Department of Informatics

Technische Universität München

Master´s Thesis in Informatics

Face Recognition with a 3D Camera On an Embedded Processor

Abdul Hannan Khan

Department of Informatics

Technische Universität München

Master's Thesis in Informatics

Face Recognition with a 3D Camera On an Embedded Processor

Gesichtserkennung mit einer 3D-Kamera auf einem eingebetteten Prozessor

Author:	Abdul Hannan Khan
Supervisor:	Prof. Dr. Daniel Cremers
Advisor:	Nikolaus Demmel
Advisor:	Jens Harnisch
Submission Date:	November 15th, 2019

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

November 15th, 2019 Abdul Hannan Khan

Acknowledgments

I am thankful to my parent who supported me in my masters financially and morally. I am also thankful to Jens Harnisch and Nikolaus Demmel for guiding me through my master thesis. Furthermore, I am thankful to my friends who made this stay in foreign country feel like home.

"Don't be satisfied with stories, how things have gone with others. Unfold your own myth"

- Rumi, The Essential Rumi

Abstract

In the recent years there have been significant advances in 2D face recognition by using deep neural networks. One of the potential next steps is to develop optimized 3D facial recognition. Shifting from 2D to 3D increases complexity of the problem by adding another dimension to data, making possible solutions more resource hungry. In this thesis we investigate different depth camera based facial recognition techniques and test their performance by deploying them on an embedded processor. We focus on applications for embedded systems and use a small low-resolution time of flight (ToF) camera with Raspberry Pi to keep overall system portable and compact. The first approach involves feature enrichment using an auto encoder and uses a Support Vector Machine (SVM) to train a classifier on these encoded features. The second and third approach are based on two different variant of Iterative Closest Point (ICP) along with a face detector and a facial landmark detector. In order to evaluate the performance of the developed techniques, we created a dataset of 20 identities captured with a low-resolution depth camera and use it for both training and testing. The presented results show that the ICP based techniques perform better in terms of accuracy and run several times faster than SVM based classifier. We achieved almost real-time performance on Raspberry Pi for face recognition.

Contents

Part I.

Introduction and Background Theory

1. Introduction

2D face recognition has improved significantly in recent years. The improvement was mainly based on the success of deep neural networks combined with tremendous computational power of GPUs and large-scale data. Following the success, one of the potential next steps is to develop 3D facial recognition and deploy it on an embedded processor. 2D face recognition is prone to image attacks, and there is no potential algorithm based on single image which can detect these image attacks. 3D face recognition provides a solution against spoof by utilizing the geometry of the scene. Further, the constraint that the method should be able to run on an embedded processor, makes sure that the method is computationally optimal. However, utilizing 3D information drastically increases computational complexity and Single Board Computers (SBCs) with an embedded processors have limited computational resources.

Deep Neural Networks are data and resource hungry, they require a good amount of system memory and multiple processing cores. In computers of daily use, these requirements are fulfilled by powerful dedicated GPUs. However, in SBCs the integrated GPUs share limited system memory with the CPU. When the system memory is exhausted, the CPU starts using secondary storage space as the virtual memory. The hard drives used as secondary storage are thousand times slower than the system memory, and as soon as the CPU starts using it, the performance hits the floor due to latency of disk read/write operations.

Furthermore, the SBCs are made with top priorities being portability and small form factor. Small form factor makes SBCs suitable for DIY and IoT projects. If a 3D face recognition method with reliable accuracy can run on an SBC with satisfactory running time, it will be a huge breakthrough in the embedded industry.

1.1. Problem Statement

Facial Recognition has been one of the most researched topics in recent years in the field of Human and Computer Interaction. The goal of a facial recognition algorithm is to create a face model using multiple training samples for each identity, and to use these models to recognize the faces in the samples afterwards. The face recognition problem can be divided into two sub problems. The first is to create face models using image samples and later is to identify the faces in the images using created face models. Identifying the faces seems to be more important step however, it relies on creation of the face models.

In the 3D face recognition domain, the face model creation problem is a well defined problem since 3D scene data is already given and only few processing steps are required to extract models from the scene. In this particular case when 3D scene data is given, the focus is to register multiple faces of a single identity and find suitable weights for different facial features to have an acceptable face model. Further, the 3D data reduces the problem of identifying faces in images to the 3D face registration problem.

The 3D face registration problem is one of point cloud registration problems. In addition to point cloud registration, 3D face registration has prior information of face structure and non-rigid motion. The face structure prior helps in reducing complexity of the problem by opening possibilities of sparse feature registration instead of dense point cloud registration. On the other side, non-rigid face motions or facial expressions indicate that the 3D structure of faces can be misleading sometimes, and hence relative positioning of features has more importance.

For the scope of this dissertation, another layer of complexity is added in the problem by restricting the computational resources to be of an SBC to make it highly portable and optimized. This limits the possibility of using deep neural networks based methods due to their huge data and computational power requirements.

1.2. Related Work

Deep learning, as it has left behind other techniques in many areas of data processing, has also interesting solutions for the face recognition problem. There exists different deep neural network based solutions, some of them take 2D data as input while other operate on 3D data. One of the famous deep learning based face recognition methods is FaceNet proposed by Schroff, Kalenichenko and Philbin (2015). In their research Schroff et al. (2015) used feature encoding in order to increase information to noise ratio [19]. The feature encoding was used to produce concise face embeddings. Schroff et al. (2015) also trained a classifier using triplet loss on the top of these face embeddings to achieve good results [19]. The triplet loss not only makes sure that weights for feature embedding of an identity are same but also ensures that they are distant from other identities [19]. However, the FaceNet was trained on 260 million images with 7.5 million parameters and it takes roughly 1.6 billion flops for one forward pass [19].

Face recognition problem is also addressed using classical methods of computer vision, most of them use the point cloud registration for this purpose. Also, there exist many techniques to handle the point cloud registration, one of which is the Iterative Closest Point (ICP) as used by Min, Choi, Medioni and Dugelay (2012) [13]. The ICP takes two point clouds and tries to find a transform which register first point cloud to second by finding correspondences and minimizing the distances between them. The closed form solution of ICP exists if the correspondences are known but one of the major challenges of ICP is its dependence on good initialization.

Before ICP can be applied on the face models, they must be created out of raw scene data. The DLib-ml machine learning toolkit introduced by Davis (2009) contains a great face detector for raw images. The DLib-ml face detector yields a 2D bounding box for each face in the image. The 2D bounding box based cutout, when projected to 3D, can contain 3D points other than of face and also it can miss some of them as well. In their research, Mian, Bennamoun and Owens (2006) proposed interesting method to improve raw and noisy face cutout [12]. Mian et al. (2006) used coarse to fine approach to find nose in face point cloud and once located, it is used to improve the face cutout by drawing a sphere around nose. The points within the sphere are considered valid while other points are neglected [12].

There are three ways the ICP can be used for facial recognition i.e. one-to-all or direct ICP based facial recognition, indirect ICP involving intermediate model formation steps and a hybrid method as proposed by Min et al. [13]. Direct face recognition includes matching probe face to all of the gallery faces and returning the most matched face with identity. This problem can also be visualized as nearest neighbor search where distance two models is distance between their corresponding face features or points. This approach is known to be accurate, but it is slow since for n faces in gallery registration process has to be done n times [13]. In terms of time bounds, running time of the direct ICP based facial recognition becomes $O(n \times T(p))$ for one query, where $T(p)$ is runtime of ICP for probe face p. Upon generalizing we get asymptotic bound $O(nm \times T(p))$ where m is number of query faces.

Indirect ICP based method for facial recognition mostly have an intermediate step which is registration to a face model or atlas usually Average Face Model (AFM) [13]. Registration to an Intrinsic Coordinate System (ICS) also falls in this category. All these indirect methods only require ICP to run once and hence are faster than one-to-all ICP method. However, in order to obtain performance with respect to time these methods compromise on accuracy [13].

Hybrid technique proposed by Min et al. (2015) aimed to achieve best of both direct and indirect registration technique by introducing multiple intermediate face models or canonical faces. These canonical faces are chosen from gallery faces [13]. All the gallery faces are registered to all canonical faces and on query, the probe face is only registered to these canonical faces. The probe face is then matched to the gallery faces based on their registration results. In this way hybrid technique achieves better running time which has asymptotic bound of $O(c \times T(p))$ where c is number of canonical faces and $T(p)$ represents running time of ICP for probe face$p$. Here it can be seen that by taking $c = n$ hybrid method becomes one-to-all ICP method and also by taking $c = 1$ it becomes indirect ICP. This technique is powerful since it has flexibility to adjust compromise between performance in terms of running time and accuracy.

Another yet interesting work is presented by Park, Zhou and Koltun (2017). In their research Park et al. (2017) propose combining photometric and geometric objective together in a manner which is normally used in RGB-D image alignment to add photometric information in ICP and improve convergence. RGB-D image alignment proposed by Steinbruecker, Sturm and Cremers (2011) [20] provides promising results for 3D reconstruction. Feng and Zhao (2018) in their research present novel depth-to-depth face recognition method using Deep Convolutional Neural Networks (DCNN) [5]. Feng and Zhao (2018) use two different networks to obtain normalize 3D reconstruction and feature encoding respectively. And, on the top of extracted feature encoding they use softmax classifier to classify faces into identities.

The datasets frequently used by researchers for face recognition using the depth images are dataset of 3D images created for Face Recognition Grand Challenge (FRGC) and Lock3DFace dataset created by Zhang, Huang and Sun (2016) [15, 21]. Both datasets contain colored and depth images, having multiple sample per identity with different environment conditions and facial expression.

1.3. Challenges

1.3.1. Embedded Processor and Computing Power limitations

Major objective of this dissertation is to deploy a solution on an Embedded processor. For choice of the embedded processor, Raspberry Pi was chosen, since it is easily available, economical and most power full among Single Board Computers (SBCs) of its form factor. The latest model of Raspberry Pi at the time of this dissertation is Raspberry Pi 3 B+. This version of Raspberry Pi has ARM Cortex-A5 Quad-Core-CPU with core clocks frequency up to 1.4 GHz. The Random Access Memory is 1 Gigabyte, which is shared between CPU and GPU. For secondary memory a USB drive or external SD cards is used. Raspberry Pi on one side is portable and economical but it comes with a compromise on computing power. Raspberry Pi officially supports the Raspbian Stretch Operating System which is Debian Stretch based; the third-party operating systems available for Raspberry Pi include Windows 10 IoT, Lubuntu and Ubuntu Mate. While it support other operating systems as well but with significantly slow performance.

