Technische Universität München
Fakultät für Elektrotechnik und Informationstechnik

Perception for Human Action: 3D Hand Pose Estimation and Camera Localization

Shile Li

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines Doktors der Ingenieurwissenschaften (Dr.-Ing.) genehmigten Dissertation.

Vorsitzender:	Prof. Dr.-Ing. Eckehard Steinbach
Prüfer der Dissertation:	
	1. Prof. Dongheui Lee, Ph.D.
	2. Prof. Dr. Angela Yao
	3. Priv.-Doz. Dr. Rudolph Triebel

Die Dissertation wurde am 03.03.2021 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am **30.09.2021** angenommen.

Abstract

For Learning from Demonstrations, the most intuitive way to show human demonstrations to a robot is through computer vision. For a successful demonstration of daily tasks, such as object grasping, human hand and objects are two major components to understand the actions, observing them accurately is a vital precondition for the robot to successfully imitate human demonstration. To obtain object models, this thesis investigates how to localize the camera from image sequences for the purpose of object scanning. To achieve camera localization, a novel Iterative Closest Point based visual odometry method is proposed, which improves the selection, matching and weighting stages of conventional methods. On top of that, dynamic environment can be robustly handled by the proposed static point weighting algorithm. Experiments on public dataset have shown state-of-the-art performance of proposed camera localization methods, which is also tested in an object scanning application. To estimate 3D hand pose, this thesis explores deep learning based methods, which use different types of input data: depth image, point cloud, multi-modal data (RGB, depth image, point cloud). In particular, this thesis contributes in the following aspects for hand pose estimation: i) embedded physical constraint in a deep learning framework, ii) novel network design for unordered point cloud input and iii) network design for multi-modal input. Furthermore, this thesis investigates how to utilize clean hand dataset and scanned object model to design hand pose estimation method for hand-object interaction cases, where an object augmentation strategy has shown its feasibility to solve this task. As result, this thesis provides a fast and accurate camera localization pipeline for object scanning application, and proposes several accurate deep learning based methods for different types of input data.

Zusammenfassung

Für das Lernen aus Demonstrationen ist die intuitivste Möglichkeit, einem Roboter menschliche Demonstrationen zu zeigen durch Computer Vision. Für eine erfolgreiche Demonstration einer alltäglichen Aufgabe wie das Griff von Objekten sind menschliche Hand und Objekte zwei Hauptkomponenten, um die Aktionen zu verstehen. Eine genaue Beobachtung ist eine wichtige Voraussetzung für die erfolgreiche Nachahmung menschlicher Demonstrationen durch den Roboter. Um Objektmodelle zu erhalten, wird in dieser Arbeit untersucht, wie die Kamera aus Bildsequenzen zum Zweck des Objektscannens lokalisiert werden kann. Um eine Kameralokalisierung zu erreichen, wird ein neuartiges Visuelle Odometrie auf der Basis der Iterative Closest Point Methode vorgeschlagen, das die Auswahl-, Anpassungs- und Gewichtungsstufen herkömmlicher Verfahren verbessert. Darüber hinaus kann die dynamische Umgebung durch den vorgeschlagenen statischen Punktgewichtungsalgorithmus robust gehandhabt werden. Experimente mit öffentlichen Datasets haben gezeigt, dass die vorgeschlagenen Methoden für Kameralokalisierung auf dem Stand der State-of-the-art sind und auch in einer Objekt-Scan-Anwendung getestet werden. Um die 3D Posen der Hand abzuschätzen, werden in dieser Arbeit Methoden untersucht, die auf Deep Learning basieren und verschiedene Arten von Inputdaten verwenden: Tiefenbild, Punktwolke, multimodale Daten (RGB, Tiefenbild, Punktwolke). Insbesondere trägt diese Arbeit zu den folgenden Aspekten der Handposenschätzung bei: i) eingebettete physikalische Einschränkung in einem Deep-Learning-Framework, ii) neuartiges Netzwerkdesign für ungeordnete Punktwolkeneingaben und iii) Netzwerkdesign für multimodale Eingaben. Darüber hinaus wird in dieser Arbeit untersucht, wie mithilfe eines sauberen Handdatensatzes und eines gescannten Objektmodells eine Handposenschätzmethode für Hand-Objekt-Interaktionsfälle entworfen werden kann, bei denen eine Data Augmentation gezeigt hat, dass diese Aufgabe machbar ist. Als Ergebnis bietet diese Arbeit eine schnelle und genaue Kamera-Lokalisierungs-Pipeline für die Objekt-Scan-Anwendung und schlägt mehrere genaue Deep-Learning-basierte Methoden für verschiedene Arten von Inputdaten vor.

Acknowledgements

First, I would like to thank my supervisor Prof. Dongheui Lee, for her patient and consistent guidance during the running of this project.

Moreover, I would like to thank all members from the HCR lab, past and present, for their companionship. Working in a heterogeneous team helped me to think things from different perspectives. Special thanks to Jan Wöhlke and Haojie Wang, for their collaborative effort to this project.

I would also like to thank Linlin Yang and Prof. Angela Yao, for the fruitful discussions and collaboration in the hand pose estimation project.

Thank you to Yueli Chen, for all her love and support.

Special thanks to my parents, who set me off on this road a long time ago.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations

Nowadays, robots don't just repeat hard-coded motions anymore, they start to gain the ability to learn and adapt to novel tasks more quickly. The most important way to learn a novel task without complicated engineering is the technique called Learning from Demonstration (LfD) [13], in which the human demonstrates the robot how to perform a task and the robot learns to encode the demonstrated skills to its own "language", or representation, such that it can later reproduce a certain skill with the encoded representation. The research field of LfD is flourishing, where researchers are trying to make the learning process as intuitive as possible for the end-users. For an intuitive demonstration, one can use kinaesthetic teaching to physically move the robot [70], or use a simulation environment with a haptic device as input [96], or use a computer vision system to capture human's movement directly with a camera [69, 46]. Among all the demonstration options, computer vision is the most intuitive option for the end-users as it only requires a camera, where the human can perform the task as he/she usually does in daily life. In such a computer vision system, the task relevant environment information has to be observed robustly, which includes human motions, object identities, object poses etc..

Among all the environment information, human hand pose plays the key role in LfD, where it is usually the imitation target of the robot's gripper for many tasks, such as object grasping [103], object handling [2] (Figure 1.1(c)), in which the hand could be interacting with other objects (*hand-object interaction case*) or floating in the air without touching other objects (*clean hand case*). Besides that, hand pose estimation is also used in many other applications. For example, in Augmented Reality, the head-mounted camera captures the image of human hand, then different kinds of avatar hands can replace the human hand's appearance (Figure 1.1(a)), with the accurate hand pose estimation as a prerequisite. Another example is the usage in gesture recognition, where the robot can understand human's intention from specific gestures. For example, Figure 1.1(b) shows a single hand can posture differently for the number from 0 to 9 [1]. Furthermore, hand pose estimation can be also used for a controlling interface between human and machine, e.g. leap

[1]https://i.pinimg.com/originals/9d/37/13/9d3713bd57ab331f85ceb473b553cf2d.jpg

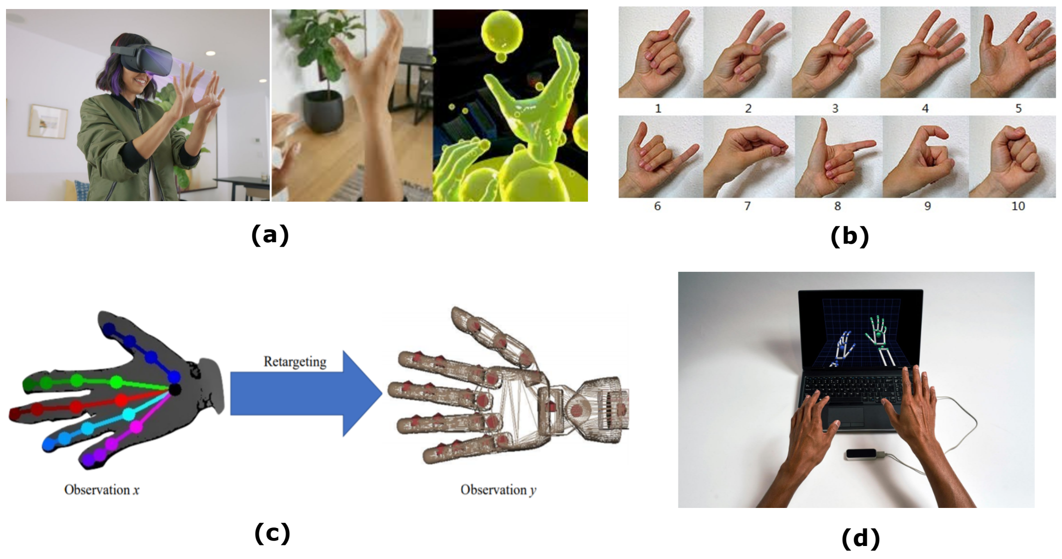


FIGURE 1.1: Applications of hand pose estimation. (a) Oculus VR. (b) Chinese gestures for numbers. (c) Hand pose retargeting to robotic hand [2]. (d) Leap Motion.

motion system can be used a "virtual mouse" to control the laptop (Figure 1.1(d)).

Apart from the hand pose, object also plays a vital role in LfD, where semantic information can be extracted from object's identity and pose. Furthermore, if the hand is interacting with the object, a known object model can benefit the hand pose estimation as well. For many other algorithms related to object observation, the object model needs to be known in prior. To generate the object's model, the first step is to scan the object with the camera, where the object is captured from different viewpoints to cover the whole surface and the observations are merged into the final object model. The key-step in the scanning process is to estimate the camera's relative pose between the different viewpoints. In this thesis, visual odometry is used to estimate the its ego-motion, where the camera is moved around the static object to capture image sequences and each image frame's camera pose relative to the first frame has to be estimated.

The need of observation of human hand and object motivates us to investigate the following research topics in this thesis:

- How to estimate camera's ego-motion from image sequences, in order to generate object models?
- How to estimate human hand pose using different types of sensor data, for a clean hand without object?
- How to utilize clean hand dataset and scanned object model to design hand pose estimation method hand-object interaction cases?

1.2 Challenges

Hand pose estimation using image data is a challenging task. Firstly, the human hand is a complex and articulated object. If we consider it as a robotic gripper, its own Degrees of Freedom (DoF) is more than 25 [145], on top of that, the global position and orientation of the hand also take additional six DoF. This means at least 31-dimensional parameter has to be estimated from a single image. In addition to the high degrees of freedom, each individual person has a different hand shape, where the hand scale, bone length, skin color, finger length ratio etc. can be very different among the population. Another difficulty is that the hand's appearance in the image suffers from occlusion issue, where not all hand's surface can be observed from single image. The fingers occlude each other in many cases, moreover, if the hand is grasping an object, the object also creates severe occlusion to the hand. The object occlusion makes the problem even harder, where the appearance of hand is mixed up with the object. To tackle the above mentioned difficulties, traditionally, researchers have used tracking based methods [67, 39, 94], which rely on sequence of data and a good initialization pose for this problem, then each frame's estimation is optimized from the previous frame's estimates. However, these methods are prone to tracking failure, which cannot be recovered because tracking methods themselves cannot provide initialization pose. In recent years, deep learning proves to be successful in different research fields. Hand pose estimation is no exception, as state-of-the-art results are all coming from deep learning based methods [127, 152, 75, 33, 101]. Relying on large annotated datasets, deep learning methods are suitable to find out the the highly non-linear mapping function between a single input image and the hand pose output. This thesis also follows the usage of deep learning, in which different types of input data are explored, including depth image, point cloud and RGB image. Using these input data, 2 different scenarios are considered: i) a "clean" hand alone in the air without touching an object, and ii) the hand is interacting with an object.

