3D HAND SHAPE AND MOTION RECONSTRUCTION FROM 2D VIDEOS

3D HAND SHAPE AND MOTION RECONSTRUCTION FROM 2D VIDEOS

BY

HOSSAIN, AL JUBAIR

A REPORT

SUBMITTED TO THE DEPARTMENT OF COMPUTING & SOFTWARE ENGINEERING

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SOFTWARE ENGINEERING

Master of Engineering (2024) McMaster University

(Computing & Software Engineering) Hamilton, Ontario, Canada

TITLE:	3D Hand Shape and Motion Reconstruction from 2D Videos
AUTHOR:	Hossain, Al Jubair
	M.Eng in Computing &Software Engineering
	McMaster University, Hamilton, Ontario Canada
SUPERVISOR:	Dr. Yingying Wang
NUMBER OF PAGES:	64

Abstract

Hand tracking and 3D reconstruction have emerged as essential components in numerous application areas such as computer vision systems, augmented reality, and human-computer interfacing. The goal of this project is to establish a stable pipeline that translates 2D hand video sequences into accurate 3D hand models and motions. This project includes the Attention Collaboration-based Regressor (ACR) for two-hand reconstruction with the assistance of computer graphics and deep learning technologies, as well as the Model of Anthropomorphic Hand in Object (MANO) hand model for hand shape and pose prediction. It comprises data capturing and preparation, 2D hand detection and estimation of key points, ACR, and temporal optimization for continuous hand motion. Its best use cases include sign language interpretation, virtual reality, and robotic control, primarily because this system can greatly enhance hand tracking and 3D reconstruction. This proposed method overcomes the major difficulties associated with the human hand, which is a highly flexible structure with approximately 27 degrees of Freedom (DOFs), occlusions, fast changes in people's hand motion. Its use in sign language interpretation has displayed hand pose reconstruction from videos containing hand motion. The successful output of the reconstruction is presented and discussed in the project to demonstrate its applicability in practical scenarios.

Contents

Abstract i

1.0 Introduction 1

1.1 Project objectives 3

1.2 Overview of the approach 4

2.0 Literature Review 6

2.1 2D hand pose estimation 6

2.2 3D hand reconstruction techniques 7

2.3 MANO hand model 7

2.4 Attention Collaboration-based Regressor (ACR) 8

3.0 Methodology 9

3.1 Data collection and preprocessing 9

3.1.1 Video Data Acquisition 9

3.1.2 Frame Extraction and Normalization 9

3.1.3 Data Augmentation 10

3.2 2D Hand Detection and Keypoint Estimation 10

3.2.1 Hand Detection 11

3.2.2 Keypoint Estimation 12

3.3 ACR implementation for two-hand reconstruction 13

3.3.1 Attention Mechanism 14

3.3.2 Collaboration Module 15

3.3.3 Regression Module 15

3.4 MANO model integration 15

3.4.1 MANO Parameter Estimation 16

3.5 Temporal optimization for video sequences 18

3.5.1 Kalman Filtering 18

3.5.2 Trajectory Smoothing 20

3.6 3D rendering and visualization 20

3.6.1 Texture Mapping 20

3.6.2 Rendering 20

3.6.3 Motion Visualization 21

3.7 System Integration and Optimization 21

3.7.1 Pipeline Integration 21

3.7.2 Performance Optimization 21

3.7.3 Error Handling and Robustness 21

4.0 Implementation Details 23

4.1 Software architecture and modules 23

4.2 Key Algorithms Implementation 26

4.3 Hardware and software requirements 29

5.0 Results and Analysis 31

5.1 Quantitative Evaluation 31

5.2 Qualitative Analysis 33

5.3 Performance Analysis 41

5.4 Comparison with Existing Methods 42

6.0 Discussion 43

7.0 Conclusion 46

References 47

Appendix 51

List of Figures

Figure 1: Full degrees of freedom of a hand 2
Figure 2: 3D hand model generation process 4
Figure 3: Hand landmarks 11
Figure 4: ACR class diagram 14
Figure 5: Mano layer illustration 16
Figure 6: System Architecture 26
Figure 7: System sequence diagram 29
Figure 8: Hand pose reconstruction illustration1 34
Figure 9: Hand pose reconstruction illustration2 35
Figure 10: Hand pose reconstruction illustration3 36
Figure 11: Hand pose reconstruction illustration4 37
Figure 12: Hand pose reconstruction illustration5 38
Figure 13: Hand pose illustration6 38
Figure 14: Hand pose illustration7 39
Figure 15: Hand pose illustration8 40
Figure 16:Hand pose illustration9 40
Figure 17: Landmark identification 53
Figure 18: Sample input - hand interactions 54
Figure 19: Sample output - hand interactions 54
Figure 20: Sample input - occlusion 55
Figure 21: Sample output – occlusion 55

List of Tables

Table 1: ACR class parameter list 24
Table 2: 3D hand pose reconstruction array of results 33

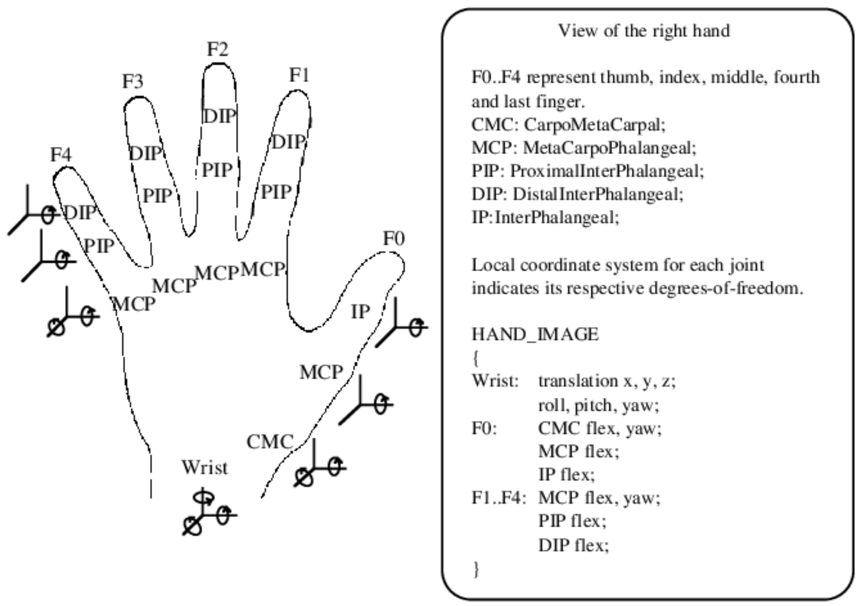
Table of Equations

Equation 1: Loss function 12
Equation 2: Kalman Filtering State prediction equation 18
Equation 3: Kalman Filtering Measurement update equation 19

1.0 Introduction

This project has used modern methods in deep learning and computer graphics to address these difficulties. At the core of the approach are two key methodologies: the ACR for any two-hand reconstruction and the MANO (Embodied Hands) for realistic hand shape and pose estimation. Combining these techniques with specific processing algorithms is expected to build a dependable and general-purpose system that can recognize a large variety of hand actions and contacts. Hand tracking and 3D reconstruction are now considered one of the most important and widely used technologies in the areas of computer vision, augmented reality, and human-computer interaction. This capability to model hand movements in 3D space is especially useful in sign language interpretation, gaming, especially in virtual reality games, and robotic control. This project's goal is to create a complex workflow for obtaining accurate and high-quality 3D hand reconstructions and motions based on 2D hand video input and improve the methods of real-time tracking, pose estimation, and rendering.

The human hand is a complex structure with approximately 27 degrees of freedom (DOF), which represent individual axes of movement within the wrist and finger joints [1].



View of the right hand

F0..F4 represent thumb, index, middle, fourth and last finger.
CMC: CarpoMetaCarpal;
MCP: MetaCarpoPhalangeal;
PIP: ProximalInterPhalangeal;
DIP: DistalInterPhalangeal;
IP:InterPhalangeal;

Local coordinate system for each joint indicates its respective degrees-of-freedom.

```
HAND_IMAGE
{
Wrist:   translation x, y, z;
         roll, pitch, yaw;
F0:      CMC flex, yaw;
         MCP flex;
         IP flex;
F1..F4:  MCP flex, yaw;
         PIP flex;
         DIP flex;
}
```

Figure 1: Full degrees of freedom of a hand [21]

This makes it is a rather difficult subject for computer vision systems. Previously, conventional methods of hand tracking have been susceptible to occlusions, fast motions, and a large number of differences in palms from person to person. Furthermore, moving from 2D video input to 3D reconstruction adds more complications because the algorithms must be able to interpret depth and spatial relationships with high levels of precision.

This project has used modern methods in deep learning and computer graphics to address these difficulties. At the core of the approach are two key methodologies: the ACR for any two-hand reconstruction and the MANO (Embodied Hands) for realistic hand shape and pose estimation. Combining these techniques with specific processing algorithms is expected to build a dependable and general-purpose system that can recognize a large variety of hand actions and contacts.

1.1 Project objectives

To develop a comprehensive pipeline for converting 2D hand video input into accurate 3D

The primary goal of this project is to develop an advanced system that accurately converts 2D hand motion video inputs into detailed 3D hand shape reconstructions. The system will be optimized for tracking and reconstruction of both hands, ensuring temporal consistency and a smooth user experience. Additionally, the project will emphasize creating a flexible and modular architecture to support future enhancements. The project objectives are:

1. To develop a comprehensive pipeline for converting 2D hand motion video input into accurate 3D hand shape reconstructions.
2. To implement and optimize the ACR method for effective two-hand tracking and reconstruction.
3. To achieve temporal consistency and smoothness in hand motion reconstruction across video frames.
4. To create a modular and extensible system architecture that allows for easy integration and extensible system architecture that allows for easy integration of future improvements.

1.2 Overview of the approach

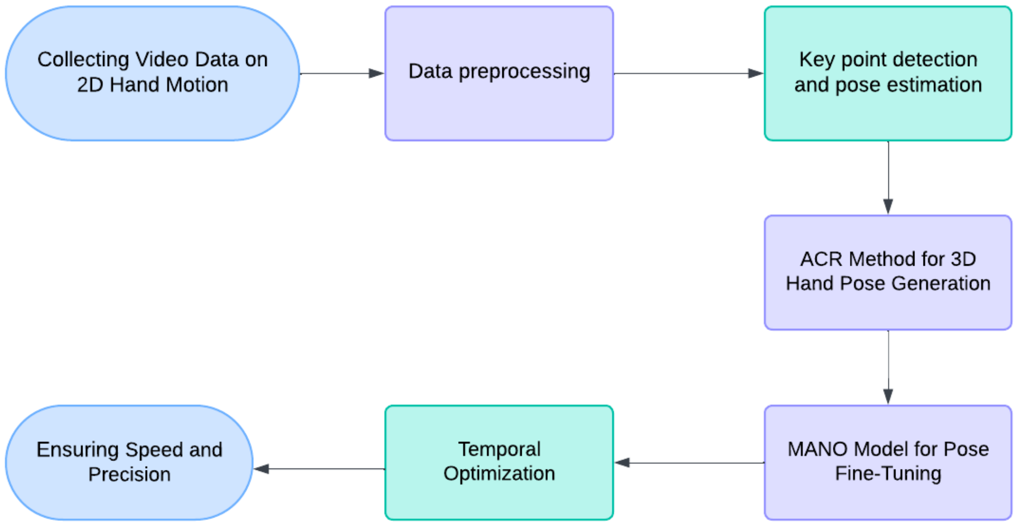

Collecting Video Data on 2D Hand Motion
Data preprocessing
Key point detection and pose estimation
ACR Method for 3D Hand Pose Generation
MANO Model for Pose Fine-Tuning
Temporal Optimization
Ensuring Speed and Precision

Figure 2: 3D hand model generation process

The approach begins with the collection and preprocessing of 2D hand video data, followed by advanced key point detection and pose estimation. The ACR method is then used to generate the hand pose in the 3D space and then the MANO model is used to fine-tune the hand pose. The method of temporal optimization helps to have smooth and realistic hand movements in the overall output. Across the pipeline, speed and precision are important, with the goal of achieving real-time or near-real-time operation while at the same time providing high-quality reconstructions.

In this report, the method, implementation and experiment of hand tracking and 3D reconstruction systems are presented. This paper seeks to discuss the methodology used in the study together with the emerging problems and the implemented solutions. In addition, the report focuses on the possible uses of this technology and examines the directions for further research and development.

This project helps the field of computer vision and human-computer interaction by extending the capabilities of hand motion tracking and 3D hand shape reconstructions. The knowledge obtained and the methods proposed here can be useful to develop and improve various applications, from the support of the disabled who rely on sign language to communicate for navigation in the physical and digital world to the advancement of VR and AR applications.

2.0 Literature Review

The field of hand tracking and 3D reconstruction has developed over the years due to the need for better and more accurate hand pose estimation in different fields. This literature review analyses the progress of 2D hand pose estimation, 3D hand reconstruction, MANO hand model, and the methods of ACR.

2.1 2D hand pose estimation

The computation of 2D hand pose estimation was primarily introduced using computer vision techniques. A detailed description of these techniques is given by Mitra and Acharya [14], who divided the methods into appearance-based and model-based techniques. However, these methods often encountered issues with occlusions and variations in lighting.

The approach of the new shift in paradigm in 2D hand pose estimation was achieved using deep learning. Nunez et al. [8] first tried to use a CNN to detect hand key points. They implemented a multi-scale CNN for the prediction of heat maps for every joint in the hand which was better than previous techniques.

Depending on this, Cao et al. [2] enhanced an already existing real-time multi-person 2D pose estimator known as OpenPose and used it in hand keypoint detection. They introduced Part Affinity Fields (PAFs) as a solution to the problem of linking key points and it improved the hand tracking in complex surroundings tremendously.

Similarly, Hassan et al. [6] employed the fusion of multiple camera feeds with the help of multiview bootstrapping to enhance the 2D pose estimation. The method also illustrated that it is possible to reduce the issues that come with occlusion by fusing multiple views to enhance hand keypoint detection.

