A VR-based System for Remote Medical Inspection

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

Master of Technology

IN

Faculty of Engineering

BY

Jakkani Sai Manoj Kumar

Computer Science and Automation
Indian Institute of Science
Bangalore – 560 012 (INDIA)

July, 2023

Declaration of Originality

I, **Jakkani Sai Manoj Kumar**, with SR No. **19382** hereby declare that the material presented in the thesis titled

A VR-based System for Remote Medical Inspection

represents original work carried out by me in the **Department of Computer Science and Automation** at **Indian Institute of Science** during the years **2021-2023**.
With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date: Student Signature

In my capacity as supervisor of the above-mentioned work, I certify that the above statements are true to the best of my knowledge, and I have carried out due diligence to ensure the originality of the report.

Advisor Name: Vijay Natarajan Advisor Signature

DEDICATED TO

My parents

and The Student Community
who are striving hard to reach new heights.

Acknowledgements

I want to express my gratitude to my advisor, Prof. Vijay Natarajan, from the Visualization and Graphics Lab (VGL) at the Department of Computer Science and Automation, IISc Bangalore. His support and advice were instrumental in the completion of this project. He regularly checked on my progress and questioned every aspect of my work, which greatly contributed to the success of this project.

I am extremely thankful for the assistance and encouragement I received from Nithin Shivashankar of Mimyk. Nithin provided invaluable insights, constructive criticism, and helpful suggestions regarding the various aspects of my work. His guidance steered me in the right direction and greatly influenced the outcome of my project.

Lastly, I want to thank my friends at the Indian Institute of Science and my family for always being there to motivate and support me. They have been a constant source of encouragement, and I am truly grateful for their understanding and help.

Abstract

Due to technological advancements and the impact of the COVID-19 pandemic, remote medical inspection has gained significant importance. Many hospitals and medical institutions are showing great interest in this field due to its numerous advantages. Remote medical inspection (RMI) enables the treatment of patients located in remote villages or unable to visit hospitals due to medical or personal reasons. Similarly, RMI provides doctors with a safe means to treat patients with highly infectious diseases. Currently, several re- mote medical inspection systems are available, utilizing online video meetings and robotic medical systems [8]. These systems enable doctors to conduct procedures such as ultrasonography and physical examinations using robotic arms and sensors remotely. Virtual Reality (VR) has gained significant popularity due to its immersive experiences, offering users the ability to interact with computer-generated artificial environments. VR provides exceptional visualizations and interactions, making it ideal for medical inspections, which surpass the capabilities of traditional video call setups. Inspired by these advantages, this project aims to combine computer vision techniques with virtual reality technologies (e.g., Oculus Quest 2) to enable more realistic and interactive remote inspections of patients' hands.

Contents

Acknowledgements i

Abstract ii

Contents iii

List of Figures v

1 Introduction 1

2 Project outline 3

3 Temporal 3D hand generation 5

3.1 RGB-D based 3D hand reconstruction . . . 5

3.1.1 Generation of depth-maps . . . 6

3.1.2 Point cloud generation . . . 7

3.1.3 Point cloud pre-processing . . . 7

3.1.4 Point cloud registration . . . 8

3.1.5 Surface reconstruction . . . 8

3.1.6 Drawbacks . . . 8

3.2 Pose and texture based 3d hand reconstruction . . . 9

3.2.1 Pose extraction and attachment . . . 10

3.2.2 Texture extraction . . . 13

3.2.2.1 Complete texture map prediction . . . 15

3.2.2.2 Partial texture map prediction . . . 19

4 Experiments 25

4.1 Results and analysis . . . 25

4.1.1 RGB-D based 3d hand reconstruction 25
4.1.2 pose and texture based 3D hand reconstruction 27
4.2 Application-level implementations 32

5 Conclusions and future work **34**
5.1 Conclusion 34
5.2 Future work 34

Bibliography **36**

List of Figures

2.1 Outline of a VR-based remote medical inspection system. 3

3.1 Architecture of GLP Network [5] 6
3.2 Pose estimation and mesh attachment 10
3.3 Extracting texture from the hand image and applying texture to generic hand mesh 10
3.4 Complete architecture of MRHSMoCap [10] 12
3.5 Left image: MANO hand mesh; Right image: UV Map of it 13
3.6 Left Image: Hand image; Right image: texture map of it 13
3.7 Textured 3D MANO hand mesh 14
3.8 Samples of the dataset: Hand raw scan and Texture map 15
3.9 Network architecture used for complete texture map prediction 18
3.10 Left image: input hand image; Right image: output texture map 19
3.11 Network architecture used for model type 1 21
3.12 Network architecture used for model type 2 21
3.13 Network architecture used for model type 3 22
3.14 Network architecture used for model type 4 23

4.1 Left image: depth map estimated by OAK-D; Right image: depth map estimated by the Monocular Depth Estimation 25
4.2 Depthmap based 3D hand generation for a single frame of the recording 26
4.3 Pose extraction and mesh attachment of a frame 27
4.4 Left: input back hand images; Middle: Ground truth texture maps; Right: Predicted texture maps 28
4.5 Left: inputs palm hand images; Middle: Ground truth texture maps; Right: Predicted texture maps 29
4.6 Left: front and back hand images; Right: Predicted texture map 30

4.7 Left: input hand image; Middle: Ground truth texture map; Right: Predicted texture map 31
4.8 Left: input hand image; Middle: Ground truth texture map; Right: Predicted texture map 31
4.9 Left: input hand image; Middle: Ground truth texture map; Right: Predicted texture map 32

Chapter 1

Introduction

The healthcare industry has undergone significant changes due to the impact of the COVID-19 pandemic, including a shift towards remote medical examinations. A comprehensive medical examination typically involves four major steps: Inspection, Auscultation, Percussion, and Palpation. During the inspection stage, healthcare professionals utilize visual, olfactory, interactive, and auditory cues to assess the conditions and deviations of various body parts. The last three stages are the physical examinations. This project specifically focuses on the remote inspection stage of medical examinations.

Virtual Reality (VR) is a technology that creates computer-generated environments and objects, providing users with a sense of presence and immersion. VR allows users to visually explore and interact with virtual features or items, creating a highly engaging and realistic user experience. Leveraging the advantages of VR, this project aims to enhance the realism of remote medical inspections.

In the context of a medical inspection, doctors conduct examinations of various body parts of the patient. To enable these inspections remotely, a typical approach involves three essential steps: recording the actions of the patient's body parts, transmitting the recorded data to the doctor's location, and displaying these actions in Virtual Reality. Computer vision techniques [9], [1] are employed to extract 3D recordings from the captured data, which are then presented to the doctor using VR technologies. Another important aspect of this system is the effective design of the 3D recording transfer process to the doctor's location, ensuring minimal transmission wait times, delays, and data loss.

Within the scope of this project, the focus was specifically directed towards remote medical inspection (RMI) of the patient's hand. As an initial step, the project aimed to develop a smaller version of RMI, concentrating solely on the inspection of the hand. This approach allowed for a more manageable and focused exploration of remote medical inspection techniques.

The decision to begin with the hand as a target area for remote inspection was driven by the need to establish a proof-of-concept and gain valuable insights into the implementation of RMI. By limiting the scope to a single body part, the project was able to narrow down the complexities involved in capturing, transmitting, and displaying hand actions within a virtual reality environment.

By accomplishing the goals and objectives of the hand-focused RMI, the project lays the groundwork for future advancements and potential expansion into remote inspection of other body parts. The knowledge and experience gained from this initial phase will serve as a solid foundation for addressing the challenges associated with more comprehensive remote medical inspections.

Chapter 2

Project outline

As previously mentioned, the initial step of the system entails the patient recording a 2D video capturing the actions of the hand. The next step involves converting the recorded information into a 3D mesh representation, specifically a collection of vertices and polygons for each time step, resulting in a temporal 3D mesh. Subsequently, the temporal mesh is transmitted to the doctor's location over the network. Finally, the temporal mesh is rendered within the Oculus Quest 2 headset and presented to the doctor, thus facilitating the medical inspection process.

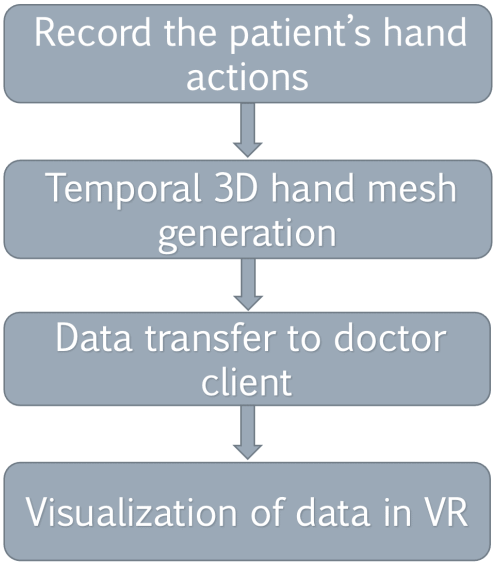

Figure 2.1: Outline of a VR-based remote medical inspection system.