Latest version of Raspbain OS available at the time of this dissertation is Raspbian 9 which ships with Linux kernel version 4.14. Official website allows to choose between 3 different images including Rasbian Stretch Lite, Raspbian Stretch Desktop and Raspbian Streatch Desktop with recommended software. Raspbian Stretch Desktop was chosen for this dissertation, since it contains Desktop environment to test things and extra software are not included to spare disk space for other package and compilation. Raspbian Stretch 9 has limited software repositories and it leaves the user with no other option but to compile required repositories for themselves.

Compilation on Raspberry Pi is a time taking process, due to RAM limited to 1 Gigabyte, which is shared between CPU and GPU. It runs out of memory quickly, relying on slow swap. As soon as it hits swap, due to low Read/Write speed of SD card, performance hits the floor. The compilation of OpenCV took around 6-8 hours when environment setup was done. Installing Python packages requires even more patience because if a dependency is missing or a package is incompatible it throws an exception after completely installing other dependencies. In most cases after 2-3 hours of dependencies installing user runs into an exception. The solution to avoid these issues is to follow installation guides alone and install dependencies beforehand to save time. The best and quick solution is to use APT package handling utility *apt-get* to install python packages instead of *pip* as it installs prebuilt packages as well as system dependencies saving compilation time.

1.4. List of Contribution

Following is the list of contributions made by this dissertation:

- Search for suitable face recognition algorithms which can run on low-resolution depth and intensity images.
- Training of SVM for face recognition using the FaceNet embeddings.
- Search for multiple preprocessing steps, which can crop out faces optimally from the point clouds and their implementation.
- Development and implementation of 3D face recognition algorithms by combining Face Detector, Facial Landmark Detector and ICP based registration methods.
- Comparison of ICP followed by 2.5D projection and Colored ICP.
- Development and implementation of nearest neighbour search algorithm for faces to avoid brute force search.
- Optimization of algorithms to make them run on almost real-time on Raspberry Pi.
- Compilation of prerequisite Python and system libraries for Raspberry Pi, which does not have default installation packages.
- Deployment, testing and insights of algorithms on Raspberry Pi.
- Collection of small datasets of low resolution depth and intensity images.

2. Background

In this chapter some basic concepts, which are necessary in order to understand the methods, results and the conclusions are explained. Reader with knowledge of the domain can skip this section.

2.1. Perspective Projection

Perspective projection is a linear image formation model where a 3D scene is projected on the image plane of the camera, the depth information is lost, and closer objects appear to be larger. In perspective projection, the image of parallel lines intersect at a point called vanishing point (See Fig. 2.1). Compared to other projection models, the perspective projection looks most realistic since, lens in the human eye as well as the lenses in the digital cameras use same projection model. Equation 2.1 and 2.2 show the projection and the inverse projection of perspective projection model.

$$\pi(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}) = \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.1}$$

$$\pi^{-1}(\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}) = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{2.2}$$

where:

f_x = horizontal focal length of camera
f_y = vertical focal length of camera
c_x = horizontal center of camera in pixel coordinates
c_y = vertical center of camera in pixel coordinates
λ = depth and scale of point

2.2. Rigid Body Transformation

The Rigid body transformation is a subset of 3D transformations which preserves the scale and the shape of the body, hence called rigid body transformation. The rigid body transformation is highly used in the field of multi-view geometry. It is composed of a rotation

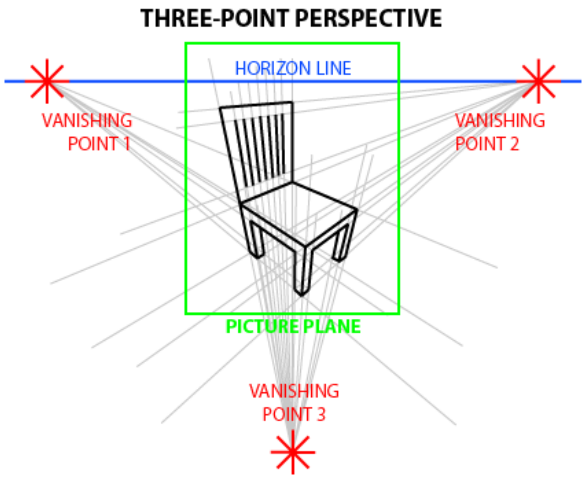

Figure 2.1.: Example of vanishing points[1]

and a translation part where rotation matrix $R \in \mathbb{R}^{3\times3}$ and translation vector $T \in \mathbb{R}^3$. In order to perform the rigid body transformation on a point, first the rotation matrix is multiplied with the point and then the translation vector is added as shown in Equation. 2.3.

$$g_t(x) = Rx + T \tag{2.3}$$

It is important to note that the rotation matrix R is orthogonal, and determinant of R is +1. Being orthogonal matrix means that if the matrix is multiplied with a vector, the scale of the vector is preserved and $R^{-1} = R^T$ hence, $R^T R = I$. These properties show that the rotation matrix R belongs to the special orthogonal group or the Lie group SO(3) (See Equation. 2.4). Using the homogeneous coordinates both rotation and the translation can be combined in to one matrix (See Equation. 2.5).

$$SO(3) = \{R \in \mathbb{R}^{3\times3} | R^T R = I, det(R) = +1\} \tag{2.4}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.5}$$

2.3. RGB-D Image Alignment

Given two RGB-D images, the goal of the RGB-D image alignment is to find optimal pose ξ which is defined as relative transformation between first and second camera. Here, first and second camera doesn't mean there must be a different camera for every single image, but instead cameras represent the position of camera when images were taken, making

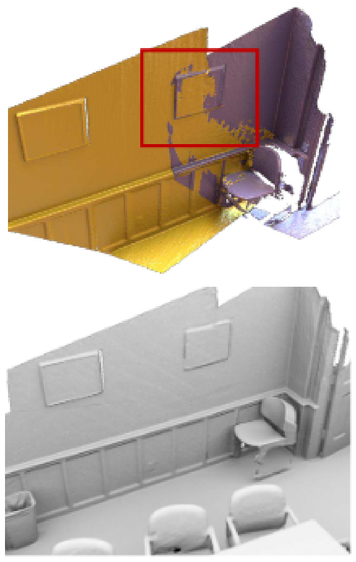

Figure 2.2.: a) 2 different scene reconstructed from RGB-D Images. b) Resultant reconstructed scene from RGB-D Image Alignment [14]

it one camera for each image. ξ is not just relative transformation of camera but it can also be seen as transformation between captured scenes and thus can be used to combine information captured by different cameras into one big scene. 3D Reconstruction problem using RGB-D cameras can be reduced to RGB-D image alignment problem on consecutive pairs of images.

The RGB-D image is composed of colored as well as depth information for each pixel in image. Unlike RGB images RGB-D image itself is enough for reconstructing captured scene. In order to do so, the RGB values are unprojected using camera intrinsic matrix K, which is defined as

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

to obtain image on focal plane of camera. On image plane each pixel corresponds to a 3D vector with values in focal units where depth value for each vector is 1 focal unit. In order to reconstruct the scene, the vectors are multiplied with their corresponding depths.

Since, RGB-D image contains sufficient information of the scene from one particular perspective, multiple images are taken from different perspectives and are later combined using RGB-D image alignment to obtain a better reconstructed scene (See Fig. 2.2).

2.4. Iterative Closest Point

The process of minimizing the distance between two point clouds is called point cloud registration. One of the famous methods of the point cloud registration is Iterative Closest Point or ICP [3, 2, 4]. ICP is used in 3D reconstruction and localization of robot. Given two point clouds it iteratively finds best rigid body transformation which minimizes the distance between corresponding points. In each iteration, ICP finds point correspondences between two point clouds and based on the found correspondences it finds optimal rigid body transformation which minimizes the distance between corresponding points. In most cases, the closest point to a point in target point cloud is considered its corresponding point. The method keeps running unless maximum number of the iterations is reached or the resultant error between correspondences is smaller than some threshold (See Source Code. 2.1).

```
def find_point_correspondences(P: list, Q: list):
    correspondences = dict()
    for p in P:
        q = closest_point_to(p, Q)
        correspondences[p] = q
    return correspondences

def icp(P: list, Q: list):
    $C_p$ = center_of_mass(P)
    $C_q$ = center_of_mass(Q)

    $t = C_q - C_p$
    P.translate(t)

    error = $\infty$
    for i in range(max_iterations):
        if error $\leq \epsilon$:
            break
        correspondences = find_point_correspondences(P, Q)
        transformation T = $\text{argmin}_T \sum_i \|q_i - T(p_i)\|^2$
        P.transform(T)
        error = euclidean_dist(correspondences)
```

Source Code 2.1.: Iterative Closest Point Algorithm

$$T = \underset{T}{\operatorname{argmin}} \sum_i \|q_i - T(p_i)\|^2 \tag{2.7}$$

In case when correspondences are known ICP has a close form solution as they reduce the problem to estimation of transformation T which minimizes Equation. 2.7. To do so, first both point clouds are mean normalized i.e.

$$\begin{aligned} \tilde{P} &= \{p_i - \mu_p\} = \{\tilde{p}_i\} \\ \tilde{Q} &= \{q_i - \mu_q\} = \{\tilde{q}_i\} \end{aligned} \tag{2.8}$$

$$W = \sum_i \tilde{p}\tilde{q}^T \tag{2.9}$$

Afterwards, using normalized point clouds, W is calculated given by Equation. 2.9 and if the $rank(W)$ is 3, Single Value Decomposition SVD of W is defined as

$$W = U \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} V^T \tag{2.10}$$

where $U, V \in \mathbf{R}^{3\times 3}$ and $\sigma_1 \geq \sigma_2 \geq \sigma_3$. Rotation matrix R and translation T are calculated using Equation. 2.11.

$$\begin{aligned} R &= UV^T \\ T &= \mu_p - R\mu_q \end{aligned} \tag{2.11}$$

There are different variants of ICP available today, they differ from each other based on different methods for point cloud sampling, correspondence weighting, different matching algorithms and outlier handling techniques. Which one is to be used, depends on scenario.