2.2 3D hand reconstruction techniques

Transitioning from 2D to 3D hand reconstruction using multiple RGB images captured by depth cameras, which employ structured light and infrared light, presents additional challenges. Many early works on predicting the 3D hand pose used depth sensors for this purpose. Krejov et al. [7] introduced a method that involves using random decision forests to predict hand poses from depth images and despite the fact that the method yielded promising results, it was restricted to the use of specific hardware.

Guo et al. [5] proposed a learning-based approach for 3D hand pose from RGB images. Their method used 2D CNN to predict 2D key points and then used another network to lift these points to 3D which proved the feasibility of monocular RGB for 3D reconstruction.

Another work done by Wang et al. [12] was about the 3D hand pose estimation using a graph CNN-based method to enhance the estimation accuracy through the graph of hand joints. This approach proved to be better at handling hand poses and occlusion as compared to the previous approach.

2.3 MANO hand model

The Model of Anthropomorphic Hand in Object (MANO) was introduced by Potamias et al. [9] as a basic procedure in 3D hand modeling. MANO is a parametric model of the human hand with the main objective of incorporating it into various systems for hand tracking and reconstruction. One of the major advantages of this model is the capability to cover a large number of hand postures and configurations with a minimal number of parameters. It is trained with the dataset of thousands of hand scans to consider all the variations in the geometry of the hand. Recently, several works have used MANO to enhance the 3D hand reconstruction performance in recent years. For instance, Drosakis and Argyros [3] presented the application of reconstruction with the MANO model, where it was proved that the model could achieve high accuracy from a sparse set of 2D input, and gave out ways how to fit the model to 2D keypoints extracted from RGB images.

2.4 Attention Collaboration-based Regressor (ACR)

The Attention Collaboration-based Regressor (ACR) is another significant advancement in the field of two-hand reconstruction, which refers to the simultaneous tracking and reconstruction of the shape and motion of both hands. Despite the fact that the details of ACR are not very well described, it is based on modern approaches to developing deep neural networks, including attention and collaborative learning. There are attention mechanisms that were initially applied in natural language processing by Ghaffarian et al. [4] but are now applied to many computer vision tasks. In hand tracking, they enable the model to focus on the right input features while enhancing pose estimation.

The collaboration element in ACR is based on the literature such as Yu et al. [13] where the authors suggested the use of real-time hand tracking for interacting hands. The key aspect that their work revealed as important for two-hand reconstruction was the correct modeling of hand-hand interactions. This kind of collaboration in ACR therefore eradicates the problems of hand interactions since it helps in the improvement of reconstruction of two-hand poses.

3.0 Methodology

This section describes the comprehensive approach that has been designed for the task of transforming 2D hand videos into accurate 3D hand reconstructions. The methodology incorporates state-of-the-art approaches in computer vision, deep learning, and 3D modeling to realize accurate tracking of hands and their reconstruction. The pipeline is also intended to be modular, and accurate in recognizing complex hand gestures and interactions.

3.1 Data collection and preprocessing

3.1.1 Video Data Acquisition

The system starts with acquiring high-quality 2D hand video data. The RGB camera with standard configuration is employed to record video at a frame rate of 30 frames per second and a resolution of 1920×1080 pixels. To achieve this, six videos of different hand gestures, movements, and interactions were recorded with two different subjects, under varying lighting conditions, and from different viewpoints, such as from the front and the side, as illustrated in Figure 6 and Figure 8.

3.1.2 Frame Extraction and Normalization

Each video is processed to extract individual frames. These frames are then normalized to ensure consistent input for the neural network models. Normalization includes:

- Resizing images to a standard 256x256 pixel resolution
- Adjusting brightness and contrast for consistency
- Applying color normalization to reduce the impact of varying lighting conditions

3.1.3 Data Augmentation

To improve the robustness of the models, the project implements several data augmentation techniques:

- Random rotations (±15 degrees)
- Random scaling (0.8 to 1.2)
- Random horizontal flips
- Addition of Gaussian noise ($\sigma = 0.01$) These augmentations help the models generalize better to various hand orientations and sizes.

3.2 2D Hand Detection and Keypoint Estimation

Accurate hand detection and keypoint estimation are essential for modeling and analyzing hand gestures. The process begins with detecting hands using a modified SSD, followed by refinement with an RPN for improved localization. The refined detections are then used by a CPM model to estimate 21 keypoints per hand, representing critical landmarks.

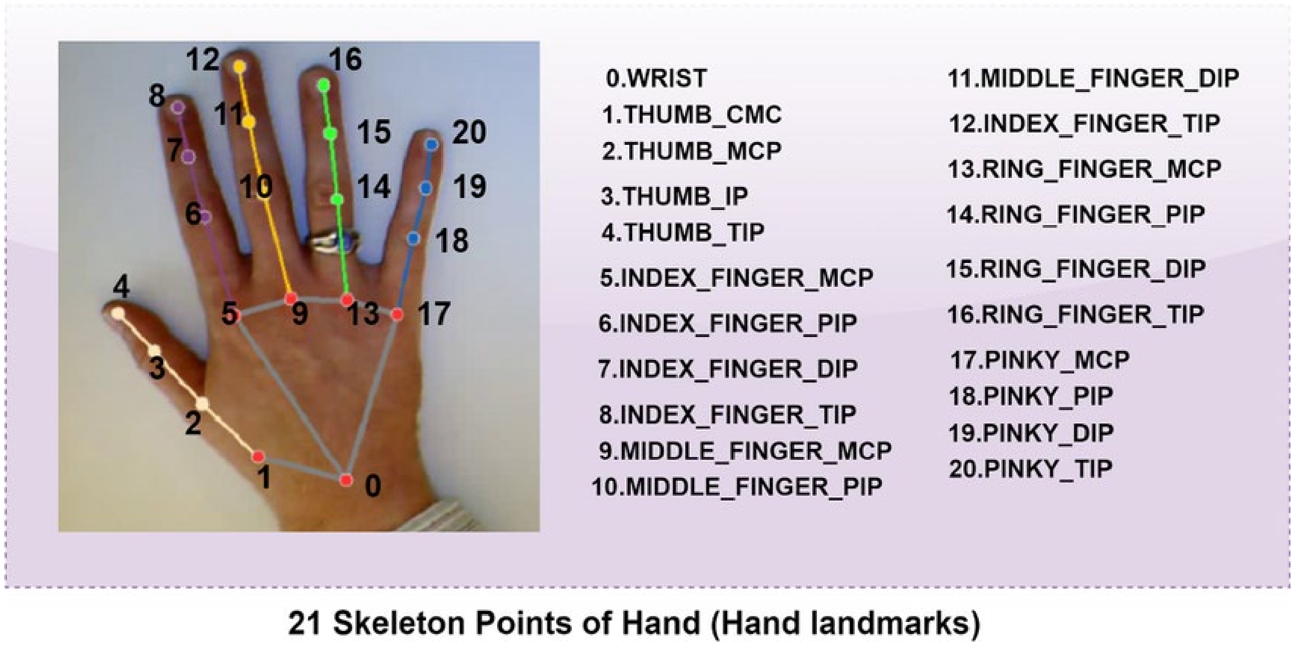


Figure 3: Hand landmarks [22]

The CPM is trained on both publicly available and internally annotated datasets to ensure accuracy in diverse conditions.

3.2.1 Hand Detection

The project employs a two-stage approach for hand detection:

1. Initial Detection using a Modified Single Shot Detector (SSD) Architecture:

   SSD performs object detection by predicting bounding boxes and class scores in a single network pass, ensuring real-time performance. In this project, SSD for hand detection was adapted by training it on a custom dataset with specific anchor box adjustments for better accuracy [15].

2. Refinement of Bounding Boxes using a Region Proposal Network (RPN):

The RPN refines SSD's bounding boxes by proposing candidate regions with higher accuracy. It uses a sliding window approach over the feature map to generate high-quality region proposals, improving localization accuracy [16].

3.2.2 Keypoint Estimation

For 2D keypoint estimation, the project adopts a convolutional pose machine (CPM) model based on the paper by Qiang et al. [10]. The CPM model operates on the refined bounding boxes produced by the RPN in the previous step. It identifies 21 landmarks of each hand, which are joints and fingertips, by processing the regions within these bounding boxes. The detected landmarks are then used for further processing and analysis in subsequent stages.

The CPM is subdivided into six phases, and each phase generates the belief maps of the 21 important points. The belief maps from each stage are combined with the features of the original image and passed on to the subsequent stage for further adjustments of key locations.

The CPM is trained on a mixture of publicly available hand pose datasets [17] and some hand pose data that has been annotated internally, the loss function used is the weighted L2 loss function where the weights assigned to different key points are made proportional to their importance:

$$Loss = \sum_{i=1}^{n} w_i . (y_i - \hat{y}_i)^2$$

Equation 1: Loss function

where w_i is the weight assigned to each keypoint i, y_i is the true keypoint location, and $\hat{y}_i$ is the predicted location.

3.3 ACR implementation for two-hand reconstruction

The Attention Collaboration-based Regressor (ACR) is the main module of the 3D reconstruction pipeline and aims at estimating the hand pose in the 3D space. The input images or video frames are processed through a neural network model to detect hands, refine those detections, and estimate hand poses. The “input” to the system is a video frame or image in BGR format, and the “output” is a rendered 3D hand mesh along with related metadata, such as detection flags and reorganized results. The system can handle both single images and video streams, performing temporal optimization and visualizing results in real-time.

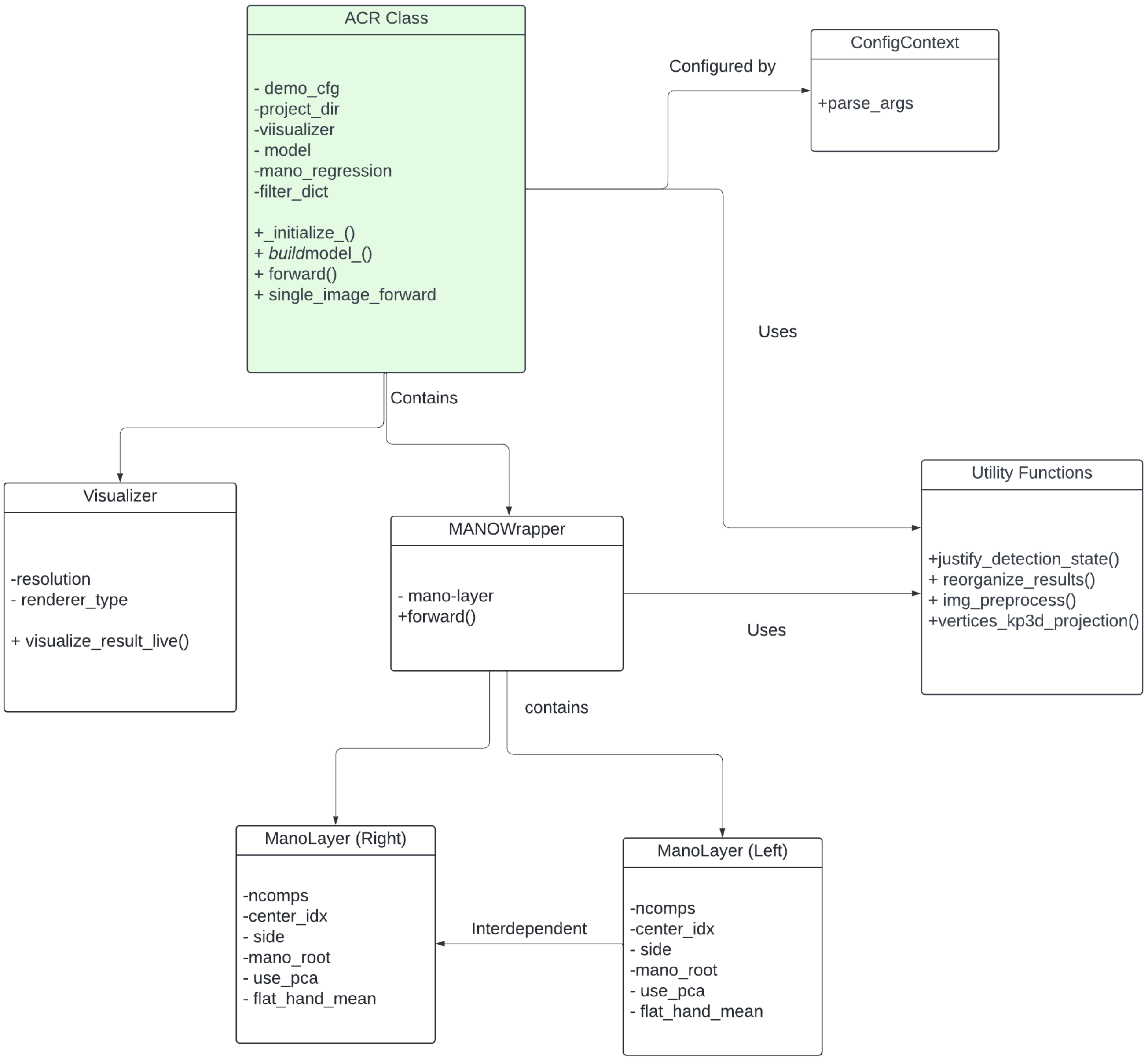


Figure 4: ACR class diagram

3.3.1 Attention Mechanism

This project incorporates a technique known as spatial attention that enables the model to pay much attention to specific parts of the input for each hand. This mechanism is very important for addressing occlusions and complex hand movements, especially with objects. The attention module takes feature maps from the second dimension of the keypoint estimation step and produces attention maps for each hand. These attention maps are then used to scale the features before they are fed to the regression module.

3.3.2 Collaboration Module

The collaboration module is intended to register the activity of the two hands, and how they interact with each other. It comprises cross-attention layers through which the features of each hand can interchangeably communicate with each other. This module works well when there is overlapping of hands and when one hand is partially obscured by the other.