VR Remote Medical Inspection systems (VRMI) consist of two clients: the Patient client and the Doctor client. The project pipeline outlined above can be divided into two stages. The first stage involves extracting the temporal 3D hand mesh representation of the patient's hand actions and processing it at the patient-client. The second stage focuses on transmitting the

generated 3D mesh to the doctor-client through the network and visualizing it in the VR environment.

The core component of the pipeline is the first stage, which involves the generation of the 3D hand mesh. Initially, a method was proposed to reconstruct the entire 3D hand for each time step of the video. However, this approach encountered various challenges, including time consumption and hardware requirements. Consequently, an alternative approach was adopted for 3D mesh generation. Instead of reconstructing the complete 3D hand for each time step, the pose and texture of the hand were extracted from each frame of the video recording. These extracted parameters were then applied to a generic 3D hand model, resulting in an efficient and effective representation of the hand recording in the 3D. In the subsequent sections, both approaches to Temporal 3D hand generation will be discussed, along with their respective shortcomings.

Chapter 3

Temporal 3D hand generation

3.1 RGB-D based 3D hand reconstruction

This approach involved reconstructing the entire 3D hand for each time step of the video. Different ways exist for extracting 3D models, each with its own advantages and limitations. Expensive devices such as LIDAR and 3D lasers offer accurate and efficient large-scale direct 3D model extraction. However, their high cost may restrict their accessibility for certain applications. On the other hand, devices like RGB-D cameras provide a more affordable option, but the resulting 3D model construction may be less precise.

In this project, the OAK-D RGB-D camera was utilized for reconstructing the 3D hand using the recorded data. The OAK-D camera offers a camera with a 30 frames per second capability, providing a cost-effective solution for capturing the necessary visual and depth information. While the 3D model construction results may not be perfect, the use of the OAK-D camera strikes a balance between affordability and capturing the essential features of the hand in a satisfactory manner.

The process of 3D model extraction using an RGB-D camera typically involves several sequential activities:

1. Generation of Depth-maps.
2. Point Cloud Generation.
3. Point Cloud Pre-processing.
4. Point Cloud Registration.
5. Surface Reconstruction.

The steps in 3D model extraction can be tackled using non-data-driven or data-driven methods. Non-data-driven approaches use traditional computer vision techniques and mathematical algo-

rithms. Data-driven approaches, on the other hand, employ machine learning or deep learning models.

3.1.1 Generation of depth-maps

The depth map assigns each pixel an appropriate depth value based on its real-world location. OAK-D, an RGBD camera without a dedicated depth sensor, utilizes stereo vision to estimate depth maps. However, this non-data-driven approach has certain limitations. Depth estimation by OAK-D, known as Stereo Depth, can be prone to inaccuracies in certain scenarios. Occlusions, where the camera's view is obstructed, and difficulties in matching pixels between the two cameras can lead to incorrect depth values. Additionally, surfaces that are shiny (like the eyes and teeth) or have similar appearances can cause variations in depth estimation due to different perspectives. To overcome these limitations, the implementation of a paper called Global-Local Path Networks for Monocular Depth Estimation [5] was carried out.
GLPN is a state-of-the-art deep learning technique for monocular depth estimation, which predicts depth maps based on input images. The decision to utilize this paper was driven by the shortcomings of the depth by OAKD and its relatively lower parameter count of 62 million. The reduced number of parameters allows for faster results, which is crucial in this project considering the sequential steps involved in generating a 3D model.

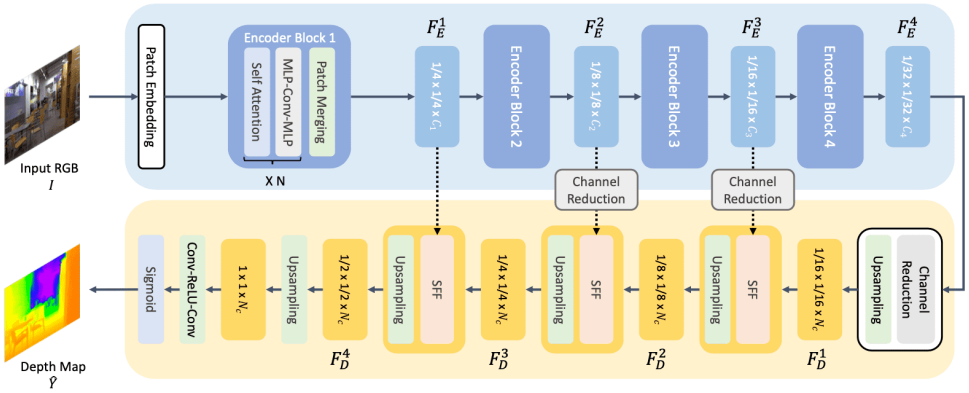

Figure 3.1: Architecture of GLP Network [5]

GLPN is an encoder-decoder network that takes an RGB image (I) of size H x W x 3 as input. It encodes the image through a series of encoder blocks containing visual transformers and skip connections, resulting in a bottleneck feature representation of size (1/32)H x (1/32)W x C4. The decoder then reconstructs the depth map of size H x W x 1 by upsampling the feature map and applying convolutional layers. To exploit local structures and fine details, the paper introduces skip connections and a fusion module called Selective Feature Fusion.
The fusion module integrates the output feature maps of the encoder, along with the output

from the previous decoder block. It constructs attention maps to adaptively select and integrate local and global features. The selected feature maps are concatenated and processed through convolutional layers. The resulting attention maps are multiplied with the local and global features, which are then added element-wise to construct a feature map for the next decoder block. To train the network, the NYU Depth V2 dataset, containing 24k images with corresponding depth maps, was utilized.

3.1.2 Point cloud generation

A point cloud is a collection of points in 3D space that represents an object. Each point in the cloud contains various associated data, such as colour values (R, G, B), vertex normals, and more. Point clouds can be generated using non-data-driven or data-driven approaches. The generation of Point clouds using depth maps is a non-data-driven approach. By knowing the camera's intrinsic calibration matrix (K), perspective projection can be applied to transform 2D depth map coordinates (u, v) into 3D coordinates (x, y, z). The intrinsic matrix contains parameters such as horizontal and vertical focal lengths (fu, fv) and the optical center (ou, ov) of the image plane. By taking the inverse of the intrinsic matrix, each pixel in the 2D depth map can be converted into a 3D coordinate system. This process results in a point cloud consisting of N points, where N corresponds to the number of pixels in the 2D depth image.

3.1.3 Point cloud pre-processing

In point cloud preprocessing, several essential techniques were applied to the generated point clouds. These steps include:

Down sampling: The point cloud generated for each frame contains a significant number of points, approximately 250,000. Handling such large point clouds can be computationally intensive and time-consuming. Additionally, transmitting the entire temporal 3D mesh of each frame over the network to the doctor's end would also consume considerable time. Downsampling was performed to reduce the number of points. Voxel downsampling, specifically, was utilized, where points within each voxel were averaged to form a representative point, preserving the 3D essence of the cloud.

Background points removal: Secondly, background points were removed by conducting image segmentation on the input frame and eliminating corresponding points in the point clouds. This not only eliminated irrelevant information but also reduced the point count for subsequent steps.

Outlier removal outlier removal was employed to eliminate points that deviated significantly

from the expected data distribution. OpenCV functions were employed for this purpose.

Vertex normal estimation vertex normal estimation was performed to determine the normal at each vertex. This step was crucial for surface reconstruction techniques used in the pipeline.

3.1.4 Point cloud registration

Point cloud registration involves combining two point clouds from different spaces into a single point cloud. In the context of hand inspection, if we record actions from one side of the hand and then capture actions from the other side, we can merge the resulting point clouds. However, point cloud registration is not always necessary. For inspections involving specific areas like eyes, teeth, or wounds, a complete 3D view may not be required or feasible. Therefore, while point cloud registration is not essential for this project, it could be considered for future development.

3.1.5 Surface reconstruction

A 3D mesh consists of vertices and triangles that define the surface of a 3D object. In the current pipeline, we have obtained the point cloud data, which represents the vertices of the object. The next step is surface reconstruction, which involves generating triangles between the points in the point cloud to create the object's surface. Surface reconstruction can be performed using either a non-data-driven or data-driven approach.

Non-data driven approach

MeshLab provides classic surface reconstruction techniques such as Ball Pivoting [1] and Screened Poisson surface reconstruction . These techniques are effective when the point clouds are of good quality. The key task is to accurately adjust the parameters of these algorithms. I utilized the Python bindings from MeshLab to directly generate surfaces and fine-tune the parameters.

Data driven approach

I experimented with deep-learning models for surface reconstruction, specifically Points2Surf [3] and Points2Mesh [4] However, these models were time-consuming compared to the Screened Poisson technique and did not yield superior results. The deep learning models were more complex than the classical algorithms mentioned earlier.