2.5. Hoteling Transform

Karhunen-Loeve Transform (KLT), Eigenvector Transform or Hoteling Transform is used for the rotation normalization/alignment with principal axes of 2D or 3D data [12, 16]. Let $P \in \mathbf{R}^{3\times n}$ a matrix with x, y and z coordinates of pointcloud (See Equation 2.12).

$$P = \begin{bmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & y_n \\ z_1 & z_2 & \dots & z_n \end{bmatrix} \tag{2.12}$$

The mean vector $m \in \mathbf{R}^3$ and covariance matrix $C \in \mathbf{R}^{3\times3}$ is given by Equation. 2.13 and 2.14 respectively.

$$m = \frac{1}{N}\sum_{k=1}^{n} P_k \tag{2.13}$$

$$C = \frac{1}{N}\sum_{k=1}^{n} P_k P_k^T - mm^T \tag{2.14}$$

Let $V \in \mathbf{R}^{3\times3}$ be the matrix of eigen vectors of C given by Equation. 2.15.

$$CV = DV \tag{2.15}$$

Where D is diagonal matrix with eigen values as diagonal entries in descending order i.e. $\lambda_1 \geq \lambda_2 \geq \lambda_3$. P can be aligned with principal axes using Equation. 2.16 known as Hoteling Transform [12, 16].

$$\tilde{P} = V(P - m) \tag{2.16}$$

Hoteling Transform can be used along with sampling and hole filling to align the point clouds with their principal axes.

2.6. Support Vector Machine

In Machine Learning, Support Vector Machines or SVMs or Support Vector Networks are supervised learning algorithms which are associated with classification as well as regression problems. The SVMs classifiers with few exceptions are non-probabilistic classifiers capable of dealing with both linear and non-linear classification.

To run an SVM for labeled data, first feature vectors are created and preprocessed using different encoding techniques e.g. Auto Encoder Network. Preprocessing is necessary for dimensionality reduction and information enrichment. These processed vectors along with the labels are then fed into an SVM classifier for training. Once the training is finished with suitably low error the trained network can be used for classification of unlabeled data. It is important to note that before querying, the unlabeled data must be preprocessed in a similar way as training data was preprocessed.

Part II.

Methods and Implementation

3. Methods

In this section, different experimental methods are discussed in detail. 2D based techniques process only gray scale images leaving depth data unused while 3D based techniques utilize both gray scale and depth data to predict facial identities. However, 2D based techniques are lighter in terms of memory and computational power which is desirable for solutions which must run on an embedded processor.

3.1. Dataset

Since, specific ToF camera is used, with no standard dataset there is a need of creating a dataset just for the scope of this dissertation. The dataset should contain 3D face models of 6-10 people with 2-5 samples each. Moreover, the algorithms applied for the face recognition are not based on supervised learning so whole dataset can be used for validation. Furthermore, ToF cameras are passive cameras with strong enough illumination source to light up the subject in scene. Due to direct light of sensor, shading artifacts from other sources of light in scene are negligible for the subject close enough to the camera with the assumption that the scene is indoor. This negates the need of differently illuminated samples per identity hence, reducing number of samples required per identity.

3.1.1. Model Format

Each face model is saved as numpy file in .npy format [7]. The face model contains cropped face points along with their intensities as well as their corresponding indices in pixel coordinates, which are concatenated in column-wise fashion as

$$F = \begin{bmatrix} P & I_p^T & i_p^T \end{bmatrix} \tag{3.1}$$

where:

P = 3D points $\in \mathrm{R}^{N\times 3}$
I_p = intensity values $\in \mathrm{R}^{N\times 1}$
i_p = indices in pixel coordinates $\in \mathrm{R}^{N\times 1}$
N = number of points

The resultant numpy array is of form $F \in \mathrm{R}^{N\times 5}$ which is then saved as pickled object with .npy extension [7].

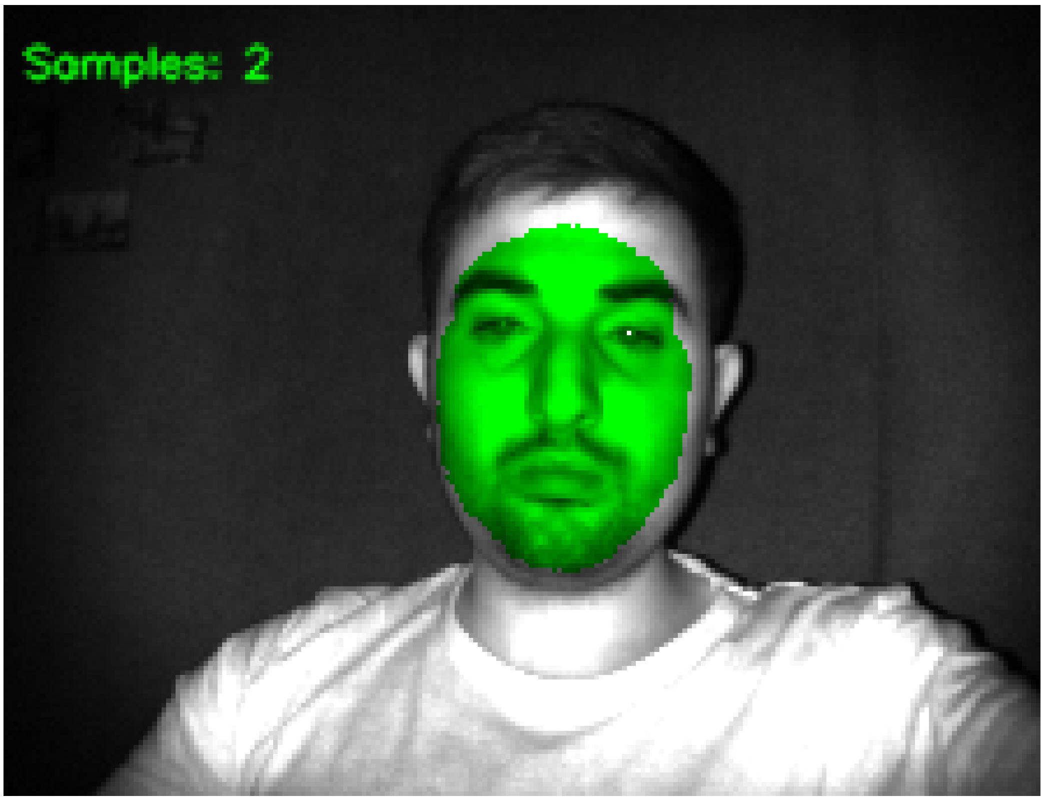

Figure 3.1.: Capturing Samples for Database

3.1.2. Capturing Models

For the purpose of collection of data, the user is asked to stand between 0.5, 1.5m distance away from the camera with their face directed towards the camera and hence mostly visible. Additional condition which can be imposed for better is that there should not be any direct light on the user's face. Since, direct light on face introduces intensity variantions which in turn introduces huge noise in photometric objective.

Using the annotated image with facial region as visual guide, as shown in Figure. 3.1, the 3D point cloud, the depth image and the gray image are captured. Normally around 10-15 samples are taken, out of which 5 samples are randomly taken as the test models and from the rest, 5 best models which represent other samples well are taken as training models. The selection of best models is based on ICP registration, each model is registered against rest of models and then models are sorted in descending order of average registration score. This setup helps in quick search for best match which will be discussed later in 3.5.2. In addition to sorting, all models under one identity are registered to model with best average registration score in order to make it canonical face/model of other models [13]. Although, the intensity and the depth images are saved in the database they are not needed by either Colored ICP Registration or 2.5D Registration. However, the depth and the intensity images are used in the 2D Face Recognition using FaceNet.

3.2. Recognition Classes and Criteria

The output of the face recognition algorithm is confidence/probabilities against each label. There are multiple ways to interpret this information and classify raw probabilities into different classes such as recognized or fake. In this section, the criterion for the different classes is briefly described.

3.2.1. Confidence from ICP Based Algorithms

The output of iterative closest point algorithm is two measures, namely fitness and RMSE (Root Mean Squared Error). The fitness is the ratio of the number of points which have correspondences over total number of points. Since, not all points in source point cloud can always be matched to points in destination point cloud, the fitness ratio is almost always less than one. The RMSE is the root mean squared error based on mean of distance measure between transformed source points and their corresponding points in the target point cloud. In order to find correspondence for one point in source point cloud, ICP looks for the closest point in destination point cloud. ICP allows a configuration where search for closest point in destination point cloud can be constrained hence, enabling the definition of maximum possible distance between corresponding points. This maximum possible distance, when set according to resolution and the precision of depth values from the camera, allows fitness ratio to be considered as percentage match between source and destination point clouds. Further, the resultant percentage match can be used as match confidence for ICP based recognition methods.

3.2.2. Recognized or Uncertain

A result of face recognition is classified as ″Recognized″ when there is only one label with confidence above some threshold. The cases where there are multiple labels above threshold, the result is classified as ″Uncertain″.

3.2.3. Verified or Fake

Once a face is recognized, it is passed to the spoof detector and if spoof detection is positive, the result is classified as ″Fake″, otherwise ″Verified″. In case of ICP based methods, spoof detector and face recognizer are not separate. Spoof attacks like pictures or video are easily caught by ICP, since it registers 3D geometry and geometry to pictures doesn't match geometry of face. Also, if the face is printed on a curved surface, the actual geometry of different part of the face, for example nose, doesn't match and results in an ″Unknown″

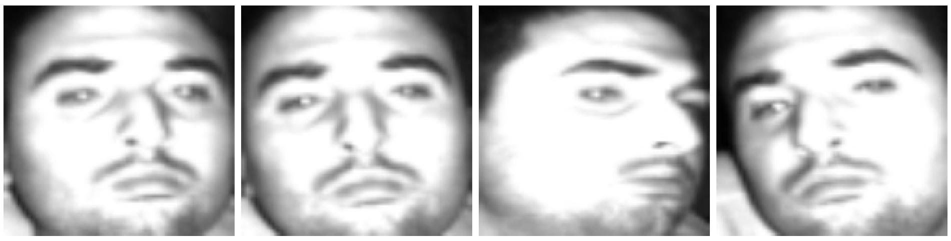

Figure 3.2.: 2D intensity image samples cropped using MTCNN face detector[22]

label. Mask attacks however, can depict exact geometry as original but the reflectance of any material used for mask can barely match the reflectance of skin.

3.2.4. Unknown

If there is no label with confidence above the defined threshold, the result is classified as "Unknown". It happens in the case where the face is not registered in the database. It also happens when the face is not fully visible to the camera or it is a spoof attack. The 2D Face Recognition using FaceNet can detect spoof explicitly since it has a separate module for the spoof detection but in the case of ICP based methods, it cannot explicitly tell if the "Unknown" is an unregistered face or a spoof attack.