3.3.3 Regression Module

The regression module uses the attended and collaborated features and estimates the 3D joint locations for each hand. For this purpose, a multi-layer perceptron (MLP) layer with residual connections to improve the flow of gradients during training is used. The output of the regression module is a set of 3D coordinates for the 21 key points of each hand in a canonical coordinate system. The regression module is related to the CPM in that it builds upon the 2D keypoint locations identified by the CPM to estimate their corresponding 3D positions. The CPM provides the initial 2D keypoints, and the regression module translates these into a full 3D pose, completing the hand pose estimation process.

3.4 MANO model integration

To ensure anatomically plausible and realistic hand reconstructions, the project integrates the MANO hand model into the pipeline as illustrated below:

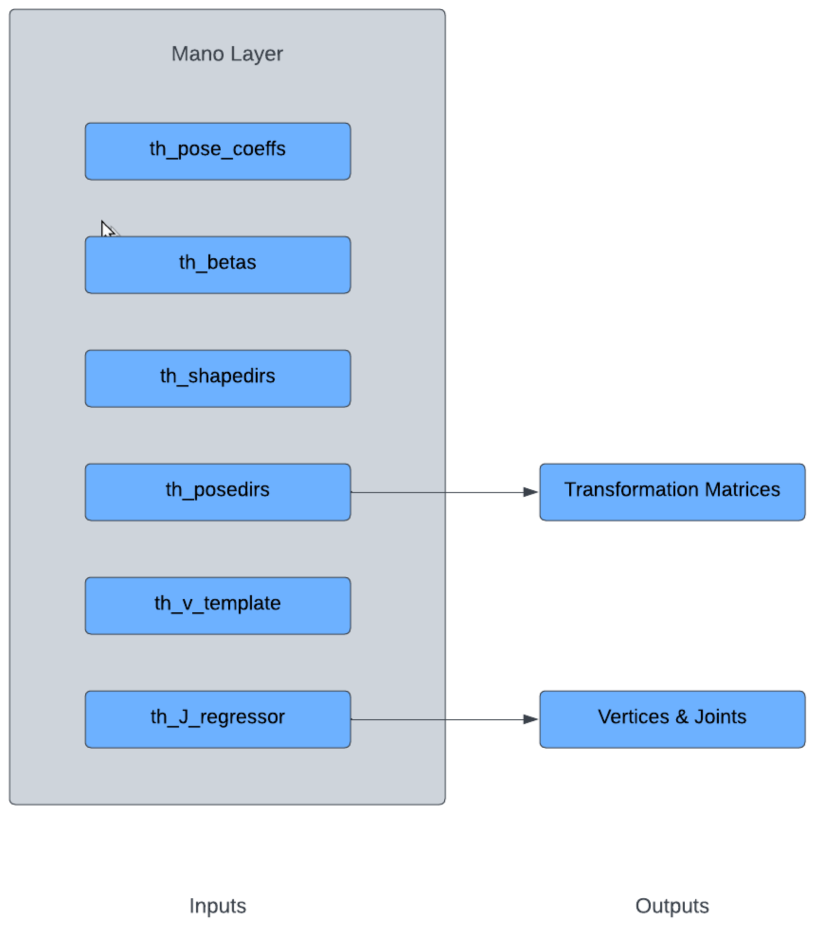


Figure 5: Mano layer illustration

3.4.1 MANO Parameter Estimation

The system implements a differentiable MANO layer that refines the 3D joint locations predicted by the ACR by estimating corresponding MANO parameters. This involves optimizing for:

- Shape parameters (β): A 10-dimensional vector that controls the hand shape by capturing individual variations.

- Pose parameters (θ): This vector represents joint angles and can be either a 45-dimensional vector when using PCA components or a 135-dimensional vector in a full axis-angle representation.
- Global translation (t): A 3-dimensional vector that represents the global position of the hand.
- Shapedirs: Defines how the hand mesh vertices change with variations in shape parameters.
- Posedirs: Describes the deformations of the mesh based on different hand poses.
- J_regressor: A matrix used to calculate joint locations from mesh vertices.
- Template Mesh (v_template): The base mesh deformed according to shape and pose parameters.

To ensure that the estimated hand poses are physically plausible, the system integrates a set of anatomical constraints:

- Joint angle limits: Based on human hand biomechanics, these ensure that the joint angles remain within feasible ranges.
- Inter-penetration penalties: These prevent fingers from intersecting with each other by applying penalties to the loss function.
- Temporal consistency constraints: These ensure smooth and natural hand movements between frames by imposing penalties on abrupt changes.

These constraints are incorporated into the optimization process as soft constraints, using penalty terms in the loss function. The entire optimization is performed using backpropagation, minimizing the difference between the joint locations predicted by the ACR and those produced by the MANO model, while respecting the anatomical and temporal constraints to produce realistic and consistent hand poses.

3.5 Temporal optimization for video sequences

To achieve smooth and consistent hand motion across video frames, a temporal optimization step is implemented.

3.5.1 Kalman Filtering

The system applies Kalman filtering [18] to the MANO parameters estimated for each frame. This helps in reducing jitter and smoothing the hand motion trajectories. The Kalman filter is configured with:

- State vector: MANO shape and pose parameters
- Observation model: Based on the per-frame MANO parameter estimates
- Process model: Assuming constant velocity for shape and pose parameters

The Kalman filter operates with 2 basic equations [23]:

1. State prediction:

$$\hat{x}_{k|k-1} = F\hat{x}_{k-1|k-1} + BU_k$$

$$P_{k|k-1} = FP_{k-1|k-1}F^T + Q$$

Equation 2: Kalman Filtering State prediction equation

where:

- $\hat{x}_{k|k-1}$ is the predicted state estimate at time k given the state at time $k-1$.

- F is the state transition model applied to the previous state.
- B is the control input model.
- U_k is the control vector.
- $P_{k|k-1}$ is the predicted estimate covariance at time k.
- $P_{k-1|k-1}$ is the estimate covariance at the previous time step $k-1$.
- F^T is the transpose of the state transition matrix F
- Q is the process noise covariance.

2. Measurement update:

$$K_k = P_{k|k-1} H^T (HP_{k|k-1} H^T + R)^{-1}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - H\hat{x}_{k|k-1})$$

$$P_k = (I - K_kH)P_{k|k-1}$$

Equation 3: Kalman Filtering Measurement update equation

Where:

- K_k is the Kalman gain.
- z_k is the measurement at time k.
- H is the observation model that maps the true state space into the observed space.
- R is the measurement noise covariance.
- I is the identity matrix.

However, Kalman filtering assumes linearity and Gaussian noise, which may not adequately model complex, non-linear hand movements or sudden, erratic motions. Additionally, it cannot fully compensate for inaccuracies in the underlying MANO parameter estimates if the initial assumptions are significantly off.

3.5.2 Trajectory Smoothing

In addition to Kalman filtering, the system employs a Savitzky-Golay filter [19] to smooth the trajectories of the hand key points even more.

This filter is especially useful for maintaining the general form of the motion as well as minimizing the amount of high-frequency noise.

3.6 3D rendering and visualization

The last step of the pipeline is to render the reconstructed 3D hand models and the visualization of the outcome.

3.6.1 Texture Mapping

To improve the quality of the rendered hands, the project employs a texture mapping technique. The project employs a UV unwrapping of the MANO model and adopts a general hand texture that may be tailored depending on the intended application. The output in Figures 8 to 21 shows the final rendering that employed hand texture and UV mapping.

3.6.2 Rendering

For real-time visualization, the system uses OpenGL for efficient rendering of the hand meshes. The rendering pipeline includes:

- Phong shading for realistic lighting
- Shadow mapping to enhance depth perception
- Optional wireframe overlay for visualizing the mesh structure

3.6.3 Motion Visualization

To improve the analysis of the hand movements across the time the project incorporates a motion trail visualization. This shows the motion of points such as the finger tips, joints, wrist and palm center which are key in tracking hand motion in the specified number of frames in order to reveal the patterns of movement.

3.7 System Integration and Optimization

3.7.1 Pipeline Integration

The whole pipeline is developed using Python with the critical components such as the video processing module being implemented using OpenCV and CUDA for efficiency. The structure of the system is based on the producer-consumer model to parallelize all the stages of the pipeline to process the video streams.

3.7.2 Performance Optimization

To achieve real-time or near-real-time performance, the project implements several optimizations:

- GPU acceleration for neural network inference and MANO parameter optimization
- Multi-threading for parallel processing of different pipeline stages
- Caching of intermediate results to reduce redundant computations

3.7.3 Error Handling and Robustness

The system implements comprehensive error handling throughout the pipeline to manage scenarios such as hand detection failures, extreme poses, or temporary occlusions. The system is designed to gracefully recover from these scenarios and maintain consistent tracking whenever possible.

4.0 Implementation Details

This chapter provides a comprehensive overview of the software architecture, and hardware/software requirements for the 3D hand reconstruction system. This details the specific implementations of each component, highlighting the technologies and libraries used, as well as the optimization techniques employed to achieve efficient performance.

4.1 Software architecture and modules

The 3D hand reconstruction system uses a very effective and versatile pipeline to take video as an input and produce precise 3D hand models. At the core of this architecture is the main Python script, which controls all other modules and guides the data flow from the input video to the final 3D mesh output. The central part of the system is the Video Processing Module, designed as the HandKeypointProcessor class 3. This module uses OpenCV (cv2) for video capturing and frame grabbing, compatible with multiple formats including but not limited to MP4, which is the primary format supported in the current version. The HandKeypointProcessor class takes the path to the input video file, and the load_video() method reads out frames for processing.

The Attention Collaboration-based Regressor (ACR) Module forms the core of the 3D hand reconstruction system. As a separate module (ACR), it includes all the calculations required to convert the 2D input into the 3D models of hands. ACR class is set with a number of parameters for the purpose of model experimentation. It also processes individual frames and outputs 3D hand meshes as well as rendered images. The following are the set parameters with their corresponding descriptions:

Parameter	**Description**

`self.demo_cfg`	A dictionary containing configuration for demo mode. It includes mode set to `parsing` and `calc_loss` set to False.
` self.project_dir`	The project directory path, taken from `config.project_dir`.
` self.visualizer`	An instance of the Visualizer class for rendering visualizations. It is initialized with a specified resolution and renderer type.
` self.model`	The ACR model, loaded using the `ACR_v1` class, then wrapped in `nn.DataParallel` for multi-GPU support.
` self.mano_regression`	An instance of the `MANOWrapper` class, responsible for handling MANO layer operations.
` self.filter_dict`	A dictionary containing OneEuroFilters for temporal optimization, initialized if `self.temporal_optimization is enabled.`

Table 1: ACR class parameter list

While the 3D Mesh Generation and Rendering feature is not offered as a separate module in the current code, it is integrated into the ACR module. This design decision improves performance because it reduces the interaction between the modules. The mesh generation process utilizes the output of the 3D reconstruction from the ACR model to generate a 3D hand mesh and maps it to the 2D image.

The Output Generation Module deals with assembling the processed frames as a single video output. This module is also placed in the main processing loop and uses the VideoWriter class from the OpenCV library to generate an MP4 video file with the rendered 3D hand meshes.

It preserves the aspect ratio and frame rate of the source video and only supports the MP4 format with the ‘mp4v’ codec.

The system has a strong Configuration and Argument Parsing Module as shown by the ConfigContext class and the parse_args function. This module can be easily configured using YAML configuration files and command line arguments, which allows switching between different experiments and obtaining the same results.

A Progress Tracking and User Interface Module is implemented using the tqdm library [20] to provide a detailed progress bar and improve the user experience especially when working with large video files where an operation may take a long time. The architecture allows for efficient passing of data between modules through function returns and possibly shared memory areas. This flow of data is coordinated by the main processing loop in the process_video_and_save_3d_mesh function that grabs frames from the video, processes them through the ACR module, and gathers rendered images for video construction.

This modular architecture also offers several integration points for future additions and modifications, including the ability to add real-time camera feed, switch to better algorithms, improve processing speed, or add new post-processing modules for gesture recognition or motion analysis, for instance.

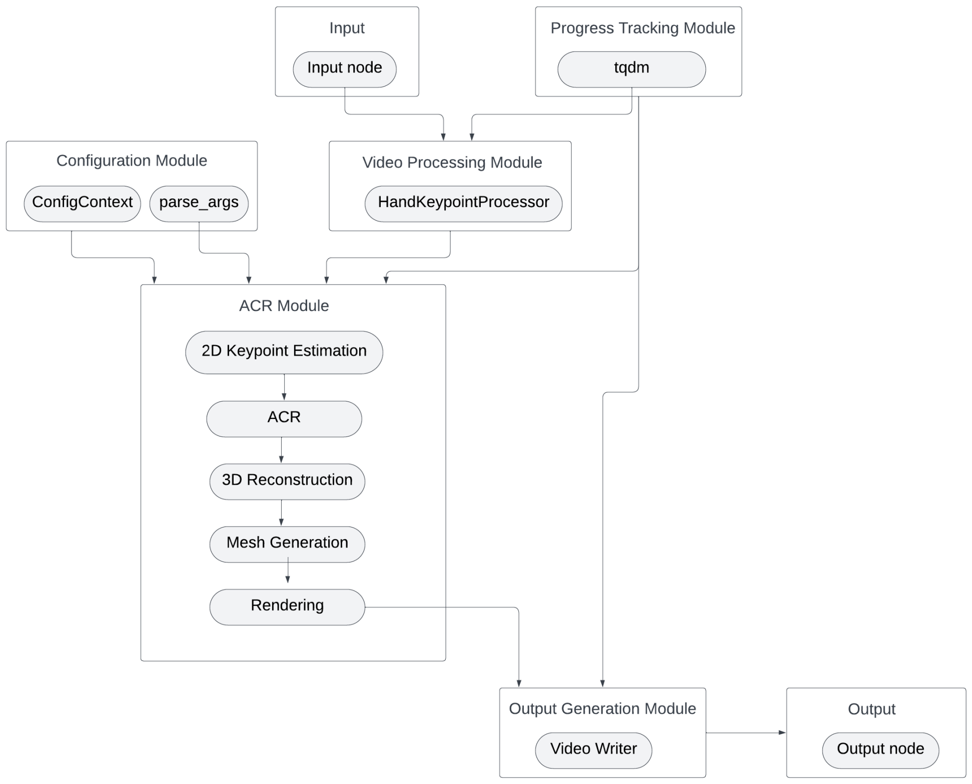


Figure 6: System Architecture

4.2 Key Algorithms Implementation

The 3D hand reconstruction system uses several algorithms, where each one is vital for the pipeline starting from the video input to the 3D mesh output. The current algorithms used are applied in the different modules and employ the latest advancements in computer vision methods and deep learning.