3.1.6 Drawbacks

The following challenges were encountered in this approach:

1. Time and Data Volume: The processing time required to generate a 3D mesh from a single image is significant, taking around 2 seconds per frame. Consequently, generating

a temporal 3D mesh for an entire minute-long video can take up to 30 minutes. Additionally, the amount of data generated per frame is approximately 500 MB, resulting in a substantial data size for a minute-long video. This presents significant challenges for my real-time application, as latency plays a crucial role. The extended processing duration and data transfer times introduce undesirable latency, which is not acceptable in this case.

2. Trade-off between Processing Time, 3D Output Quality, and Data Size: There exists a trade-off between reducing processing time, maintaining high-quality 3D output, and minimizing the amount of data generated. Aggressive vertex downsampling is performed to reduce processing time, but this can lead to a loss of detail in the resulting 3D mesh.

3. Hyperparameter Tuning Complexity: Various algorithms involved in surface reconstruction, voxel downsampling, outlier removal, and normal estimation require careful tuning of multiple hyperparameters. Ensuring optimal parameter settings for each algorithm becomes a time-consuming and challenging task.

These drawbacks have prompted the adoption of approach 2 as an alternative solution.

3.2 Pose and texture based 3d hand reconstruction

In the previous approach, the time-consuming process of reconstructing everything for each frame posed a challenge. To overcome this, an alternative solution is proposed. Instead of reconstructing the entire part for each frame, a generic 3D hand mesh, closely resembling the patient's hand in terms of size and dimensions, is utilized. By extracting and attaching only the poses from each frame to this pre-existing mesh, the temporal 3D mesh of the hand recording can be generated much more quickly. This approach significantly reduces the processing time required for the recording and minimizes the amount of data to be transmitted to the doctor's end, as only the poses need to be sent. Furthermore, if a patient's 3D hand can be rapidly reconstructed at the beginning of the inspection, it can be directly used instead of the generic 3D hand mesh for posing. These modifications aim to address the limitations of the previous approach, improving efficiency and streamlining the process.

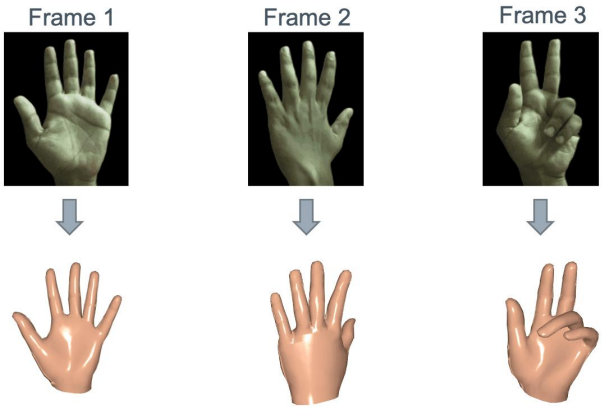

Figure 3.2: Pose estimation and mesh attachment

Now, considering that each patient's hand may have unique textures or specific characteristics like injuries, it is important to make the generic hand mesh appear realistic. To achieve this, the texture of the patient's hand can be extracted and applied to the mesh. By incorporating the patient's specific texture onto the generic hand mesh, the output of this approach can closely resemble the results obtained from the previous approach. This ensures that the reconstructed hand in this approach retains the visual details and realistic appearance achieved in the previous approach, effectively addressing the limitations of the previous approach.

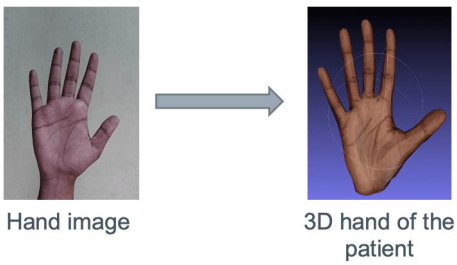

Figure 3.3: Extracting texture from the hand image and applying texture to generic hand mesh

So approach 2 has two steps:

1. Pose Extraction and Attachment
2. Texture Extraction

The following sections provide detailed explanations of the methods utilized for each step in approach 2, including extracting pose from a single image, attaching it to a 3D hand, and the technique employed for texture extraction.

3.2.1 Pose extraction and attachment

Before delving into the methods, it is important to introduce MANO (Hand Model with Articulated and Non-rigid defOrmations) [7].

MANO is a 3D hand computer graphics model derived from 3D hand scans of 32 individuals. The MANO model is a widely used computer graphics model specifically designed for representing and animating human hands. The model encompasses two key parameters: shape parameters and pose parameters.

1. **Shape parameters** control the overall shape and proportions of the hand, allowing for customization to match different hand sizes and shapes.

2. **Pose parameters** determine the joint angles and rotations, enabling the model to accurately replicate various hand poses and movements.

By manipulating these shape and pose parameters, we can accurately deform the 3D MANO hand model to match the desired configurations. So, in order to address the need for a realistic 3D hand representation specific to each patient, the MANO (Model for Articulated Hands) model is utilized. This model serves as a generic 3D hand model that can be customized by the pose and shape parameters unique to each patient's hand. By accurately estimating these parameters, we can generate a 3D hand shape that closely resembles the patient's actual hand and accurately replicate its pose.

Pose and shape estimation In recent literature, the majority of papers [2], [10] focused on 3D hand generation from images rely on MANO as the foundational 3D model. The paper titled 'Monocular Real-time Hand Shape and Motion Capture using Multi-modal Data(MRHSMoCap)'[10] is utilized to estimate the shape and pose parameters of the MANO model. This paper showcases state-of-the-art results in this domain. Given that the objective was to develop a VR Medical inspection system, existing implementations are leveraged instead of devising new approaches for each component of the system.
The approach proposed in the paper 'Monocular Real-time Hand Shape and Motion Capture using Multi-modal Data' involves a combination of neural networks within a single architecture: DetNet and IKNet.

1. DetNet is responsible for predicting 2D and 3D hand joint positions from a single RGB image, employing a multi-task scheme.
2. IKNet, on the other hand, takes the 3D joint predictions and converts them into a joint rotation representation in an end-to-end manner.

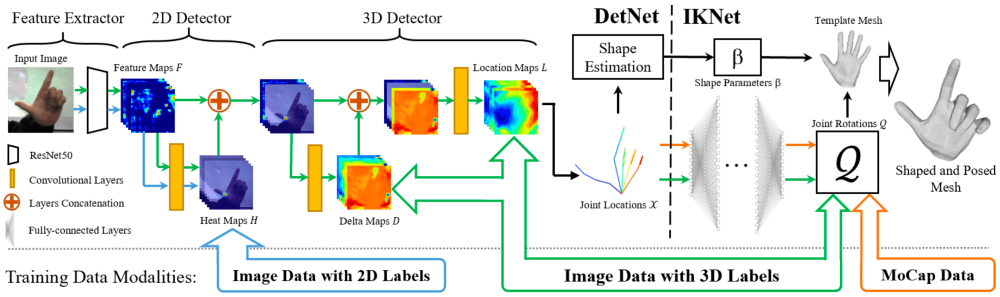

Figure 3.4: Complete architecture of MRHSMoCap [10]

DetNet takes a single RGB image as input and generates both 3D hand joint predictions and 2D joint predictions in image space. It employs ResNet50 as the feature extractor, producing a 32x32x256 feature volume from 128x128 input images.

It has a 2D detector, a compact 2-layer CNN, takes the feature volume and outputs 21 heatmaps representing hand joints. In each heatmap, each pixel encodes the confidence of that pixel being associated with joint j. The heatmaps are used for 2D pose estimation, and this subtask is supervised using ground truth 2D annotations.

The 3D detector estimates location maps, where each pixel describes the 3D coordinates of a joint, by processing heatmaps, feature maps, and delta maps through a CNN. The location maps and delta maps are supervised using 3D annotations. The joint positions are retrieved from the location maps based on the maxima of the heatmaps.

The DetNet loss function consists of four terms, including the heatmap loss between predicted and ground truth heatmaps. Two additional loss terms measure the difference between predicted and ground truth location maps and delta maps.

During training, the network is trained jointly using data with both 2D and 3D annotations. This multi-task scheme allows the network to learn 2D pose estimation from in-the-wild images, enhancing its generalization ability and capturing 3D spatial information from 3D labelled data.

IKNet is a 7-layered feed-forward neural network designed to solve the inverse kinematics (IK) problem by predicting joint rotations from joint locations. These are used to pose MANO model

IKNet is trained using motion capture (MoCap) data collected from 3D scanners and advanced hand-tracking gloves. Additionally, the pre-trained DetNet is utilized to generate 3D predictions for all training samples. The resulting joint rotations of those 3D predictions estimated by the IKNet are given to the forward kinematic layer to reconstruct the joint positions. The reconstructed positions are compared to the generated 3D joint annotations for training supervision. This approach enables IKNet to effectively handle the 3D predictions of DetNet.

The loss function for training IKNet involves the l2 norm between the ground truth rotation

angles and predicted rotation angles.
Using the predicted shape and pose parameters for each frame of the video, we attach them to the MANO model to obtain the temporal 3D hand mesh.