3.3. 2D Face Recognition using FaceNet

Thanks to deep neural networks, 2D face recognition provides promising results however, the deep neural networks are data and resource hungry. In 2D Face Recognition using FaceNet, FaceNet presented by Schroff et al. (2015) is used as a feature encoder to enrich raw face data and increase signal to noise ratio [19]. The face embedding obtained from FaceNet are then used to train an SVM classifier.

3.3.1. FaceNet

In their research Schroff et al. (2015) pointed out that with the better encoding of face model accuracy of face recognition and verification can be improved [19]. Schroff et al. (2015) used deep convolutional network to obtain features in euclidean space where lesser distance between models represent face similarity [19]. Schroff et al. (2015) introduced triplet loss in order to improve recognition results [19]. This representation when used for face recognition achieves the accuracy of 99.63% on Labeled Faces in the Wild (LFW) dataset[19, 6].

3.3.2. Spoof Detector with 2D Face Recognition using FaceNet

2D Face Recognition alone is not enough since, most of the face recognition algorithms are optimized to identify the face in image and it is hard to tell whether the image contains another image of a face. In order to cope with this problem, spoof detector is introduced to verify if the image contains real face or just another image. The spoof detector runs along with face recognition, once an identity is established the spoof detector gives confidence about it being real (See Fig. 3.3). The spoof detector is pre trained and does not need any further training specific to face identities.

3.3.3. Training

For 2D Face Recognition, 200 samples of intensity images per identity are captured and processed by MTCNN face detector to extract the region of interest (See Fig. 3.2)[22]. Further, these region of interest patches are resized to 160x160 images and processed through facenet to obtain face embeddings [19]. Once, all embeddings are obtained and SVM classifier is trained with names as labels to recognize the faces.

Training SVM requires computational, when tried on raspberry pi 3 with 1 Giga Byte of RAM, the process runs out of system memory limiting the training only for computers with greater resources.

3.3.4. Classification

For classification, face is cropped out of intensity image using MTCNN and processed through FaceNet to obtain embeddings [22, 19]. These embeddings are then fed forward to the SVM classifier to obtain a label. If the confidence of the classifier is below 0.60 the identity is considered unknown. Instead of running classifier and spoof detector on every frame, once an identity is established, it is tracked using MTCNN tracker and new identity is established only after the first one is lost by tracker [22].

3.3.5. Verification

For verification, raw intensity and depth images are provided to spoof detector without cropping out any part. Spoof detector returns score in interval [0-1]; the score represents confidence of spoof detector for face being real. Due to separate spoof detection, the method can explicitly differentiate between ”Unknown” and ”Fake” faces.

3.4. Preprocessing for ICP

A series of preprocessing steps are applied to increase signal to noise ratio in images. All preprocessing steps discussed below are applied after obtaining point cloud from depth image. For the purpose of unprojecting image into point cloud, pinhole camera model

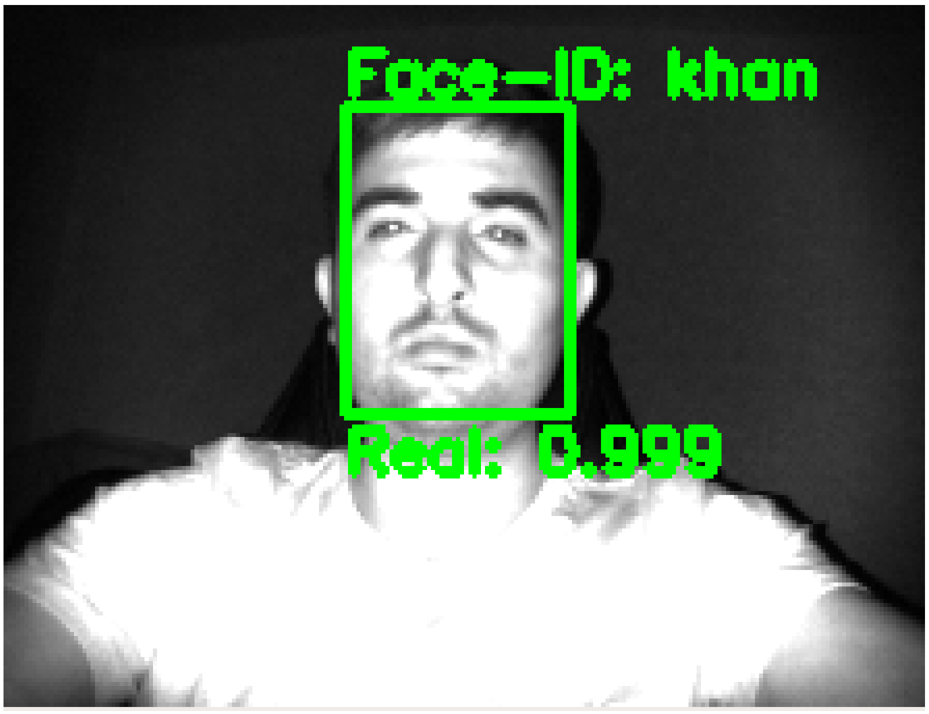

Figure 3.3.: Output of 2D Face Recognition and Verification

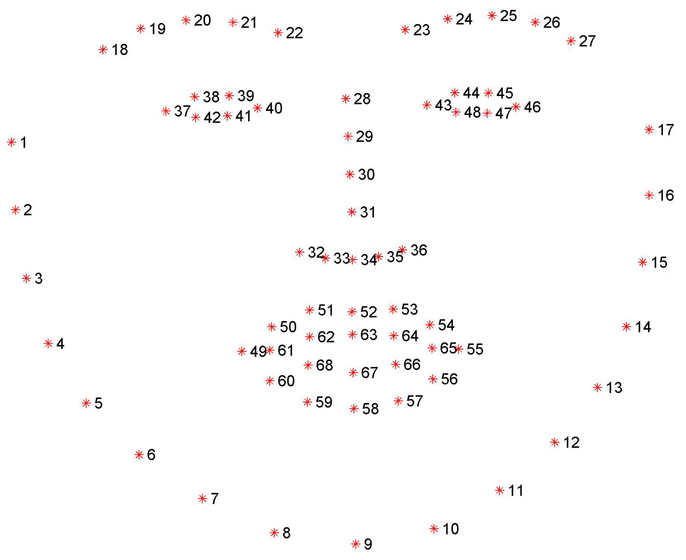

Figure 3.4.: Output of Dlib Shape Predictor [17]

was used due to its simplicity. The camera intrinsics were obtained from API sources which came along with camera as an SDK. The depth and the gray images obtained from the ToF camera were already registered making color assignment to the points, simple and straight forward.

3.4.1. Face Detection and Shape Prediction

The DLib-ml face detector was used to obtained face bounding box from gray image along with DLib-ml shape predictor which is an implementation of face alignment with an ensemble of regression trees, to predict the facial landmarks[8, 10]. The predictor outputs 68 (x, y)-coordinates which map to the facial structures on the face as shown in Fig. 3.4. The corresponding 3D points of facial landmarks are obtained by using corresponding index of the facial landmarks in images since, order stays same while projecting 2D image pixels to 3D point cloud.

$$x_{\mu} = X - \sum_{i=0} x_i \tag{3.2}$$

3.4.2. Mean and Scale Normalization

The unprojected point cloud has non-consistent scale due to varying distance of the person from the camera. In order to normalize the obtained point cloud, points are first mean

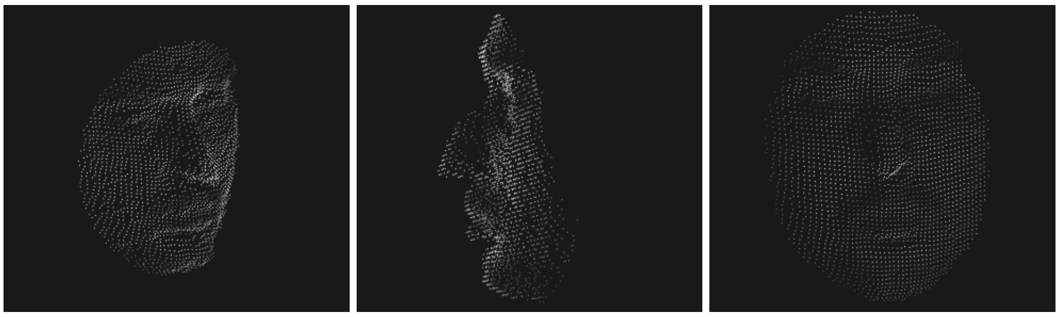

Figure 3.5.: Point Cloud after Extraction of Facial Points

normalized using equation 3.2 and then scale normalized using equation 3.3.

$$X_\gamma = \frac{X}{\frac{1}{|X|}\sum_{i=0}\|X_i\|} \tag{3.3}$$

3.4.3. Facial Points Extraction

As Face Detector only runs on 2D gray scale image, the detection output is not consistent due to varying poses and sometimes includes areas like hair, ears and neck. These areas are undesired because either they're highly dynamic or they're not always part of detected regions. In order to deal with inconsistent detection, the point cloud is filtered again for facial region based on the nose tip. For this purpose, technique proposed in research by Mian et al. (2006), is used, which takes a sphere of fixed radius with nose tip as center and all points which lie within this sphere are considered part of facial points [12]. In their research Mian et al. (2006) apply a facial geometry based coarse to fine approach to detect nose, which in proposed methods is detected using DLib-ml shape detector [8, 10, 12]. Once facial points are extracted mean and scale normalization discussed in section 3.4.2 are applied again. An Example of extracted facial points is shown in Figure. 3.5.

3.4.4. Pose Normalization

The iterative closest point algorithm needs a good initialization in order to avoid local minima. To ensure that, the pose of the faces must be normalized so that before they're presented to the ICP both source and target faces are of almost similar pose hence, reducing the work of ICP and decreasing chances of hitting local minima. There are two techniques which are considered effective for pose normalization in such scenarios, the Hoteling Transform (See section. 2.5) or finding the face plane using facial landmarks. The Eigenvector or Hoteling Transform sometimes results in mirrored point clouds which is undesired for ICP, making it not a viable solution. Also, finding the face plane using facial landmarks didn't work well due to unstable facial landmarks.