Camera localization for object scanning is also a non-trivial task, where the relative motion between frames has to be estimated from the change of values of the high-dimensional camera sensor data. In particular for this thesis, in order to scan the object, RGB-D (RGB-Depth) data is used for indoor camera localization. The challenges for camera localization are following. Firstly, the dimensionality of input data is high, e.g. for Kinect sensor, at each time step, two 640x480 images (color and depth) are received. To achieve the real-time online performance, the algorithm has to be designed sophisticatedly to handle this large amount of data stream. Furthermore, in most cases, there is no global marker or map as reference system because there is no prior knowledge of the environment. Therefore, only relative motion between consecutive frames can be directly estimated. This leads to the drift problem, as the frame-to-frame estimation error can be accumulated. The solution for this problem is to use a SLAM (Simultaneous Localization And Mapping) system to incrementally create a global map and correct the potential drift problem. In this thesis, for the purpose of object scanning, a novel fast visual odometry

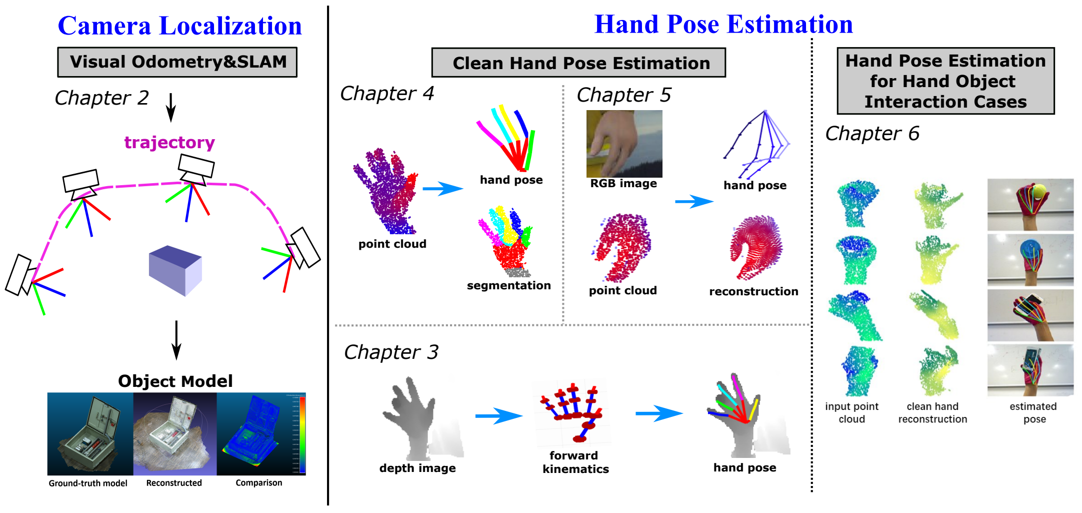


FIGURE 1.2: An overview of this thesis. (Camera localization): this part presents method for estimating camera's ego-motion, in order to generate object models. (Hand pose estimation): this part investigates deep learning based method of human hand pose estimation from camera data. For (Clean Hand Pose Estimation), different types of input data are used, including depth image, point cloud and multi-modal input. For (Hand Object Interaction Cases), object models are used to train a hand pose estimation system that is invariant to objects.

method is developed and integrated to a SLAM system.

1.3 Overview of the thesis

Figure 1.2 gives an overview of the thesis and the connection between different parts.

Camera localization

Chapter 2 presents approaches for indoor camera localization with a RGB-D camera. The first part presents a visual odometry method to estimate camera's ego-motion. A novel visual odometry method, Intensity assisted Iterative Closest Point (IAICP), is proposed. The second part presents two extensions to IAICP, which includes handling of dynamic objects and integration to SLAM system. In the end, this chapter also describes how to merge the observation from different viewpoints to construct object models. The material presented in this chapter is published in:

- *Shile Li and Dongheui Lee. "RGB-D SLAM in dynamic environments using static point weighting." IEEE Robotics and Automation Letters 2.4 (2017): 2263-2270.*
- *Shile Li and Dongheui Lee. "Fast visual odometry using intensity-assisted iterative closest point." IEEE Robotics and Automation Letters 1.2 (2016): 992-999.*

Clean Hand Pose Estimation

Chapter 3-5 present deep learning based hand pose estimation methods for the clean hand cases using different types of input data.

Chapter 3 introduces a depth image based method, in which physical constraint of the hand is embedded into a deep learning structure. The material presented in this chapter is published in:

- *Shile Li*, Wöhlke Jan* and Dongheui Lee. "Model-based hand pose estimation for generalized hand shape with spatial transformer network." European Conference on Computer Vision (ECCV), Extended Abstract Presentation in 4th International Workshop on Observing and Understanding Hands in Action (HANDS2018). 2018. *equal contribution*

- *Wöhlke Jan*, Shile Li* and Dongheui Lee. "Model-based hand pose estimation for generalized hand shape with appearance normalization." arXiv preprint arXiv:1807.00898 (2018). *equal contribution*

Chapter 4 proposes a 3d point cloud based hand pose estimation method, in which a deep learning structure, Residual Permutation Equivariant Layer (ResPEL), is proposed. The network structure is designed to be invariant to the input points' order and still able to be able to extract complex and meaningful features from input data. The material presented in this chapter is published in:

- *Shile Li and Dongheui Lee. "Point-to-pose voting based hand pose estimation using residual permutation equivariant layer." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2019.*

Chapter 5 presents a hand pose estimation method using multi-modal data as input, in which different modalities of hand data (*e.g.* RGB images, depth maps, point clouds, 3D poses, heat maps and segmentation masks) are considered to formulate a cross-modal inference problem. In particular, a multi-modal variational autoencoder (VAE) is used to encode and decode different modalities, where the latent space of individual modalities are aligned using Gaussian product. The developed method is flexible such that it can be trained using multi-modality data (RGB and point cloud) and can be deployed using only RGB data. The material presented in this chapter is published in:

- *Linlin Yang*, Shile Li*, Dongheui Lee and Angela Yao. "Aligning Latent Spaces for 3D Hand Pose Estimation." Proceedings of the IEEE International Conference on Computer Vision (ICCV). 2019. *equal contribution*

Hand pose estimation for hand object interaction cases

Chapter 6 presents a hand pose estimation method for hand-object interaction cases. The backbone of this method is the point cloud based method from Chapter 4. For training this method, augmented autoencoder is used to add artificial random object model to clean hand dataset, such that the trained model can be used for real hand-object cases.

- *Shile Li*, Haojie Wang* and Dongheui Lee, Hand Pose Estimation for Hand-Object Interaction Cases using Augmented Autoencoder, in Proc. IEEE International Conference on Robotics and Automation (ICRA), 2020. *equal contribution*

1.4 Contributions

The contributions of this work are listed as following:

Camera localization

The first section of Chapter 2 presents a fast and robust ICP based visual odometry method that uses both intensity and depth data. The proposed method improves the conventional ICP [8] significantly. In the second section, the proposed visual odometry method is extended to dynamic environment, in which a novel efficient static weighting method is proposed to reduce the influence of dynamic objects on pose estimation. Experiments on TUM RGB-D benchmark [115] shows that for both static and dynamic sequences, the proposed method outperforms previous state-of-the-art methods: DVO [57], Motion-Removal [116] and BaMVO [61].

Clean Hand Pose Estimation

Chapter 3 presents a deep learning based method using depth image as input. For handling of different human hand shapes, it uses a novel kinematic layer, where the hand shape parameters (palm shape, bone lengths) become variables whose values are regressed from the input image.

Chapter 4 presents a deep learning based method for an unordered point cloud. To deal with the unordered structure of the point cloud, the Permuation Equivariant Layer (PEL) [105] is used as the basic element to construct a residual network version of PEL to train the hand pose estimation task. Compared to previous point cloud based methods, our method doesn't require tedious steps such as normal estimation, nearest neighbour estimation. A point-to-pose voting scheme is proposed to merge the information from point-wise local features, which also generates weakly-supervised segmentation results without the need of segmentation ground-truth.

Chapter 5 presents a method using multi-modal data, which includes 2D RGB image and 3D point cloud. It learns a joint latent representation that leverages other modalities as weak labels to improve RGB-based hand pose estimation, where experiments on public benchmarks: STB and RHD dataset [149, 152] show that encoding and decoding the point cloud of the hand surface can improve the quality of the joint latent representations.

Hand pose estimation for hand object interaction cases

Chapter 6 presents a method for handling hand-object interaction cases. It uses an augmentation strategy to simulate hand object interaction cases utilizing existing large clean hand datasets. An auxiliary clean hand reconstruction decoder is proposed to improve the quality of the latent space, which in turn improves the hand pose accuracy.

Chapter 2

Camera localization for object modelling

In order to scan an object to generate object models, camera's ego-motion has to be estimated, such that observation from different viewpoints can be merged in a common coordinate system. This chapter presents camera localization methods for a RGB-D camera. In the first section, visual odometry method for static environment will be introduced. Then in the second section, the extension for dynamic environment, as well as the integration to a SLAM system will be presented. Finally, the application usage in object modeling will be shown. This chapter includes the content of following publications:

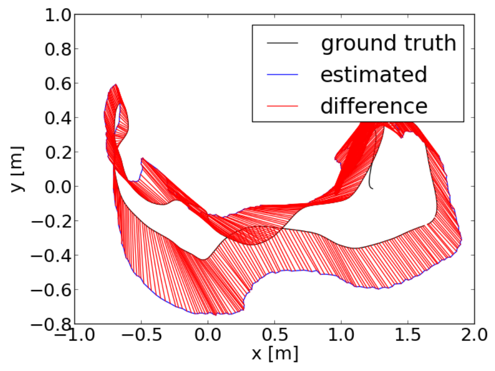
- *Shile Li and Dongheui Lee. "RGB-D SLAM in dynamic environments using static point weighting." IEEE Robotics and Automation Letters 2.4 (2017): 2263-2270.*
- *Shile Li and Dongheui Lee. "Fast visual odometry using intensity-assisted iterative closest point." IEEE Robotics and Automation Letters 1.2 (2016): 992-999.*

2.1 Fast visual odometry

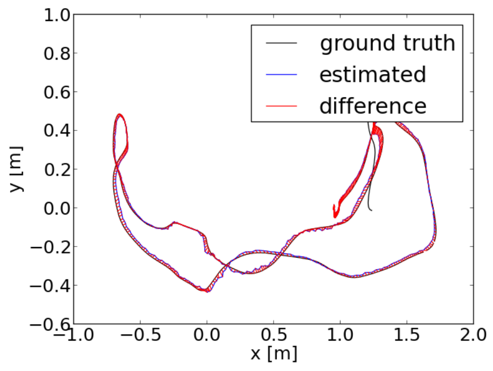
2.1.1 Motivation

Six degrees of freedom (DOF) odometry estimation from visual data, i.e. estimation of camera's position and orientation from images, is one of the most active research area in the last decade [57, 88, 87, 38, 64, 135, 114]. Visual odometry is important for a wide range of robotic applications such as localization and mapping tasks. Furthermore, understanding ego-motion from visual odometry can provide additional sensor information for tasks such as obstacle avoidance [110], object pose estimation [65], scene flow estimation [130]. Especially visual odometry provides 3D pose, whereas wheel odometry or GPS navigation provide only 2D pose. In the context of this paper, the estimated 3D pose of camera serves as the prerequisite for object scanning application.

This section presents a novel method for visual odometry estimation from a RGB-D camera. The camera motion is estimated by aligning a source to a



(a) Visual odometry estimated from the conventional ICP method [8]



(b) Visual odometry estimated from the proposed ICP method

FIGURE 2.1: Comparison of estimated camera trajectories on "fr1/desk" sequence from TUM RGB-D benchmark [115]. Our method improves the conventional ICP greatly.

target RGB-D frame using an intensity assisted Iterative Closest Point (ICP) algorithm. As the example shown in Figure 2.1, the proposed method improves the conventional ICP significantly. The proposed method differs from the conventional ICP with following aspects:

- An intelligent salient point selection method for the source frame is proposed, where points that provide valuable information for ICP are selected. With the reduced point number, substantial computation time is saved.
- With robust statistics on the real data [48], intensity values are integrated into correspondence estimation and correspondence weighting stages of ICP.
- The proposed method runs with a single CPU thread in real-time (78 Hz) with state-of-the-art accuracy on the TUM dataset [115].

The proposed method runs real-time with single core CPU thread, hence it is suitable for applications with limited computation resources. The evaluation on TUM RGB-D benchmark shows that in the majority of the tested sequences, the proposed method outperforms state-of-the-art accuracy in terms of translational drift per second with a computation speed of 78 Hz.

