Scene Flow based Hand Tracking using Deep Learning

By

Muhammad Adnan Anwer

Fall-2021-MS-EE 363845 SEECS

Supervisor

Dr. Muhammad Jameel Nawaz Malik

Department of Electrical Engineering

A thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science in Electrical Engineering with Specialization in Artificial Intelligence and Autonomous Systems (MS EE AI & AS)

In

School of Electrical Engineering & Computer Science (SEECS) ,

National University of Sciences and Technology (NUST),

Islamabad, Pakistan.

(November 2023)

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS/MPhil thesis entitled "Scene Flow based Hand Tracking using Deep Learning" written by Muhammad Adnan Anwer, (Registration No 363845), of SEECS has been vetted by the undersigned, found complete in all respects as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS/M Phil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.

Signature: [illegible]

Name of Advisor: Dr. Muhammad Jameel Nawaz Malik

Date: 23-Sep-2023

HoD/Associate Dean: [illegible]

Date: 18/12/2023

Signature (Dean/Principal): ____________

Date: ____________

Approval

It is certified that the contents and form of the thesis entitled "Scene Flow based Hand Tracking using Deep Learning" submitted by Muhammad Adnan Anwer have been found satisfactory for the requirement of the degree

Advisor : Dr. Muhammad Jameel Nawaz Malik

Signature: [illegible]

Date: 23-Sep-2023

Committee Member 1:Dr. Ahmad Salman

Signature: [illegible]

Date: 19-Sep-2023

Committee Member 2:Mr. Muhammad Imran Abeel

Signature: [illegible]

Date: 02-Oct-2023

Signature: ______________________

Date: ________________________

<u>Certificate for Plagiarism</u>

It is certified that PhD/M.Phil/MS Thesis Titled "Scene Flow based Hand Tracking using Deep Learning" by Muhammad Adnan Anwer has been examined by us. We undertake the follows:

a. Thesis has significant new work/knowledge as compared already published or are under consideration to be published elsewhere. No sentence, equation, diagram, table, paragraph or section has been copied verbatim from previous work unless it is placed under quotation marks and duly referenced.

b. The work presented is original and own work of the author (i.e. there is no plagiarism). No ideas, processes, results or words of others have been presented as Author own work.

c. There is no fabrication of data or results which have been compiled/analyzed.

d. There is no falsification by manipulating research materials, equipment or processes, or changing or omitting data or results such that the research is not accurately represented in the research record.

e. The thesis has been checked using TURNITIN (copy of originality report attached) and found within limits as per HEC plagiarism Policy and instructions issued from time to time.

<u>Name & Signature of Supervisor</u>

Dr. Muhammad Jameel Nawaz Malik

Signature : [illegible]

Certificate of Originality

I hereby declare that this submission titled "Scene Flow based Hand Tracking using Deep Learning" is my own work. To the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics, which has been acknowledged. I also verified the originality of contents through plagiarism software.

Student Name:Muhammad Adnan Anwer

Student Signature: [illegible]

Dedication

This thesis is dedicated to all the deserving children who do not have access to quality education especially young girls.

Acknowledgments

Glory be to Allah (S.W.A), the Creator, the Sustainer of the Universe. Who only has the power to honour whom He please, and to abase whom He please. Verily no one can do anything without His will. From the day, I came to NUST till the day of my departure, He was the only one Who blessed me and opened ways for me, and showed me the path of success. Their is nothing which can payback for His bounties throughout my research period to complete it successfully.

Muhammad Adnan Anwer

Contents

List of Tables

List of Figures

List of Abbreviations and Symbols

Abbreviations

SF	Sceneflow
MVPE	Mean Vertex End Point Error
EPE	Mean End Point Error
SF	SceneFlow
SOTA	State of The Art

Abstract

Tracking of human hand is an integral part of many key computer vision systems with diverse applications such as human computer interaction, gesture recognition and augmented reality etc. However this task is very challenging due to the non rigid nature of hands, heavy occlusions due to interaction and it's smaller size relative to the camera frame. Sceneflow estimation is also one of the key algorithms used in many applications such as image processing, navigation and visual surveillance etc. This task is also very difficult due to the one-to-many correspondence problem.In this project, we will develop a deep learning based system for estimation of sceneflow and extend it to solve the hand tracking problem.

CHAPTER 1

Introduction and Motivation

Human hand tracking and pose estimation serve as a natural method for human computer interaction in augmented / virtual reality applications. This is an active research area in computer vision with vast industrial applications [49, 50].

Hand shape estimation or mesh reconstruction has been a tough problem and continues to challenge the computer vision community especially for images in the wild [25, 29]. Multiple deep learning based efforts have been made in solving this issue in a monocular setup [8, 26, 27, 10, 21, 41] with exceptional performance. However, the monocular approach (depth or RGB) has few limitations due to following challenges:

- **1. Depth image uncertainty:** The depth images are very noisy and have high level of uncertainty that causes erroneous prediction.
- **2. Limited perspective:** Any monocular image, whether it be depth or RGB, captures only one perspective. So any deep learning technique is biased towards the limited perspectives available in the datasets.
- **3. Occlusion:** Naturally, single view does not provide any information about the occluded portion of the hands causing a reduction in robustness.

Multiview approach seems to be the most suitable and intuitive way of solving above issues. In this research project we will focus on developing a similar approach.

1.1 Problem Statement and Contribution

In this research project, we have developed a state of the art scene flow based hand tracking network by resolving the depth image uncertainty, limited perspective and occlusion problems inherited by the state-of-the-art methods due to their monocular nature by following ways:

- **Multiview Fusion:** Fusing multiple views in a global 3d frame of reference.
- **3d deeplearning:** Designing a deep learning model with direct 3d to 3d mapping for hand mesh reconstruction.
- **Pointcloud input:** Using 3d point cloud input for smaller network size.
- **Realworld images:** We have used a realworld hand image dataset [25] instead of senthetic images which makes the task more challenging.

Our main contributions are:

- **2d to 3d data pipeline:** A data preprocessing pipeline for fusing multiple views into a singular point cloud.
- **3d hand shape estimator:** A state of the art hand mesh estimator.
- **3d sceneflow estimator:** Optimization / finetuning of the state of the art scene flow estimation network for hand point clouds.
- **3d hand shape tracker:** A sceneflow based state of the art hand mesh tracker.
- **3d hands dataset:** Our version of DexYcb with merged views containing point cloud and corresponding joints/vertices of hand.
- **3d hands sceneflow dataset:** Our version of DexYcb with merged views containing point cloud pairs and corresponding sceneflow for hand flow problems.

For the scope of this project, hand mesh or shape in 3d is defined by the standard MANO parametric model [42] comprising of 778 vertices.

CHAPTER 2

Literature Review

Hand mesh reconstruction or shape estimation task can be broadly classified into two distinct forks, namely monocular and multiview. Multiview approaches either merge multiple views to form a pointcloud or merge the embeddings generated through some feature encoder. Here we will focus on the single view and point cloud based approaches.