3.4.5. Voxel based Down Sampling for Lower Resolution

In voxel base down sampling, a voxel grid is applied to the point cloud so that all points in the points cloud are contained in voxels. The size of the voxels defines the resolution of resultant point cloud, lower the voxel size is higher the resolution will be. Once, voxel grid is applied for each voxel, voxel centroid is selected by checking which point is closer to mean of points contained in voxel. Voxel centroid is given by Equation 3.4. Finally, a point cloud containing only centroids of voxel is created.

$$\zeta = \underset{x_i}{\operatorname{argmin}} \left\{ \left(x_i - \frac{1}{N} \sum_{i=0}^{N} x_i \right)^2 \right\} \tag{3.4}$$

Selecting voxel centroid instead of voxel center is less efficient in terms of running time however, point cloud formed by voxel centroids represents geometry much better than voxel centers [12].

3.5. Colored ICP

Colored ICP is a variant of the ICP algorithm for registration of 3D point clouds introduced in research by Chen and Medioni (1992) and Besl and McKay (1992) [2, 3]. Colored ICP generalizes photometric objectives used in RGB-D image alignment by introducing virtual image on tangent plane of every point to define a photometric objective for point cloud registration [14]. The virtual images on tangent planes provide a local approximation of color variation. Efficient joint photometric and geometric optimization is enabled by integrating resultant photometric objective with geometric objective obtained from virtual images on tangent planes [14]. Combined geometric and photometric objective is given by

$$E(T) = (1 - \lambda)E_I(T) + \lambda E_D(T) \tag{3.5}$$

where:

λ = weight of geometric objective $E_D(T)$ over photometric objective $E_I(T)$, $\lambda \in [0, 1]$.

Open3D [24] implementation of ICP was used which is an implementation of research by Park et al. (2017) [14].

3.5.1. Photo-Geometric ICP

Unlike classical ICP which only uses geometric error, colored ICP tries to find the best transformation which minimizes both photometric and geometric error [14]. Equation 3.5 shows the joint objective function which is minimized in order to get optimal T. λ hyper parameter weighs geometric error over photometric error with range (0,1) where 0 means photometric error only and 1 means geometric error only. The technique is applied in coarse to fine approach since the minimization problem is not fully convex. Although, coarse to fine approach deals with local minima better but it is expensive in terms of computation power.

Instead of dealing with point cloud directly, Park et al. (2017) try to simplify it by casting it into a RGB-D image alignment problem [14]. Given two RGB-D images, the goal of RGB-D image alignment is to find a transformation/pose which registers one image to other [20, 9]. RGB-D image alignment uses both geometric and photometric objectives and hence perform much better than state of the art ICP [20]. The only problem which prevents casting the point cloud registration problem directly into RGB-D image alignment problem is discrete intensity values. However, for photometric objective given by 3.6 intensity information should be continuous. In order to cope with the issue Park el al. (2017) create virtual intensity image on the tangent plane of each point which contain interpolated intensity values based on intensity values of point and its neighbors [14].

$$E(\xi) = \int_{\Omega} (I_2(\tau(\xi, x_i)) - I_1(x_i))^2 \tag{3.6}$$

When point clouds are not pose normalized, traditional ICP gets stuck into local minima. To solve this issue convex relaxation methods are tried and one of those methods is coarse to fine scale pyramids. In this approach, point cloud is sampled with multiple resolution and ICP registration is done for sampled point clouds from coarse to fine resolution. The solution found for low resolution is used to initialize the registration of higher resolution and in this way good initialization which is necessity for ICP is found automatically using coarse to fine scale pyramids.

3.5.2. Nearest Neighbor Search

Face Recognition can also be viewed as nearest neighbor search. At first, query model is registered to first model under each label. It is important to note that the first model is best representative of label since, while adding models to database they were sorted in descending order with respect to average registration score against all other models under same label. Once, the first model of each label is scored, possible candidate labels are filtered out based on how close they are compared to maximum score. Labels with scores greater than 90% of maximum score proved to be good criteria to find out candidate labels. In case where, there is only one candidate label it is straight forward that models under this label match the query model most. However, if there are more than one candidate labels, the query model is again matched with next model under each candidate label and candidate labels are filtered again based on logic stated before. The process repeats until there is only one candidate label remaining or there are no more models for candidate labels (See fig. 3.6).

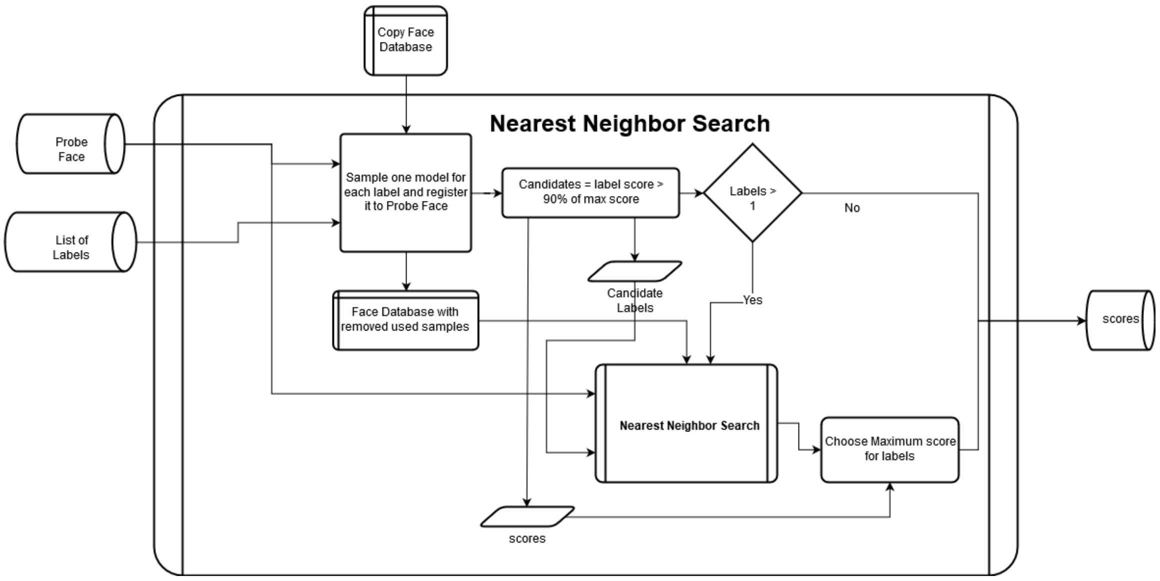
Copy Face Database
Nearest Neighbor Search
Probe Face
List of Labels
Sample one model for each label and register it to Probe Face
Candidates = label score > 90% of max score
Labels > 1
No
Candidate Labels
Yes
Face Database with removed used samples
scores
Nearest Neighbor Search
Choose Maximum score for labels
scores

Figure 3.6.: Nearest Neighbour Search Algorithm for ICP.

3.6. 2.5D projection with ICP

Another approach to efficiently register 3D faces while addressing both geometric and photometric error is presented in research by Min et al. (2012) [13]. Min et al. (2012) argue that 3D face registration using ICP is a time-consuming technique and effects overall running time badly when done on larger scale. They insist that matching a query with all gallery faces (faces in database) is not a viable option when the number of gallery faces is large, so in order to reduce number of registrations they introduce so called canonical faces which can be seen as centroids with a cluster of faces which are registered to it. Instead of registering query to every gallery face, query is only registered to the canonical faces. The registered canonical face with lowest RMSE is picked and gallery faces registered to it are then searched for best match.

3.6.1. Point-to-plane based ICP registration

Point-to-plane ICP introduced by Chen and Medioni (1992) has ICP error metric based on distance between source point and tangent plane of destination point (see Figure 3.8)[3, 11]. If M and M_{opt} are 4×4 3D rigid-body transformation matrices in homogenous coordinates then loss function of Point to Plane ICP becomes

$$M_{opt} = \underset{M}{\operatorname{argmin}} \left\{ \sum_i ((M \cdot s_i - d_i) \cdot n_i)^2 \right\} \tag{3.7}$$

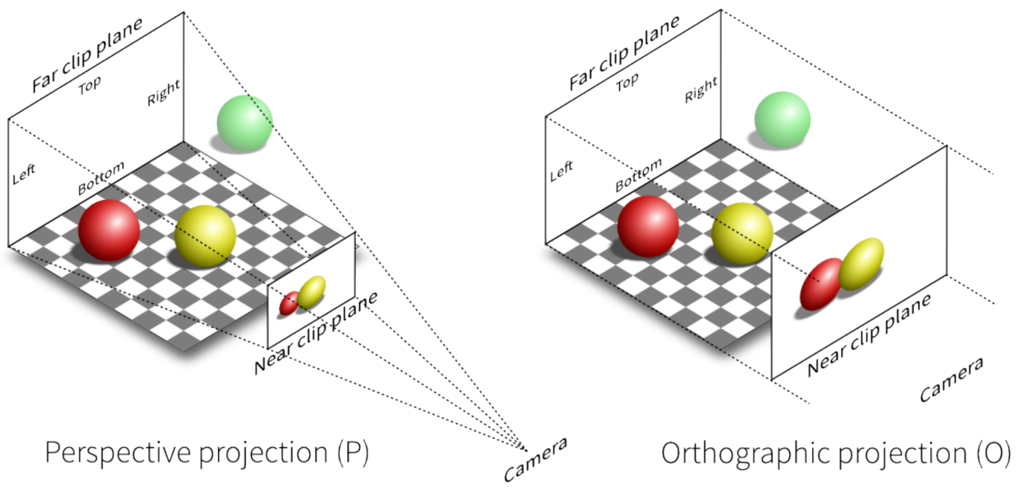

Figure 3.7.: Perspective vs Orthographic Projection [23]

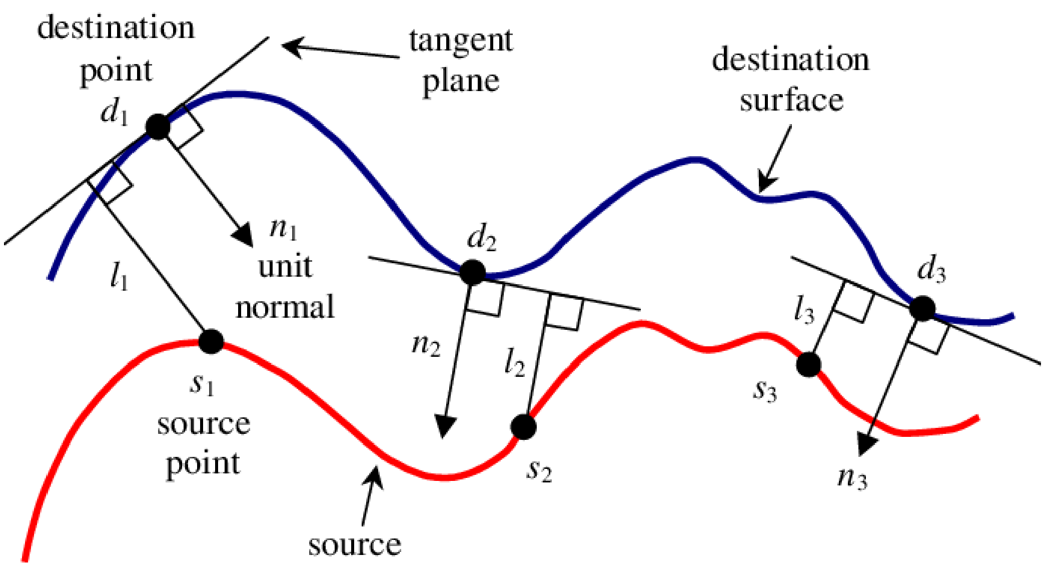

Figure 3.8.: Point-to-plane error between two surfaces. [11]

where:

s = corresponding points from source point cloud
d = corresponding points from destination point cloud
n = normal vector at corresponding points from destination point cloud

Point-to-plane ICP is used due to its fast convergence and since, it is more robust to sensor noise along the tangents of surface[3]. However, point-to-plane ICP can get stuck in local minima since, no steps were taken to ensure good initialization for point-to-plane ICP.