2.1.2 Related works

In the last decade, the availability of lightweight RGB-D sensors such as Asus Xtion raised the popularity of visual odometry estimation from both color and depth images or depth images alone. The recent proposed depth based visual odometry algorithms can be categorized into two groups.

Pixel-wise energy minimization

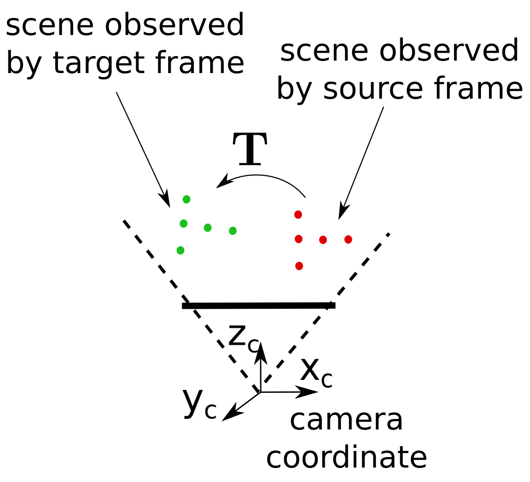
The first group formulates the task as an energy minimization problem [58, 57, 124, 83]. The energy function is devised from pixel-wise photometric and/or depth residual error between the target image and the warped source image, where "warping" is performed by rigid transforming the source image and projection onto the target image. This problem is iteratively solved by numerical optimization such as gradient descent method, where linearized Jacobian matrix with respect to the 6 DOF motion is required for each optimization iteration. These methods are efficient to solve, however, it is based on a strong assumption that the energy function is locally smooth with respect to the 6 DOF pose, which is often not true due to the non-linear nature of image and sensor noises. Therefore a coarse-to-fine strategy is used in most methods, where large motion is recovered using coarser level and small motion is estimated using finer level, but this will require extra computation time to build the image pyramid and time to estimate the image gradient for each pyramid level.

Iterative Closest Point based methods

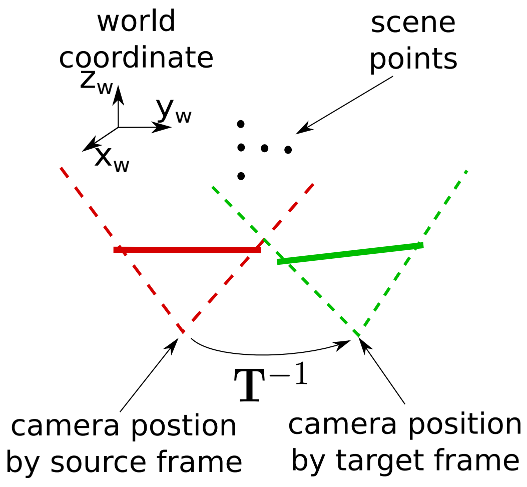
The second group [88, 43] relies on the classic registration method: Iterative Closest Point (ICP) [8]. Corresponding points between the source and the target frames are first estimated based on a certain metric. Then the relative transformation is estimated with a closed-form solution to minimize the distances between correspondences. The above two steps are performed iteratively until a convergence criterion or the maximum iteration number is reached. However ICP suffers from the risk to be trapped in a local minimum, especially in cases of large camera motion or lack of 3D structure in the observed scene. One option to avoid local minimum is to use a feature (such as SIFT [78], SURF[7]) based alignment first and use ICP only as a refinement step [43, 23]. The feature based method improves the probability of convergence in the global minimum, however detection, description and matching of sophisticated keypoint require substantial computation time that hinders the performance to keep up the camera frame rate. Another option is to establish correspondences using normal projection instead of finding nearest points [88, 134, 135], which helps ICP to avoid some local minimum. However surface normal estimation requires even more computation than a feature based method, because the surface normal is densely needed for each pixel. Meilland et. al. [83] choose 3D pixels which best condition the 6 degrees of freedom of the camera, but their method still needs to compute the image Jacobian.

Real-time requirement

Dealing with a large amount of color and depth data, achieving real-time performance that keeps up the camera frame rate (30 Hz) is challenging. In order to perform online visual odometry, some approaches that only use single core CPU [58, 57, 52], need to perform their algorithms on lower resolution


scene observed by target frame
scene observed by source frame
T
z_c
x_c
y_c
camera coordinate

(a) static scene observed by camera in two different frames


world coordinate
scene points
z_w
y_w
x_w
T^{-1}
camera postion by source frame
camera position by target frame

(b) corresponding camera motion in the world coordinate

FIGURE 2.2: Visual odometry method estimates the camera motion by aligning the scene points observed from different frames. (a): The scene points are rigidly transformed with **T** in the camera coordinate. (b): The corresponding camera motion in the world coordinate is the inverse of **T**.

images than the original sensor data to compromise between accuracy and processing time. Some other approaches that directly perform on the original resolution [88, 38, 135] require state-of-the-art Graphics Processing Unit (GPU) to parallelize their algorithms, however not every mobile platform is equipped with a GPU.

2.1.3 Preliminaries

Camera model

Given a 3D point $\mathbf{p} = (x, y, z, 1)^T$ in homogeneous coordinate relative to the camera, the image pixel coordinate $\mathbf{x} = (u, v)^T$ ($u \in [0, height - 1], v \in [0, width - 1]$) of $\mathbf{p}$ is calculated with the camera projection function π:

$$\mathbf{x} = \pi(\mathbf{p}) = (\frac{x f_x}{z} + o_x, \frac{y f_y}{z} + o_y)^T, \tag{2.1}$$

where *height* and *width* are the pixel number in image's x- and y- direction, f_x, f_y are the camera focal lengths and o_x, o_y are the camera center coordinates.

As shown in Figure 2.2, in the camera coordinate, if the scene points are rigidly transformed with a transformation matrix $\mathbf{T} \in \mathbb{R}^{4\times4}$, then a camera motion $\mathbf{T^{-1}}$ in the world coordinate is implied. In the camera coordinate, the point $\mathbf{p}$ is then changed to: $\mathbf{p}' = \mathbf{Tp}$, and its pixel coordinate becomes $\mathbf{x}' = \pi(\mathbf{p}') = \pi(\mathbf{Tp})$.

At time step t, an intensity image I_t and an organized point cloud P_t with the resolution $width \times height$ are obtained, where $P_t(i)$ indicates the ith point in P_t. Intensity value of pixel $\mathbf{x}$ is $I_t(\mathbf{x}) \in [0, 255]$, the pixel $\mathbf{x}$'s corresponding

3D point $\mathbf{p}$ is indicated as $P_t(ind(\mathbf{x}))$, where $ind()$ is a function that maps a image coordinate to a point index of a one-dimensional list of organized point cloud:

$$ind(\mathbf{x}) = ind((u,v)^T) = u \times width + v, \tag{2.2}$$

To retrieve the image coordinate $\mathbf{x}$ of a point index i, the inverse function is:

$$\mathbf{x} = ind^{-1}(i) = (\lfloor \frac{i}{width} \rfloor, i - \lfloor \frac{i}{width} \rfloor width)^T. \tag{2.3}$$

Conventional ICP

Let us consider two subsequent frames that need to be aligned as $< I_1, P_1 >$ for the source frame and $< I_2, P_2 >$ for the target frame. The conventional ICP method [8] uses the point cloud pair P_1, P_2 to iteratively find the optimal relative rigid transformation matrix $\mathbf{T}^*$, such that after transforming P_1 with $\mathbf{T}^*$, the source observation will be aligned with the target observation (Figure 2.2). ICP is an iterative method, the matrix $\mathbf{T}^*$ is initialized as an identity matrix, the kth iteration of the ICP algorithm can be summarized as follows:

1) Search for each point of the source cloud a closest point in the target cloud as correspondence. For the ith point in P_1, the index of the corresponding point in P_2 is denoted as $c(i)$, where

$$c(i) = \underset{j}{\operatorname{argmin}} \|\mathbf{T}^* P_1(i) - P_2(j)\|. \tag{2.4}$$

2) Compute the optimal incremental transformation $\mathbf{T}_k$ that minimizes the distances between the established correspondences:

$$\mathbf{T}_k = \underset{\mathbf{T}}{\operatorname{argmin}} \sum_i \|\mathbf{T}\mathbf{T}^* P_1(i) - P_2(c(i))\| \tag{2.5}$$

This is usually solved with a closed-form solution [22] such as Singular Value Decomposition [4].

3) Update $\mathbf{T}^*$ as:

$$\mathbf{T}^* \leftarrow \mathbf{T}_k \mathbf{T}^*. \tag{2.6}$$

The above steps are performed iteratively until the incremental transformation is smaller than a threshold or the maximum allowable iteration number has reached.

2.1.4 Intensity assisted iterative closest point

Overview

Rusinkiewicz et. al. discussed in [108] about ICP variants, where the variants differ in the following stages: selection, matching, weighting, rejection, error metric and minimizing. Based on the categorization from [108], the proposed ICP method differs from the conventional ICP in the following stages.

- **Selection** - Salient points selection is performed on the source frame, where points that provide valuable information for ICP are selected.

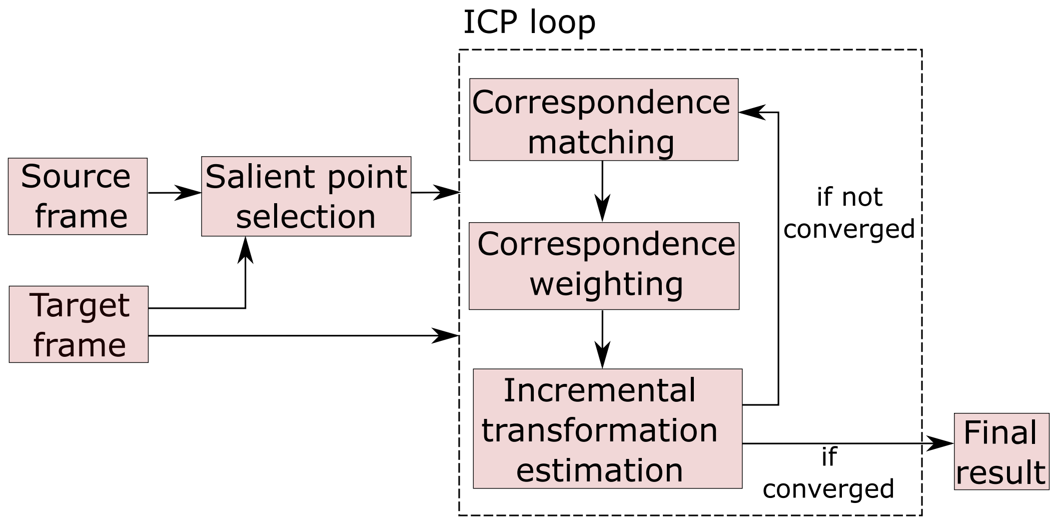


FIGURE 2.3: Intensity assisted ICP overview

For the target frame, the original image resolution is kept without any sampling.

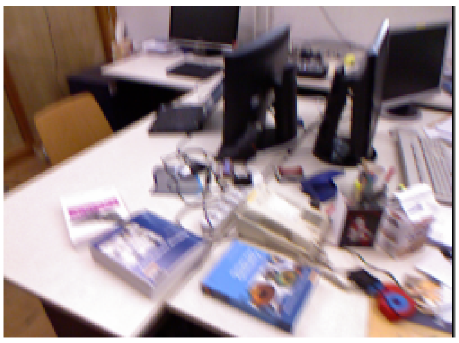
- **Matching** - The search of correspondences is performed by examining nearby points in the image coordinate, where the matching point is determined by considering both intensity and geometric distance.
- **Weighting** - Weighting of corresponding pairs is performed based on robust statistic [48]. This improves the robustness of ICP against false correspondences. Additionally, the sensor noise model of the depth camera is also considered for the weighting.

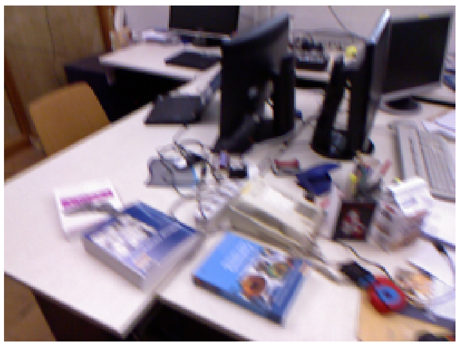
The overview of the proposed method is illustrated in Figure 2.3.