3.2.2 Texture extraction

An essential step in approach 2 is to precisely apply the hand's texture onto the MANO hand mesh. While this task can be performed manually, it is highly time-consuming, especially when doing it for each frame of the video. In this real-time setup, we need to automate this process. However, there is limited existing literature in this area, posing a challenge for hand texture mapping. The complex shape of the hand further complicates the task, making it a non-trivial task to achieve realistic and visually appealing results. Therefore, the focus was on developing automatic methods to achieve accurate texture mapping onto the hand mesh in this real-time scenario.
Mapping a 2D image onto a 3D mesh is a challenging task, particularly when it involves converting from 2D to 3D. so instead of that, First, the 3D hand model is unwrapped into a 2D space, creating a UV mapping. This step essentially flattens the 3D mesh into a 2D representation. Then, the goal is to map the 2D image onto this UV mapping, which can be considered as mapping from one 2D space to another. This 2D-to-2D mapping is comparatively easier than direct 2D-to-3D mapping, allowing for a more feasible and manageable process.

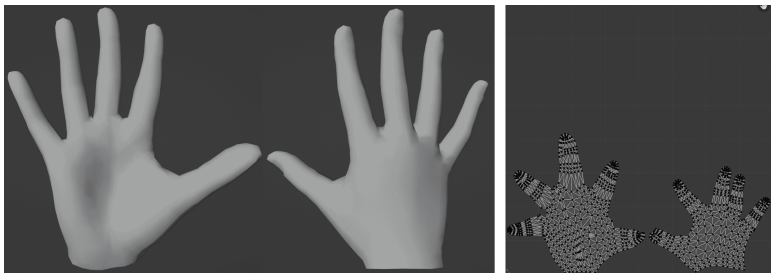

Figure 3.5: Left image: MANO hand mesh; Right image: UV Map of it

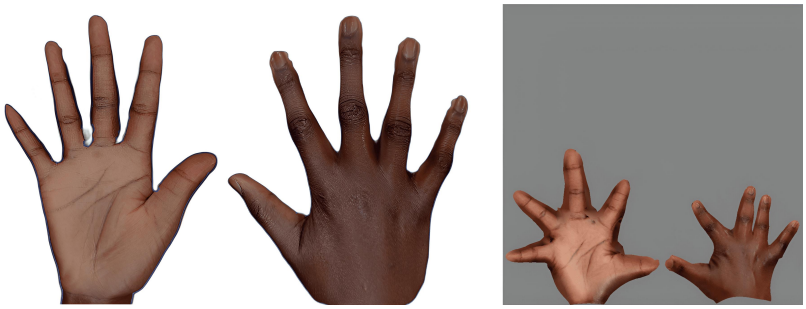

Figure 3.6: Left Image: Hand image; Right image: texture map of it

In Figure 3.5, the first image is the MANO hand mesh, while the second image represents the UV map of it. The left image of Figure 3.6 shows the patient's hand images. If we can align

and position the patient's hands in the same manner as shown in the UV map of the MANO mesh, which is the right image of Figure 3.6, then we can apply the texture onto the MANO hand mesh, as shown in Figure 3.7.

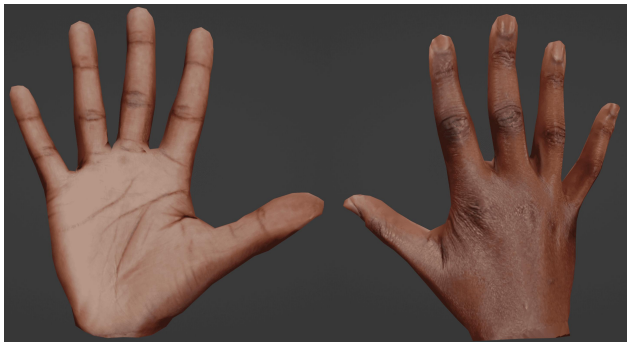

Figure 3.7: Textured 3D MANO hand mesh

This process allows us to map the patient's hand appearance onto the 3D hand model, enabling realistic and accurate texture rendering.

The task at hand is to find a way to convert the hand images to that of the texture map shown as the right image in Figure 3.6.

The paper titled "HTML: A Parametric Hand Texture Model for 3D Hand Reconstruction and Personalization" [6] presents a dataset comprising high-resolution hand scans from 51 individuals with diverse characteristics such as gender, age, and ethnicity. The authors utilized these scans to generate 102 texture maps, representing the left and right hands of the subjects in the dataset. However, the process described in the paper for transforming the hand scans into corresponding texture maps is time-consuming and not suitable for this real-time setup.

The paper introduces a parametric model constructed using PCA on the 102 texture maps derived from the dataset of hand scans. However, when estimating the parameters for a patient's hand texture map and applying those parameters in this parametric model, it does not yield an exact replica of the patient's hand texture. Instead, the resulting texture map represents a combination of the 102 texture images from the dataset. Unfortunately, this does not align with the task of applying the exact texture of the patient's hand on the MANO model.

Despite the inability to directly utilize the paper's texture extraction approach, the focus remained on leveraging the dataset of raw scans and the texture map created by the authors. By exploring alternative methods and techniques, a way was sought to make use of this valuable resource.

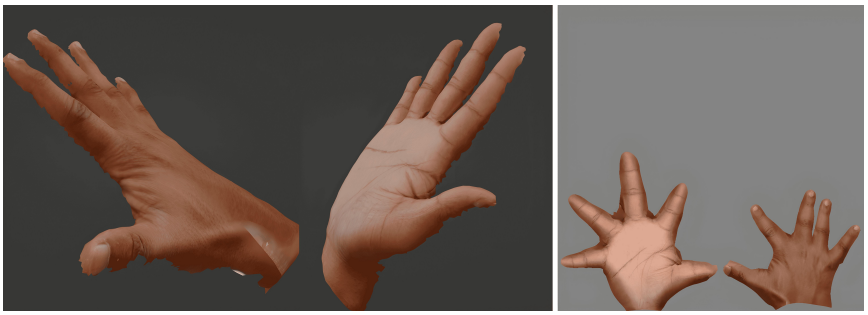

Figure 3.8: Samples of the dataset: Hand raw scan and Texture map

A strategy was proposed which involves constructing a neural network capable of learning the mapping between hand images and texture maps. Specifically, the network aims to accurately position the hands in the images relative to the UV Map of the 3D Hand mesh.
Two approaches have been formulated for this task:

1. **Complete texture map prediction:** The objective is to predict the texture map given the back and front images of the hand.
2. **Partial texture map prediction:** The goal is to predict the visible texture map of the hand for each frame of the hand recording. These partial texture maps are combined using an online combination algorithm to create the complete texture map.

In both approaches, the neural network takes an image as input and produces an image as output. To accomplish this, a dataset is created using the dataset of raw scans and the texture map, and then a neural network is formulated and trained using this dataset to obtain the desired results in each of the approaches. The following sections provide detailed explanations of the experiments conducted for each approach.

3.2.2.1 Complete texture map prediction

This approach focuses on predicting a complete texture map by using both the front and back images of the hand. Before recording the hand actions of the patients, the front and back images of their hands are requested. Utilizing these images and the neural network, the texture map prediction is acquired at the beginning of the process, unlike the Partial Texture Map Prediction approach. To facilitate the task at hand, a dataset is created following the format illustrated in Figure 3.6, where the left image is designated as the input and the right image serves as the output. Raw scans of the hands from 51 subjects, as depicted in Figure 3.8, are employed for dataset generation.

Dataset creation: Pictures of the raw scans are taken as if the subjects are resting their hands on a table, resulting in a total of 102 data points by considering both hands of each individual.

Despite the limited size of the created dataset, due to the nature of working with images, there are numerous image augmentation techniques available to augment the dataset. However, it is important to consider that not all augmentation techniques are suitable for the task at hand, as the specific requirement is to focus on techniques that can effectively modify the tone of the images.

Since the objective is to accurately position the hand depicted in the input images onto the UV area of the UV map, it can be intuitively understood as mapping the pixels of the hand to specific positions in the texture map. In this context, even if the tone of the images is modified, it primarily affects the pixel values, which is acceptable and desired for the task. By leveraging appropriate image augmentation techniques, the diversity and variability of the dataset can be effectively increased, thereby enhancing the robustness and generalization capabilities of the model.

Another aspect worth mentioning is the possibility of applying the "data acquisition" method outlined in the HTML paper to generate additional hand scans and texture maps in the future. However, the implementation of the data acquisition part has been postponed at present due to the insufficient details provided in the paper. Instead of investing time and resources in further research on the data acquisition process, the decision has been made to proceed with the available data points.

A total of 75 image augmentation techniques were thoroughly tested and evaluated to assess their suitability for the task of texture prediction from hand images. It is important to consider that the dataset consists of hand images from individuals of various ethnicities, including white and black individuals. Consequently, certain tone-changing augmentation techniques may not yield desired results for all types of hands. Hence Each augmentation method was carefully applied and examined to ensure its effectiveness in achieving the desired outcomes. The careful execution of this process required a substantial investment of time and effort within this approach.