3.6.2. 2.5D projection

2.5D or orthographic projection is widely used in computer games to give an experience of 3D in 2D images. Image formation in human eye is based on perspective projection which is another more commonly used type of projection. Figure 3.7 shows comparison of perspective projection with orthographic projection. Compared to perspective projection, orthographic projection maintains relative scale of objects, there is no vanishing point and parallel lines never intersect, the orthographic projection matrix K is given by

$$K = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bottom} & 0 & -\frac{top+bottom}{top-bottom} \\ 0 & 0 & -\frac{2}{far-near} & -\frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

Once, point clouds are registered via ICP registration as described in 3.6.1, images of their 2.5D projection are taken. The projected images are scaled down to get intensity values in interval [0,1]. These images are then compared with each other in pixel-wise manner to generate root mean square difference (see Equation 3.9), which is then used as difference score or subtracted from 1 to generate score of match.

$$e = \sqrt{\mu(\Delta^2)} \tag{3.9}$$

where:

$\mu(x)$ = mean of x
Δ = pixel-wise image difference

4. Implementation

In this chapter, implementation of the methods, mentioned in chapter. 3 is explained in detail. Only implementation of 2.5D Projection with ICP and Colored ICP and explained below, FaceNet with SVM is discarded in this section because its implementation is straight forward.

4.1. Solution Architecture

The solution architecture is designed as simple as possible. Both algorithms, the 2.5D Projection with ICP and the Colored ICP are represented by different Registration classes, Register25D and RegisterICP respectively. Both classes inherit from the Register base class. The Face class manages loading, storing, processing and manipulation of face models. The logic for registration and classification is handled within Registration classes and preprocessing is done in Face class. Virtual functions were used to keep the consistency in registration classes.

4.2. Programming Language and System Libraries

Python was chosen as programming language for the solution due to its vast package database and quick prototyping possibilities. Another major reason behind choosing python was the fact that certain required libraries like OpenCV can be compiled directly in python packages. The system libraries which were compiled from scratch are OpenCV, VTK and Camera driver modules.

4.3. Python Libraries

The solution is dependent on certain python libraries. All these libraries are essential to run full pipeline of solution. Following is the list of required python libraries:

- Open3D
- DLib
- VTK
- Numpy

- Imutils

4.4. Environment Setup for Ubuntu

In order to run the solution on the Ubuntu Operating System, some system libraries as well as python packages needed to be installed. These libraries and packages were installed using Ubuntu Advance Packaging Tool (APT) and if the libraries and packages were not found in the APT database, they were manually compiled from scratch. OpenCV 4.0 and python 3.6 were used in solution as they were the latest versions available at the point when solution development was started.

4.5. Environment Setup for Raspbian

Running solution on Raspbian is bit more complicated on Raspberry Pi, since most of python packages used were not available in Raspbian APT database and installation from scratch was the only option. Due to small embedded processor and limited system memory Raspberry Pi took long time to compile libraries. Most of this delay time was consumed by swap latency since, compilation of libraries requires system memory and Raspberry Pi had only 1 GB. OpenCV 4.0 and python 3.5 were used on Raspberry Pi because at the time of development, latest version of the Raspbian used to come with python 3.5.

Part III.

Results and Conclusion

5. Results and Discussion

In order to judge how well the proposed techniques scale, datasets of three different sizes have been used with 5, 10 and 20 identities. Test samples are picked at random, removing any kind of bias. In this chapter, PC represents Intel 8th generation i7 CPU with 6 cores and 12 threads, and Raspberry Pi represents Raspberry Pi 3 B+ model unless stated otherwise.

5.1. 2D Face Recognition using FaceNet

2D Face Recognition using FaceNet did not provide reliable results in terms of f1-score which was around 75% in case of dataset of 20 identities. Also, it does not scale well with increasing size of database. This is due to the fact that 2D face recognition has more false positives with increase in number of identities (See fig. 5.2). The running time of the method was bit higher than running times of the other methods for smaller datasets however, running time seemed to scale better than running time of other methods for larger datasets by staying almost constant for datasets with up to 20 identities. This is because the computation in neural networks does not depend on number of labels but instead it depends on network size. On the contrary the loading time of the method was around 45 seconds on Raspberry Pi and the size of dataset seemed to have no effect on loading time. Further, with increase in dataset size, the training step took huge memory and time most of which was used for calculating image features by processing it through the FaceNet network. The training of algorithm did not run on Raspberry Pi due to low system memory. The method achieved precision around 88% on average however, recall of the method was low, being around 65%. The running time of method is given based on face recognition alone without spoof detection. When only face recognition was run on Raspberry Pi, system ran out of memory and started using swap as virtual memory slowing down the process. Further, Raspberry Pi had 1 Giga Byte of RAM and the face detection used all of it, showing it's hunger for resources.

5.2. 2.5D projection with ICP

2.5D projection with ICP runs faster than both 2D Face Recognition and Colored ICP, in terms of recognition time. However, its f1-score was better than the 2D Face Recognition but not as good as Colored ICP (See fig. 5.4, fig. 5.2).

Since, in 2.5D projection with ICP does not use coarse to fine approach and the algorithm has running time of $\mathcal{O}(n)$, it ran quicker than Colored ICP. Coarse to fine approach

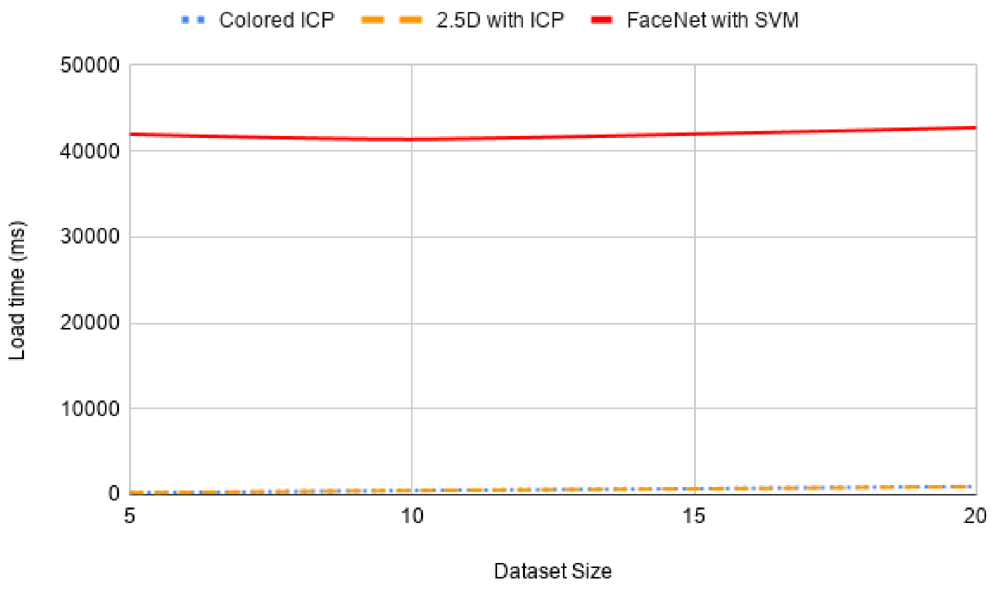

Figure 5.1.: Load time of methods with different dataset sizes on Raspberry Pi. The loading time of FaceNet with SVM is huge due to the fact that, it contains two neural networks and their size exhausts the system memory of Raspberry Pi.

requires solution to run over different scale of problem, which takes more time compared to running ICP once. Although, Colored ICP converged faster than traditional Point-to-Plane ICP but due to coarse to fine approach, it took more time [14]. 2.5D achieved around 94% precision and f1-score of 81% on dataset with 20 identities (see fig. 5.2).

Moreover, 2.5D projection with ICP ran at the rate of 1100ms per frame which is almost 1 fps on dataset of 20 identities.

5.3. Colored ICP

In experiments, Colored ICP showed promising results. In terms of f1-score, it outperformed both 2D Face Recognition and 2.5D projection with ICP (See fig. 5.2). This is because the Colored ICP uses both colored and depth information compared to 2D Face Recognition and the fact that ICP based algorithms need a good initialization and coarse to fine approach ensures that Colored ICP finds a good initialization [14]. Colored ICP achieves approximately 0.93 % f1-score which is good for a method which around 1 fps on Raspberry Pi. Further, the performance is even better than what f1-score shows, since in face recognition false positive have more impact than false negatives. Colored ICP achieved almost 98.8% precision which means there were fewer false positives.