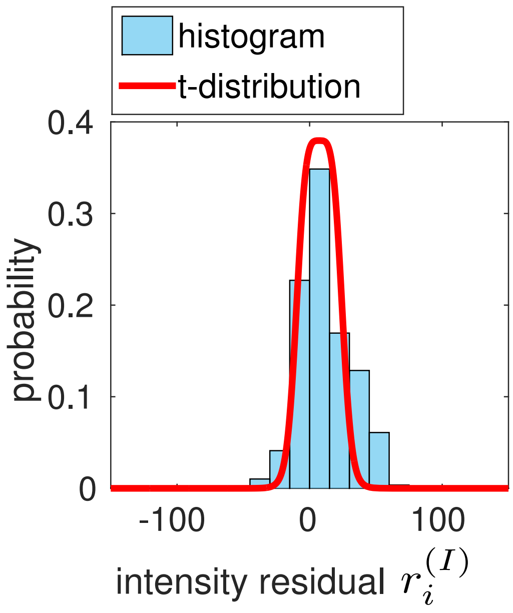
Robust correspondence weighting

With the proposed correspondence matching algorithm, correspondences between source and target frames are established, where corresponding point of the ith point in source frame $P_1(i)$ is $P_2(c(i))$. In practice, not every correspondence is determined correctly. The resulting outliers have a bad influence on the accuracy of the estimated transformation. Moreover the precision of depth value depends on the distance to the camera, where a more distanced point has lower precision for depth value. In the proposed method, robust weighting function is applied to reduce the influence of outliers and to adapt to the sensor noise model, where the ith corresponding pair a weight $w(i)$, thus Equation (2.31) changes to:

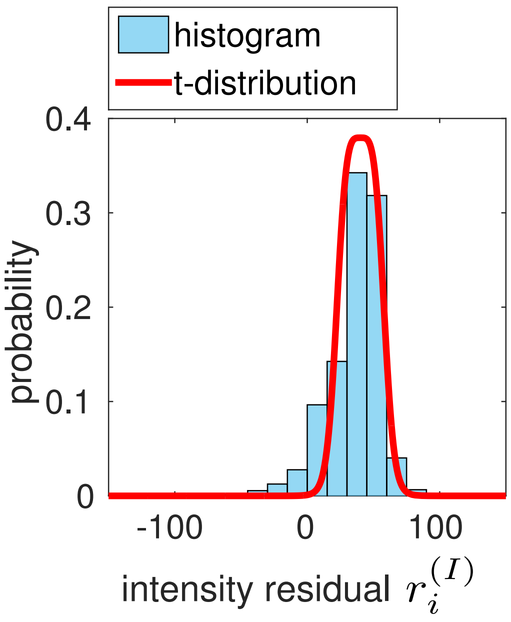
$$\mathbf{T}_k = \operatorname*{argmin}_{\mathbf{T}} \sum_i w(i) \| \mathbf{T}\mathbf{T}^* P_1(i) - P_2(c(i)) \|. \tag{2.7}$$


(a) 220th frame of 'fr1/room'

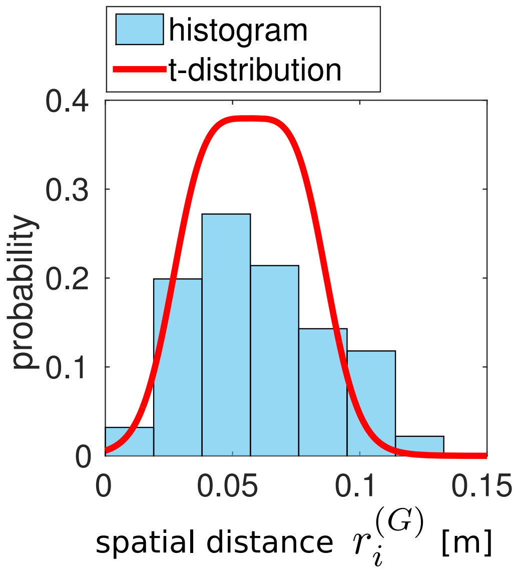

(b) 221st frame of 'fr1/room'



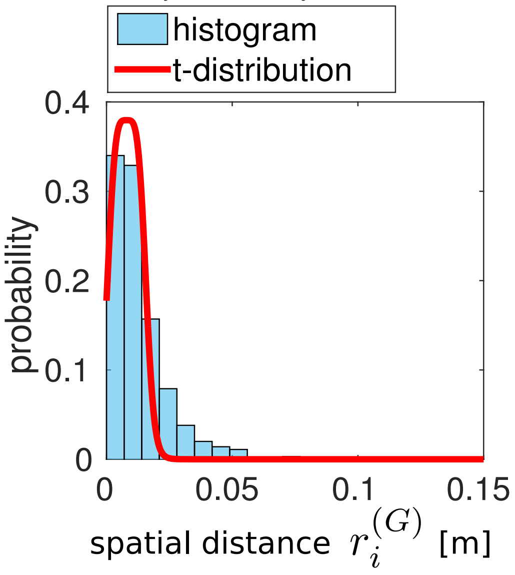
(c) intensity residual by 1st iteration



(d) intensity residual by 30th iteration



(e) spatial distance by 1st iteration



(f) spatial distance by 30th iteration

FIGURE 2.4: (a)(b): The illumination of the 221st frame is lower than the 220th frame due to auto-exposure, the average intensity has dropped by 43 in 221st frame. (c)(d): The intensity residual distribution changes towards the true illumination model as iteration number grows. (e)(f): The distribution of spatial distance between correspondences moves from larger value toward zero as iteration number grows.

Intensity residuals between the correspondences are considered for weighting, the intensity residual $r_i^{(I)}$ of the ith pair is calculated as:

$$r_i^{(I)} = I_1(ind^{-1}(i)) - I_2(ind^{-1}(c(i))). \tag{2.8}$$

Inspired by [58], the Student's t-distribution (a M-Estimator function) is used to weight each intensity residual. Following [68], the derived weighting function based on the t-distribution is:

$$w_I(r_i^{(I)}) = \frac{\nu + 1}{\nu + ((r_i^{(I)} - \mu^{(I)})/\sigma^{(I)})^2}, \tag{2.9}$$

where ν is the degree of freedom of t-distribution, $\nu = 5$ is used as the same in [58]. The mean $\mu^{(I)}$ of all intensity residuals, is estimated as the median value:

$$\mu^{(I)} = Med(\{r_i^{(I)}\}_{i=1}^N), \tag{2.10}$$

where N is the number of correspondences. The standard deviation $\sigma^{(I)}$ is estimated using the median absolute deviation:

$$\sigma^{(I)} = 1.4826\ Median(\{|r_i^{(I)} - \mu^{(I)}|\}_{i=1}^N). \tag{2.11}$$

The t-distribution fits the real data nicely as shown in Figure 2.4(c)(d), which gives outliers very small weights. An important reason to weight the correspondences based on the statistics over all residuals is to consider the changing illumination caused by auto-exposure of the camera. Figure 2.4 gives an example from 'fr1/room' sequence of TUM dataset, the average pixel intensity value has dropped by 43 from Figure 2.4(a) to Figure 2.4(b). By first iteration, the mean value of intensity residuals is still almost zero (Figure 2.4(c)), as iteration number grows, the intensity residual distribution changes towards the correct illumination difference value (Figure 2.4(d)). Then the statistic based weighting ensures larger weights for the correct intensity residuals.

Another component of the weighting function is based on the spatial distance between correspondences. It follows the same procedure as intensity based weighting, where spatial distance is considered as residual:

$$r_i^{(G)} = \|\mathbf{T}^* P_1(i) - P_2(c(i))\|. \tag{2.12}$$

The computation of $w_G(r_i^{(G)})$ follows the same as the weighting for intensity residual.

$$w_G(r_i^{(G)}) = \frac{\nu + 1}{\nu + ((r_i^{(G)} - \mu^{(G)})/\sigma^{(G)})^2}. \tag{2.13}$$

The t-distribution also fits the distribution of spatial residuals nicely as shown in Figure 2.4(e)(f). By the first iteration (Figure 2.4(e)), the spatial distance is relative large due to the camera motion, the variance is also large due to the rotation component of the camera motion. By later iteration (Figure 2.4(f)), the current transformation estimate $\mathbf{T}^*$ gets closer to the true transformation, thus the mean spatial residual value is closer to zero and the variance is also much smaller.

To compensate the noises caused by depth sensor model, another weight $w_S(i)$ is assigned. The sensor noise model devised from [89] is used without considering the influence of surface normal. The weight for the ith corresponding pair is computed based on the average depth value of $P_1(i)$ and $P_2(c(i))$:

$$w_S(i) = \frac{1}{0.0012 + 0.0019(\frac{\mathbf{e}_z^T P_1(i) + \mathbf{e}_z^T P_2(c(i))}{2})^2}, \tag{2.14}$$

where $\mathbf{e}_z^T = (0, 0, 1, 0)$.

The total weight $w(i)$ for ith corresponding pair combines the three weights from intensity difference, spatial distance and sensor noise model:

$$w(i) = w_I(r_i^{(I)}) w_G(r_i^{(G)}) w_S(i), \tag{2.15}$$

where the three weight components are complementary to downweight outlying correspondences and correspondences with higher sensor noise. The multiplication in Equation (2.15) ensures that only corresponding pairs that have relative larger weight in all three components, are assigned with large total weight.

Intensity assisted point matching

The conventional ICP method establishes correspondences only based on spatial distance (Equation (2.4)). In this paper, intensity difference and spatial distance are both used to determine "closest" point by considering the robust statistic obtained from Equation (2.10)(2.11). Given the query point pair $< P_1(i), P_2(j) >$, the score function for this query pair is:

$$\begin{aligned} s(i,j) &= w_I(I_1(ind^{-1}(i)) - I_2(ind^{-1}(j))) \\ &\quad w_G^{(0)}(\|\mathbf{T}^* P_1(i) - P_2(j)\|), \end{aligned} \tag{2.16}$$

$w_I()$ and $w_G^{(0)}()$ are the robust function derived from the previous ICP iteration (Equation (2.9) (2.13)). $w_G^{(0)}()$ has the form:

$$w_G^{(0)}(r) = \frac{\nu + 1}{\nu + ((r - 0)/\sigma^{(G)})^2}, \tag{2.17}$$

where the mean value is set to zero, because we adopt the assumption from the conventional ICP [8], where closer point is more probable to be the correspondence. In case of a large camera motion, where closest point assumption does not necessarily hold, the intensity weight term in Equation (2.16) provides an additional constrain compared to the conventional ICP. For the first iteration, where the robust statistic is not available, $\mu^{(I)}$ is set to zero, and $\sigma^{(I)}, \sigma^{(G)}$ are initialized with 10 and 0.04m in our implementation. The matching point for $P_1(i)$ is then the point that results in the highest score:

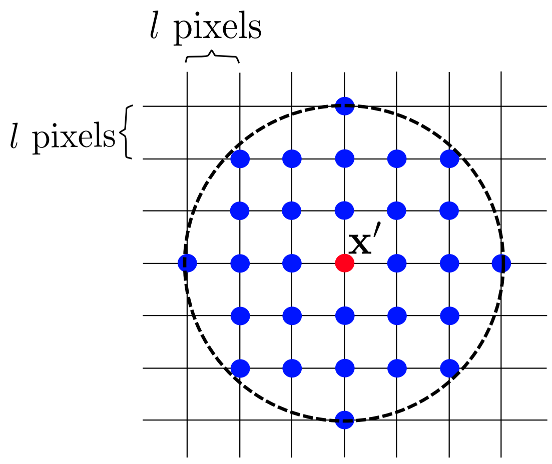


FIGURE 2.5: The search pattern for matching point: the blue points within the circle of the radius $3l$ in the image coordinate are examined

$$c(i) = \operatorname*{argmax}_{j} s(i, j). \tag{2.18}$$

Taking advantage of the organized point cloud, the search of matching point is performed in the image coordinate. Based on the current known transformation T^* from last iteration, $P_1(i)$ is warped onto target frame to determine the area of search, where $P_1(i)$'s image coordinate in the target frame is $\mathbf{x}' = \pi(\mathbf{T}^* P_1(i))$. With $\mathbf{x}'$ as the center, the search pattern is illustrated in Figure 2.5, where the blue points within the circle of the radius $3l$ are examined.