Among the considered techniques, the following were identified as potentially beneficial:

1. Rotation: Given that subjects' hands can be positioned at various angles, applying rotation enables the model to handle different orientations effectively.

2. Flipping: The mirroring of right and left hands is a valid technique as it effectively expands the dataset by treating the mirrored hands as additional samples.

3. White balance: This is used to adjust the colour temperature of an image to make it appear more visually appealing. By applying this augmentation, images can be made to look more natural and lifelike.

4. Grayscale conversion: Converting both the input and output images to grayscale facilitates the model's understanding of texture patterns.

5. Bilateral blur: Introducing blur to the input images and predicting the unblurred output texture images enhances the model's robustness by accounting for potential image blurring issues.

6. Hue adjustments: Modifying the hue shifts the tone of the hands and texture maps, enabling the model to handle variations in skin colour and shading, such as yellow shading.

7. Brightness adjustments and exposure: Fine-tuning the brightness and exposure of the hand images and corresponding texture maps allows the model to adapt to varying lighting conditions.

8. Lens distortion: It is the distortion that occurs in a camera lens, causing straight lines to appear curved or distorted. Augmentation techniques replicate this effect to introduce diversity in the input hand images. It is not applied to the Output texture image since the model should learn to rectify the distortions.

9. Adding artificial dust: This technique introduces a simulated camera artifact, thereby improving the model's resilience when predicting unaltered output images.

A few other image augmentation techniques employed include random distortion, colour channel dropout and perspective projection on input hand images and gamma, saturation, exposure and contrast adjustments on both input and output images. By these, the dataset is effectively increased from 102 to 6120 data points.

Modelling: Multiple neural network models were designed and trained using this dataset. Among the various network models that were trained, the following network yielded better results.

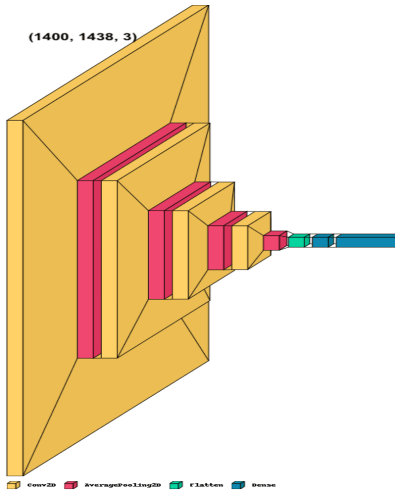

Figure 3.9: Network architecture used for complete texture map prediction

Two distinct networks with identical architectures same as in Figure 3.9 were constructed for the palm and backside hand images, respectively. The network trained on the backside hand images predicts the values for 42,016 pixels in the texture map, which represents the UV mapping of the back of the MANO mesh. On the other hand, the network trained on the palm side hand images predicts the values for 61,326 pixels in the texture map, corresponding to the UV mapping of the palm of the MANO mesh. Given that the predicted values fall within the range of 0 to 255, mean squared error and mean absolute error were employed as the loss functions, respectively.

One of the challenges encountered was the escalation of training parameters, leading to overfitting. Despite this issue, deliberate training was pursued, considering that the input images during testing would be similar to those used for training. However, the achieved results were not satisfactory. While the predicted texture maps showcased tones resembling the input images, they failed to accurately reflect the exact details of the hands depicted in the input images. The objective was for the model to learn the mapping between the input image and the texture image, but instead, the model learned the hand features and attempted to replicate them closely to the input hand. Adjusting the parameters by reducing the size of the input and output images was considered, as this approach is commonly employed in models dealing with images to control the number of parameters. However, resizing the images to a smaller size was unsuitable for our case as it compromised image clarity, which is undesirable for texture prediction. Such resizing techniques are typically more applicable when predicting pose or similar factors. Thus, these challenges were encountered in this approach. The obtained results and analysis are detailed in the Experiments section.

3.2.2.2 Partial texture map prediction

An alternative approach was considered to address the texture extraction challenge. During the recording of hand actions, only a partial view of the hand is visible in each frame. Consequently, the patient's hand is captured from different views throughout the entire recording. By designing a neural network capable of predicting the partial texture of the visible hand area of a given image, we can generate partial texture maps for all frames in the recording. These partial texture maps can then be combined into a complete texture map using some kind of online combining algorithm. The underlying idea behind this approach is to provide the model with pose information, enabling it to predict the visible texture specific to the posed hand. It is anticipated that this approach may yield improved results compared to the previous method. To proceed with this approach, it is necessary to construct a dataset consisting of input and output images as the left and right images depicted in Figure 3.10.

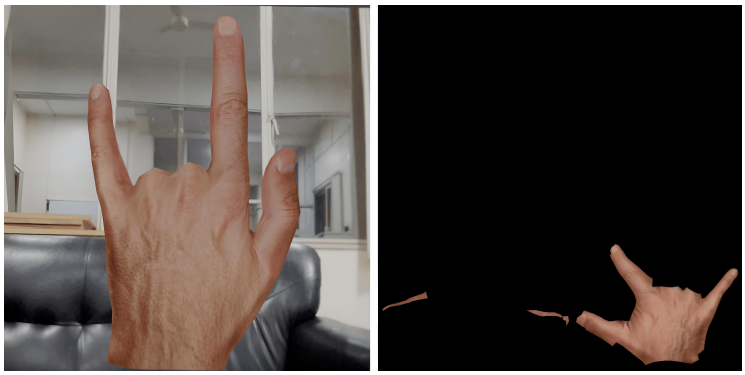

Figure 3.10: Left image: input hand image; Right image: output texture map

Dataset creation: The HTML dataset includes a collection of 102 texture maps. These are now applied to the MANO hand mesh. Pictures of the textured mesh are captured from various angles, sizes, and poses. By employing scripts executed within the Blender application, the visible texture map of the posed hand is extracted.

In order to obtain the visible texture map of the posed hand from a specific camera location, various approaches were explored. One such approach involved identifying the corresponding 3D points on the 3D hand mesh for each pixel of the texture map. Subsequently, a ray casting technique was utilized to determine the visible 3D points when viewed from that particular point. However, the computation cost associated with performing ray casting from 1.1 million points (corresponding to the 1024x1024 texture map) proved to be impractical due to its high computational demands.

To address the computational limitations, the focus was narrowed down to the 908 points present on the mesh. The ray casting approach was employed to determine the visibility of

these points when viewed through the camera positioned in 3D space. Subsequently, only the triangles on the texture map associated with the visible points were selected. Visible texture maps are then generated by selecting the texture content present only in the visible triangles. This process is automated using scripts within the Blender application.
The MANO meshes are posed, and the camera location is specified. Subsequently, the scripts are executed using all 102 texture maps to generate the corresponding visible texture maps for each posed mesh. This process is repeated for 60 different poses in the Blender software, resulting in the generation of 6,120 data points.
This process was time-consuming as it required manual handling of specific cases at the visible boundaries of the mesh. For instance, there were instances where certain points appeared close when viewed through the camera, but in reality, they were located farther on the mesh. It would be challenging for a network to predict a large visible texture area based on such a small area, so those areas are removed. Additionally, some vertices were positioned just behind the visible area, yet their inclusion was essential. On the other hand, some vertices were visible from the camera, but the corresponding triangle formed by them was not visible, necessitating their removal. These manual adjustments and considerations contributed to the overall time required for the dataset generation process. The output texture maps is of size (1024x1024) and the input images is deliberately chosen to be of size (544x544) instead of (1024x1024) due to memory constraints and limited processing capabilities.

Modelling: The neural network trained in the previous approach was not utilized for the current task, as the results were not good, necessitating the development of a new network specifically tailored to this requirement.
Several network models were experimented with, totalling more than 10 different specifications. However, the results obtained were unsatisfactory. Due to the nature of the texture maps, which consist of 95% empty space and only 5% non-empty areas, many of the models tend to predict texture maps that predominantly contain empty areas. The challenge arose from the need to accurately predict both the non-empty and empty areas, as the latter corresponded to textures that were not visible. Additionally, attempts were made to modify the texture map, including converting it to a lower resolution. Different variations of the UNET architecture were also explored, including adjusting the number of contracting paths, implementing custom loss functions, and incorporating pre-processing techniques to remove the background from input images. Unfortunately, these experiments did not yield improved results.
The following are the details of the various models utilized in this approach:

Model type 1:

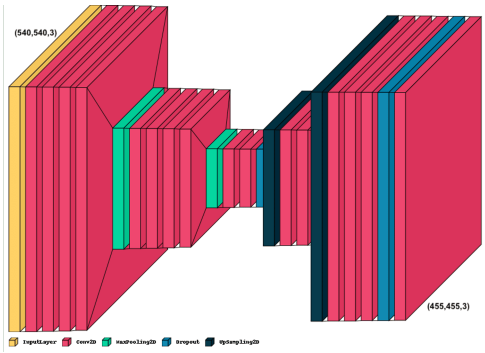

Figure 3.11: Network architecture used for model type 1

In this type of model, the pixels corresponding to the hand texture content in the texture map were selected. These pixels amounted to a total of 206,330, each having 3 channels (RGB). The data was reshaped from (206, 330, 3) to (455, 455, 3) to create an image, which served as the output for prediction.