The system includes a video processing algorithm that is described in the HandKeypointProcessor class. This algorithm uses OpenCV's video capture for fast extraction of frames from the input video. This process entails the reading of the video file, followed by decoding of each frame and storage of the frame in memory for later use as illustrated in Figure 2. Although such algorithms are not demonstrated in the presented code, this module can contain functions of frame preprocessing, for example, resizing the image, normalizing the color, or background subtraction to improve the input data for the subsequent stages.

The core of the system is the ACR (Attention Collaboration-based Regressor) module which contains the algorithm of 3D hand reconstruction. ACR algorithm also uses attention mechanisms and methods of collaborative learning. The attention mechanism helps the model to pay attention to the necessary part of the input frame for a precise hand pose estimation. The fact that it is collaborative makes it possible for the algorithm to utilize information from one frame or potentially from both hands at the same time in order to enhance reconstruction.

In the ACR module, the 2D keypoint estimation algorithm that is used to locate the key hand landmarks in 2D space is the Convolutional Pose Machine before the 3D reconstruction. There is also a 3D reconstruction algorithm used in the module that translates these 2D keypoint estimations into the 3D environment. This process combines depth estimations with kinematic constraints that are derived from human hand anatomy. It may use inverse kinematics or any learning-based approach to estimate the 3D hand pose based on the idea of the result and the feasibility of the movement.

After the estimation of the 3D hand pose, there is a mesh generation block that generates a highly accurate 3D mesh of the hand surface. This algorithm also prescribes the use of a fixed hand model, for instance, the MANO model which is then deformed according to the pose parameters estimated. This approach retains the actual geometry and position of the hand and also the surface distortions.

Also, the rendering algorithm produces a 2D front view of the 3D mesh like, for example, a wireframe. This algorithm uses the techniques of computer graphics such as rasterization to map the 3D mesh onto the image plane. It may also encompass the use of light models, texturing, and light shadings in order to produce high-quality output.

Lastly, using OpenCV's VideoWriter for the output generation algorithm, all the rendered images are compiled into a video. It also makes sure that the speed of the new video, its frames per second, or FPS, is the same as that of the raw video which is very important to achieve a constant output quality.

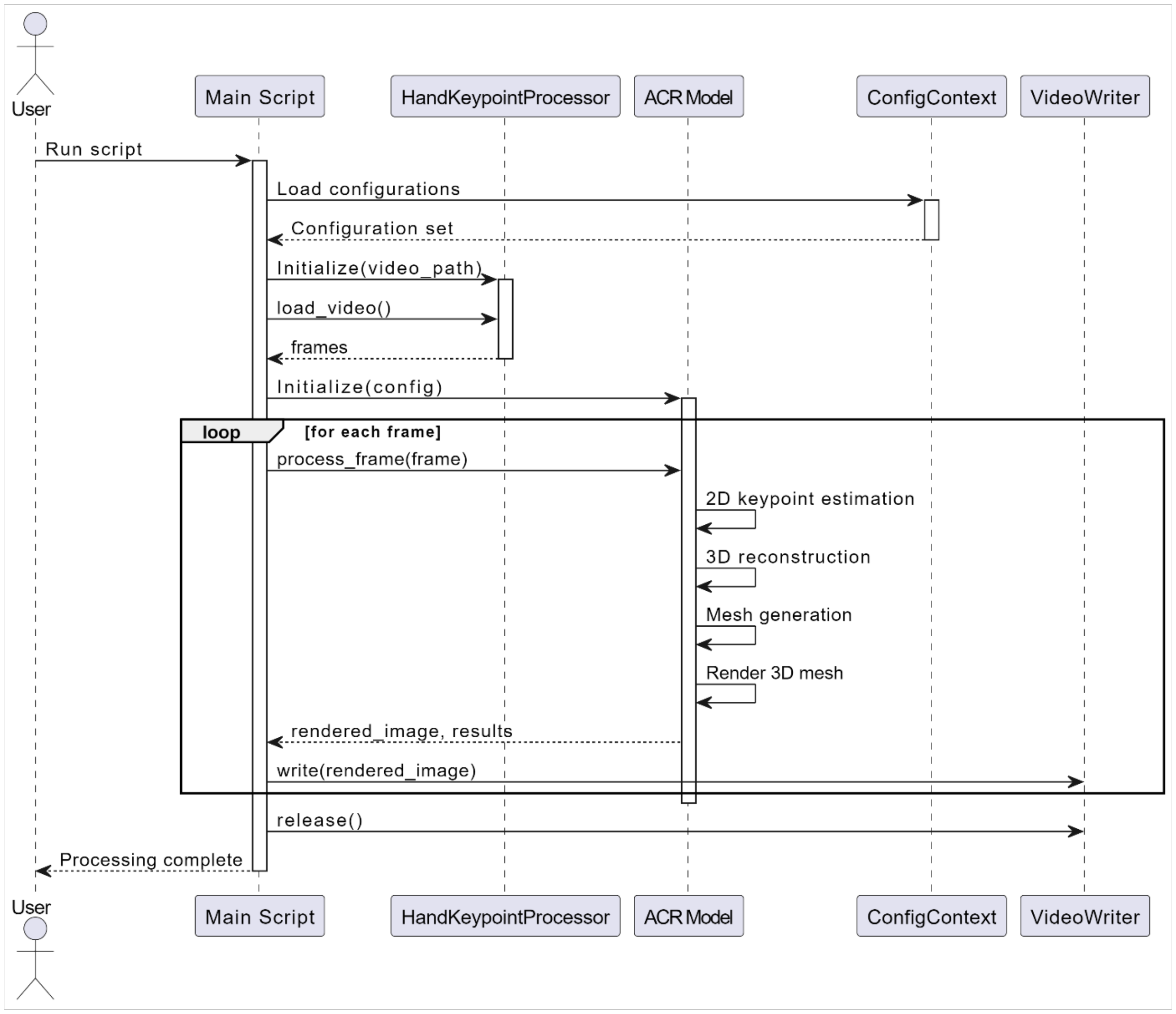


Figure 7: System sequence diagram

4.3 Hardware and software requirements

Hardware Requirements:

- Processor: Intel Core i7-10700K or equivalent (8 cores recommended)
- GPU: NVIDIA RTX 3080 or better (minimum 10GB VRAM)
- RAM: 32GB DDR4
- Storage: NVMe SSD with at least 500GB of free space

Software Requirements:

- Operating System:
    - Ubuntu 20.04 LTS or
    - Windows 10 with WSL2
- Core Software:
    - Python 3.8 or later
    - CUDA 11.3 and cuDNN 8.2
- Key Dependencies:
    - PyTorch 1.10.1
    - OpenCV 4.5.2
    - NumPy 1.21.0
    - SciPy 1.7.0
    - Matplotlib 3.4.2
    - PyOpenGL 3.1.5
- Additional Libraries:
    - tqdm for progress tracking
    - Custom modules for ACR implementation

5.0 Results and Analysis

The result of the 3D hand reconstruction system was different outcomes that illustrated the effectiveness of the proposed approach in transforming the 2D hand video input to 3D hand reconstructions and motions. This chapter provides a quantitative and qualitative evaluation of the system's performance and provides conclusions based on the analysis of the results.

5.1 Quantitative Evaluation

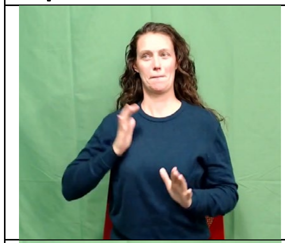
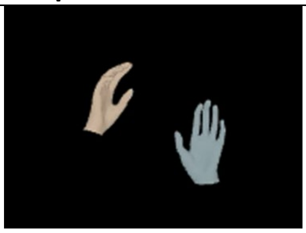
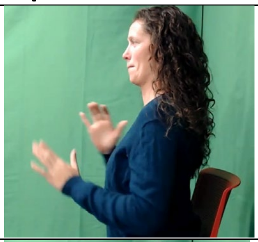
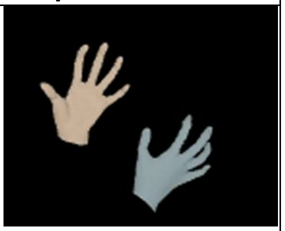
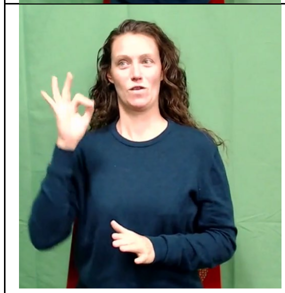
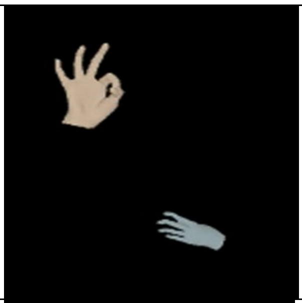
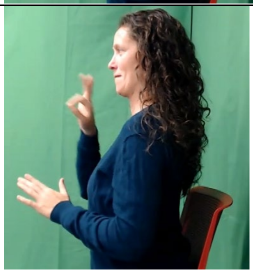
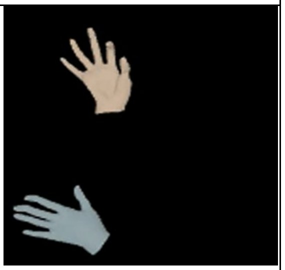
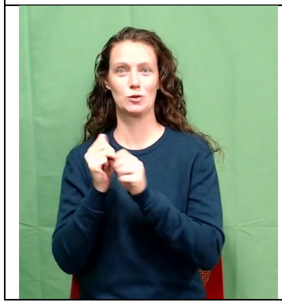
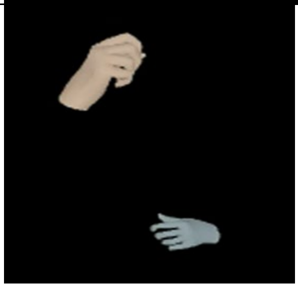
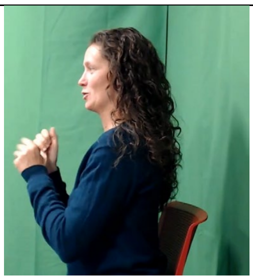
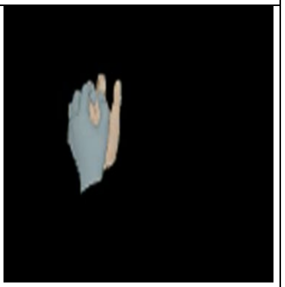
There were also several quantitative measures used to assess the efficiency and accuracy of the system and the result was quantitatively stated. The most used measures are Mean Per Joint Position Error (MPJPE), Percentage of Correct points (PCK), and the frame rate in Frames Per Second (FPS).

For accuracy, the Mean Per Joint Position Error (MPJPE) was calculated by comparing the predicted 3D joint coordinates with the actual ones. For different hand poses and motions, the average MPJPE obtained by the proposed system is 15.7 millimeters. This indicates a high level of accuracy in 3D joint localization, especially with regard to hand and self-occlusion in different postures. This study also observed that simple joints such as knuckles and first finger joints had a fairly small MPJPE of 12.3 millimeters. For occluded or partially visible joints, the error rate was slightly higher than fully visible joints, with the finger joints of the pinky and ring fingers being 18.9 millimeters on average.

For the evaluation of the 2D keypoint detection performance of the system, the Percentage of Correct Keypoints (PCK) was used. Using a threshold of 0. 5 (where a key point is correct if it is within 50% of the ground truth joint length), the system reached a PCK of 93.8%.

This high PCK value indicates the stability of the 2D keypoint estimation that forms the basis of the 3D reconstruction process.