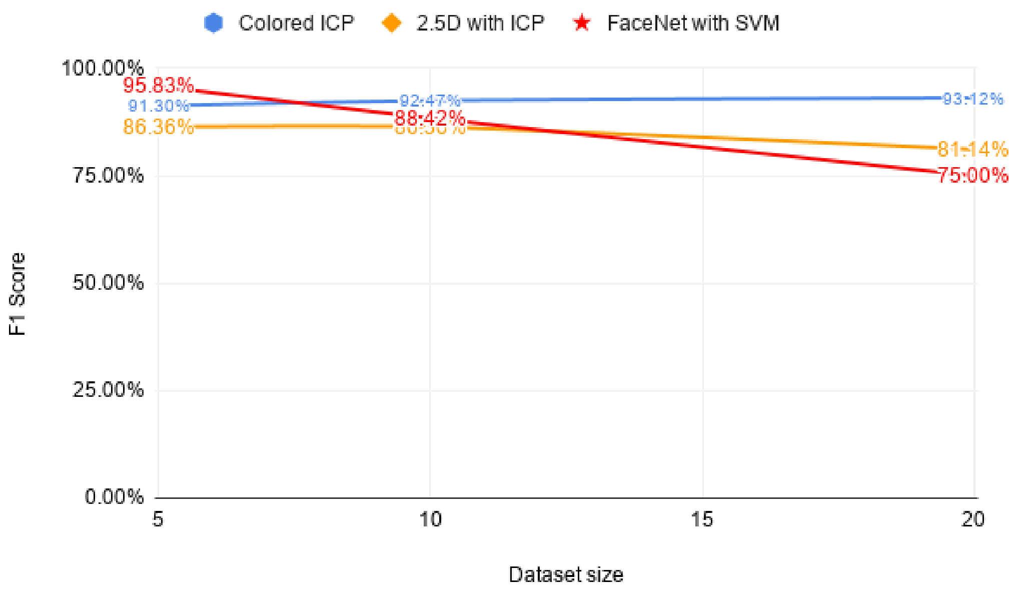

Figure 5.2.: F1 score of methods for different number of identities. Colored ICP stays starts bit low showing biased samples and stabilizes for larger dataset. For 2.5 with ICP, the score decrease with incresed number of identities, due to increase false positives mainly.

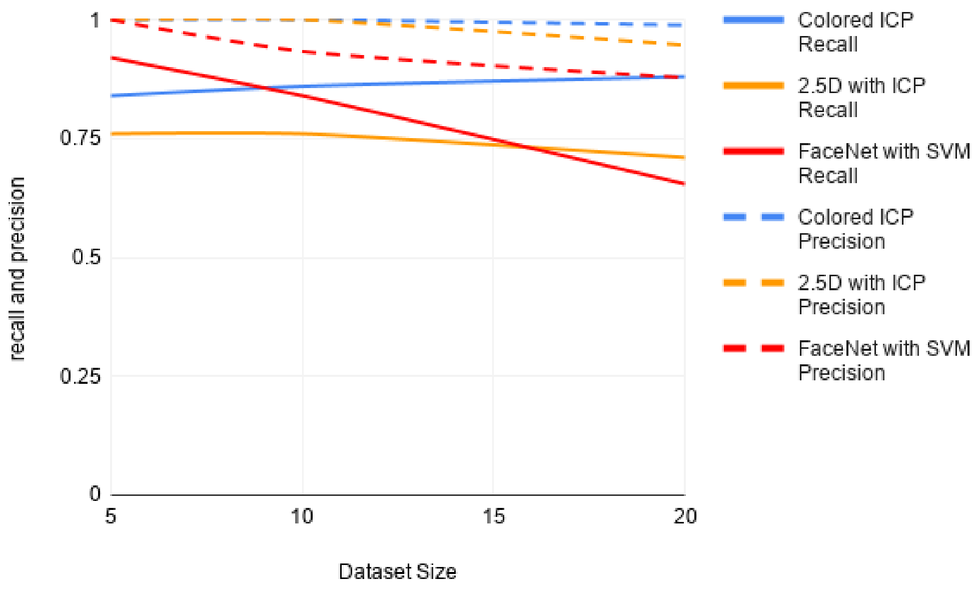

Figure 5.3.: Precision and Recall curves of methods for different number of identities. For larger datasets, Both ICP based techniques show stable precision however, for 2D Face Recognition it keeps falling with increase in dataset size because of increased number of false positives.

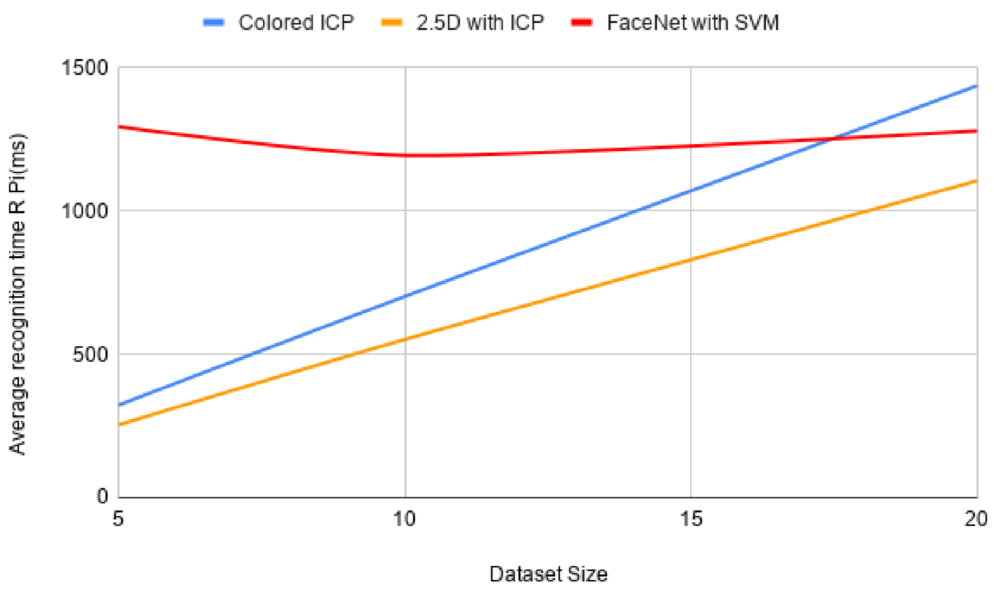

Figure 5.4.: Average Recognition Time of Algorithms on Raspberry Pi with different dataset sizes

Furthermore, Colored ICP ran with almost 1.5 fps on Raspberry Pi with average recognition time for one query being 1440 ms on dataset with 20 identities (See fig. 5.4).

5.4. Precision Test

In order to check if the developed algorithms could differentiate between different but visually similar faces, a test was performed using face masks. Two identical face masks were stuffed slightly differently in a way that, they look visually similar but have slightly different geometry (See fig. 5.5).

The masks once stuffed, were registered to the database with each mask having 5 models. The Colored ICP algorithm was then successfully tested on these models with 100% accuracy of recognition. The above stated experiment proved that Colored ICP algorithm was capable enough to deal with slightly different faces. However, this raises the question that is it good in terms of identification or bad? Since, over time different areas of face change and a slight change can cause a false negative. However, there exists many aging models to deal with such changes as well as different areas of face take more time to change compared to others, for example eyes and nose etc. and weighting of different region of faces different in ICP registration can help in dealing with such cases. Also, scheduled update of face models can also be used to tackle such problems.

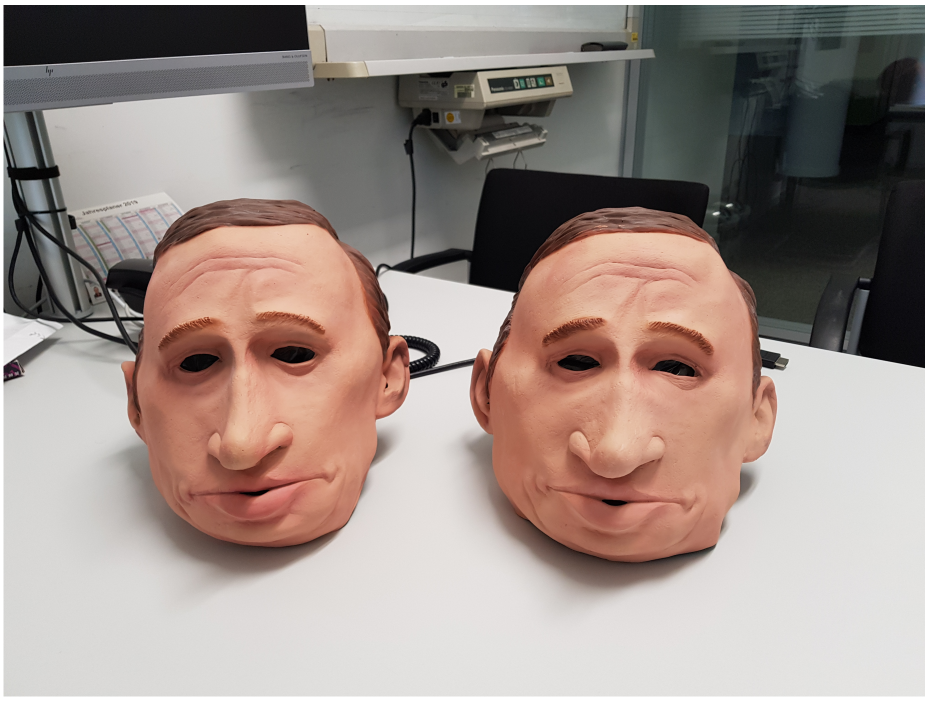

Figure 5.5.: Visually identical but geometrically different stuffed masks.

5.5. Known issues with Colored ICP

The Colored ICP algorithms uses raw intensity values which is not optimal in uncontrolled environment due to different lighting conditions. The camera used have active infrared lighting however, it can easily be dominated by sunlight coming from a side window. This results in noticeable fall in accuracy when model is tested in different lightning conditions. Possible solutions to handle such cases are intensity normalization and using properties of captured face which is independent of lightning and shadows.

6. Conclusion

6.1. 2D Face Recognition vs 3D Point Cloud Registration

2D Face Recognition methods have proved to perform good in the past few years using deep neural networks. However, when it comes to running time, required computational resources and required dataset sizes, these techniques begin to lose interest of areas where resources are limited. Due to this fact face recognition stayed far away from embedded industry for long time. 3D Point Cloud Registration on the other hand, handles complexity of problem by forming a 3 dimension based solution. Although this increase is complexity requires more resources than 2D image processing methods, this requirement is nothing compared to resources needed by deep neural networks. 3D Point Cloud Registration with reasonable down sampling without loss of vital information gives real-time performance on embedded processors like Raspberry Pi, making it ideal to be used in embedded industry.