One of the main challenges encountered in this approach is the long training time per epoch, which is approximately 60 minutes. This prolonged duration can be attributed to the computationally expensive upsampling layers, the depth of the model, the large dataset size of 6,000 images and the size of input and output images. Additionally, the model's loss is not converging, indicating difficulties in effectively learning from the data. To address this issue, a custom loss function was implemented. However, this modification has further increased the training time to around 1 hour and 25 minutes per epoch, making it impractical to train the network. Despite experimenting with different combinations of upsampling and max-pooling layers, the models failed to converge.

Model type 2:

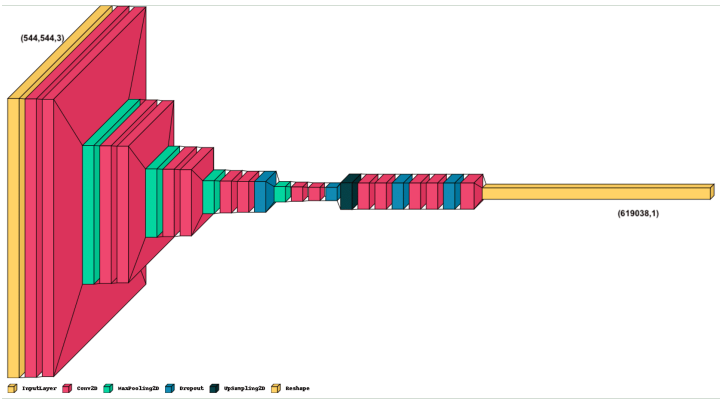

Figure 3.12: Network architecture used for model type 2

In this approach, the 206,330 * 3 values representing the hand texture content in the texture map were reshaped into a matrix of size (51, 51, 238). The objective was to predict this matrix from the given input. Organizing the data in this way, allowed for a more structured representation of the hand texture, potentially facilitating the learning process for the model. One of the main challenges encountered in this model is the extended time required for convergence. Despite training the model for 900 epochs, it has not yet reached convergence. This could be attributed to various factors such as the complexity of the network architecture, the size and diversity of the dataset, or the chosen optimization algorithm.

Model type 3:

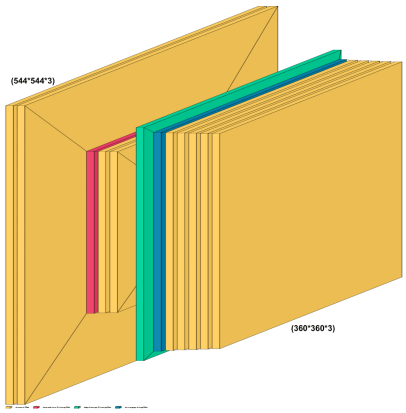

Figure 3.13: Network architecture used for model type 3

The resolution of the texture map was reduced from 1024 to 360, which helped to reduce the computational requirements and memory usage. Additionally, the background of the input images was removed to focus solely on the hand region of interest. With these adjustments in place, an image-to-image network inspired by the UNET architecture was trained. These modifications aimed to improve the training efficiency and enhance the model's ability to learn the mapping between input images and texture maps.

Model type 4:

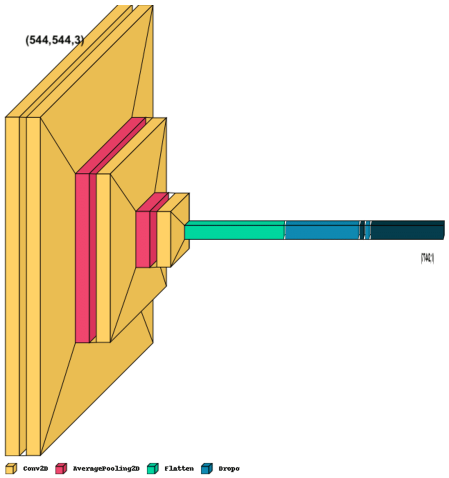

Figure 3.14: Network architecture used for model type 4

In this approach, the goal was to predict the RGB values of the pixels within the hand texture content of a low-resolution texture map. Specifically, the focus was on the 25,814 pixels that constitute the hand texture, excluding the larger number of grey pixels present in the surrounding areas. By narrowing the scope to the region of interest, the model aimed to effectively capture and reproduce the intricate details and colour variations of the hand texture.
While this model demonstrates good performance in accurately predicting the boundaries of the visible texture, it faces challenges when it comes to accurately predicting the specific content within those boundaries. Moreover, its generalization capability is limited, meaning that it may not perform optimally when presented with input images that differ significantly from those in its training dataset. In such cases, the model's predictions may not be as reliable or accurate as desired.
Regrettably, the conducted experiments did not yield improved results in comparison to previous approaches. The obtained results and analysis for each of the above models are detailed in the Experiments section.
During the process of modelling, various challenges were encountered.

1. UNET is typically designed for images of size 128x128. However, in the experiments, larger images with dimensions of 550x550 are opted. This decision was made with the intention of providing the network with higher-resolution inputs, which theoretically should improve the texture information captured. Nevertheless, training a UNET with these larger images necessitates additional time compared to the standard approach. Consequently, it is taking a longer duration to ascertain the effectiveness of the model and resulting in impractically long training times.

2. Scarcity of literature available for this specific task or related tasks, such as Image2Image, where the most focus is on image segmentation, object identification, or style transfers.

3. The texture map contains a significant proportion (60%) of empty spaces, leading models to disproportionately focus on them during training. However, accurate predictions are required for both empty and non-empty spaces. Custom loss functions have been attempted, but satisfactory results have not been achieved.

4. Challenges in hyperparameter tuning: Finding optimal hyperparameters for training the models proved difficult and time-consuming, exacerbated by the lack of guidance from existing literature or prior knowledge about the task.

5. Lack of interpretability: The models lack interpretability, making it challenging to comprehend the reasoning behind their predictions and understand the underlying factors driving those predictions.

Chapter 4

Experiments

As mentioned earlier, the aforementioned implementations were completed and the data from the generated temporal 3D meshes was transferred to the doctor's client for display in virtual reality (VR). The following section presents the results of these implementations.

4.1 Results and analysis

4.1.1 RGB-D based 3d hand reconstruction

Figure 4.1: Left image: depth map estimated by OAK-D; Right image: depth map estimated by the Monocular Depth Estimation

The depth estimated by the OAK-D camera shows some noise compared to the depth estimated by the deep learning model. However, this noise is observed only in a few frames of the clip. On the other hand, the OAK-D camera provides depth maps much faster than the deep learning model, which has a significantly larger number of parameters. The OAK-D camera's optimized code allows for efficient depth extraction. However, it may not accurately provide depth values for objects closer to the camera due to the restricted search space. While the OAK-D camera takes more time in such scenarios, it generally offers quick and reliable depth values, excluding cases where objects are closer to the camera.

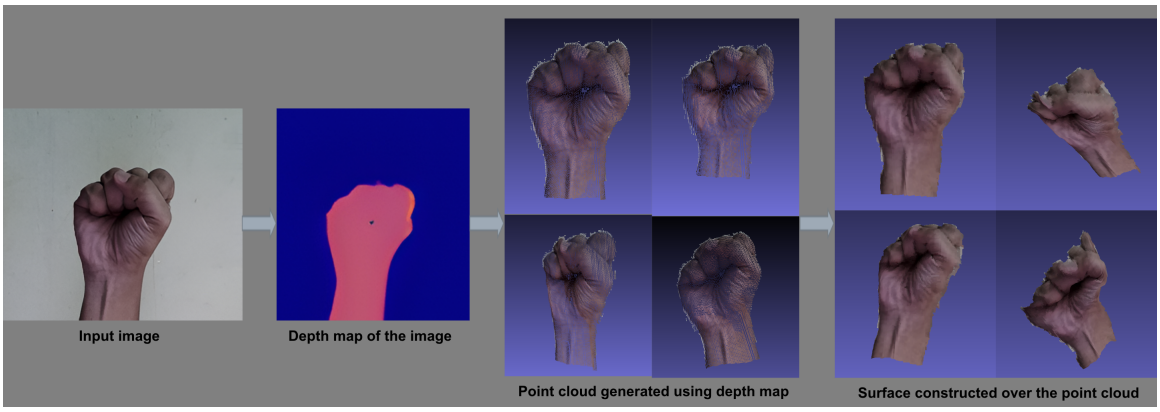

Figure 4.2: Depthmap based 3D hand generation for a single frame of the recording

The displayed result represents a single frame of the recording. To obtain the 3D result from the image, a sequence of 7 algorithms is applied. Tuning the hyperparameters of downsampling, outlier removal, vertex normal estimation, and surface reconstruction algorithms significantly influences the quality of the results. On average, the conversion process for a single image takes approximately 1.8 seconds and generates 2.6MB of data, including vertices, colors, normals, and triangles. However, for hand part inspection, additional time and space are required for point cloud registration.