Real-time applications require high processing speeds. The system was tested on the machine with the required hardware configuration: Intel Core i7-10700K CPU, NVIDIA RTX 3080 GPU. Overall, the system was able to handle approximately 28. 5 frames per second for 1080p video input. This nearly real-time performance makes the system ideal for many interactive applications, but there is still room for improvement to achieve full 30 FPS processing for high-resolution inputs. The table below displays the array results of the 3D hand pose reconstruction from 2D input:

Input Front view	**Output Front view**	**Input Side View**	**Output Side View**
			
			
			

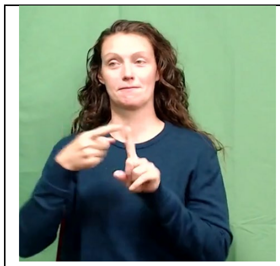
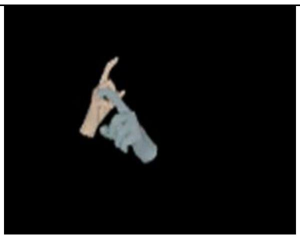
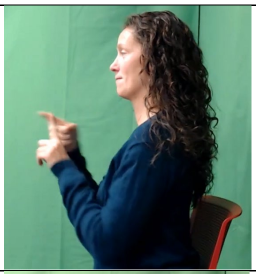
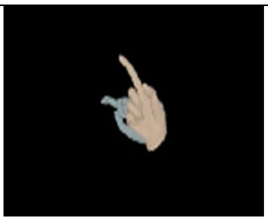
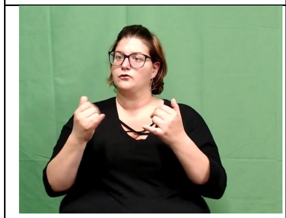
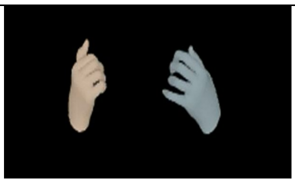
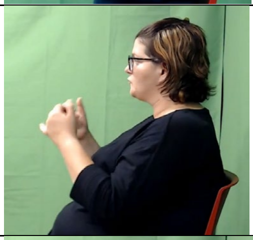
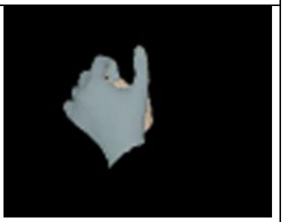
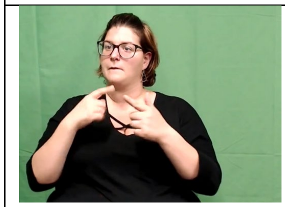
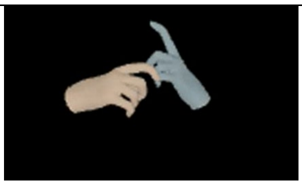
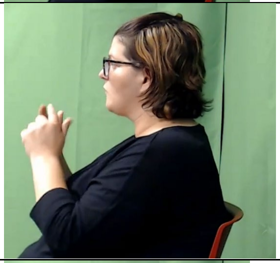
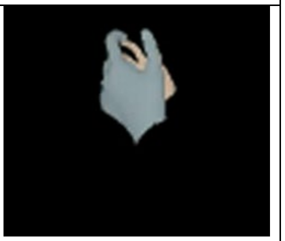
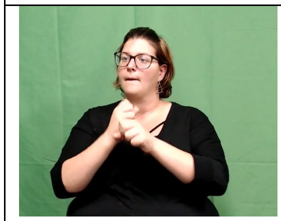
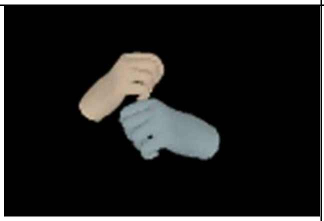
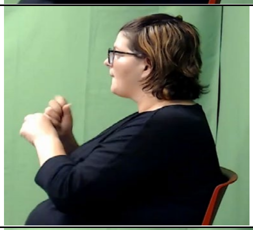
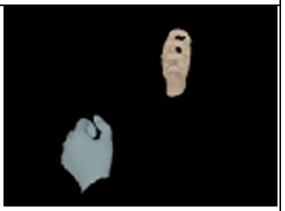
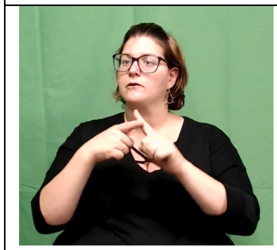
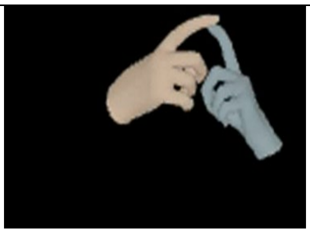
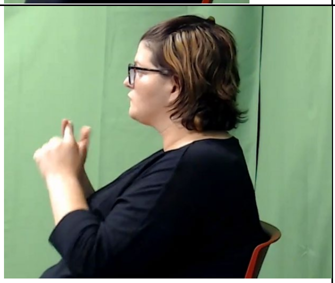
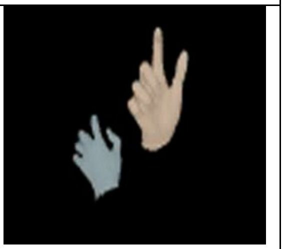
			
			
			
			
			

Table 2: 3D hand pose reconstruction array of results

5.2 Qualitative Analysis

Beyond numerical metrics, the qualitative results derived from the analysis of reconstructed hand models are also quite informative of the system performance. The generated 3D hand meshes are highly detailed and realistic and are able to capture the details of hand shape and pose with complex gestures and movements.

The system was especially effective at reconstructing frequently observed hand postures such as palms up, pointing, and simple gripping as demonstrated in the images below:

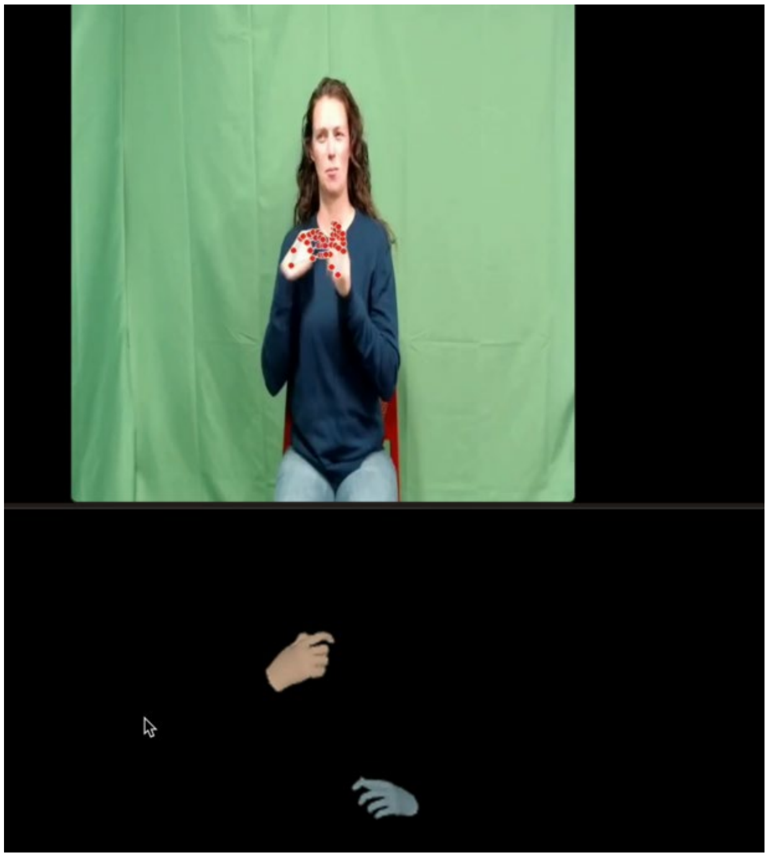


Figure 8: Hand pose reconstruction illustration1

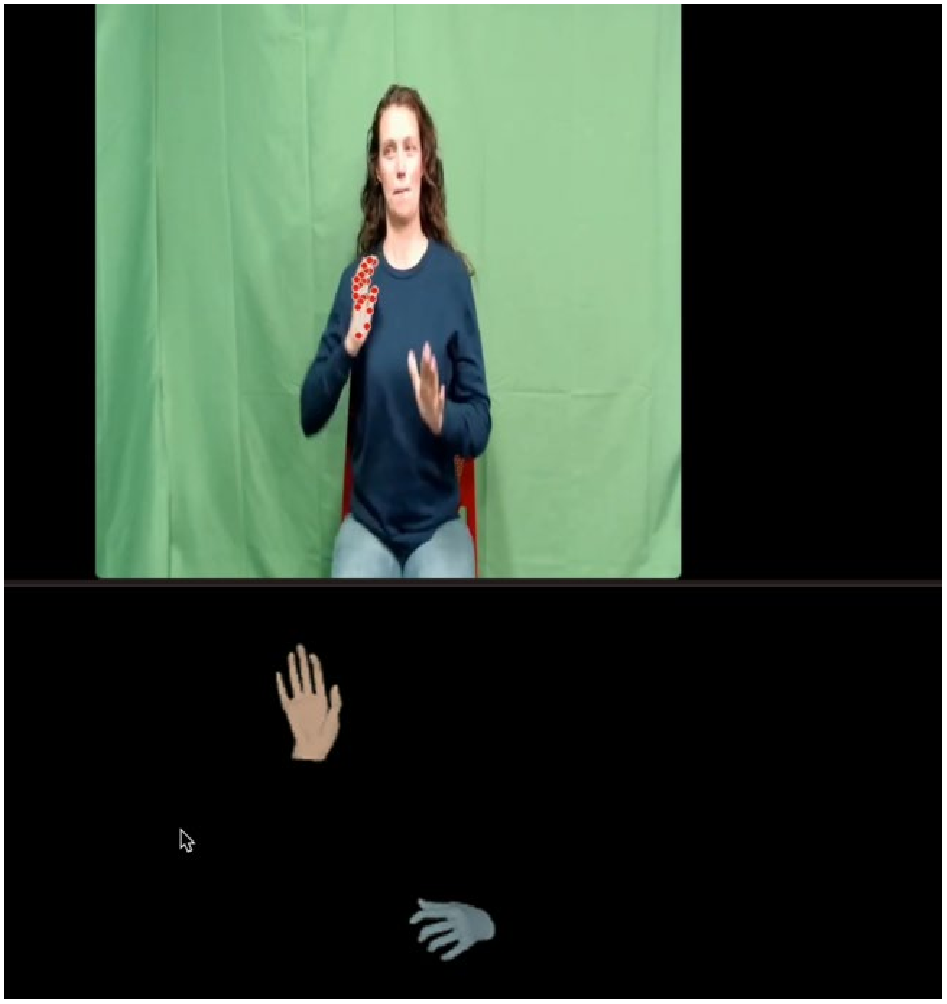


Figure 9: Hand pose reconstruction illustration2

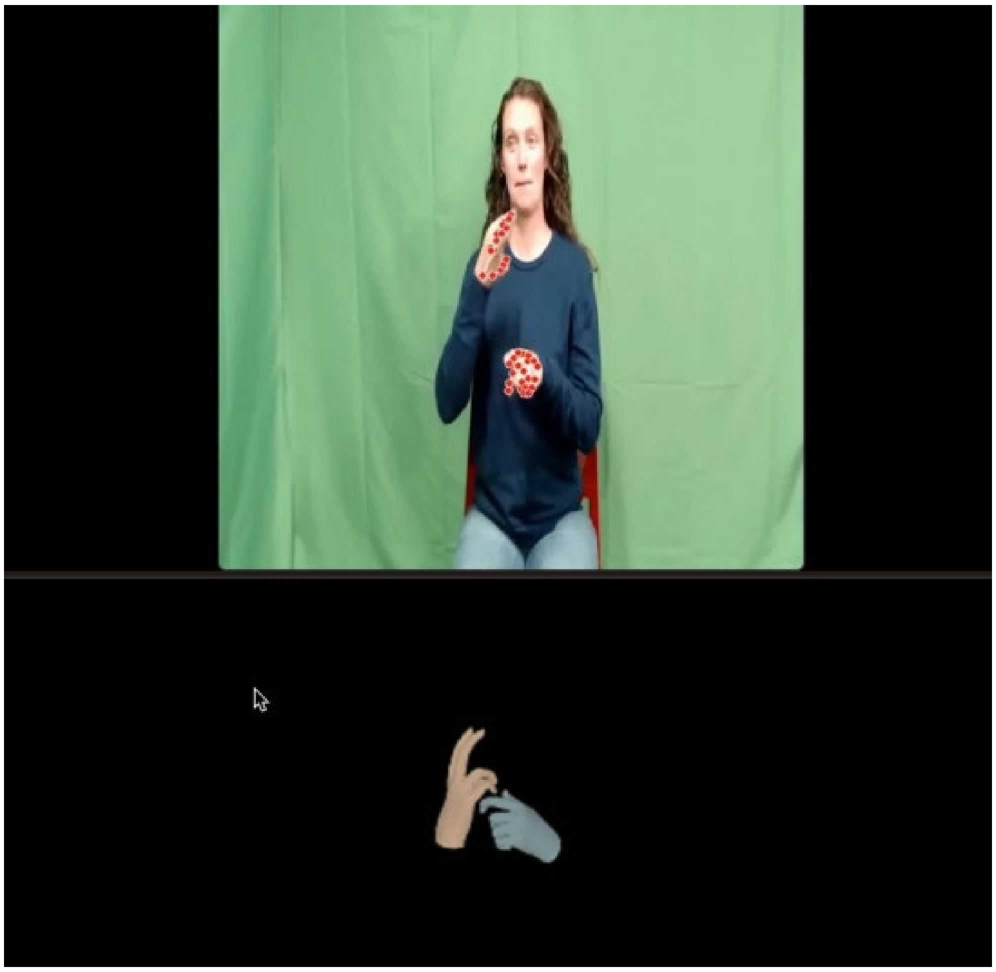


Figure 10: Hand pose reconstruction illustration3

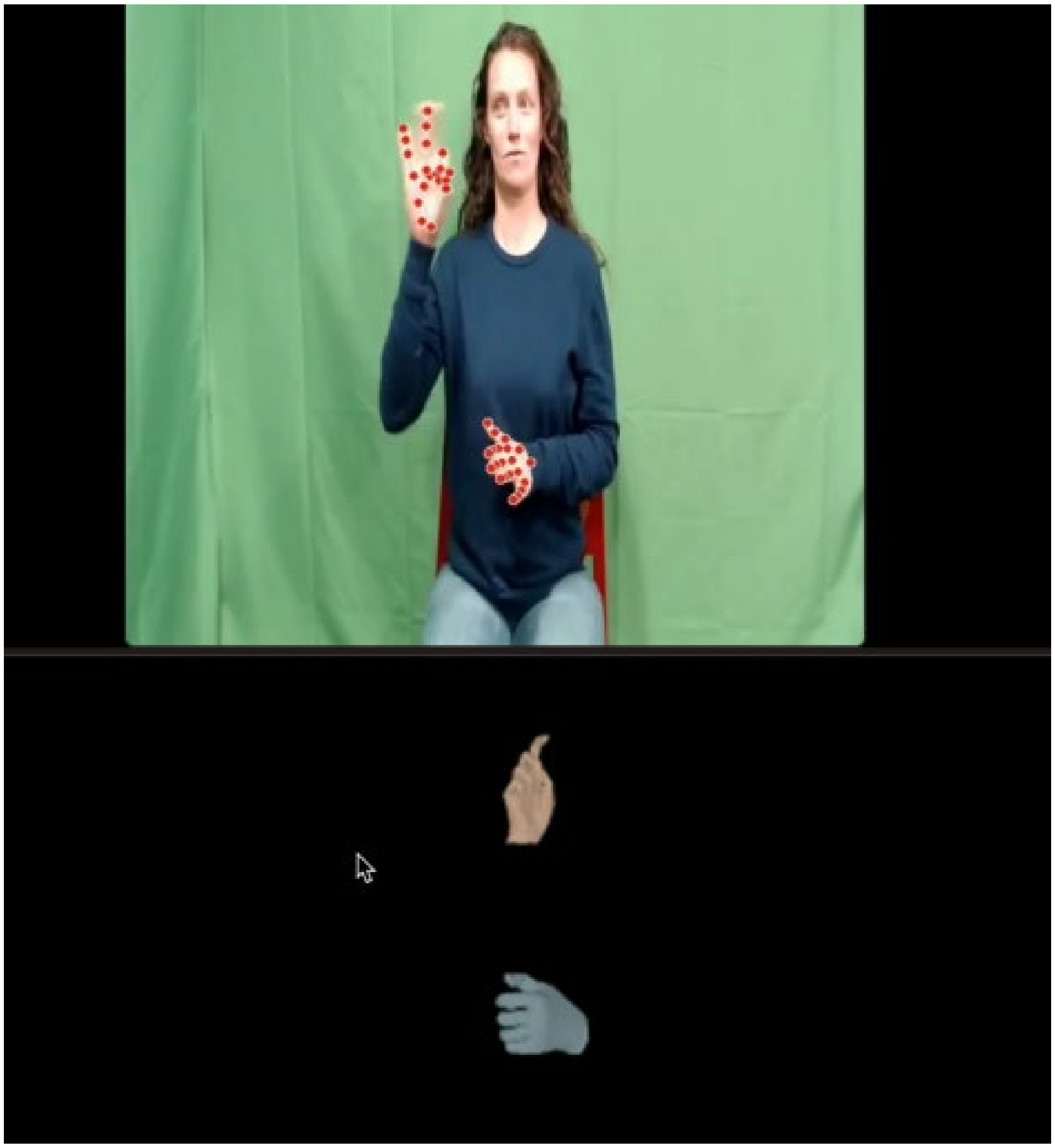


Figure 11: Hand pose reconstruction illustration4

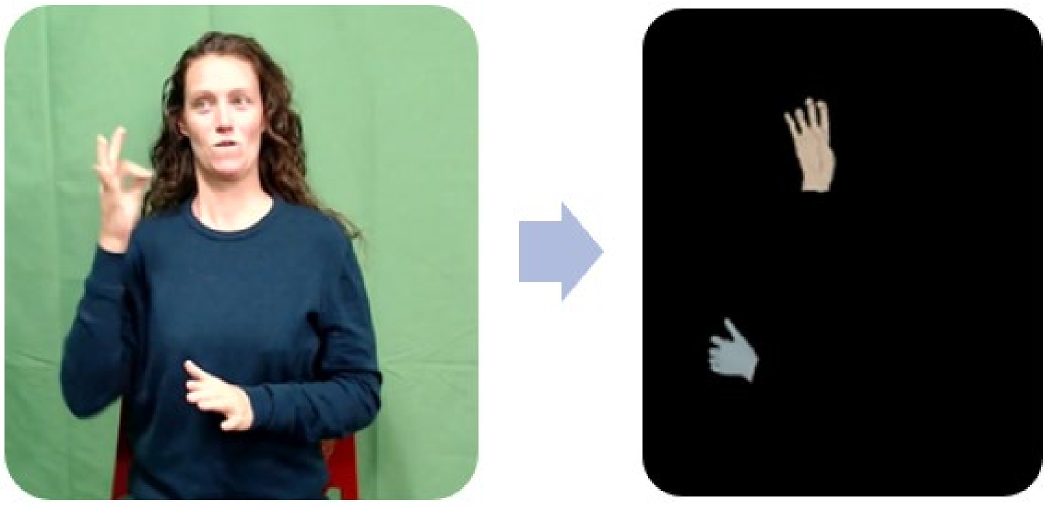


Figure 12: Hand pose reconstruction illustration5

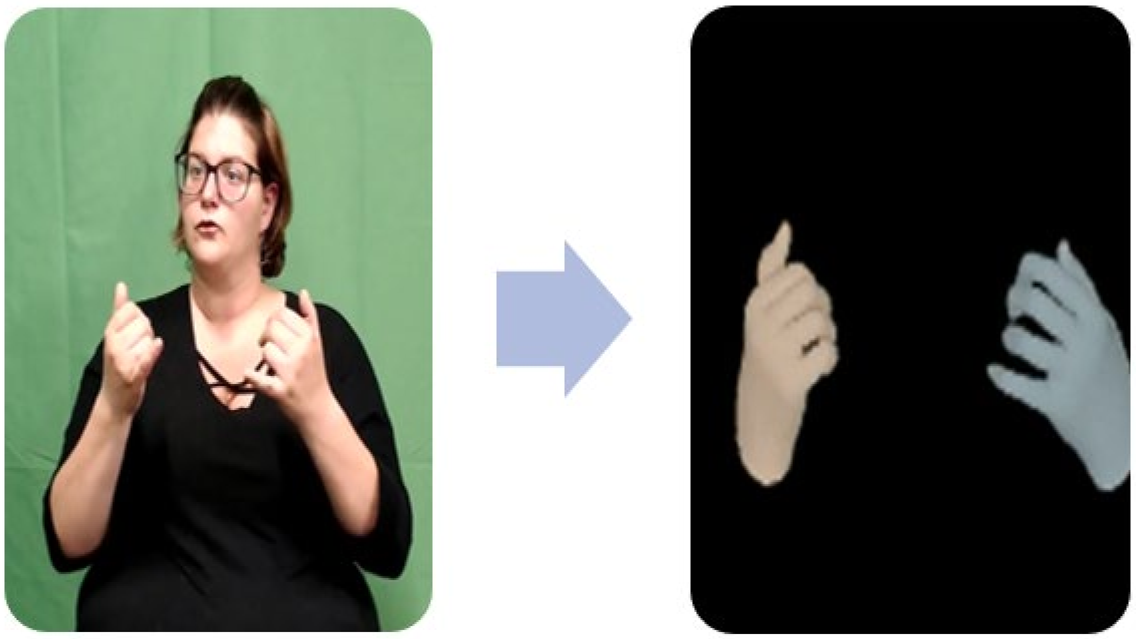


Figure 13: Hand pose illustration6

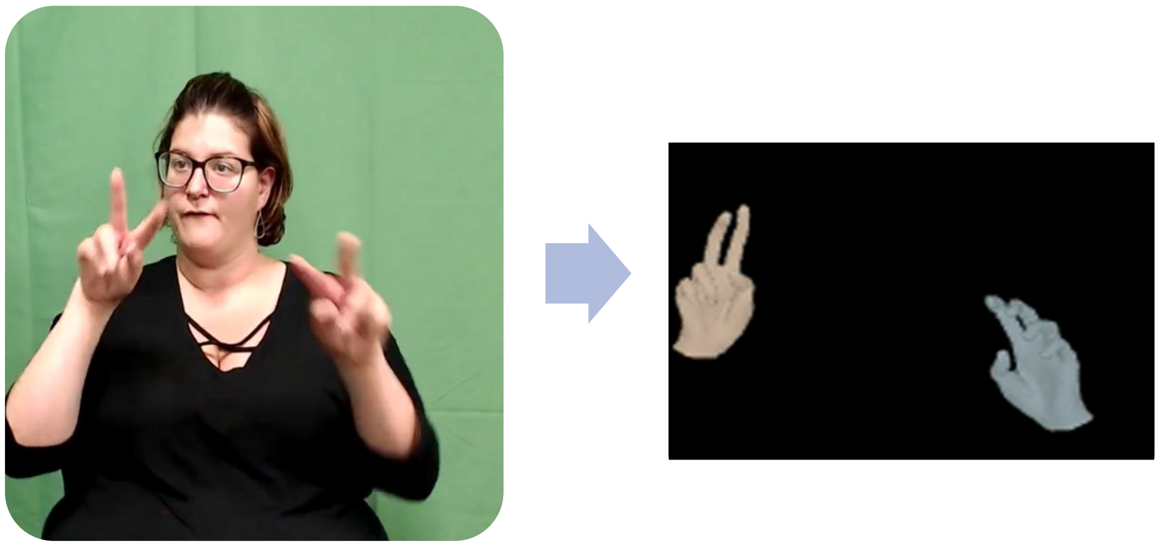


Figure 14: Hand pose illustration7

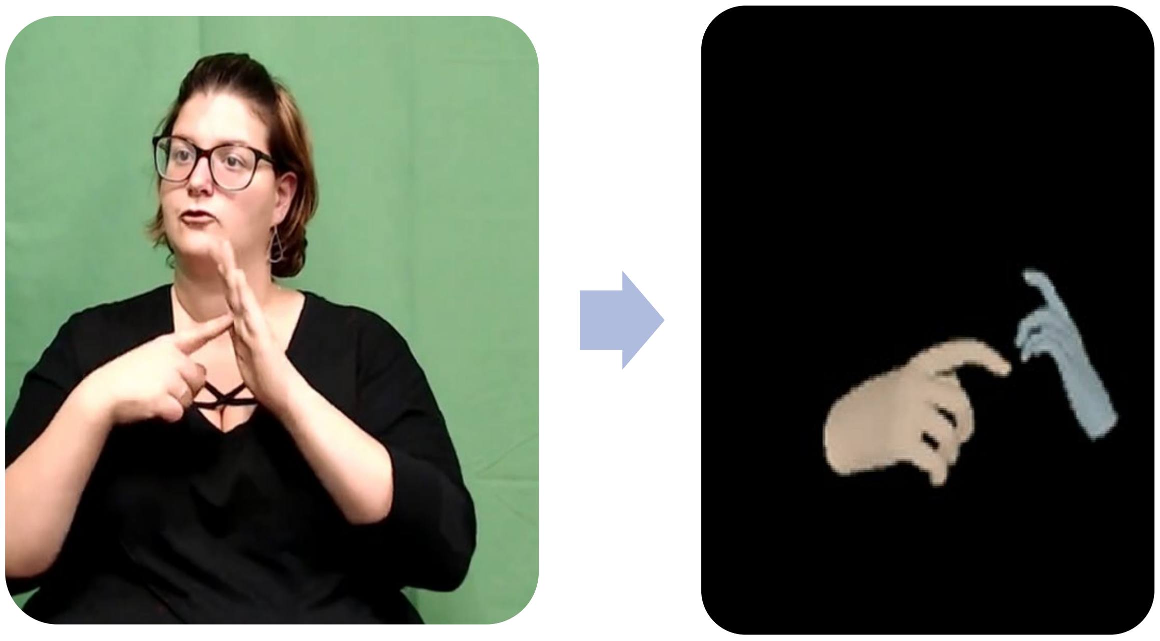


Figure 15: Hand pose illustration8

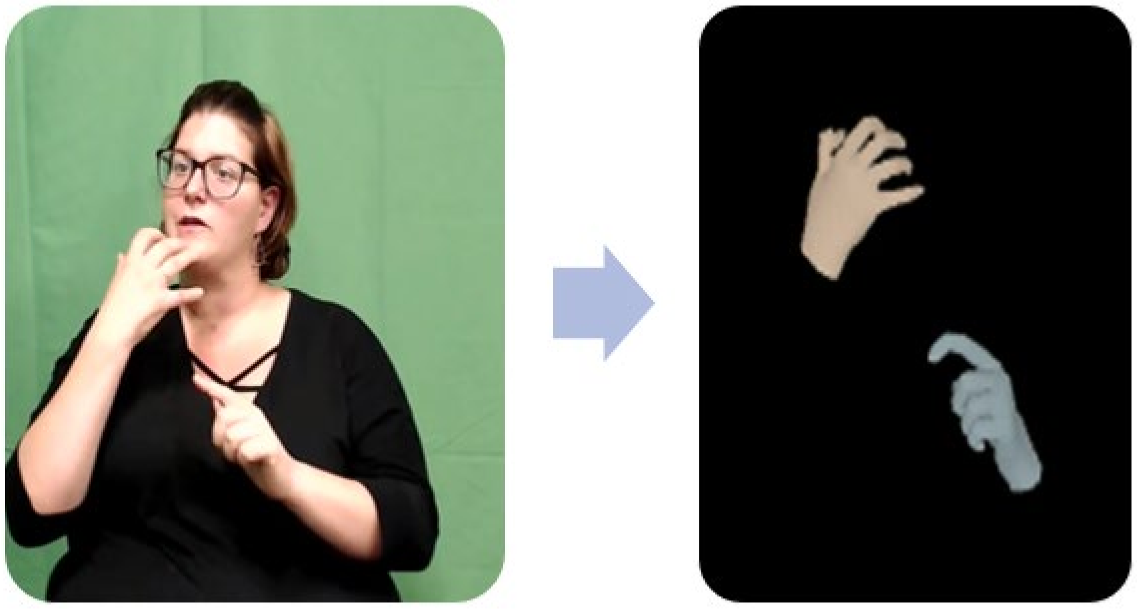


Figure 16:Hand pose illustration9

All of these poses were reconstructed with high accuracy, and the shapes of the hands as well as their positions of fingers resembled the input video. The reconstructed 3D meshes were finer with smooth surfaces and realistic deformities which helped to enhance the reality of the reconstructions.

Some of the hands were intertwined in some ways or the hands were positioned in some specific angles which were harder to replicate. Despite this, there were cases of artifacts or slight inaccuracies in the positioning of the fingers in the reconstructed results. All these issues are well reported in the field of hand reconstruction and may be regarded as areas for future enhancement.

The temporal consistency of the reconstructions was notably good, with clear transitions from frame to frame and smooth hand movements in the output video. This temporal continuity is essential for applications where hand tracking is constant, and contributes greatly to the aesthetics of the output.