- **Monocular Hand Shape Reconstruction**

  Most of the techniques [8, 11, 38, 24] focus on hand shape reconstruction through deforming a standard hand mesh template based on some parametric model such as MANO [42]. However, regression of 3d rotations is a challenging task which limits the performance. Many recent techniques [27, 10, 13, 20] try to regress the hand mesh vertices directly without any parametric model through graph based approaches for preserving geometric sanctity. 3d voxels [21], signed distance functions [35, 19] and UV positional graphs [26] have also been used successfully. [13] proposed that combining parametric model and graph convolutions are helpful in human body pose estimation. This can be extended to human hands as well. Transformer based approaches [30, 31] have also proved fruitful in extracting useful features for this problem. Building on these observations, we also base our approach on graph convolutions and transformer attention centric concepts.

- **Pointcloud Learning Networks**

  Qi et al. [3] developed a ground breaking pointcloud feature extraction deep learning netowork called PointNet. They further improved the performance in PointNet++ [4] by incorporating ball query and heirarchical clustring. PointNet++ better preserved the geometric structure of the pointcloud. There are many other approaches that try to extract local features from pointsets through convolution operators [6, 15, 16]. Zao et al. integrated the transformer architecutre into such networks and introduced Point Transformer [34] with local neighborhood attention mechanism. Guo et al. achieved similar performance in Point Cloud Transformer [28] via application of Laplacian matrix. Our network consists of a feature extractor comprising of successive PointNet++ and Point Transformer Layers.

- **Multiview Hand Shape Reconstruction**

  In literature, fusing multiple views directly or merging there features is usually termed as multi-view stereo (MVS) problem. Some approaches extract 2d image features from each view and then merge them in some latent space such as epiolar-line based position embedding [18], point based position embedding [37, 23, 33] and 3d position encoding [40]. While others extract 3d features [1, 2, 7, 9] directly from 3d pointclouds using 3D-CNN or point cloud networks. POEM [48] proposed by yang et al. is a similar structure aware approach that directly updates hand vertices by utilizing feature aggrigation and structure conscious vertex query. Since intrinsic and extrinsic calibration parameters of a camera can be easily calculated, we use a simpler approach and merge the piontclouds generated by un-projecting depth maps directly into a global or common frame of reference.

- **Sceneflow Estimation**

  Sceneflow prediction is another fundamental computer vision problem where we need to predict the movement of a pixel or a point in 3d between two consecutive frames. It's an ill posed correspondence problem where we need to identify a corresponding pixel or point in the next frame. There are a lot of state-of-the-art networks with exceptional performance predicting sceneflow using deep learning. These methods vary in the type of data they take as input. Broadly speaking, they can be classified into two categories. Ones take RGB-D images as input data [32, 39, 46]. RAFT3d [32] introduced the concept of recurrent all pairs feature transforms. Using the Lie Algebra they significantly reduced the training time as well as the prediction accuracy. Second are the ones that take point clouds as input [14, 17, 22, 43, 36].

  FlowNet3d [14] was the pioneering network that estimated sceneflow directly from pointsets using a PointNet [3] based Siamese feature extractor and a similar flow embedding layer to find correspondences. Flow embedding layer dealt with the correspondence problem in a point-to-patch manner. PointPWCNet [17] incorporated the pyramid approach to counter for different scales and improved the results significantly. FLOT [22] emplyed optimal transport approach and significantly reduced the parameters. Bi-PointFlowNet [36] introduced bidirectional flow embedding layers that learn in both forward and backward flow directions. Guangming Wang et al. [43] proposed an all-to-all flow embedding layer at the lowest level in the pyramid network. We build our own sceneflow estimation network and also use fine-tuned Bi-PointFlowNet as a off-the-shelf sceneflow estimator in our experiments.

CHAPTER 3

Methodology

Section 3.1 provides details on the data pipeline while section 3.2 explains the deep learning model. Section 3.3 explains the training methodology. Section 3.4 sheds light on the evaluation metrices.

3.1 Introduction to the Dataset - DexYcb

We are using DexYcb [25] Dataset which contains hundred video sequences of ten different subjects interacting with multiple objects one hand at a time on a table top from 8 different views or perspectives. The dataset contains 640x480 RGB-D Images, camera intrinsics and extrinsics. The ground truth includes hand mask, object mask and MANO [42] parameters which include three dimensional global translation, three dimensional global rotation, ten blend shape parameters and fourty-five local rotations.

3.2 Data Pre-processing Pipeline

The main goal of the data pre-processing pipeline is to first convert the 2D images to 3D point clouds. Then extract the points that belong to the hand. And finally merge multiple views.

3.2.1 Back-projection of Depth Images and Filtering Hand Points

The depth image directly contains the z-axis location of every pixel. Given the intrinsic parameters of the depth camera, each pixel with location (u, v) can be back-projected to the 3d world coordinates (x, y, z) using following relations:

$$X = \frac{U - c_x}{f_x} \cdot D \tag{3.2.1}$$

$$Y = \frac{V - c_y}{f_y} \cdot D \tag{3.2.2}$$

$$Z = D \tag{3.2.3}$$

Where:

- fx and fy are the focal lengths in pixels along the X and Y axes.
- cx and cy are the principal point coordinates (the optical center) in pixels.
- U and V are the pixel coordinates in the image.
- D is the depth value at pixel (U, V).
- X, Y, and Z are the resulting 3D world coordinates.

Since we also need to filter out the points that belong to the hand, we add another factor to the above equations signifying whether the pixel belongs to hand or not. This is obtained from the hand mask ground truth available in the DexYcb Dataset.

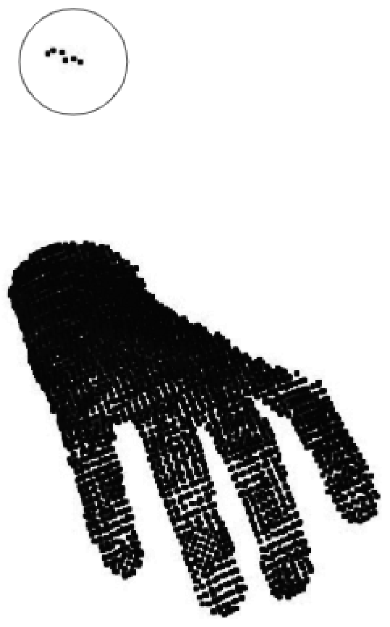

Figure 3.1: Hand Point Cloud generated from Single View (Before DBSCAN)

After masking out the required points, we observe that some outliers are observed that are clearly not a part of the hand when we observe qualitatively. These pixels arise due to the small errors in hand mask ground truth. Such points are filtered out using DBSCAN algorithm [5]. DBSCAN algorithm generates a set of clusters or groups of points based on their vacinity to each other. Their are two parameters that control output of this algo:

1. **EPS** which is the maximum distance between two points in the same cluster. We set this parameter equal to 0.005 m or 5 mm.
2. **S** which is the minimum number of points required to form a cluster. We set this parameter equal to 5.