The offset parameter $l \in \mathbb{Z}^+$ (Figure 2.5) is used to control the size of the search area. To cope with large camera motion, the search area should be larger in the initial iterations, thus l is set with larger values. As the iteration number grows and ICP converges, l decreases accordingly until 1, which brings a decreasing search area and increasing precision as ICP proceeds. Notice although a coarse-to-fine scheme is used here, no subsampling is performed on the target image, therefore precision is retained even for coarser levels.

Salient point selection

Due to 3D motion between two frames and projective nature of image, some points in P_1 might be occluded in P_2. If occluded points from P_1 are used for correspondence estimation, the established correspondences is definitely wrong, which influences the accuracy of estimated pose (Equation (2.31)). Therefore, by correspondence estimation, the points from P_1 which have high probability to be occluded in P_2, are discarded. These points are background points that are near the edges of foreground region, where a subtle motion might cause the occlusion. The point $P_1(ind(\mathbf{x}))$ is considered to have high

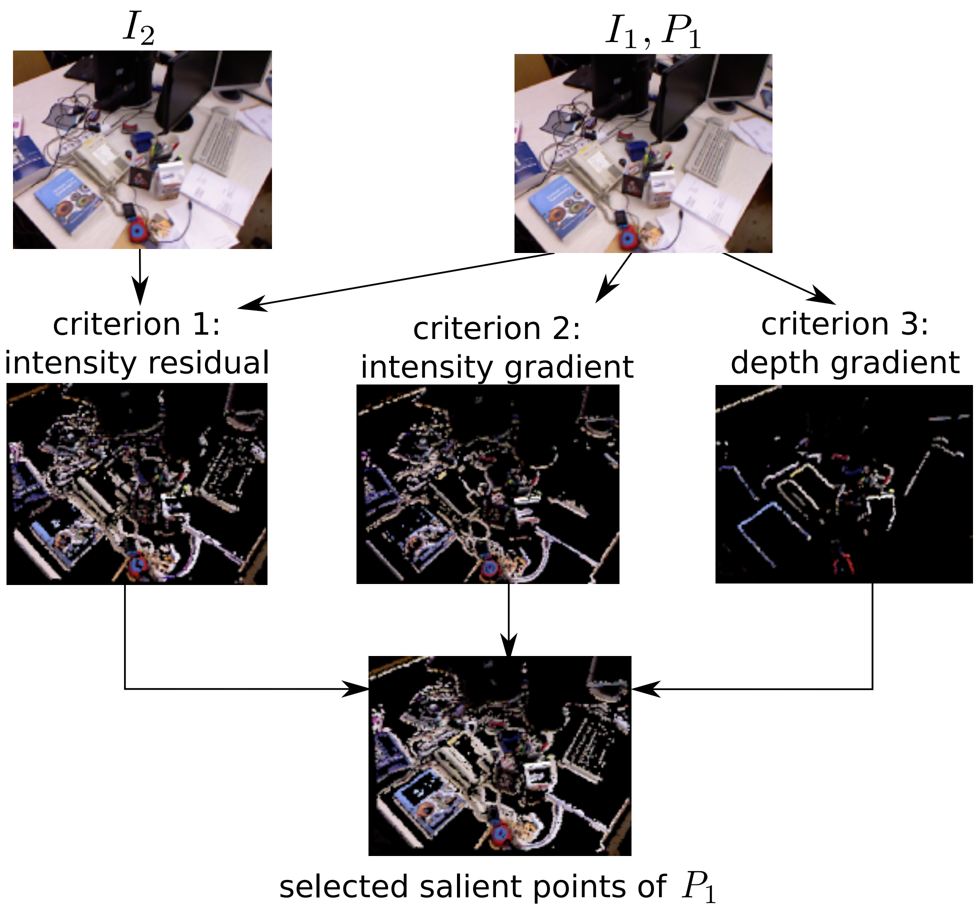

I_2
I_1, P_1
criterion 1: intensity residual
criterion 2: intensity gradient
criterion 3: depth gradient
selected salient points of P_1

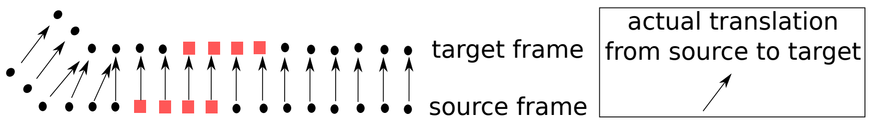
FIGURE 2.6: Salient point selection. Top: the source and the target frame. Middle: selected salient points based on three different criteria. Down: all salient points combining three criteria.

occlusion probability, and is discarded if it satisfies:

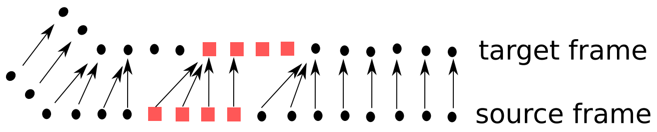
$$\begin{array}{rll} & e_Z^T P_1(ind(\mathbf{x})) - e_Z^T P_1(ind(\mathbf{x}+(0,5)^T)) & > \tau_r, \\ or & e_Z^T P_1(ind(\mathbf{x})) - e_Z^T P_1(ind(\mathbf{x}+(0,-5)^T)) & > \tau_r, \\ or & e_Z^T P_1(ind(\mathbf{x})) - e_Z^T P_1(ind(\mathbf{x}+(5,0)^T)) & > \tau_r, \\ or & e_Z^T P_1(ind(\mathbf{x})) - e_Z^T P_1(ind(\mathbf{x}+(-5,0)^T)) & > \tau_r, \end{array} \quad (2.19)$$

which implies that $P_1(ind(\mathbf{x}))$ is discarded if it has a much larger depth value than a nearby point.

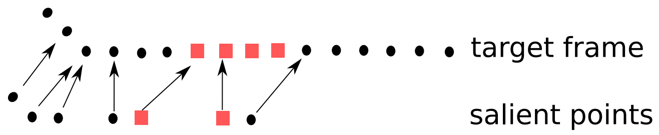
After background point rejection, a lot of the remaining points in the source frame still might fail to find their correct correspondences in the target frame. In particular, the source frame points that lie in homogeneous regions of intensity image or depth image, have higher probability to be matched to false correspondences. Figure 2.7 illustrates example cases of correspondence matching result for a translated scene. For a source frame point in homogeneous region of depth image, the target frame points near the correct match are sometimes closer to the query point (Figure 2.7a), thus many false matches can occur. Even with intensity assisted matching term (Equation (2.18)), a lot of source frame points in homogeneous region of intensity image can still be matched to wrong points (Figure 2.7b). Figure 2.7c shows that by neglecting homogeneous regions, the ratio of correct matches can be increased. With higher ratio of correct matches, the correspondences based incremental transformation result (Equation (2.31)) will be more accurate. To avoid homogeneous regions, three selection criteria are used to determine whether a point in the source frame should be considered.



(a) Conventional ICP



(b) Conventional ICP + weighting with Equation (2.18)



(c) Conventional ICP + weighting with Equation (2.18) + salient point selection

FIGURE 2.7: Point pair matching for a translated scene. (a): Correspondences are established using all points with nearest neighbour criterion, the ratio of correct matches is low. (b): Using our matching score function (Equation (2.18)), some mismatches are corrected, because the matching score includes an intensity term. (c): The ratio of correct matches is further increased, if the homogeneous regions in depth or intensity image are avoided.

Intensity residual based: The first criterion is intensity residual. By computing intensity residual of same pixel between two frames, pixels in homogeneous regions of intensity image do not result in large intensity difference. In contrast, pixels in textured regions or border of homogeneous regions, might result in large intensity residual from camera motion. Therefore the first criterion for salient points is:

$$|I_1(\mathbf{x}) - I_2(\mathbf{x})| > \tau_1. \tag{2.20}$$

Intensity gradient based: Points inside homogeneous region have low intensity gradient, therefore points with high intensity gradient are selected:

$$\begin{aligned} & |I_1(\mathbf{x} + (0,2)^T) - I_1(\mathbf{x} + (0,-2)^T)| && > \tau_2, \\ or\quad & |I_1(\mathbf{x} + (2,0)^T) - I_1(\mathbf{x} + (-2,0)^T)| && > \tau_2, \end{aligned} \tag{2.21}$$

where either gradient in x-direction or y-direction of the image coordinate should be greater than the gradient threshold τ_2. Based on the intensity term from Equation (2.16), these points have higher probability to distinguish their

Algorithm 2.1 Salient point selection

```
Require: - Source frame $< I_1, P_1 >$ and target frame $< I_2 >$
Ensure: - List $list$ that contains indexes of salient points from $P_1$.
  - $list \leftarrow \emptyset$
  for $u = 1:4:width$ do
    for $v = 1:4:height$ do
      - get the point to be checked: $P(ind((u,v)^T))$ with intensity $I((u,v)^T)$
      if $P(ind((u,v)^T))$ satisfies rejection criterion (Equation (2.19)) then
        continue;
      end if
      for c=1:3 do
        if $P(ind((u,v)^T))$ satisfies $c$th criterion (Equation (2.20) or (2.21) or (2.22)) then
          - $list.append(ind((u,v)^T))$;
          - continue;
        end if
      end for
    end for
  end for
```

correct correspondence.

Depth gradient based: Similar to intensity, points inside a homogeneous depth space also have ambiguity to find its correct correspondence. Therefore the third criterion is depth gradient, where points that contains variation in the depth space are selected:

$$\begin{aligned} & \frac{|e_Z^T P_1(ind(\mathbf{x}+(0,2)^T)) - e_Z^T P_1(ind(\mathbf{x}+(0,-2)^T))|}{e_Z^T P_1(ind(\mathbf{x}))} > \tau_3, \\ or \quad & \frac{|e_Z^T P_1(ind(\mathbf{x}+(2,0)^T)) - e_Z^T P_1(ind(\mathbf{x}+(-2,0)^T))|}{e_Z^T P_1(ind(\mathbf{x}))} > \tau_3. \end{aligned} \tag{2.22}$$

In summary, for salient point selection, one rejection criterion and three acceptance criteria are used. The algorithm to retrieve the salient points is described in Algorithm 1. The salient point selection is efficient, because a salient point only needs to meet one of the three criteria. As soon as the point fits one criterion, the algorithm skips the rest of criteria and proceeds to check next point. The salient points selected based on different criteria are illustrated in Figure 2.6.

As in Algorithm 2.1, only every 16th source frame point is checked for its saliency by only checking every 4th row and 4th column. This reduces the computation time for salient point selection stage by a factor of 16, but the lost of accuracy is not too much because no subsampling is performed on the target frame $< I_2, P_2 >$. In contrast, optimization methods that involves dense pixel-wise energy minimization have to [57][52] perform subsampling on both source and target frames to reduce computation cost, which results in a relative higher accuracy lost.

For each ICP iteration, a different subset of salient points is randomly selected. Then only the subset of salient points is used for registration. The reason of the random selection for each iteration is: firstly, it further reduces the computation cost by using less points per iteration; secondly, all selected salient points should have equal possibility to contribute in registration. The number of this randomly selected subset per iteration is set as ca. $100 - 200$, where it is enough to generate accurate 6 DOF pose from this amount of correspondences.

2.1.5 Experimental results

The proposed method is evaluated using the TUM RGB-D benchmark [115]. The benchmark contains 89 RGB-D video sequences, for each video sequence, accurate ground truth for camera motion is provided by a motion capture system. We evaluated our method on 14 video sequences, which are commonly used in the previous publications. For evaluation of visual odometry accuracy, the root mean square error (RMSE) of translational drift in m/s is used, which is a standard metric to measure accuracy of visual odometry method [115].

In the following, the proposed method is first compared with other RGB-D based visual odometry methods [57, 135, 38] that use both intensity and depth data. Then further comparison with depth only based methods [98, 52] is performed, where all intensity related process are turned off. The computation performance is also evaluated with different parameter settings. Finally we show some qualitative result of reconstructed scenes based on our visual odometry method.