Further, 2D Face Recognition is a standalone method. On the other hand, 3D Point Cloud Registration need certain preprocessing steps to extract the region of interest and normalize it in such a way that it is easy for the registration method to use. Since, 2D Face Recognition uses deep neural networks, for classification, the whole network needs to be loaded in memory making it slow and inconvenient. Also, to add or remove a label whole networks needs to be retrained. On contrary, in case of the 3D point cloud registration, models are nothing but point clouds and can be loaded in quickly. Further, 3D point cloud registration uses lesser memory and lesser disk space and hence, is easy to move from one system to another. Removing or adding a label only requires moving files in or out of the database, no explicit training step is required.

6.2. RGB-D Image Alignment vs Point-to-Plane ICP

ICP is prone to getting stuck in local minima since it requires a good initialization to reach near global optima. Coarse to fine approach provides good initialization and guides ICP toward global minima with small cost of running time. RGB-D Image Alignment is highly used in reconstruction of scenes using multiple point clouds and locating an object in big point cloud. Unlike ICP, RGB-D Image does not rely on good initialization however, if provided it can improve the convergence time. This is since RGB-D Image Alignment relies strongly on geometric and photometric gradients. Using photometric information gives RGB-D Image Alignment an edge over ICP and improves convergence rate.

6.3. Future Work

There are several improvements which were left due to time and scope constraint. These improvements if implemented might improve results as well as running time of Colored ICP and ICP followed by 2.5D projection on Raspberry Pi.

6.3.1. Replacement of 68-Facial Landmark Detector

One of the most time-consuming steps in 3D Face Recognition using ICP was detection of facial landmarks. However, only location of nose tip is required in algorithm which makes other 67 facial landmarks useless. DLib-ml offers a yet faster facial landmark detector namely 5-point facial landmark detector [18, 10]. 5-point facial landmark detector contains a nose landmark, so it can be used to replace 68-Facial Landmark Detector. Also, it is 8-10% faster and model size is 9.2 Mega Bytes compared to 68-Facial Landmark Detector with size 99.7 Mega Bytes, which is almost 10 times smaller. Another possible replacement for 68-Facial Landmark Detector is method proposed by Mian et al. (2006) in their research, they use coarse to fine search to localize nose in slices of faces.

6.3.2. Removing spikes and hole filling

No explicit noise removal method is implemented in proposed algorithms, which cause random mismatch score reduction. The problem can handle by removing noisy spikes in the face point clouds. Removing spikes create holes in model which cause same issues as noisy spikes, however they can easily be filled using a hole filling method [12]. In their research Mian et al. (2006) use Gaussian based hole filling which produce satisfactory results [12].

6.3.3. Pose Normalization

One of the major reasons of low f1-score for 2.5 projection with ICP is that ICP is not properly initialized, which is important to reach close of global minimum. To handle this issue, pose normalization method like Hoteling Transform can be used [12]. The pose obtained from Hoteling transform can be used to initialize the ICP to have better results. Another possible pose normalization solution is fitting of 3D plane to the point cloud. The RANSAC should be used handle noisy outliers.

6.3.4. Eigen Value based Clustering for Larger Dataset

The current ICP based algorithms work for up to 20 identities. However, once the number of identities is increased the methods will not run in real-time. It happens because each query is registered to at least one sample for each identity. One way to reduce number of identities to register is by using eigen value-based clustering. Mean eigen values can be obtained for each identity and can be used to form 3 dimensional clusters of identities. Further, these clusters could be used to reduce search space by looking into entries which are part of same cluster as query face or query face can be registered to N nearest neighbors, where N is significantly smaller than number of identities M.

6.3.5. Integration of Weights in ICP

Face masses can also vary due to environmental changes. These changes are mainly geometrical in nature and most affected areas are around cheeks and mouth. However, area of nose, eyes and around eyes and nose remains mostly unchanged. It only makes sense for nose and eyes to have more weight in ICP compared to cheeks and mouth. These weighting can significantly improve the recognition and add some flexibility towards such changes in faces [13]. Additionally, it can also help in recognizing faces with different emotions.

6.3.6. Better Face Extraction

Currently, a fix radius is used to cut face from point cloud with center of sphere being nose tip. However, with change in demographics the size of the face changes and hence the face recognition algorithms should incorporate flexibility towards such changes.

Appendix

A. Experiments

A.1. Validation of ICP Registration

In order to validate the registration results of ICP algorithm, an experiment was setup with two slightly different plastic heads (See fig. A.1). The heads were identical except one had dent around both cheeks. These heads were then registered with each other using ICP registration to validate the RMSE and fitness.

The results of ICP registration can be seen in fig. A.2, the blue color indicates unfit points which did not found correspondences in destination point cloud. All other point which are not colored blue are fit points with color indicating the euclidean distance with their respective correspondences. Further, the green color indicates close to zero distance and red indicates distance equal to maximum possible correspondence distance. In fig. A.2 the left most sub figure shows cluster of unfit points around dent area which validates the ICP registration results. The other small clusters of blue points are due to holes in point cloud caused by perspective occlusion or noise.

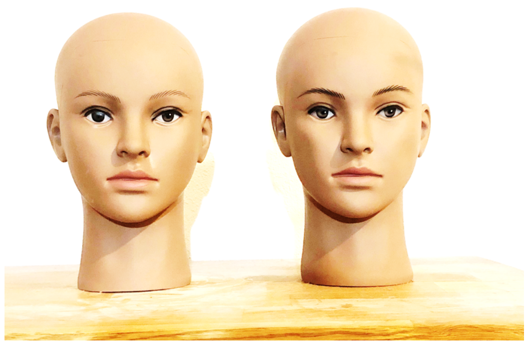

Figure A.1.: Miss Dent and Miss No Dent for Validation of the ICP Registration.

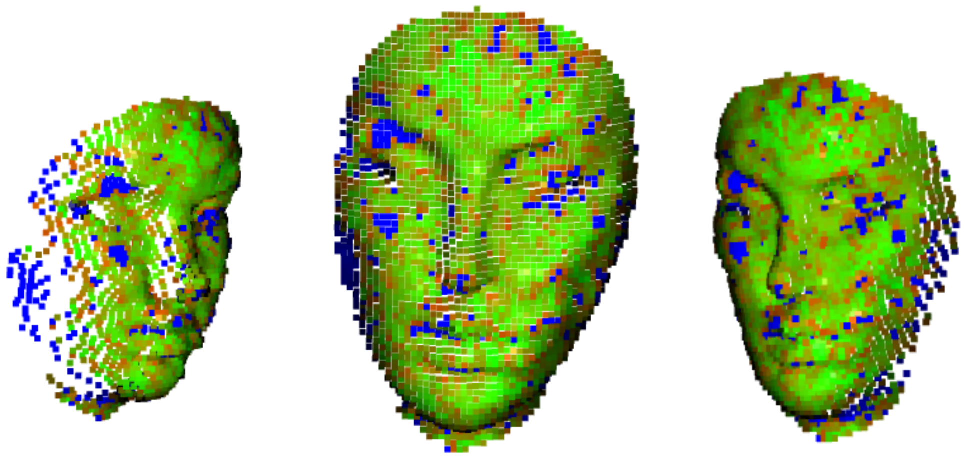

Figure A.2.: Registration results of Miss Dent and Miss No Dent

Bibliography

[1] A. Kator. The illusion of depth, 2003. [Online; accessed September 6, 2019].

[2] Besl, P., and McKay, H. A method for registration of 3D shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 14* (03 1992), 239–256.

[3] Chen, Y., and Medioni, G. Object modeling by registration of multiple range images. *Image and Vision Computing 10* (04 1992), 145–155.

[4] De Recherche, E., En Automatique, E., Antipolis, S., and Zhang, Z. Iterative point matching for registration of free-form curves. *Int. J. Comput. Vision 13* (07 1992).

[5] Feng, Z., and Zhao, Q. Robust face recognition with deeply normalized depth images. *ArXiv abs/1805.00406* (2018).

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

[7] Jones, E., Oliphant, T., Peterson, P., et al. SciPy: Open source scientific tools for Python, 2001–.

[8] Kazemi, V., and Sullivan, J. One millisecond face alignment with an ensemble of regression trees. *2014 IEEE Conference on Computer Vision and Pattern Recognition* (2014), 1867–1874.

[9] Kerl, C., Sturm, J., and Cremers, D. Robust odometry estimation for RGB-D cameras.

[10] King, D. E. DLib-ml: A machine learning toolkit. *Journal of Machine Learning Research 10* (2009), 1755–1758.

[11] Low, K.-L. Linear least-squares optimization for point-to-plane ICP surface registration.

[12] Mian, A., Bennamoun, M., and Owens, R. Automatic 3D face detection, normalization and recognition. 735 – 742.

[13] Min, R., Choi, J., Medioni, G., and Dugelay, J.-L. Real-time 3D face identification from a depth camera. *Proceedings - International Conference on Pattern Recognition* (01 2012), 1739–1742.

[14] Park, J., Zhou, Q.-Y., and Koltun, V. Colored point cloud registration revisited. 143–152.

[15] Phillips, P. J., Flynn, P. J., Scruggs, T., Bowyer, K., Chang, J. K., Hoffman, K., Marques, J., Min, J., and Worek, W. Overview of the face recognition grand challenge. *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1 1* (07 2005), 947– 954 vol. 1.

[16] R. Wang. Karhunen-loeve transform (KLT), 2016. [Online; accessed September 7, 2019].

[17] Rosebrock, A. Facial landmarks with DLib, OpenCV, and python, 2017. [Online; accessed July 10, 2017].

[18] Rosebrock, A. (faster) facial landmark detector with DLib, 2018. [Online; accessed September 11, 2019].

[19] Schroff, F., Kalenichenko, D., and Philbin, J. Facenet: A unified embedding for face recognition and clustering. *Proc. CVPR* (03 2015).

[20] Steinbruecker, F., Sturm, J., and Cremers, D. Real-time visual odometry from dense RGB-D images. In *Workshop on Live Dense Reconstruction with Moving Cameras at the Intl. Conf. on Computer Vision (ICCV)* (2011).

[21] Zhang, J., Huang, d., and Sun, J. Lock3dface: A large-scale database of low-cost kinect 3D faces. pp. 1–8.

[22] Zhang, K., Zhang, Z., Li, Z., and Qiao, Y. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters 23*, 10 (Oct 2016), 1499–1503.

[23] Zhang, P. 3D building models , production and application.

[24] Zhou, Q.-Y., Park, J., and Koltun, V. Open3D: A modern library for 3D data processing. *arXiv:1801.09847* (2018).