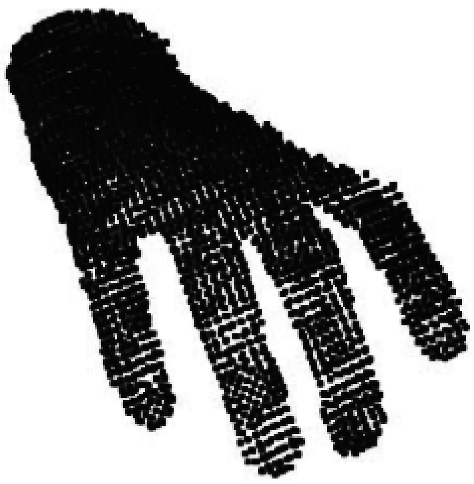

Figure 3.2: Hand Point Cloud generated from Single View (After DBSCAN)

3.2.2 Generating Ground Truth for Pose/Shape Estimation

DexYcb Dataset has annotated MANO [42] parameters which include three dimensional global translation, three dimensional global rotation, ten blend shape parameters and fourty-five local rotations. The 3D hand mesh (778x3) and joints (21x3) can be obtained by using these paramters and inserting them into the differentiable pytorch MANO layer [12]. The 3D hand mesh contains 778 vertices while there are 21 joints in standard MANO parametric model.

3.2.3 Generating Ground Truth for Dense Sceneflow Estimation

DexYcb does not contain any sceneflow information. Therefore, generating the sceneflow ground truth was a great challenge. This problem was solved by utilizing the hand shape ground truth as a guide.

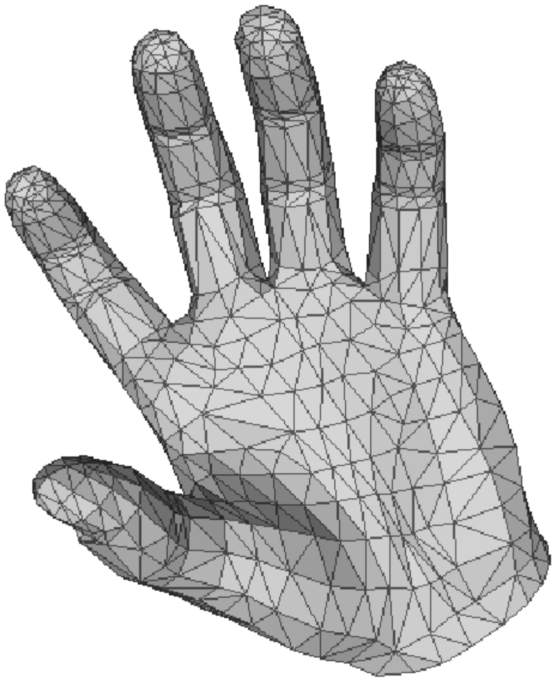

Figure 3.3: Hand Shape Ground Truth (778 Mesh Vertices)

To find accurate sceneflow, we need to know the point to point correspondences between each 3D point. Once we know these correspondences, the sceneflow of any 3D point 'p' is simply the vector directed from the location of this point in first frame 'p1' to the location in second frame 'p2'. So finding the sceneflow is simply a problem of finding the correspondences or movement of points.

Now consider the ground truth hand shape which was generated in previous section. Let 'Vi' be the 3D location of the ith vertex in the ground truth hand mesh. If 'Vi1' is the location of ith vertex in frame 0 and 'Vi2' is the location of ith vertex in frame 1, then the sceneflow can be evaluated using following formula:

$$SceneFlow_i = Vi2 - Vi1 \tag{3.2.4}$$

Here we assume that the sceneflow of a local patch or group of 3D points remains same due to the rigid body movement which is a fairly reasonable assumption. Since we already have the known sceneflow at vertex locations, we can find the the sceneflow at any arbitrary point 'n' in the hand point cloud (Figure 3.2) by simply averaging the sceneflow of the nieghbouring ground truth vertices.

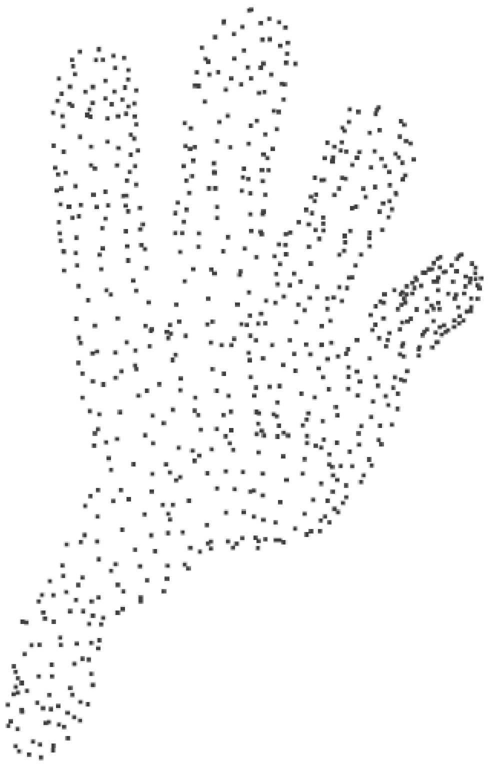

Figure 3.4: Hand Shape Ground Truth (778 Mesh Vertices)

$$SceneFlow_n = mean(\text{ sceneflow of 03 nearest nieghbouring MANO mesh vertices }) \quad (3.2.5)$$

After generating the dense sceneflow in such a way the results were qualitatively evaluated. Following figure shows two point clouds superimposed on each other. Blue points represent the original point cloud while red points depict points of the next frame minus the dense sceneflow. It can be seen that the two superimpose perfectly.

3.2.4 Merging Multiple Views and Filtering

The 3D points generated so far are in camera frame of reference or camera coordinate system. To convert them to global coordinate system, we need to use camera extrinsic parameters. These parameters describe the camera's position and orientation in the world coordinate system. We'll need these parameters to transform from camera coordinates to world coordinates.

- First we convert the 3D point in camera frame to homogenous coordinate system by ap-

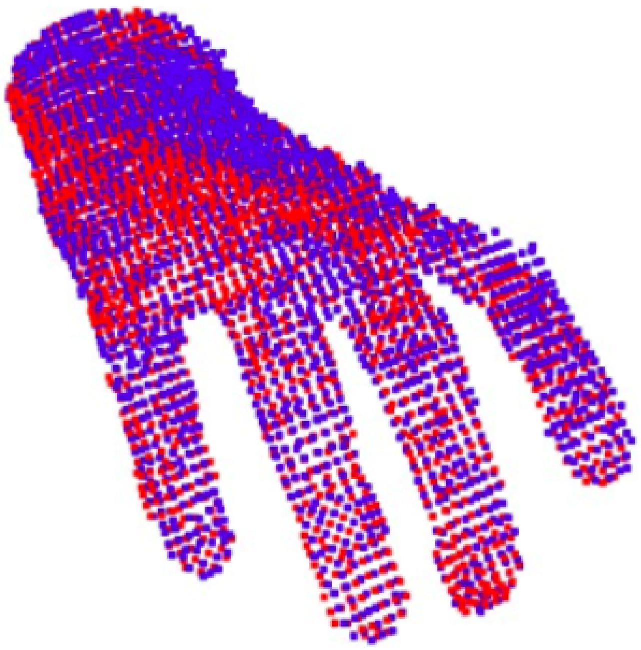

Figure 3.5: Dense Sceneflow Ground Truth of Hands (colored image)

pending 1 at the end.