The experiments are performed on a desktop computer with Ubuntu 12.04, equipped with Intel Core i7-4790K CPU (4 GHz) and 16GB RAM. Notice that our implementation only runs on a single CPU thread.

Accuracy

For evaluation, commonly used video sequences from TUM datasets in previous publications are selected. For each ICP iteration, a subset of 100 salient points are used. In total, 30 iterations of ICP are performed: 3 levels with 10 iteration per level. For the first 10 iterations, the offset l for correspondence search is set to 6, for the second 10 iterations l is set to 3 and for the last 10 iterations l is set to 1. Every 5th frame is used as the source frame ($< I_1, P_1 >$), and the current frame is used as the target frame ($< I_2, P_2 >$). The threshold values for selecting salient points: $< \tau_r, \tau_1, \tau_2, \tau_3 >$ (Equation (2.20-2.22)) are empirically set as $< 0.02, 30, 30, 0.03 >$ and are fixed in all experiments.

To prove the contribution of each component in our method, 'fr1/desk' and 'fr1/desk2' sequences are used. Selection (s), matching (m) and weighting (w) component of our method are incrementally added to the conventional ICP method, Table 2.1 shows that each component provides an incremental improvement on the accuracy. Among the three components, the selection component contributes the most by neglecting the source frame

TABLE 2.1: On the two 'fr1' desk sequences, each of the three algorithm components provides incremental contribution to the accuracy.

Method	RMSE of translational drift[m/s]		Average improvement
	fr1/desk	fr2/desk	
ICP	0.175	0.182	0%
ICP+s	0.045	0.056	71.7%
ICP+s+w	0.044	0.051	73.4%
ICP+s+m	0.029	0.041	80.39%
ICP+s+m+w	**0.021**	**0.038**	**83.25**%

TABLE 2.2: RMSE of translational drift (m/s): comparison with other RGB-D based method

Sequence	average velocity [m/s]	average angular velocity [°/s]	**Our Method [74]**	RGB +D [57]	RGB +D +KF [57]	RGB +D +KF +Opt [57]	ICP +RGBD [135]	Inverse depth [38]
fr1/desk	0.413	23.327	**0.0217**	0.036	0.030	0.024	0.0393	0.026
fr1/desk2	0.426	29.30	**0.0381**	0.049	0.055	0.050	-	0.0387
fr1/room	0.344	29.882	**0.0416**	0.058	0.048	0.043	0.0622	0.0491
fr2/desk	0.193	6.388	0.0204	-	-	-	0.0208	**0.0121**
fr1/xyz	0.244	8.920	**0.018**	0.026	0.024	0.018	-	-
fr1/rpy	0.062	50.147	**0.031**	0.040	0.043	0.032	-	-
fr1/360	0.210	41.600	**0.072**	0.119	0.119	0.092	-	-
fr1/floor	0.258	15.071	0.10	fail	**0.090**	0.232	-	-
fr1/teddy	0.315	21.320	0.048	0.060	0.067	**0.043**	-	-
fr1/plant	0.365	27.891	**0.018**	0.036	0.036	0.025	-	-
fr3/office	0.249	10.188	**0.016**	-	-	-	-	-
fr3/ nostructure _texture _far	0.299	2.890	**0.047**	0.073	-	-	-	-
fr3/ structure _notexture _far	0.166	4.000	**0.034**	0.038	-	-	-	-
fr3/ structure _texture _far	0.193	4.323	**0.023**	0.039	-	-	-	-

TABLE 2.3: RMSE of translational drift (m/s): comparison with other depth alone based methods

	fr1/desk	fr1/desk2	fr1/room	fr2/desk
Our method (depth only)	**0.0297**	**0.384**	**0.0484**	0.0330
Sparse depth [98]	0.058	0.073	0.073	**0.028**
Fast 3-D [52]	0.0366	0.0528	0.0489	0.0313

points that have higher probability to be incorrectly matched. The matching component provides the second largest contribution by combining intensity term for more accurate correspondence matching process. Although the weighting component provides a small contribution for accuracy, it is essential for providing the parameters of matching score function (Equation (2.16)), which are needed for the matching component.

Table 2.2 shows the comparison of our method and other state-of-the-art visual odometry methods based on both depth and intensity, where in Table 2.2, '-' means that result is not provided in the corresponding literature. Since the optimal parameter settings of other methods are unknown, for fair comparison, we only listed the reported values from the original publications [57, 135, 38]. The proposed method, IAICP, provides the best accuracy in 11 of the 14 tested sequences, and even outperforms the method RGB+D+KF+Opt from [57] that uses loop closure detection and global pose optimization. This is because the proposed method can avoid some local minimum by using sparse and distinctive salient points, while energy minimization based methods rely on all image pixels.

There is another group of methods that only uses depth information, the comparison with two recent ones [98, 52] are conducted. For comparison, depth only based method is simulated by setting all pixels' color to a same value. As shown in Table 2.3, the proposed IAICP method outperforms in most sequences. This indicates that IAICP can also handle poor lightening condition and textureless scene.

Computation time vs. Performance

Real-time capability is crucial for online application, therefore some algorithms require GPU for parallelization and some others operate on lower resolution image and hence loose some accuracy. In contrast, our method uses intelligent salient point sampling method without sacrificing accuracy. Due to sparse sampling in the source image, a high frame rate of 78 Hz using only single CPU thread is achieved. The frame rate is expected to be even higher if GPU programming is used, however 78 Hz is enough for the real-time requirement. The reported computation time of different methods and the hardware settings are compared in Table 2.4, where the computation time of our method is obtained by using parameter settings from the previous "Accuracy" subsection. The '-' symbol in Table 2.4 means that GPU programming is not used. Table 2.4 shows that our method is the only one that performs on

TABLE 2.4: Comparison of computation time per frame [ms] vs. hardware setting vs. image resolution

	Time	CPU	GPU	Resolution
Our method	12	i7-4790K @4.0GHz	-	640×480
RGB+D+KF [57]	32	i7-2600 @3.4GHz	-	320×240
ICP+RGB-D [135]	18	i7-3960X @3.3GHz	NVIDIA GeForce 680GTX	640×480
Inverse depth [38]	47	i5-2500 @3.3GHz	NVIDIA GeForce 660GTX	640×480
Sparse depth [98]	67	i7-2860QM @2.5GHz	-	VoxelGridFilter voxel size: 1cm
Fast 3-D [52]	28	i7-3820 @3.6GHz	-	320×240

the original image resolution (640 $\times$ 480) without GPU programming which can keep up the camera frame rate.

Furthermore, different number of iterations and amount of salient points are tested and the influence for accuracy and computation performance are compared. We use three fixed levels of offset l in the correspondence searching stage. For each level, the number of iterations varies. And for each iteration, the number of salient point used for correspondence estimation is also varied.

Table 2.5 shows that our method performs well with a small number of iteration and a small number of salient points. Our method achieved reasonable result even using only 10 salient points per ICP iteration, achieving 100 Hz, which is much higher than the camera frame rate, saving a lot of computational resource for additional tasks such as scene reconstruction. As the number of iteration grows, the drift error does not change much. As the number of salient point per iteration grows, the drift error decreases. This implies that the number of salient point is more important than the number of iteration for visual odometry.

Qualitative result

To illustrate the performance of our method qualitatively, Figure 2.8 shows four reconstructed scenes from TUM dataset. The quality of scene reconstruction from video sequence is sensitive to the accuracy of visual odometry, the error of frame-to-frame registration result can be accumulated to a large drift. Therefore the usual remedy to correct large drift error is to use loop closure detection and global pose graph optimization [57]. However, to show the accuracy of the proposed visual odometry method, no loop closure detection

TABLE 2.5: Different parameters vs. precision vs. computation time

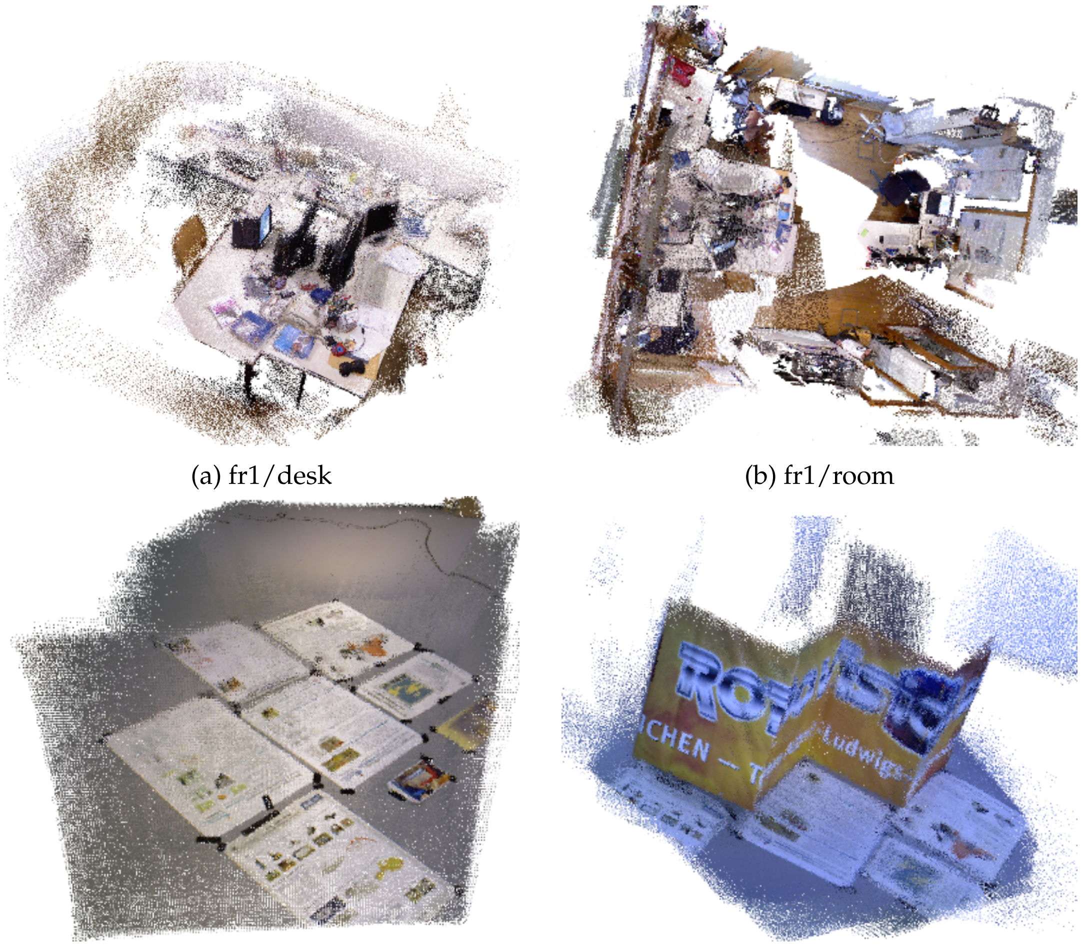
Iterations per level	#Salient points per iteration	RMSE of translational drift [m/s]				Time [ms]	
		fr1/ desk	fr1/ desk2	fr1/ room	fr2/ desk	mean	max
10	100	0.0217	0.0381	0.0416	0.0204	12.8	17.9
2	100	0.0255	0.0450	0.0425	0.0240	9.3	13.6
5	100	0.0219	0.0369	0.0442	0.0225	10.9	15.0
20	100	0.0218	0.0389	0.0440	0.0184	17.8	22.9
50	100	0.0219	0.0373	0.0455	0.0188	30.4	34.0
100	100	0.0226	0.0389	0.0554	0.0182	58.7	67.6
10	10	0.0274	0.0400	0.0781	0.0312	9.5	12.9
10	20	0.0260	0.0382	0.0467	0.0236	9.9	13.7
10	50	0.0226	0.0384	0.0466	0.0200	11.8	13.6
10	200	0.0219	0.0379	0.0430	0.0195	16.7	22.4
10	500	0.0217	0.0367	0.0415	0.0196	27.3	31.9

and no global pose graph optimization are performed for generating the results of Figure 2.8. The colored point cloud of each frame is simply added into a global point cloud based on the estimated visual odometry result. The estimated visual odometry result is still accurate enough to reconstruct the scene without large drift.