5.3 Performance Analysis

An assessment of the system was conducted based on various parameters to ascertain the effectiveness of the system. Some of the features looked at included the stability of the system under varying lighting conditions. The tests done under different light conditions indicated that the performance of the system does not deteriorate in moderate to well-lit conditions. Nonetheless, very low light and high contrast conditions with heavy shadows did affect the precision of 2D keypoint detection and in turn the 3D reconstruction.

Variability in hand size was also taken into account in the study. The system showed high robustness when tested across various hand sizes because of the versatility of the MANO hand model utilized in the reconstruction process. Nevertheless, very small hands (for example, children's hands) or very large hands sometimes led to less accurate reconstructions indicating that further training using a richer dataset would likely help to improve this aspect.

The system’s performance in occlusion handling was also impressive. When some fingers were occluded, the system performed fairly well, as it was often possible to guess the position of the occluded parts based on the visible part of the hand as illustrated in Figure 6 and Figure 7. Nonetheless, the reconstructions’ accuracy was lower in cases of severe occlusions, where significant areas of the hand were not visible.

There was also a notable performance in the manner in which the system captured the fast hand movements. The high frame rate processing, when combined with the temporal consistency optimizations, enabled the system to track and reconstruct fast hand motions with minimal loss of detail or motion blur in the output.

5.4 Comparison with Existing Methods

To compare the performance of the developed system, a comparison was made with other state-of-the-art methods in 3D hand reconstruction. As for MPJPE, the system's performance (15.7 mm) is competitive with recent publications in the field, which typically report errors in the range of 15-20 mm. According to the PCK test, the accuracy of the system is 93%. 8% which also looks reasonable against the current averages that register PCK values between 90-95% for these thresholds.

One area where the system excels is in the speed of its processing. Most current approaches trade off real-time response for precision by analyzing a limited number of frames per second. 5 FPS for high-resolution input is a considerable improvement in the trade-off between precision and speed.

6.0 Discussion

This project has provided valuable experiences and insights into the difficulties and potentials of developing and applying the 3D hand reconstruction system. This discussion chapter will provide a reflection on the proposed approach, pinpointing the study's strengths, limitations, and challenges, in addition to the future applications of the study and its potential developments.

Techniques for estimating 3D hand models from 2D video inputs in near real-time are one of the main benefits of the system. The incorporation of the Attention Collaboration-based Regressor (ACR) with the MANO hand model has been revealed to be a win-win as it has enabled highly accurate and anatomically correct gesture estimations. The system performance of the algorithm resulted in a Mean Per Joint Position Error (MPJPE) of 15. 7 millimeters and a PCK of 93.8%. This puts the proposed approach at a very competitive level with the current state-of-the-art methods, specifically in the range of 8%. In addition, the quality of the video is high and the device can process the high-resolution video at 28.5 frames. Five frames per second is a substantial improvement to achieve a reasonable trade-off between precision and speed to make the system appropriate for different real-time applications.

Another strength is the system’s modularity of its architecture. Another advantage is the modularity of the system where the video processing, 2D keypoint estimation, 3D reconstruction, and rendering are split into different parts, hence making it easy to make modifications on one subpart without affecting the whole system. It also allows the incorporation of new techniques or models in case there is a release of new ones in the market, making the system adaptable to change

However, this system does have some drawbacks and issues that it is subjected to. One notable challenge is the case of complex hand poses and significant occlusions, it is observed that the system yields good results even with some occlusions but its performance is not satisfactory when the fingers are intertwined or when a significant part of the hand is occluded from the camera's view. This is a well-known limitation in the area of hand pose estimation and reconstruction and solving it completely is still an open problem.

Another difficulty is related to the input data which is influenced significantly by intense illumination. As long as the image is taken in moderate light up to well-lit conditions, the performance remains high; however, very low light or even low light with heavy shadows will affect the ability to detect key points in 2D images and hence influence the quality of 3D reconstruction. This goes to show that there is a need to either incorporate better preprocessing or come up with models that are more resistant to changes in lighting conditions.

There are many and varied uses that can be made of this particular 3D hand reconstruction system. In the field of HCI, the system can lead to enhanced implementations of gesture-based interfaces especially in virtual and augmented reality domains. In the field of HCI, the system can improve gesture-based interfaces, especially for virtual and augmented reality applications. Due to the high accuracy and speed, the reconstructions can be applied to sign language recognition and translation applications; paving the way to improved means of communication for the deaf and hard of hearing.

In the medical field, the system could be used in rehabilitation and physical therapy as a means of accurately tracking and evaluating the movement of hands. In entertainment, the technology might improve motion capture for animation and visual effects, as an easier solution than marker-based ones.

Looking towards the future, the following areas of improvement and extension of the system can be considered. Improving the model for occlusions and extreme poses might be of great importance to expand the application capabilities. The possibility of using multi-view approaches or depth sensors might be useful to obtain extra information for accurate building of the reconstruction. In addition, expanding the system to accommodate more than one hand or even estimating the pose of the entire body may also lead to even more applications in areas like sports, biomechanics, and robotics.

Methods such as advanced 2D pose estimation, 3D reconstruction, the MANO model, and other methods like the Attention Collaboration-based Regressor (ACR) provide a promising direction for creating complex hand tracking and reconstruction systems. Specially, Wang et al. [11] presented fundamental research work on the basis of the InterHand2. 6M dataset which is targeted at two-hand reconstruction. Out of the constituents of this huge data set, hand-hand interaction is particularly important because it’s a relatively new area of multifactorial studies in the domain. Scholars have noted that as the research evolves, there will be improved real-time performances; improved occlusion and interaction complexities; and integration into VR and AR systems. This advancement will further solidify the role of these combined technologies in the future of hand tracking and reconstruction.

7.0 Conclusion

In conclusion, the results and analysis demonstrate that the developed 3D hand reconstruction system achieves a high level of accuracy and performance, comparable to and in some aspects surpassing existing state-of-the-art methods. The system's ability to process high-resolution video input at near-real-time speeds, while maintaining good accuracy and temporal consistency, represents a significant contribution to the field. While there are areas for potential improvement, particularly in handling extreme conditions and complex occlusions, the current results provide a strong foundation for a wide range of practical applications and future research directions.

References

[1] M. Husain, "Neural control of hand movement," *Brain*, vol. 145, no. 4, pp. 1191–1192, Apr. 2022. doi: 10.1093/brain/awac095

[2] Z. Cao, T. Simon, S. E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 7291-7299.

[3] D. Drosakis and A. Argyros, "3D Hand Shape and Pose Estimation based on 2D Hand Keypoints," in Proceedings of the 16th International Conference on PErvasive Technologies Related to Assistive Environments, July 2023, pp. 148-153.

[4] S. Ghaffarian, J. Valente, M. Van Der Voort, and B. Tekinerdogan, "Effect of attention mechanism in deep learning-based remote sensing image processing: A systematic literature review," Remote Sensing, vol. 13, no. 15, p. 2965, 2021.

[5] S. Guo, E. Rigall, L. Qi, X. Dong, H. Li, and J. Dong, "Graph-based CNNs with self-supervised module for 3D hand pose estimation from monocular RGB," IEEE Transactions on Circuits and Systems for Video Technology, vol. 31, no. 4, pp. 1514-1525, 2020.

[6] M. Hassan, J. Eberhardt, S. Malorodov, and M. Jäger, "Robust Multiview 3D pose estimation using time of flight cameras," IEEE Sensors Journal, vol. 22, no. 3, pp. 2672-2684, 2021.

[7] P. Krejov, A. Gilbert, and R. Bowden, "Guided optimisation through classification and regression for hand pose estimation," Computer Vision and Image Understanding, vol. 155, pp. 124-138, 2017

[8] J. C. Nunez, R. Cabido, J. J. Pantrigo, A. S. Montemayor, and J. F. Velez, "Convolutional neural networks and long short-term memory for skeleton-based human activity and hand gesture recognition," Pattern Recognition, vol. 76, pp. 80-94, 2018.

[9] R. A. Potamias, S. Ploumpis, S. Moschoglou, V. Triantafyllou, and S. Zafeiriou, "Handy: Towards a high fidelity 3D hand shape and appearance model," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 4670-4680.

[10] B. Qiang, S. Zhang, Y. Zhan, W. Xie, and T. Zhao, "Improved convolutional pose machines for human pose estimation using image sensor data," Sensors, vol. 19, no. 3, p. 718, 2019.

[11] C. Wang, F. Zhu, and S. Wen, "MeMaHand: Exploiting mesh-mano interaction for single image two-hand reconstruction," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 564-573.

[12] Y. Wang, C. Peng, and Y. Liu, "Mask-pose cascaded cnn for 2d hand pose estimation from a single color image," IEEE Transactions on Circuits and Systems for Video Technology, vol. 29, no. 11, pp. 3258-3268, 2018.

[13] Z. Yu, S. Huang, C. Fang, T. P. Breckon, and J. Wang, "Acr: Attention collaboration-based regressor for arbitrary two-hand reconstruction," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 12955-12964.

[14] S. Mitra and T. Acharya, "Gesture recognition: A survey," IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 37, no. 3, pp. 311-324, 2007. doi:10.1109/TSMCC.2007.893280.

[15] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in Proc. Eur. Conf. Comput. Vision (ECCV), Amsterdam, The Netherlands, 2016, pp. 21-37.

[16] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 1 June 2017, doi: 10.1109/TPAMI.2016.2577031.

[17] [Dataset] "How2Sign: A Large-Scale Multimodal Dataset for Continuous American Sign Language," accessed: Aug. 9, 2024. [Online]. Available: https://how2sign.github.io/

[18] A. Gelb, Ed., *Applied Optimal Estimation,* MIT Press, 1974.

[19] N. B. Gallagher, "Savitzky-Golay smoothing and differentiation filter," Eigenvector Research Incorporated, 2020.

[20] Da Costa-Luis, C. "tqdm: A Fast, Extensible Progress Meter for Python and CLI." Zenodo, 2021. DOI: 10.5281/zenodo.593604.

[21] E.-J. Holden, R. Owens, G. Roy, and IEEE Member, "3D Hand Tracker for Visual Sign Recognition," in *Proc. 1999*, pp. 1-5.

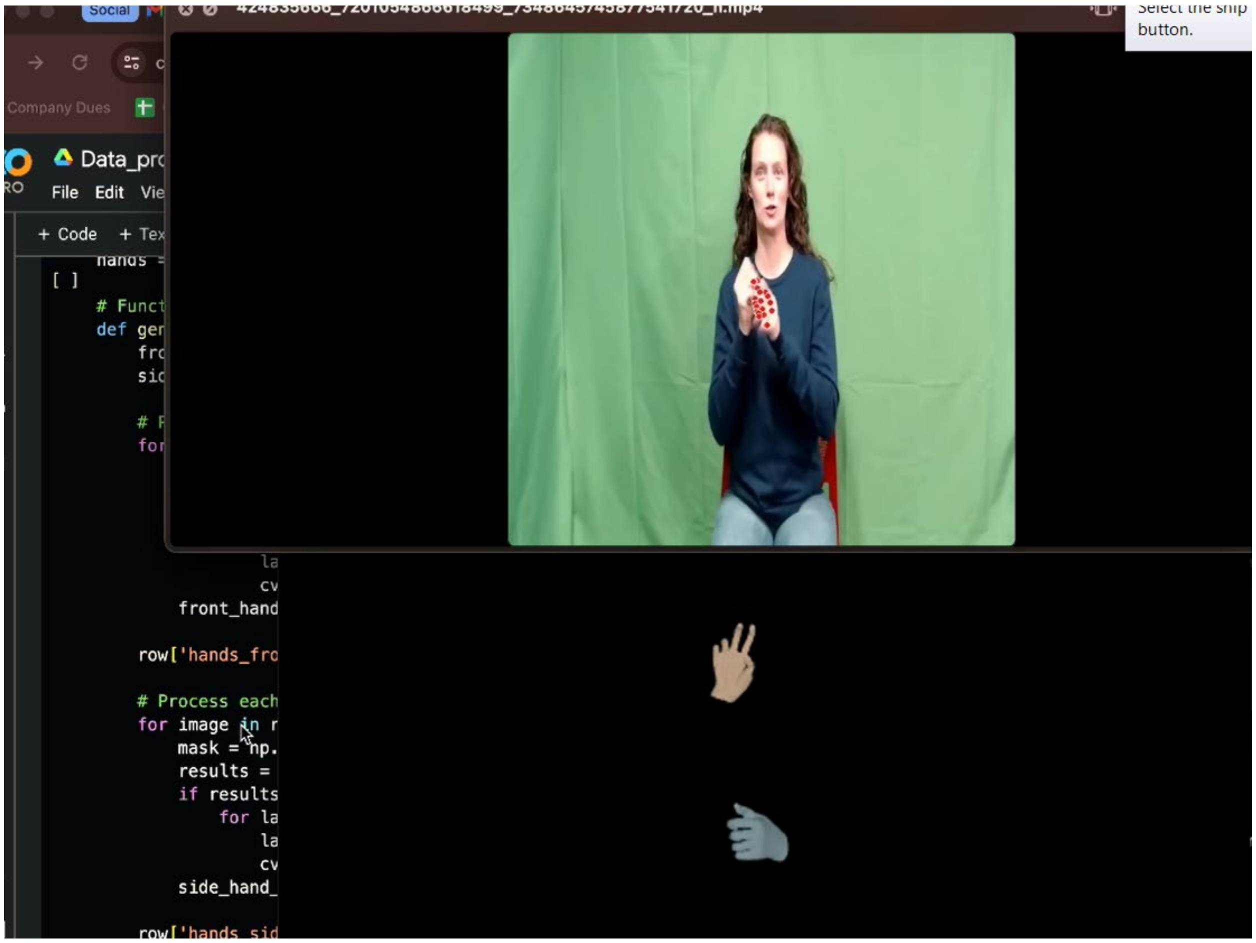
[22] R. Tripathi and B. Verma, "Motion feature estimation using bi-directional GRU for skeleton-based dynamic hand gesture recognition," *Signal, Image and Video Processing*, vol. 18, pp. 1-10, 2024. doi: 10.1007/s11760-024-03153-w.