Recording a 10-second video at 30 frames per second would demand a maximum processing time of 9 minutes and the transfer of 630MB of data to the doctor's client. This is a considerable challenge in real-time scenarios, especially considering that not all patients will have high-end devices capable of parallelization. Furthermore, reducing the data size for transfer is not feasible due to the importance of detailed hand mesh in medical inspections. Estimating the time required to transfer this data at a transfer speed of 10 Mbps, would take approximately 6 minutes, which is impractical for low-latency systems.

Given the drawbacks of high processing time, large data transfer requirements, and the need for high-end devices, the approach of pose and texture-based 3D hand reconstruction was devised to address these limitations.

4.1.2 pose and texture based 3D hand reconstruction

Pose and shape estimation:

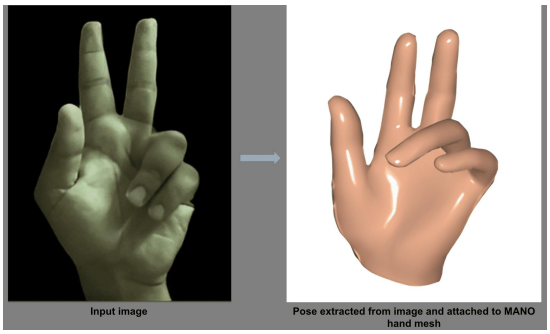

Figure 4.3: Pose extraction and mesh attachment of a frame

The approach described in MRHSMoCap was evaluated using four public datasets: RHD, STB, DO, and ED. Evaluation metrics included the percentage of correct 3D key points (PCK) and the area under the PCK curve (AUC) with thresholds from 20mm to 50mm. The evaluation of the above datasets resulted in the following AUC values for PCK: DO = 0.891, ED = 0.721, STB = 0.732, and RHD = 0.796. These values slightly differ from those described in the paper, but they still demonstrate state-of-the-art performance as mentioned in the paper.

The above-displayed result in Figure 4.3 pertain to a single frame. The joint rotations were extracted from the frame using the MRHSMoCAP network, which took approximately 0.15 seconds. The network predicts rotations for 21 hand joints, resulting in a (21,4)-shaped matrix representing quaternion angles. This matrix is compact, measuring only 2KB, significantly less than the generated data for each frame in the previous approach. This predicted matrix for each frame of the video is transferred to the doctor-client. Then that is applied to the MANO hand mesh in the doctor-client application to obtain the temporal 3D hand actions. Additionally, a texture map is generated for the entire recording, occupying no more than 20MB of space.

Compared to the previous approach, this new method offers several advantages. The data to be sent for a 10-second recording is reduced to 20MB, which is 30 times less than the previous approach. Moreover, the processing time is significantly decreased, making it more efficient. However, this approach has its own drawback, as the size and shape of the generated 3D hand may not precisely match that of the patient's hand. Furthermore, the texture map generated is not sensitive to the muscle movements i.e. it does not account for the dynamic changes in hand texture caused by muscle movements during hand motion. When a human hand moves, the muscles are also in motion, leading to variations in the texture of the hand at different time instances. However, in this approach, only a single texture map is extracted for the entire video, failing to capture the intricate texture changes that occur in each frame. Addressing

this limitation requires the development of sophisticated methods to generate a texture map for each frame, incorporating the texture changes caused by muscle movements. This aspect can be considered as future work for this project. The results of the Texture Extraction step are listed below:

Texture extraction:

The results from complete texture map prediction are as follows
Predictions of the model on the back side hand images of the test data:

Figure 4.4: Left: input back hand images; Middle: Ground truth texture maps; Right: Predicted texture maps

Predictions of the neural network on the palm side hand images of the test data:

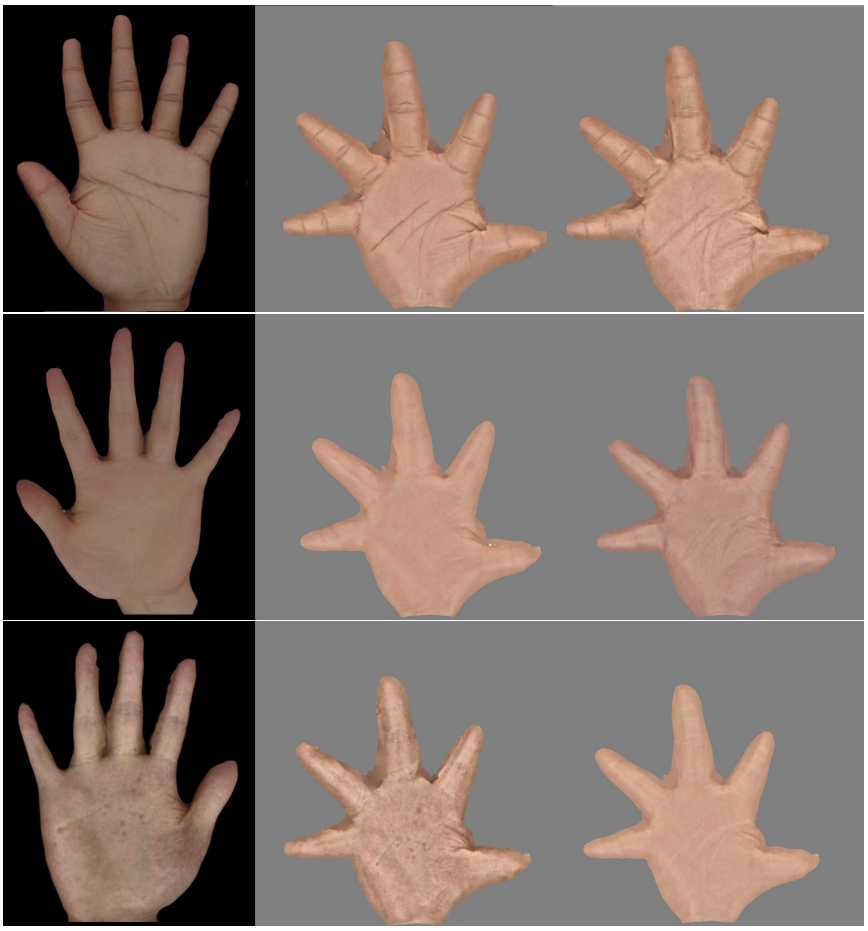

Figure 4.5: Left: inputs palm hand images; Middle: Ground truth texture maps; Right: Predicted texture maps

The model for predicting the texture map of the back of the hand is trained for 300 epochs, and the Mean Squared Error loss on the test data for the best model is 28.306.
The model for predicting the texture map of the palm of the hand is trained for 300 epochs, and the Mean Absolute Error loss on the test data for the best model is 4.193. Both models show limited effectiveness in learning, as their loss values do not decrease significantly beyond the reported values.
The above results indicate that the predicted texture does not precisely match the input hand image. The comparison between the predicted texture maps and the ground truth reveals certain discrepancies. In a medical inspection, it is crucial to accurately represent the patient's hand in 3D, capturing the precise details and texture for a thorough examination by the doctor. Texture mapping plays a vital role in achieving this goal. It is important to note that patients may have specific wounds or conditions on their hands, and it is essential to faithfully depict these elements in the texture map.

Upon analyzing the results in the middle row of Figure 4.4, it is evident that the predicted texture map does not align precisely with the ground truth. Hair, skin patterns, and nerve highlights are missing from the predicted texture maps, which are vital for a comprehensive representation. Similarly, in the first row of Figure 4.5, the positioning of palm lines in the predicted texture map does not match the actual hand images. Additionally, in the second row of Figure 4.5, the falsely predicted palm lines in the texture map for images where they are not present in the input hand images are problematic.

The last row of Figure 4.5 demonstrates a particular medical scenario, where the patient's hand exhibits pimples. However, the predicted texture map fails to accurately represent these pimples, which is unacceptable since clear visibility of such dermatological features is crucial for diagnosis and examination.

Considering these observations, it is evident that the above results are unsatisfactory and do not meet the necessary standards for a reliable medical inspection. Improvements and further research are required to enhance the accuracy and fidelity of the predicted texture maps to ensure precise representation of the patient's hand in medical applications.

Here the model primarily learns the general features of a human hand and attempts to predict the texture by incorporating those features and aiming to closely match their characteristics with the input hand images. However, it is important to note that the model suffers from overfitting, which occurs due to the excessive increase in parameters. Consequently, the model's performance on real-world input images is not satisfactory, as demonstrated below.

