden neurons in the hidden layer. The training and validation loss obtained after 200 epochs are 12.25 and 13.73 respectively. The resulting chart of training and validation loss is shown in Figure 5.9. The RMSE loss value measured on the test set is 13.59. All these losses are measured without alignment with the original output mesh. To align the outputs, we used the Kabsch Procrustes algorithm as a post-processing step. The loss values measured on training, validation and a test set after post-processing are 4.5379, 4.9715, 4.4474 respectively. An example prediction of this method is shown in Figures 5.7 and 5.8. This method is 8 to 18 percent worse than head and head with AWGH methods.

![Figure 5.7: Original output](image)

![Figure 5.8: Predicted Output](image)

![Figure 5.9: Training Loss vs Validation loss for Low Resolution Head with Landmarks as input](image)
CHAPTER 5. RESULTS AND DISCUSSIONS

Face with Landmarks as input

For this experiment, ANN was trained with 130 landmarks as inputs and 61743 vertices as outputs. A mask file was given to extract only the face image from the whole head. Hence, the vertices were reduced from 144153 to 61743 vertices. Therefore, with a total of 130 neurons in the input layer, 61743 neurons in the output layer and 500 hidden neurons in the hidden layer, the model was trained. The training and validation loss obtained after 200 epochs are 10.82 and 11.5 respectively. The resulting chart of training and validation loss is shown in Figure 5.10. The RMSE loss value measured on the test set is 10.81. All these losses are measured without alignment with the original output mesh. Outputs were aligned using the Kabsch Procrustes algorithm as a post-processing step. The loss values measured on training, validation and a test set after post-processing are 3.4986, 3.5119, 3.6646 respectively. An example prediction of this method is shown in Figures 5.11 and 5.12. This method is 10 to 17.5 percent better than the head method and low resolution head method.

![Figure 5.10: Training Loss vs Validation loss for Face with Landmarks as input](image)

![Figure 5.11: Original output](image)

![Figure 5.12: Predicted Output](image)
CHAPTER 5. RESULTS AND DISCUSSIONS

5.2 Summary Results

This section answers to the research question raised in Chapter 1:

**What is the lowest evaluation metric loss deep learning model can achieve when reconstructing 3D face or head?**

<table>
<thead>
<tr>
<th>Model</th>
<th>Loss on Training set</th>
<th>Loss on Validation set</th>
<th>Loss on Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Head</td>
<td>3.90</td>
<td>3.94</td>
<td>4.08</td>
</tr>
<tr>
<td>Head with AWGH features</td>
<td><strong>3.45</strong></td>
<td><strong>3.58</strong></td>
<td><strong>3.62</strong></td>
</tr>
<tr>
<td>Head (Low Resolution)</td>
<td>4.53</td>
<td>4.97</td>
<td>4.44</td>
</tr>
<tr>
<td>Face</td>
<td>3.49</td>
<td>3.51</td>
<td>3.66</td>
</tr>
<tr>
<td>Philips 2D3D algorithm (Face)</td>
<td>4.56</td>
<td>4.31</td>
<td>4.58</td>
</tr>
</tbody>
</table>

Table 5.1: Post Processing Procrustes Output Table Comparison

In table 5.1, one can find the final loss values after applying post-processing step (i.e, Kabsch Procrustes algorithm). The deep learning output model is not aligned with the original output face. Hence the loss values would be little bigger. This post-processing step aligns the predicted output with the original output face. Then the loss value is measured which in turn gives lower RMSE loss value. The head with AWGH (age, gender, weight, height) model gives the lowest loss values after applying post processing step. Quantitatively speaking, head with AWGH model is 41.04 percent better than the baseline model whose RMSE loss value was 6.14. It is also 20.96 percent better than Philips state of the art 2D to 3D reconstruction algorithm. Overall it is 1.09 to 41.04 percent better compared to all other models.

Since high resolution model itself achieves good results there is no reason to go for low resolution model.
Chapter 6

Conclusions

In this thesis, the performance of the ANN architectures in reconstructing the 3D face or head was investigated. The experiments included addition of more features, scaling of data and with different configurations of ANN. The metric used to evaluate the architectures was Root Mean Square Error (RMSE).

The real dataset was collected from thousands of people who have different ages, height, weight, gender and under different environmental conditions. From the results achieved in this thesis, the following conclusions can be done:

- A good 3D reconstruction of facial image from 2D image was achieved by just giving facial landmarks as input to the deep learning model.
- The addition of features like age, gender, height, weight increased the performance of the reconstructed output.
- Kabsch algorithm was used as a post-processing step in order to obtain the predicted head or face aligned to the real one. In this way, we can measure the RMSE in a fair way and get a value that really shows the error between expected and predicted 3D models.
- A wide range of experiments were performed on deep learning architecture to reduce the evaluation metric loss value and have a good model and this was achieved as shown in the thesis.

6.1 Future Work

The results generated in this thesis are very much useful for Philips to add them to their project framework pipeline for pre surgery planning application. The 3D data was directly predicted with landmarks and age, gender, weight, height (AGWH) as features. It would be interesting to predict directly from images to 3D data without landmarks using Convolution neural networks.

Another method to explore is to use UV mapping [15] which is quite new in this field. 3D data will be converted to a 2D texture which is called UV maps. So a neural network can be trained to learn a mapping from images to UV maps. After the prediction of UV maps, they can be converted back to 3D data. It would be fascinating to compare the RMSE metric value generated from the above two techniques with our neural network and Philips 2D to 3D algorithm.
Bibliography


[14] Pia Deitersen, Kwang Kim, Wolf Kienzle, Bernhard Schlkopf, and Volker Blanz. Automatic 3d face reconstruction from single images or video. pages 1–8, 09 2008. 11


[17] Riza Alp Güler, George Trigeorgis, Epameinondas Antonakos, Patrick Snape, Stéfanos Za-
feiriou, and Iasonas Kokkinos. Densereg: Fully convolutional dense shape regression in-the-
wild. CoRR, abs/1612.01202, 2016. 12

by reducing internal covariate shift. CoRR, abs/1502.03167, 2015. 9

[19] Aaron S Jackson, Adrian Bulat, Vasileios Argyriou, and Georgios Tzimiropoulos. Large pose
3d face reconstruction from a single image via direct volumetric cnn regression. International
Conference on Computer Vision, 2017. 12

algorithm. 20

[21] Stéphane Lathuilière, Pablo Mesejo, Xavier Alameda-Pineda, and Radu Horaud. A comprehen-

444, 5 2015. 8

[23] Chen-Hsuan Lin, Chen Kong, and Simon Lucey. Learning efficient point cloud generation for

2012. ix, 5

A Logical Calculus of the Ideas Immanent in Nervous Activity, pages 15–27. MIT Press,
Cambridge, MA, USA, 1988. 6

component_analysis. 11

[27] Marcel Piotraschke and Volker Blanz. Automated 3d face reconstruction from multiple images
using quality measures. pages 3418–3427, 06 2016. 11


ations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations

3d-learning-its-time-to-teach-in-voxels-instead-of-pixels-d644df659328,
2019. ix, 3

[31] Anh Tuan Tran, Tal Hassner, Iacopo Masi, and Gerard Medioni. Regressing robust and dis-
criminative 3d morphable models with a very deep neural network. In The IEEE Conference
on Computer Vision and Pattern Recognition (CVPR), July 2017. 12


[33] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang.
Pixel2mesh: Generating 3d mesh models from single RGB images. CoRR, abs/1804.01654,
2018. 13

[34] Ronald Yu, Shunsuke Saito, Haixiang Li, Duygu Ceylan, and Hao Li. Learning dense facial