$$\text{Homogeneous Point in Camera Frame: } (X_{\text{cam}}, Y_{\text{cam}}, Z_{\text{cam}}, 1)$$

- Then we compile the extrinsic parameters of the camera:

$$\text{Extrinsic Parameters: Rotation Matrix } \mathbf{R} \text{ and Translation Vector } \mathbf{T}$$

- Then we build the Transformation matrix:

$$\text{Transformation Matrix (4x4): } \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Finally the point is projected to world coordinates using following equation:

$$(X_{\text{world}}, Y_{\text{world}}, Z_{\text{world}}, 1) = \text{Transformation Matrix} \cdot (X_{\text{cam}}, Y_{\text{cam}}, Z_{\text{cam}}, 1) \tag{3.2.6}$$

Where:

- R is the rotation matrix representing the camera's orientation in the world frame.
- T is the translation vector representing the camera's position in the world frame.
- point camera is the 3D point you want to transform in homogeneous coordinates.
- The transformation matrix extrinsic matrix combines rotation and translation.
- The transformed point point world represents the 3D coordinates of the point in the world frame.

Once we have converted all the camera views to the global frame, we can simply concatenate or directly merge them to form a consolidated point cloud.

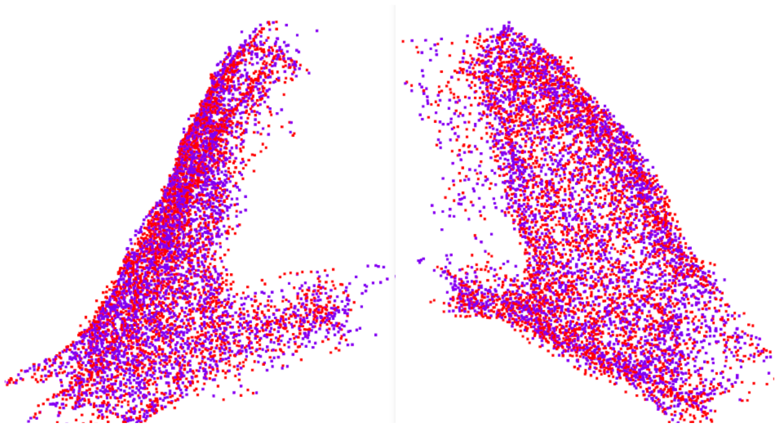

Figure 3.6: Multiview Point Cloud after merging all individual camera views

3.2.5 Removing Outliers

All frames that have less than 4096 points after merging all views are considered outliers and have been removed from the dataset.

3.2.6 Single Sided Dataset

Only right hand frames are considered for evaluation and training in line with the latest state-of-the-art papers [48] for a fair comparison.

3.2.7 Training and Validation Split

We have used standard S0 split of DexYcb [25] dataset. According to this split, 80 percent sequences are used for training while 18 percent are used for testing and 2 percent for validation. In total there are 58188, 2900 and 11658 samples in S0 for train, validation and test splits respectively. We have combined validation and test splits to form a single 20 percent validation spilt in line with [48].

In total there are 23564 training samples and 5920 validation samples in our dataset.

3.3 Deep Learning Architecture

The goal of the project is that given an initial mesh and a pair of consecutive pair of pointcloud frames, we need to estimate an accurate 3D hand mesh in the next frame by utilizing sceneflow information. Our proposed deep learning pipeline is shown in the figure below.

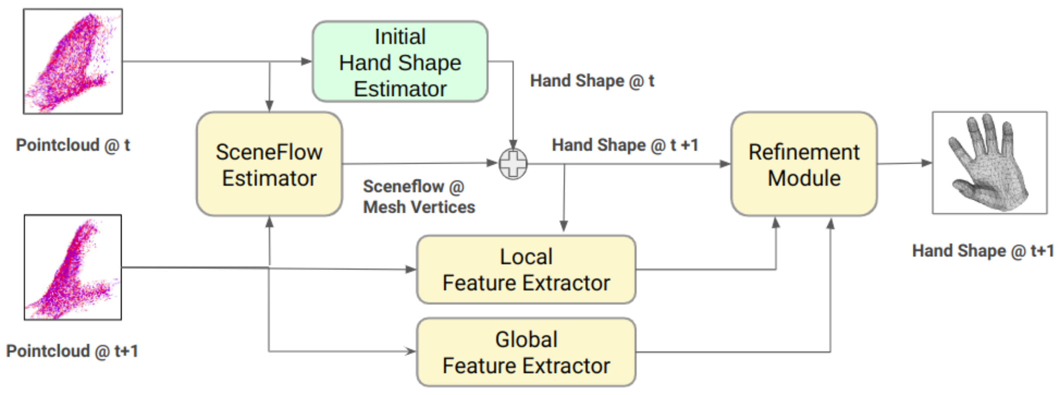

Figure 3.7: Overall Architecture of Our Hand Tracker

The architecture includes five distinct learning blocks.

- Initial Hand Shape Estimator
- Sceneflow Estimator
- Local Feature Extractor
- Global Feature Extractor
- Refinement Module

The initial hand pose estimator estimates the hand shape for the first frame from pointcloud. Sceneflow estimator estimates the sceneflow at the mesh vertices predicted by the initial hand pose estimator. We find the hand shape of the next frame by simply adding the the sceneflow into the intial shape estimated for the previous frame.

$$\text{Shape @ frame t + 1} = \text{Shape @ frame t} + \text{Sceneflow from frame t to t + 1} \tag{3.3.1}$$

Evidently, this initial shape contains error due to the error in the sceneflow which accumulates over time. This error is removed by the refinement module which uses the local and global features extracted from pointcloud of next frame through local and global feature extractors respectively.

Next sections will dive deep into these individual blocks in more detail.

3.3.1 Global Feature Encoder

The global feature extractor is a combination of Set Abstraction Layers from PointNet++ [4] and PointTransformer Layers from PointTransformer [34]. We have used three sets of these layers back-to-back and the final set abstraction layer generates a global feature using the maxpool function. Layer settings are given in the tables.