2.1.6 Summary and Conclusion

In this section, a fast and robust visual odometry estimation method based on intensity assisted ICP (IAICP) is presented. By contributing in the selection, matching and weighting stages, IAICP improves the conventional ICP significantly. Intelligent salient point selection is performed on the source frame, thus drastically reduced the computation time. Correspondences are established by searching nearby points in the image coordinate. With weighting function devised from statistics, robustness against outlying correspondences is ensured. The proposed method was evaluated on the TUM Dataset both quantitatively and qualitatively. In terms of translational drift, it outperforms state-of-the-art methods in 11 out of the 14 tested video sequences. Our method runs with an average frame rate of 78 Hz using a single CPU thread. Experimental results showed that our proposed approach achieved overall better accuracy than approaches with GPU parallelization. With changes of parameter settings, our method can even achieve 107 Hz by loosing ca. 12% precision of drift error. With the achieved high frame rate, substantial computation resources can be saved for other online tasks.

Although the proposed visual odometry method can provide reconstruction result with good quality (Figure 2.8), it still suffers from the certain drawbacks. Firstly, the frame-to-frame motion estimation still suffers from estimation error, where this error can be accumulated as the camera moves



(a) fr1/desk (b) fr1/room

(c) fr3/nostructure_texture_far (d) fr3/structure_texture_far

FIGURE 2.8: Reconstructed scene from TUM benchmark sequences based on estimated visual odometry.

around the object. The accumulated error is called the drift problem, which is not correctable with a pure visual odometry method. Therefore, if the camera traverses with a long trajectory, the accumulated drift could be extremely large to deteriorate the reconstruction result. Secondly, the success of IAICP method is based on the assumption that the entire environment is static, where the only movable object is the camera itself, thus the whole scene moves rigidly with respect to the camera. However, in practice, many dynamic objects exist in the world, such as human, robot manipulator. These dynamic objects violate the static assumption, causing inconsistent rigid motion of the scene, therefore causing failure of camera ego-motion estimation.

2.2 RGBD-SLAM in dynamic environment

In the previous section, the pure visual odometry method, IAICP (Intensity Assisted Iterative Closest Point) was introduced, which is designed for static environment. As discussed in the last section, it suffers from the "drift" problem and influence of dynamic objects. In this section, in order to handle these limitations, extension to IAICP is proposed to handle dynamic environment. Furthermore, the visual odometry method is integrated to a SLAM framework to correct the potential "drift" problem.

2.2.1 Motivation

To simplify the problem formulation, most state-of-the-art visual odometry methods only consider a static environment. However, dynamic objects, such as human, exist in many real life environments. While small portion of dynamic objects can be filtered by viewing them as noise, large proportion of dynamic objects will violate the static environment assumption strongly, thus many existing visual odometry methods are limited for the usage of real applications. This section proposes a real-time depth edge based RGB-D SLAM system for dynamic environment. The proposed method is based on frame-to-keyframe registration, where only depth edge points are used. To reduce the influence of dynamic objects, a static weighting method is proposed for edge points in the keyframes. Static weight indicates the likelihood of one point being part of the static environment. This static weight is added into the Intensity Assisted Iterative Closest Point (IAICP) method (Chapter 2.1) to perform the registration task. Furthermore, the extended visual odometry method is integrated into a SLAM (Simultaneous Localization and Mapping) system, where an efficient loop closure detection strategy is used. Both the visual odometry method for dynamic environment and the integrated SLAM system are evaluated with challenging dynamic sequences from the TUM RGB-D dataset [115]. Compared to state-of-the-art methods [57, 61, 116], the proposed method reduces the estimation error significantly.

The key features of the proposed method are:

- A novel efficient static weighting method is proposed to reduce the influence of dynamic objects on pose estimation. It calculates the likelihood of each keyframe point being part of the static environment.
- The static weighting terms are integrated into the IAICP method. This leads to a real-time RGB-D visual odometry method for dynamic environment.

2.2.2 Related Works

Current RGB-D visual odometry methods can be roughly categorized into two groups. To handle dynamic objects, different strategies are used for these two groups.

Dense visual odometry

The first group is dense visual odometry [58, 57, 124, 83]. These methods formulate the task as an energy minimization problem. The energy function is usually the sum over pixel-wise intensity/depth difference between the target image and warped source image. The camera's 6 DOF motion will be then iteratively optimized using the defined energy function. Unfortunately, this form of energy function strongly depends on the static environment assumption, where the whole scene moves rigidly with respect to the camera. In a dynamic environment, even with the correct motion, many pixels from warped target frame do not align with the source frame due to the inconsistent movement of the dynamic objects, resulting in large intensity/depth difference, thus the energy function does not have a minimum at the correct motion. To compensate potential dynamic objects, they need to be detected and their influence need to be removed from the optimization process. Wang et al. calculate dense optical flow from RGB images, and dynamic objects can be then found by clustering the image based on point trajectories [131]. The pixels of dynamic objects are then excluded for energy function minimization. Their method improves the robustness against dynamic object effectively, but the optical flow estimation and clustering require heavy computation, thus their method cannot be performed in real-time. Sun et al. [116] use intensity difference image to identify the boundaries of dynamic objects. Then dense dynamic points are segmented using the quantized depth image. Their method achieves stable performance for highly dynamic scenes, however, the segmentation process takes half second for each frame, which severely limits the real-time applicability. Kim et al. [61] propose to use depth difference between the current frame and multiple warped previous frames to calculate a static background model, where pixels with large depth difference will be considered as dynamic pixels. However, the effectiveness of their method is limited to forward-backward motion with respect to the camera. If the dynamic object moves parallel to the image plane, only the boundary of dynamic object can be found effectively using depth difference due to the aperture problem. Therefore the influence of dynamic object cannot be totally removed with their method.

Correspondence based method

The second group is correspondence based method [88, 43, 44, 23]. In these methods, correspondences are firstly matched between the source and target frames. Then the camera's ego-motion is estimated using closed-form solution from the correspondences. The correspondence can be found by matching feature descriptors of sparsely detected keypoints (such as SIFT, SURF) [44], or for Iterative Closest Point [8] based methods [88, 50], correspondences are densely established using a certain distance metric. Since the majority of points from the static environment follow the same rigid motion (inverse motion of the camera ego-motion), RANSAC regression is usually used to filter out dynamic objects [60, 71]. However, if dynamic feature points are the majority in a frame, RANSAC has a low success rate, which results in wrong estimate of the set of static points.

To compensate dynamic objects, all above mentioned methods require a correspondence matching step, where either dense or sparse correspondences are needed. While accurate dense correspondences matching is time consuming [131], fast approximation [61] suffers from the aperture problem. Accurate matching of 2D keypoints can be performed in real-time [60, 71]. However, sparse 2D keypoints can be distributed unevenly in the environment. If a dynamic object has many texture, then the dynamic keypoints will outnumber static keypoints, which may result in failure in RANSAC regression. Therefore additional IMU sensor data are often used to compensate this issue [60, 71].

2.2.3 IAICP for dynamic environment

Considering the trade-off between dense methods and sparse keypoints based methods, semi-dense depth edge is chosen to find correspondences. Depth edge contains the structure information of the environment. It was shown that accurate visual odometry can be estimated based on depth edge [9, 17]. Moreover, depth edge points are sparsely present, therefore they can be efficiently matched and the amount of depth edge points is more balanced than 2D keypoints. In the proposed method, edge points between frames will be matched by using both geometric and intensity distances [74] (Chapter 2.1). Upon matched edge points, a novel static weighting method is proposed to downweight dynamic points for the visual odometry method.

Notation

Given a 3D point $\mathbf{p} = (x, y, z, 1)^T$ in homogeneous coordinate relative to the camera, the image pixel coordinate $\mathbf{x} = (u, v)^T$ $(u \in [0, height - 1], v \in [0, width - 1])$ of $\mathbf{p}$ is calculated with the camera projection function π: $\mathbf{x} = \pi(\mathbf{p}) = (\frac{xf_x}{z} + o_x, \frac{yf_y}{z} + o_y)^T$, where *height* and *width* are the pixel number in image's x- and y- direction, f_x, f_y are the camera focal lengths and o_x, o_y are the camera center coordinates.

Due to the ego-motion of the camera, a 3D point $\mathbf{p}$ in the keyframe (timestep k) coordinate is rigidly transformed in the current frame (timestep t) with the transformation matrix $\mathbf{T}_k^t \in SE(3)$. The point's new coordinate in the current camera coordinate frame is then:

$$\mathbf{p}' = \mathbf{T}_k^t \mathbf{p} = \begin{bmatrix} \mathbf{R}_k^t & \mathbf{t}_k^t \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{p} \tag{2.23}$$

The pixel $\mathbf{x}$'s corresponding 3D point $\mathbf{p}$ is indicated as $P_t(ind(\mathbf{x}))$, where $ind()$ is the mapping from the image coordinate to the point index in the organized point cloud's one-dimensional list: $ind(\mathbf{x}) = ind((u, v)^T) = v \times width + u$. To find out the image coordinate $\mathbf{x}$ of a point index i, the inverse mapping is: $\mathbf{x} = ind^{-1}(i) = (i - \lfloor \frac{i}{width} \rfloor width, \lfloor \frac{i}{width} \rfloor)^T$, where $\lfloor \cdot \rfloor$ denotes the floor operator.

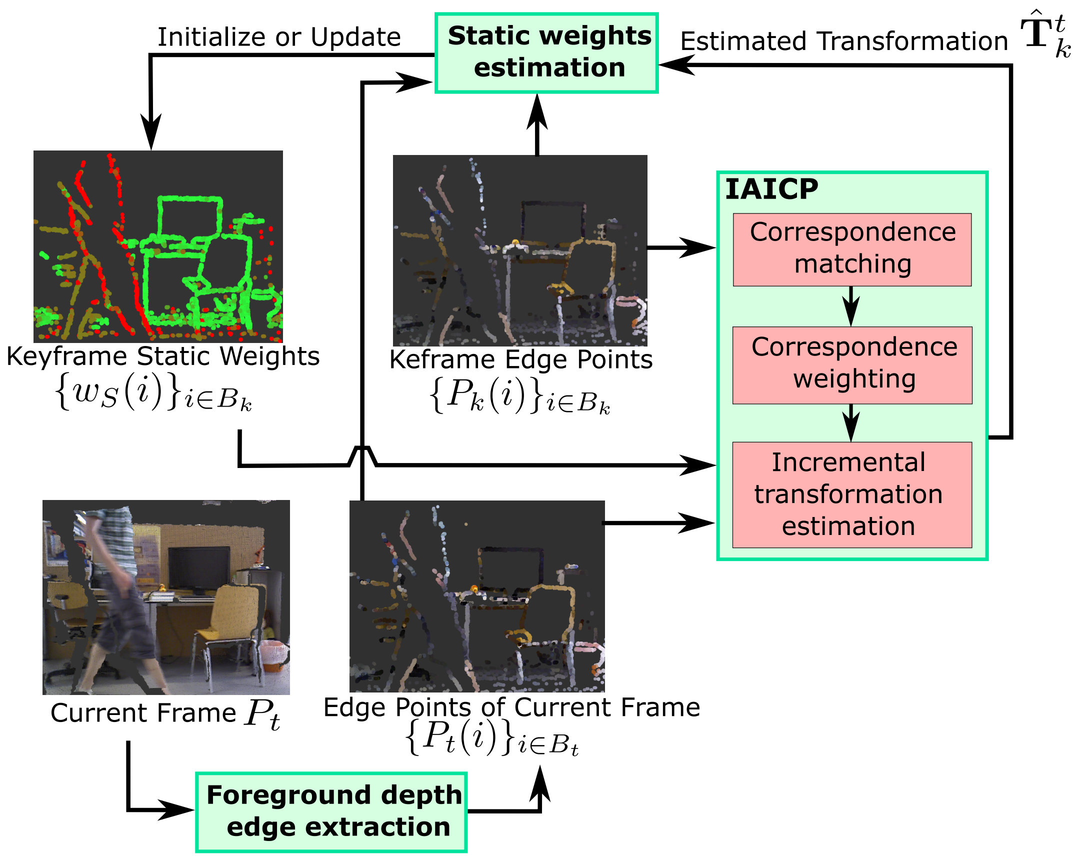

Initialize or Update
Static weights estimation
Estimated Transformation $\hat{\mathbf{T}}_k^t$
Keyframe Static Weights $\{w_S(i)\}_{i \in B_k}$
Keframe Edge Points $\{P_k(i)\}_{i \in B_k}$
IAICP
Correspondence matching
Correspondence weighting
Incremental transformation estimation
Current Frame P_t
Edge Points of Current Frame $\{P_t(i)\}_{i \in B_t}$
Foreground depth edge extraction

FIGURE 2.9: Overview of the proposed visual odometry system for dynamic environment.