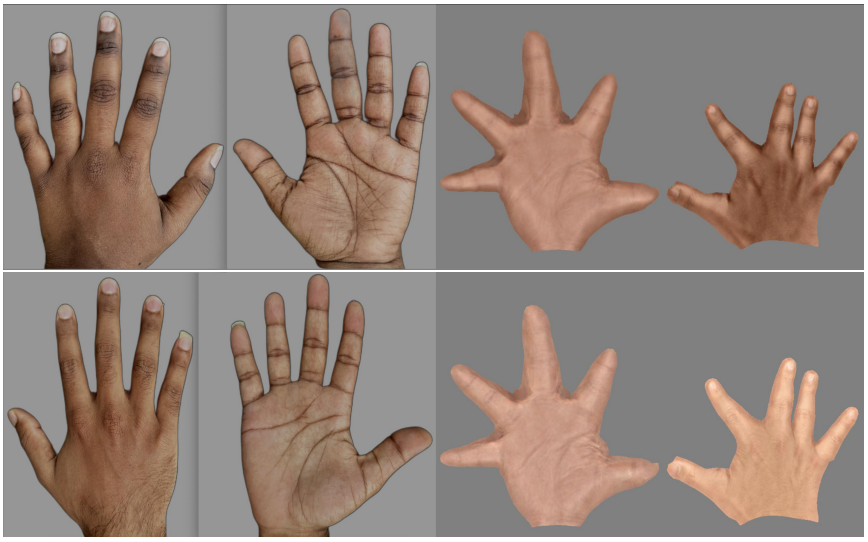

Figure 4.6: Left: front and back hand images; Right: Predicted texture map

The results from partial texture map prediction are mentioned below. The predictions of each neural network model that was built are demonstrated below.

1. Model type 1: any model of this particular type model failed to converge and suffered from impractically long training times.

2. Model type 2:

Figure 4.7: Left: input hand image; Middle: Ground truth texture map; Right: Predicted texture map

Various variations of this model were attempted, but none of them converged even after training for a maximum of 602 epochs. The loss function failed to reach a stable minimum, suggesting ongoing instability in the training process.

3. Model type 3:

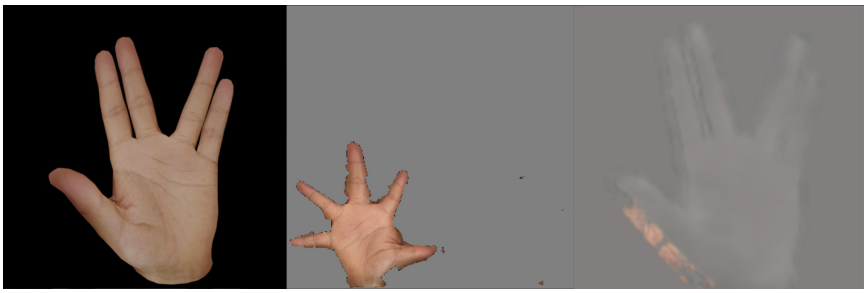

Figure 4.8: Left: input hand image; Middle: Ground truth texture map; Right: Predicted texture map

This type of model's texture map prediction shows a significant deviation from the ground truth, indicating poor performance. It utilizes a modified UNET architecture, which requires larger input image sizes and longer training times. The evaluation using Mean Squared Error loss on the test data yields a high value of 403.34, indicating substantial errors in the predicted texture maps. This falls short of the desired quality standards for accurate texture mapping.

4. Model type 4:

Figure 4.9: Left: input hand image; Middle: Ground truth texture map; Right: Predicted texture map

The training of this particular model was conducted using input images with the background removed. Additionally, a custom loss function was employed, which combines the mean absolute and mean squared loss, with an emphasis on pixels corresponding to the UV map of the MANO mesh. Among all the models within Approach 2, this particular model showed relatively better results. However, the overall performance remains unsatisfactory, as it fails to accurately predict the texture for unobservable regions and lacks detailed texture representation for visible areas. The model underwent 150 epochs of training, resulting in a loss of 56.46 on the test data, which indicates a significant deviation from the desired outcome. This outcome is considered to be of poor quality.

In all of the above models, the results are worse compared to the previous approach. The primary reason for this can be attributed to the inherent difficulty that simple CNN, DNN, and UNET architectures encounter when attempting to learn complex tasks. For instance, when both the complete front and back hand images were provided to the previous approach, it was unable to accurately generate a comprehensive texture map. This emphasizes the additional difficulty of extracting solely the visible texture from a given pose, particularly within this specific scenario.

4.2 Application-level implementations

For the patient's client application, Python was utilized, and for the doctor's client, an android application was developed which runs on the Oculus Quest 2. To ensure the smooth transmission of data from the patient's end to the doctor's application, UDP Protocol was employed. In Approach 1, where the entire 3D hand data is transmitted for each frame, the amount of data involved can be substantial, often exceeding 500MB. As a result, transferring such large volumes of data can be time-consuming and potentially hamper the real-time nature of the application. The application uses approach 2, where Upon recording and processing the hand actions of the patient, a single texture map is generated for the entire recording. Additionally, for each frame of the recording, there is a corresponding quaternion rotation angle for each of

the 21 joints of the hand and shape parameters. This significantly reduces the data size and allows for swift and efficient data transfer between the patient and doctor applications. Upon receiving the transmitted data, the doctor's application proceeds to rotate the joints of the MANO mesh based on the received rotations for each time step. Additionally, the texture map is applied to the mesh to generate a complete and realistic representation of the patient's hand movements. Finally, the resulting visualization is displayed to the doctor, enabling them to assess and analyze the patient's hand actions effectively. The application demo with a generic hand texture can be accessed at this link

Chapter 5

Conclusions and future work

5.1 Conclusion

This project aimed to develop a more realistic remote medical system by leveraging virtual reality (VR) technology. A key requirement of this system was to achieve accurate 3D hand reconstruction. To fulfil this requirement, two approaches were explored.

The first approach, RGB-D-based 3D hand reconstruction, involved reconstructing the entire 3D hand for each frame using depth maps from an RGB-D camera. While this approach showed promising results, it had limitations in terms of processing time, data transfer requirements, and computational demands, making it impractical for real-time applications. To overcome these limitations, a second approach called pose and texture-based 3D hand reconstruction was devised. This approach involved extracting and attaching poses from each frame to a generic 3D hand mesh, along with extracting a texture map to enhance the mesh's realism. This significantly reduced processing time and data transfer requirements, making it more feasible for real-time applications. However, the challenge of accurately mapping the texture from hand images to the 3D hand mesh remained, and deep learning techniques were employed to address this issue. Despite various experiments, achieving satisfactory results proved to be challenging.

5.2 Future work

As part of future work, it is crucial to develop specialized methods for accurately mapping textures from hand images to the 3D hand model. Additionally, addressing the inherent issues in Approach 2 is essential, such as ensuring the size and shape of the 3D hand matches that of the patient's hand and generating texture maps that capture the dynamic changes in hand texture due to muscle movements at any given time instance. These advancements will contribute to improving the overall fidelity and realism of the 3D hand reconstruction system enabling a more

comprehensive and reliable representation of the patient's hand in medical inspections.
This project is a significant step in the field of 3D hand reconstruction for medical inspections, paving the way for future research to refine techniques, improve texture mapping, and capture intricate hand details. With further advancements, this technology has the potential to revolutionize medical inspections, enabling enhanced diagnostics in various fields.

Bibliography

[1] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, 1999. doi: 10.1109/2945.817351. 1, 8

[2] Adnane Boukhayma, Rodrigo de Bem, and Philip H. S. Torr. 3d hand shape and pose from images in the wild, 2019. 11

[3] Philipp Erler, Paul Guerrero, Stefan Ohrhallinger, Niloy Jyoti Mitra, and Michael Wimmer. Points2surf learning implicit surfaces from point clouds. In *European Conference on Computer Vision*, 2020. 8

[4] Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Point2mesh. *ACM Transactions on Graphics (TOG)*, 39:126:1 – 126:12, 2020. 8

[5] Doyeon Kim, Woonghyun Ka, Pyungwhan Ahn, Donggyu Joo, Sehwan Chun, and Junmo Kim. Global-local path networks for monocular depth estimation with vertical cutdepth, 2022. v, 6

[6] Neng Qian, Jiayi Wang, Franziska Mueller, Florian Bernard, Vladislav Golyanik, and Christian Theobalt. HTML: A Parametric Hand Texture Model for 3D Hand Reconstruction and Personalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2020. 14

[7] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6), November 2017. 10

[8] Gerald Stollnberger, Christiane Moser, Manuel Giuliani, Susanne Stadler, Manfred Tscheligi, Dorota Szczesniak-Stanczyk, and Bartlomiej Stanczyk. User requirements for a medical robotic system: Enabling doctors to remotely conduct ultrasonography and physical examination. pages 1156–1161, 08 2016. doi: 10.1109/ROMAN.2016.7745254. ii

[9] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows, 2019. 1

[10] Yuxiao Zhou, Marc Habermann, Weipeng Xu, Ikhsanul Habibie, Christian Theobalt, and Feng Xu. Monocular real-time hand shape and motion capture using multi-modal data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 0–0, 2020. v, 11, 12