Sr.	Name	npoint	radius	nsample	in channel	mlp	out channel
1	SA1	512	0.015	4	3	32, 32, 64	64
2	SA2	128	0.030	4	64+3	64, 64, 128	128
3	SA3	64	0.060	4	128+3	128, 128, 256	256
4	SA4	None	None	4	256+3	256, 256, 1024	1024

Table 3.1: Configuration parameters of Set Abstraction Layers in Global Feature Extractor

Sr.	Name	Model Dimension	Point Feature Dimension	nsample
1	T1	64	64	4
2	T2	128	128	4
3	T3	256	256	4

Table 3.2: Configuration parameters of Transformer Layers in Global Feature Extractor

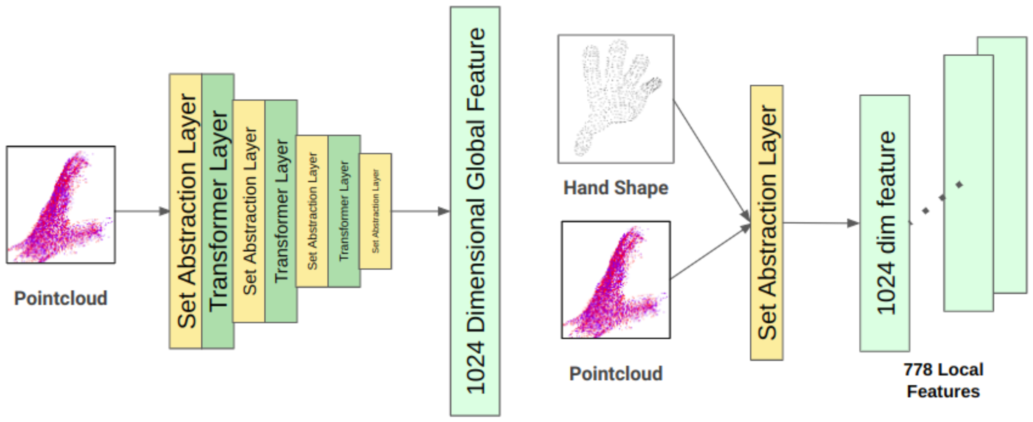

Figure 3.8: Architecture of Local (right) and Global (left) Feature Extractors

3.3.2 Local Feature Encoder

Similar to global feature extractor, local feature extractor also consists of a single set abstraction layer from PointNet++ [4] architecture. However, it is slightly different in a sense that it does not perform ball query to find the points. Rather it accepts pre-computed points as input. Another important thing to remember is that we are not grouping the features at the end. So basically we are extracting local features from the local neighbour hood for each of the 3D point provided at the input. The detail of the layer is provided in the table.

Sr.	Name	radius	nsample	in channel	mlp	out channel
1	SA1	0.030	4	3	32, 64, 128, 256, 512, 1024	778x1024

Table 3.3: Configuration parameters of Set Abstraction Layer in Local Feature Extractor

3.3.3 Graph Convolution Network for Shape Refinement Module

The heart of the shape refinement module is the graph convolution layers. The architecture of our network is similar to [13]. We have used five layers in total. Each layer has 512 channels and a feature length of 2048 (1024 global feature + 1024 local feature). First we form a graph from the input initial hand shape. Then we copy and concatenate global feature with each node. After that we concatenate corresponding local feature with each node. Then we pass this input graph from the graph convolutional layers. Finally we extract the refined shape from the output graph.

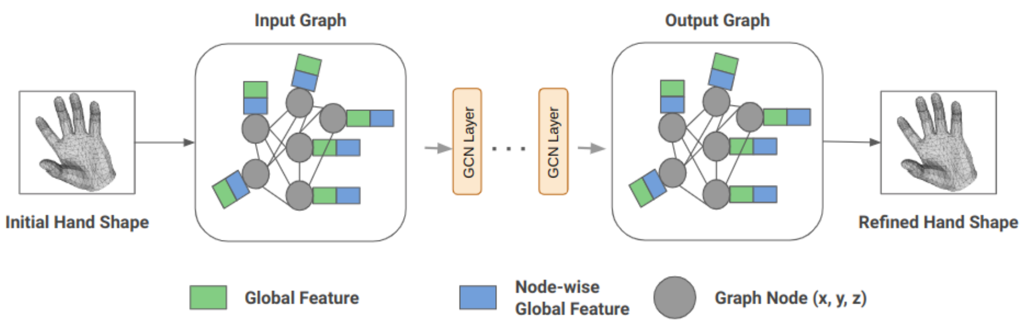

Figure 3.9: Architecture of Graph Convolution Network for Shape Refinement Module

3.3.4 Hand Shape Regressor

The input of our hand shape regressor is a piontcloud made up of 4096 3D points. We first extract 1024 dimensional global feature using the global feature extractor module. Then we use a three layer multi-layer perceptron (MLP) for regressing the initial shape of hand. The detailed architecture of this MLP is tabulated.

Sr.	Layer Name	Configuration Parameters
1	Fully Connected 1	in channel = 1024, out channel = 512
2	Batch Norm	channel = 512
3	Activation	ReLU
4	Dropout	drop percentage = 0.40
5	Fully Connected 2	in channel = 512, out channel = 256
6	Batch Norm	channel = 256
7	Activation	ReLU
8	Dropout	drop percentage = 0.40
9	Fully Connected 1	in channel = 256, out channel = 778x3

Table 3.4: Configuration parameters of MLP Layers in shape regressor

Once we have this initial shape, we can now find the local features using the local feature extractor module. After extracting local features, we can now apply the graph convolutions based refinement module to find the refined hand shape.

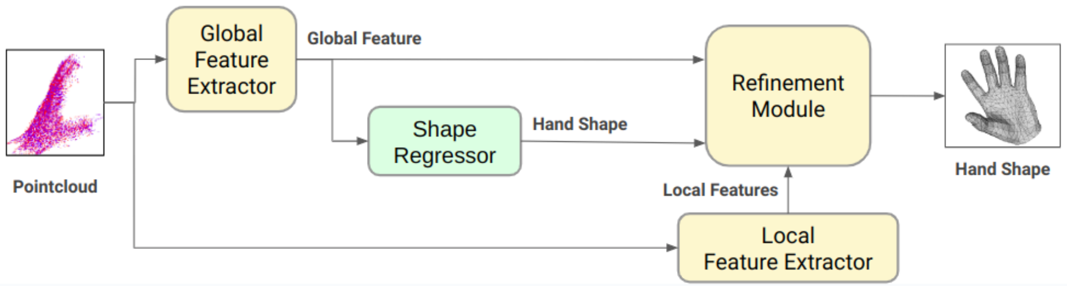

Figure 3.10: Architecture of Hand Shape Regressor

3.3.5 Scene Flow Estimator

We have used Bi-PointFlowNet [36] as our sceneflow estimator. This network generates dense sceneflow by matching a pair of pointclouds. The sceneflow generated by this network is dense meaning that a sceneflow vector is generated against each point in the first pointcloud.