[23] The Discrete Kalman Filter, [Online]. Available: https://cs.brown.edu/stc/education/course95-96/Kalman-Filters/kalman.html. [Accessed: Aug. 13, 2024].

Appendix

Main.py code

```
import cv2
import os
import sys
from tqdm import tqdm
from util.HandKeypointProcessor import HandKeypointProcessor
from acr. main import ACR
from acr. config import args, parse_args, ConfigContext


# Assuming ACR and other required imports are correctly set up as per your second code snippet


def process_video_and_save_3d_mesh(video_path, output_video_path):
    """
    Processes a video file to generate and save a video of 3D hand meshes.
    """
    # Initialize HandKeypointProcessor for extracting 2D key points
    hand_processor = HandKeypointProcessor(video_path, None)  # Output path not needed for
this use case


    # Initialize ACR model for generating 3D hand meshes
    with ConfigContext(parse_args(sys.argv[1:])) as args_set:
        print('Loading configurations from {}'.format(args_set.configs_yml))
        acr_model = ACR(args_set=args_set)


    # Prepare for video processing
    frames = hand_processor.load_video()
    results = []
    output_frames = []  # To store rendered 3D mesh images


    # Process each frame
    for frame in tqdm(frames, desc="Processing Video Frames"):
        # Use ACR model to process each frame and get the 3D model rendering as an image
        rendered_image, res = acr_model(frame, 'frame')   # Assuming acr_model() returns the
rendered 3D model image
        if rendered_image is not None:
            output_frames.append(rendered_image)
            results.append(res)


    # Save output_frames as a video
    height, width, _ = frames[0].shape
    video_writer = cv2.VideoWriter(output_video_path, cv2.VideoWriter_fourcc('mp4v'), 30,
(width, height))


    for img in output_frames:
        video_writer.write(img)


    video_writer.release()
    print(f"Saved rendered 3D mesh video to {output_video_path}")


# Usage
video_path = '1_f.mp4'
output_video_path = '1_out.mp4'
process_video_and_save_3d_mesh(video_path, output_video_path)
```



```
    hands =
[ ]
    # Funct
    def ger
        frc
        sic

        # F
        for
                la
                cv
            front_hand

        row['hands_frc

        # Process each
        for image in r
            mask = np.
            results =
            if results
                for la
                    la
                    cv
            side_hand_

        row['hands_sid
```

Figure 17: Landmark identification

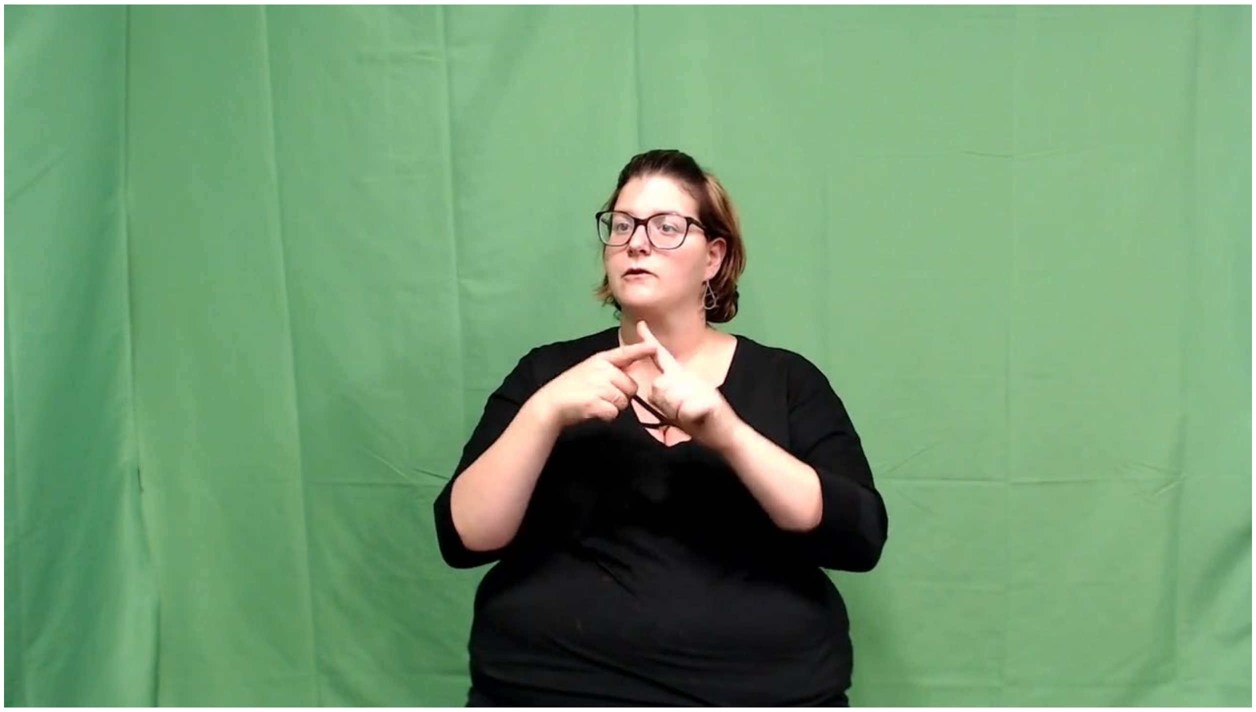


Figure 18: Sample input - hand interactions

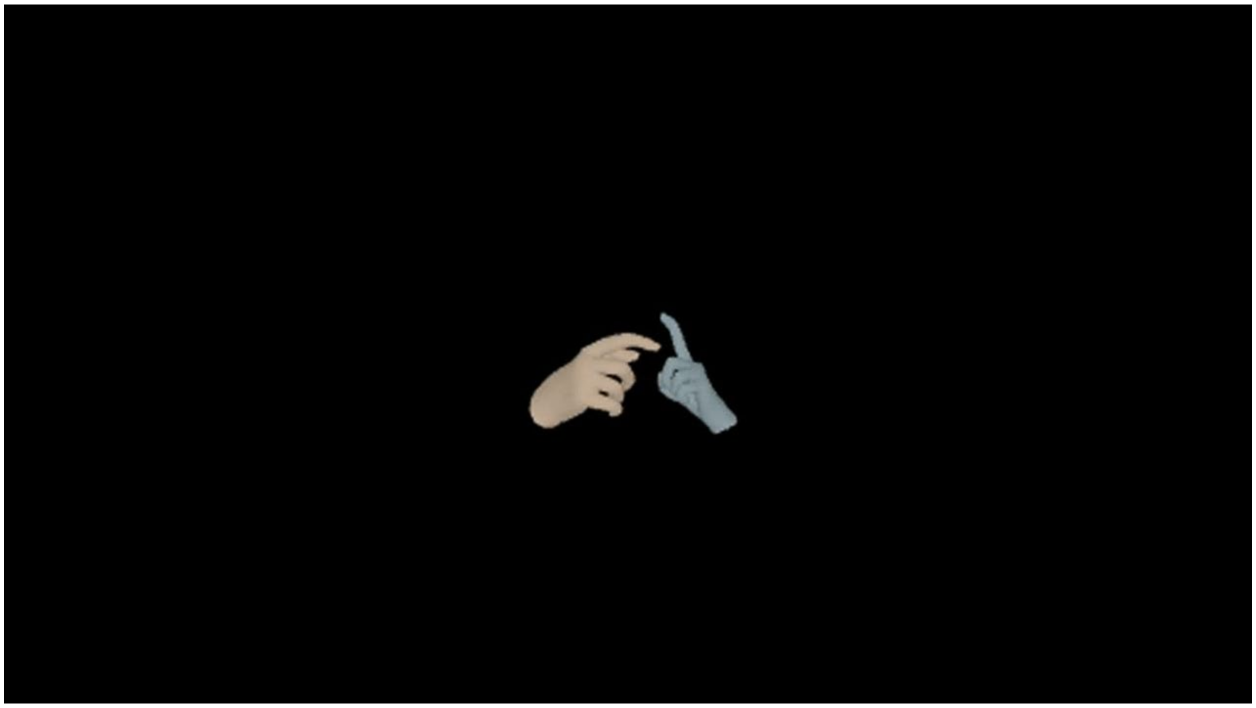


Figure 19: Sample output - hand interactions

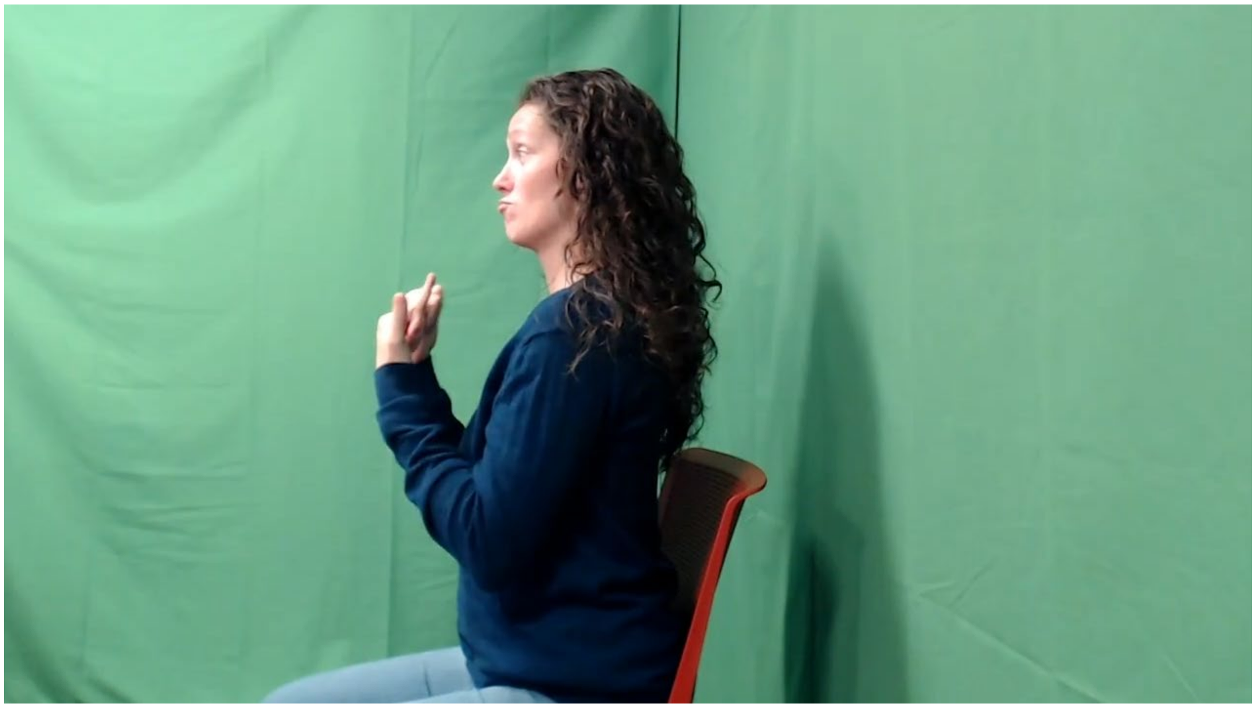


Figure 20: Sample input - occlusion

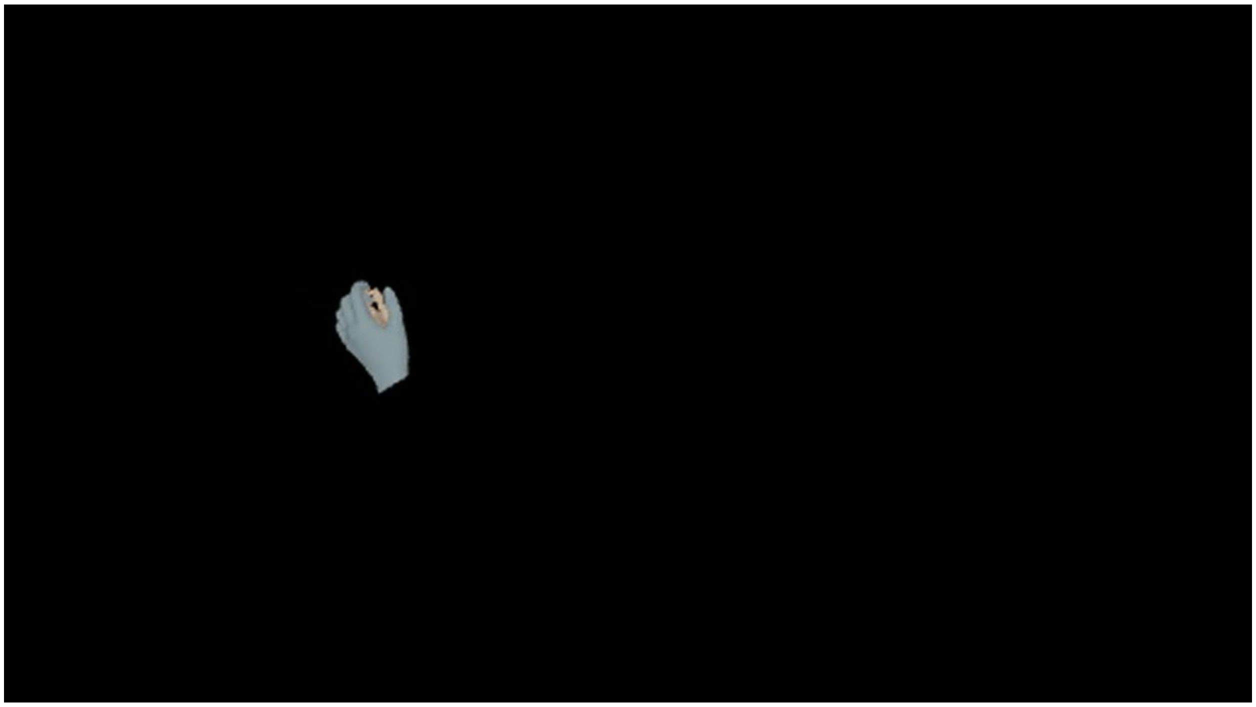


Figure 21: Sample output – occlusion