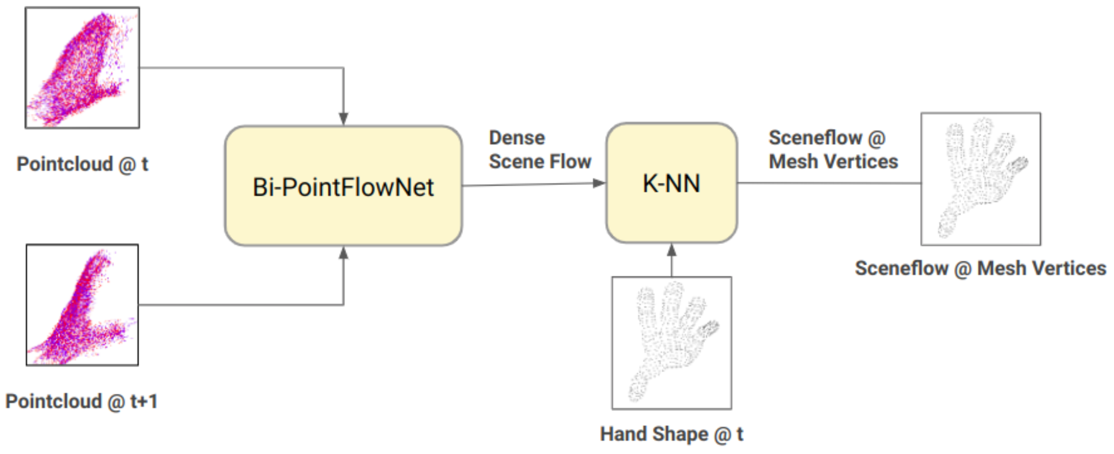

Figure 3.11: Architecture of Scene Flow Estimator

Since we need the scene flow at a specific location (meaning the hand mesh vertices), we used k-NN method to average out the sceneflow of the three nearest neighbors of the vertex location. This approach is the same as the one we used to generate ground truth for the dense scene flow.

3.4 Training Methodology

We first train the hand shape regressor and sceneflow estimator separately. Once they are ready, we then train the whole pipeline while keeping the parameters of these two networks fixed. We also fix the parameters of the global feature extractor. Here, we use the same weights as the ones used for hand shape regressor.

3.5 Evaluation Methodology

The evaluation criteria is simply the mean end point error between the ground truth hand shape mesh and the predicted one. Mean end point error is the mean L2 norm of each predicted 3d point in the evaluation (validation) dataset. Since we are only evaluating on the hand shape mesh vertices, its usually denoted by Mean Vertex End Point Error (MVPE) in literature. It is measured in millimeters. In the case of sceneflow SceneFlow (SF), simply mean end point error Mean End Point Error (EPE) is used. It is also measured in millimeters.

CHAPTER 4

Experiments and Results

4.1 SceneFlow Estimator

Here our goal is just the fine-tuning of existing Bi-PointFlowNet model on our dataset. We carried out the ablation study on subject 1 data only. The finetuning is done on complete dataset.

4.1.1 Evaluation on pre-trained weights

We performed an ablation study to find the optimum value of scaling normalization and k value for the k-NN algorithm. The results are tabulated in table below. Here we have used ground truth hand shape as input to the k-NN.

Sr.	Scale Factor	EPE of Dense SF	EPE of SF @ Mesh Vertices
1	1	522.78	418.11
2	5	4.18	5.38
3	10	2.74	4.32
4	**15**	**2.72**	**4.27**
5	20	2.71	4.30

Table 4.1: Ablation Study on scaling normalization at k = 3

Based on results in Table **??**, a scale factor of 15 was chosen for normalization. Another ablation was carried out to choose an optimum value of k in the k-NN algorithm. Results are tabulated.

As expected raising the value of k improves the error metric. However, we chose the value of 15 as an optimum value based on the knee point rule of thum used in k-NN clustring. This is

Sr.	**k**	**EPE of SF @ Mesh Vertices**
1	1	4.33
2	3	4.27
3	5	4.24
4	7	4.22
5	9	4.20
6	11	4.18
7	13	4.17
8	**15**	**4.15**
9	17	4.14
10	19	4.13
11	21	4.12
12	23	4.11
13	25	4.10
14	27	4.09
15	30	4.08
16	35	4.07
17	40	4.05
18	45	4.04
19	50	4.03
20	55	4.02
21	60	4.01
22	65	4.00
23	70	3.99
24	75	3.98
25	80	3.98

Table 4.2: Ablation Study on value of k in k-NN at Scale Factor = 15

evident from the figure.

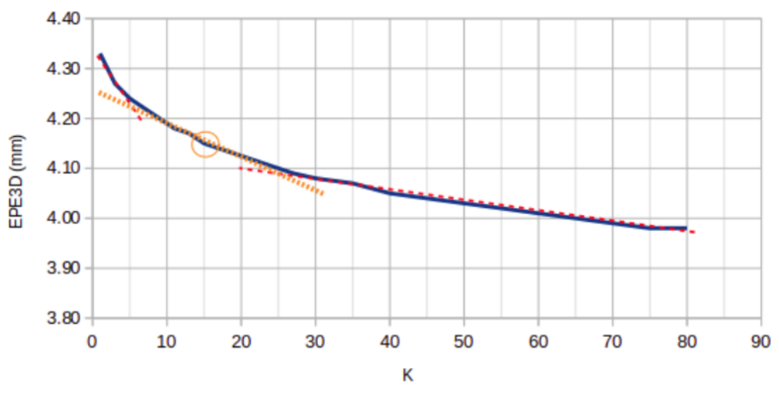

Figure 4.1: Ablation Study on value of k in k-NN at Scale Factor = 15

4.1.2 Fine-tuning the Bi-PointFlowNet Model on our dataset

We used the dense sceneflow ground truth generated during data pre-processing for fine tuning the Bi-PintFlowNet. The results are tabulated below. Note that the k value and scale factor both are fixed at 15.

Sr.	Model Weights	EPE of Dense SF	EPE of SF @ Mesh Vertices
1	Pre-Trained	7.76	7.42
2	**Fine-Tuned**	**2.86**	**3.72**
Percentage Imporovement		63.14 %	49.86 %

Table 4.3: Fine-tuning Bi-PointFlowNet

4.2 Hand Shape Estimator

We first trained the pose estimator without the refinement module. Once trained, we then trained the refinement module and local feature extractor while keeping weights of the MLP and global feature extractor fixed. Results are tabulated.

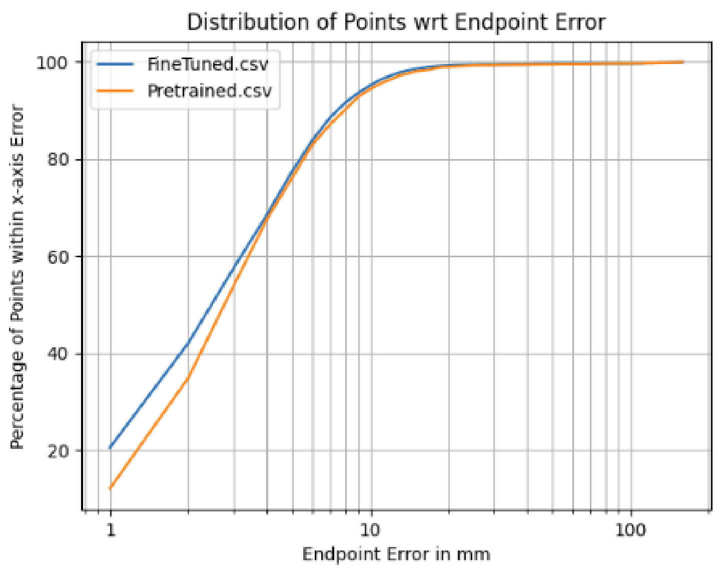

Figure 4.2: Fine-tuning Bi-PointFlowNet

Sr.	Model Type	MVPE of Hand Shape
1	Global Feature Extractor and MLP	10.74
2	**With Refinement Module**	**9.7**

Table 4.4: Evaluation Results of Hand Shape Estimator

4.3 Hand Shape Tracker

In this section, we present the results of the complete pipeline. Here we have fixed the weights of the sceneflow estimator, global feature extractor and initial shape estimator. Hence only the local feature extractor and the refinement module are trained. Results are shown below:

Sr.	Model Type	MVPE of Hand Shape
2	Hand Tracker	9.46

Table 4.5: Evaluation Results of Hand Shape Tracker

4.4 Comparison with State of The Art (SOTA) on DexYcb Dataset

Following table shows a comparison of our approach with the SOTA methods on DexYcb dataset's S0 split evaluation. All papers that are referred below are from 2023. Older papers are not shown.

Sr.	Model Type	MVPE of Hand Shape
1	POEM [48]	6.13
2	MVP [33]	9.77
3	HandOccNet [41]	13.1
4	DiffHand [44]	12.1
5	H2ONet [47]	12.7
6	HFL [45]	12.56
7	**ours**	**9.46**

Table 4.6: Comparison of our approach with SOTA on DexYcb Dataset

Qualitative Results Following figures show the qualitative comparison between the ground truth and our predicted shape. In these images, blue is the ground truth while red is the predicted one.

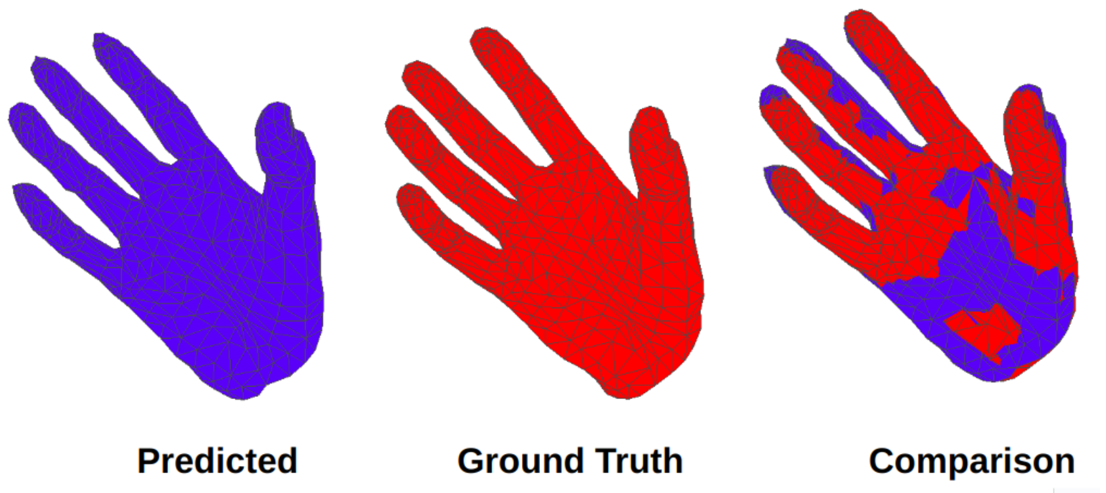

Figure 4.3: Qualitative Results of Our Hand Tracker 1

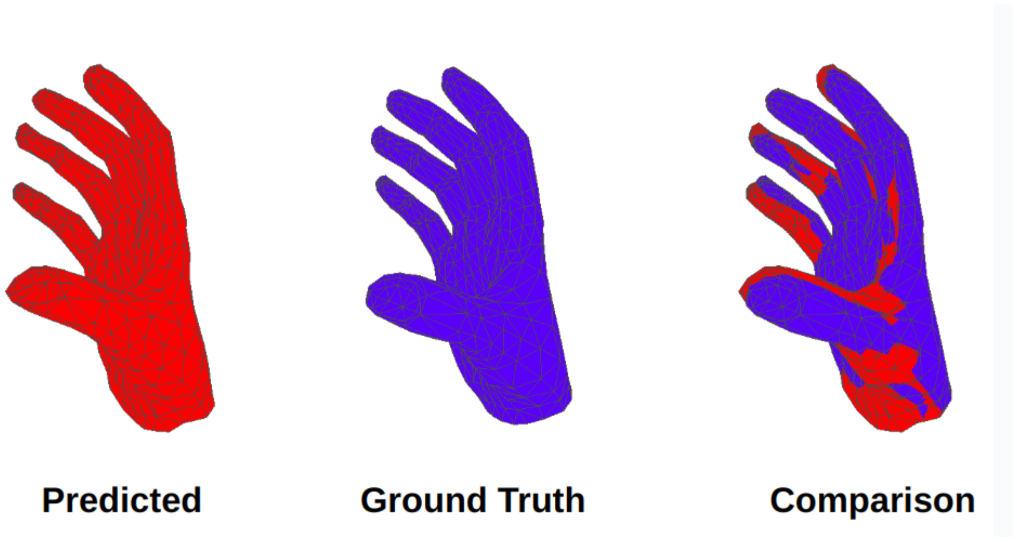

Figure 4.4: Qualitative Results of Our Hand Tracker 2

CHAPTER 5

Discussion

We have developed a hand tracker pipeline that can accurately track the hand shape in next frames given some initial shape or mesh estimate. Thre results show that the MVPE is small and comparable with the SOTA. However the quality of mesh or shape can still be improved as shown in the qualitative results. This can be achieved by introducing the parametric model such as MANO [42].

CHAPTER 6

Conclusion

The multiview hand tracker is a very challenging task and is heavilty impacted by occlusions. We have developed a occlusion resistant multiview tracker that directly predicts the hand mesh or shape in the next frame. The qualitative and quantitative results show that our approach is one of the best in terms of MVPE second only to [**POEM**].

6.1 Future Work

- Although results are good but they can still be improved further by improving the accuracy of global translation. A small network may be introduced this purpose. Another important aspect is the extrinsics of the cameras.
- It has been observed that the camera views do not align perfectly by using extrinsics alone as there is some error. A calibration procedure may be adopted or a correction technique may be introduced to improve this aspect. Resolving this issue will also improve accuracy.
- The quality of mesh or shape can be improved by incorporating a parametric model such as MANO [42]

CHAPTER 7

Recommendations

This research project shows that multiview depth images can be used to predict the 3d hand shape very accurately. Such a technique can be directly deployed in mission critical human computer interfaces.

Bibliography

[1] Mengqi Ji et al. “Surfacenet: An end-to-end 3d neural network for multiview stereopsis”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2307–2315.

[2] Abhishek Kar, Christian Häne, and Jitendra Malik. “Learning a multi-view stereo machine”. In: *Advances in neural information processing systems* 30 (2017).

[3] Charles R Qi et al. “Pointnet: Deep learning on point sets for 3d classification and segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.

[4] Charles Ruizhongtai Qi et al. “Pointnet++: Deep hierarchical feature learning on point sets in a metric space”. In: *Advances in neural information processing systems* 30 (2017).

[5] Erich Schubert et al. “DBSCAN revisited, revisited: why and how you should (still) use DBSCAN”. In: *ACM Transactions on Database Systems (TODS)* 42.3 (2017), pp. 1–21.

[6] Yangyan Li et al. “Pointcnn: Convolution on x-transformed points”. In: *Advances in neural information processing systems* 31 (2018).

[7] Yao Yao et al. “Mvsnet: Depth inference for unstructured multi-view stereo”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 767–783.

[8] Adnane Boukhayma, Rodrigo de Bem, and Philip HS Torr. “3d hand shape and pose from images in the wild”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 10843–10852.

[9] Rui Chen et al. “Point-based multi-view stereo network”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 1538–1547.

[10] Liuhao Ge et al. “3d hand shape and pose estimation from a single rgb image”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 10833–10842.

[11] Yana Hasson et al. “Learning joint reconstruction of hands and manipulated objects”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 11807–11816.

[12] Yana Hasson et al. “Learning joint reconstruction of hands and manipulated objects”. In: *CVPR*. 2019.

[13] Nikos Kolotouros, Georgios Pavlakos, and Kostas Daniilidis. “Convolutional mesh regression for single-image human shape reconstruction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4501–4510.

[14] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. “Flownet3d: Learning scene flow in 3d point clouds”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 529–537.

[15] Hugues Thomas et al. “Kpconv: Flexible and deformable convolution for point clouds”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 6411–6420.

[16] Wenxuan Wu, Zhongang Qi, and Li Fuxin. “Pointconv: Deep convolutional networks on 3d point clouds”. In: *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*. 2019, pp. 9621–9630.

[17] Wenxuan Wu et al. “Pointpwc-net: A coarse-to-fine network for supervised and self-supervised scene flow estimation on 3d point clouds”. In: *arXiv preprint arXiv:1911.12408* (2019).

[18] Yihui He et al. “Epipolar transformers”. In: *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*. 2020, pp. 7779–7788.

[19] Korrawe Karunratanakul et al. “Grasping field: Learning implicit representations for human grasps”. In: *2020 International Conference on 3D Vision (3DV)*. IEEE. 2020, pp. 333–344.

[20] Dominik Kulon et al. “Weakly-supervised mesh-convolutional hand reconstruction in the wild”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 4990–5000.

[21] Gyeongsik Moon and Kyoung Mu Lee. “I2l-meshnet: Image-to-lixel prediction network for accurate 3d human pose and mesh estimation from a single rgb image”. In: *Computer*

Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16. Springer. 2020, pp. 752–768.

[22] Gilles Puy, Alexandre Boulch, and Renaud Marlet. “Flot: Scene flow on point clouds guided by optimal transport”. In: *European conference on computer vision*. Springer. 2020, pp. 527–544.

[23] Edoardo Remelli et al. “Lightweight multi-view 3D pose estimation through camera-disentangled representation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 6040–6049.

[24] Yuxiao Zhou et al. “Monocular real-time hand shape and motion capture using multi-modal data”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 5346–5355.

[25] Yu-Wei Chao et al. “DexYCB: A benchmark for capturing hand grasping of objects”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 9044–9053.

[26] Ping Chen et al. “I2uv-handnet: Image-to-uv prediction network for accurate and high-fidelity 3d hand mesh modeling”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 12929–12938.

[27] Xingyu Chen et al. “Camera-space hand mesh recovery via semantic aggregation and adaptive 2d-1d registration”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 13274–13283.

[28] Meng-Hao Guo et al. “Pct: Point cloud transformer”. In: *Computational Visual Media* 7 (2021), pp. 187–199.

[29] Shreyas Hampali, Sayan Deb Sarkar, and Vincent Lepetit. “HO-3D_v3: Improving the accuracy of hand-object annotations of the ho-3d dataset”. In: *arXiv preprint arXiv:2107.00887* (2021).

[30] Kevin Lin, Lijuan Wang, and Zicheng Liu. “End-to-end human pose and mesh reconstruction with transformers”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 1954–1963.

[31] Kevin Lin, Lijuan Wang, and Zicheng Liu. “Mesh graphormer”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 12939–12948.

[32] Zachary Teed and Jia Deng. “Raft-3d: Scene flow using rigid-motion embeddings”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 8375–8384.

[33] Jianfeng Zhang et al. “Direct multi-view multi-person 3d pose estimation”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 13153–13164.

[34] Hengshuang Zhao et al. “Point transformer”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 16259–16268.

[35] Zerui Chen et al. “Alignsdf: Pose-aligned signed distance fields for hand-object reconstruction”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 231–248.

[36] Wencan Cheng and Jong Hwan Ko. “Bi-pointflownet: Bidirectional learning for point cloud based scene flow estimation”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 108–124.

[37] Shangchen Han et al. “UmeTrack: Unified multi-view end-to-end hand tracking for VR”. In: *SIGGRAPH Asia 2022 Conference Papers*. 2022, pp. 1–9.

[38] Deying Kong et al. “Identity-aware hand mesh estimation and personalization from rgb images”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 536–553.

[39] Haisong Liu et al. “CamLiFlow: Bidirectional camera-LiDAR fusion for joint optical flow and scene flow estimation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5791–5801.

[40] Yingfei Liu et al. “Petr: Position embedding transformation for multi-view 3d object detection”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 531–548.

[41] JoonKyu Park et al. “Handoccnet: Occlusion-robust 3d hand mesh estimation network”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 1496–1505.

[42] Javier Romero, Dimitrios Tzionas, and Michael J Black. “Embodied hands: Modeling and capturing hands and bodies together”. In: *arXiv preprint arXiv:2201.02610* (2022).

[43] Guangming Wang et al. “What matters for 3d scene flow network”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 38–55.

[44] Lijun Li et al. “DiffHand: End-to-End Hand Mesh Reconstruction via Diffusion Models”. In: *arXiv preprint arXiv:2305.13705* (2023).

[45] Zhifeng Lin et al. “Harmonious Feature Learning for Interactive Hand-Object Pose Estimation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 12989–12998.

[46] Chensheng Peng et al. “DELFlow: Dense Efficient Learning of Scene Flow for Large-Scale Point Clouds”. In: *arXiv preprint arXiv:2308.04383* (2023).

[47] Hao Xu et al. “H2ONet: Hand-Occlusion-and-Orientation-Aware Network for Real-Time 3D Hand Mesh Reconstruction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 17048–17058.

[48] Lixin Yang et al. “POEM: Reconstructing Hand in a Point Embedded Multi-view Stereo”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 21108–21117.

[49] *Meta’s Oculus Quest Hand Tracking*. URL: https://www.meta.com/blog/quest/oculus-connect-6-introducing-hand-tracking-on-oculus-quest-facebook-horizon-and-more/.

[50] *Snapchat’s Lense Studio*. URL: https://docs.snap.com/lens-studio/home.