Overview

The overview of the proposed visual odometry method is illustrated in Fig 2.9. For each incoming frame (current frame), foreground edge points are first extracted, where only extracted edge points are used for odometry estimation. Every Nth frame is chosen to be a keyframe. Alternatively one can consider to select keyframe based on camera motion. In highly dynamic environment, however, the visual content changes drastically even when the camera does not move. In this case, there might not be enough common visible points to estimate the relative pose, thus it can fail to estimate camera motion. For each keyframe, static weights for the edge points are estimated and updated. The static weight indicates how likely a point belongs to the static environment. Then the relative coordinate transformation from the keyframe to the current frame is estimated using IAICP algorithm (Chapter 2.1), where static weights are integrated to IAICP's correspondence weighting process, in order to reduce the effect of dynamic moving objects to the transformation estimation. The static weights for the keyframe will keep updating based on the estimated motion, until the next keyframe is generated.

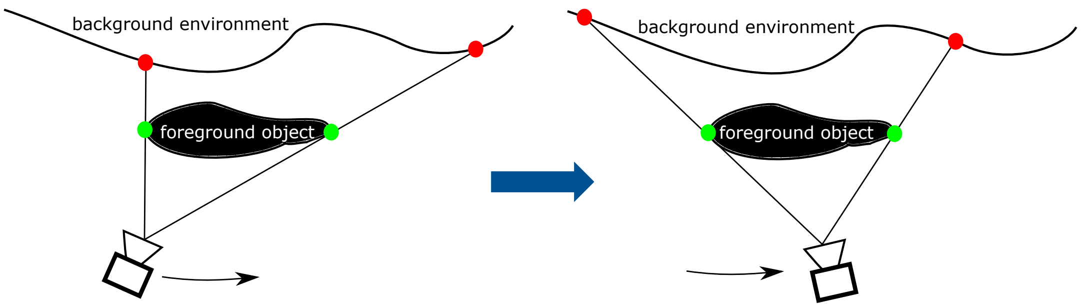


FIGURE 2.10: Two types of depth edge points exist. The first type is foreground depth edge (green points), these edges present the boundary of foreground object. By changing the camera viewpoint, the location of the foreground depth edge remains stable. Another type is occluded depth edge (red points). This type of edge is caused by occlusion of other objects. The position of occluded depth edge is very sensitive to camera viewpoint.

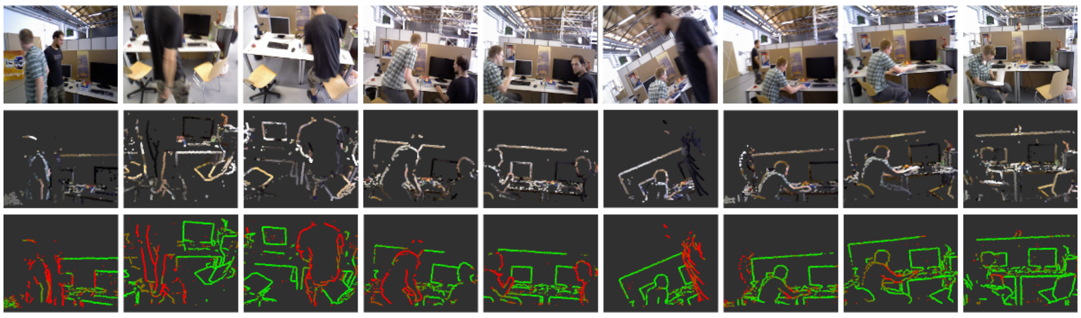


FIGURE 2.11: Foreground depth edge extraction and static weighting examples taken from "fr3/walking" sequences. First row: the original RGB image. Second row: foreground depth edge. Third row: static weighting result, where green indicates static and red indicates dynamic.

Foreground depth edge extraction

Depth edge points are points that have large depth discontinuity in their neighbourhood. Two types of depth edge points exist: foreground edge points and occluded edge points (Figure 2.10). Foreground edge points present boundaries of objects, which are in front of other objects. Occluded edge points are part of background objects, which do not represent the actual geometry boundaries, they exist due to the occlusion of foreground objects. The foreground edge points are stable to a moving camera, because they capture the actual surface boundaries of the objects. The occluded edge points are sensitive to a moving camera, therefore they need to be excluded for estimating camera trajectory.

Foreground depth edge points play an important role for iterative closest point method. As evidenced in [74, 9], using foreground depth edge points can improve the accuracy of point cloud registration task. Because by using depth edge points, the probability of finding correct correspondence is higher than using uniformly sampled points. Moreover, correct correspondences will be also crucial for the static weighting process, which will be introduced later.

Given the point cloud P_t, a set B_t consisting of foreground edge point indices is constructed. Firstly the depth differences $\{h_i\}_{i=1}^4$ between each point and its four neighbours are computed:

$$h_i = e_Z^T P_t(ind(\mathbf{x})) - e_Z^T P_t(ind(\mathbf{x} + \mathbf{o}_i)), \tag{2.24}$$

where $e_Z = (0,0,1,0)^T$ is used to extract depth value of one point and the four offset vectors $< \mathbf{o}_1, \mathbf{o}_2, \mathbf{o}_3, \mathbf{o}_4 >$ are $< (0,b)^T, (0,-b)^T, (b,0)^T, (-b,0)^T >$. An offset $b > 1$ is used here, because a lot of depth difference between direct neighours ($b = 1$) cannot be computed due to a lot of NaN (Not a Number) value pixels that exist near the depth discontinuous area. On the other hand, a larger b value causes more points to be detected as depth edge, resulting in thicker edges. Balancing between the NaN value avoidance effect and the thickness of depth edge, $b = 4$ is empirically set. A point is then considered as foreground edge points and added into the edge point set $B_t = \{B_t, ind^{-1}(\mathbf{x})\}$, if it fulfills the following conditions:

$$\begin{aligned} \max(h_1, h_2, h_3, h_4) &< e_Z^T P_t(ind(\mathbf{x}))\tau_b, \\ \max(abs(h_1 - h_2), abs(h_3 - h_4)) &> e_Z^T P_t(ind(\mathbf{x}))\tau_f. \end{aligned} \tag{2.25}$$

The first condition rejects occluded edge points, because occluded points have a much larger depth than their neighbours, where $e_Z^T P_t(ind(\mathbf{x}))\tau_b$ is the depth depended threshold. With a very large τ_b, no occluded point can be rejected. If τ_b is too small, a lot of actual foreground edge points can be also rejected due to slightly larger depth than neigbouring pixels. The second condition checks whether a point can be considered as edge point by checking the depth discontinuity with a threshold $e_Z^T P_t(ind(\mathbf{x}))\tau_f$. If τ_f is too large, then no edge points can be detected and if τ_f is too small, almost every point is detected as edge. In the experiments, τ_b and τ_f are set as $0.015m$ and $0.04m$ respectively. In the second row of Figure 2.11, some examples of foreground depth edge extraction are illustrated.

Static weight estimation

Two types of points exist in the environment: points from static object and points from dynamic moving object. Due to the ego-motion of the camera, observed points are constantly moving in the camera's coordinate. Comparing point clouds from two frames, static points are moved with same rigid transformation, which is the inverse of camera's ego-motion, while dynamic points do not follow the same rigid transformation due to their own movements.

The static weights for a source point cloud P_{src} are estimated by comparing it to a target point cloud P_{tgt}. The static weight is only estimated for the foreground depth edge points $\{P_{src}(i)\}_{i \in B_{src}}$, where B_{src} is the set of edge point indices. The static weight of $P_{src}(i)$ is denoted as $w_i^{src,tgt}$, and it is estimated based on the Euclidean distance between $P_{src}(i)$ and the corresponding point $P_{tgt}(c(i))$ in the target cloud:

$$d_i = \left\| \hat{\mathbf{T}}_{src}^{tgt} P_{src}(i) - P_{tgt}(c(i)) \right\|,$$

where $c(i) \in B_{tgt}$ is the found correspondence point index in the target cloud and $\hat{\mathbf{T}}_{src}^{tgt}$ is the estimated transformation that aligns the source cloud to the target cloud. In case that $c(i)$ cannot be found in the close vicinity of $P_{src}(i)$, d_i is set to a large constant value D.

A static point $P_{src}(i)$ after transformation becomes $\hat{\mathbf{T}}_{src}^{tgt} P_{src}(i)$. Assuming that $c(i)$ and $\hat{\mathbf{T}}_{src}^{tgt}$ are both correct, $\hat{\mathbf{T}}_{src}^{tgt} P_{src}(i)$ should be aligned perfectly on its corresponding point $P_{tgt}(c(i))$. Therefore for a static point, d_i should be zero, or a small value due to sensor noises. Taking advantage of this characteristic, the static points can be distinguished from the dynamic points based on the statistic over $\{d_i\}_{i \in B_{src}}$. Following [57], the static weight $w_i^{src,tgt}$ is estimated based on the Student's t-distribution

$$w_i^{src,tgt} = \frac{\nu_0 + 1}{\nu_0 + ((d_i - \mu_D)/\sigma_D)^2}, \tag{2.26}$$

$$\sigma_D = 1.4826 \, \text{Median}\{|d_i - \mu_D|\}_{d_i \neq D} \tag{2.27}$$

ν_0 is the degree of freedom of t-distribution, where a larger ν_0 results in steeper decrease of $w_i^{src,tgt}$ with increasing d_i. Notice that points without a valid correspondence in the close neighbourhood ($P_{src}(i)_{d_i = D}$) are not used for computing σ_D. In the experiments, ν_0 is empirically set to 10. The mean value μ_D is manually set to zero, because smaller distance indicates a more static point. The variance σ_D is estimated using the median absolute deviation.

Figure 2.12 shows an example of the histogram of correspondence distance and t-distribution. The chosen distribution fits the actual experimental data nicely. A zero valued d_i indicates highest static likelihood, a small d_i is caused by sensor noises or discrete sampling of the environment, and a large d_i is caused by dynamic movement that differs from the camera motion. The procedure to estimate the static weights $\{w_i^{src,tgt}\}_{i \in B_{src}}$ is summarized in Algorithm 2.2.

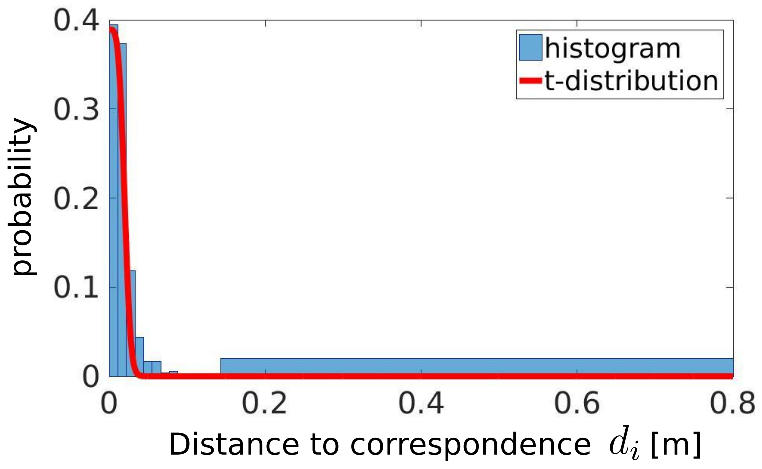


FIGURE 2.12: Histogram of correspondence distance and pdf of t-distribution. The pdf of t-distribution fits the actual data nicely.

In the proposed visual odometry method, the static weights are only estimated for keyframes, where keyframes are always set as the source frame. Assuming that the latest keyframe is P_k with k as the index of the keyframe, the static weight of keyframe points are calculated by comparing the keyframe to two other target frames: one is the previous keyframe P_{k-N}, another one is the latest frame at time step t: P_t. The static weight for the point $P_k(i)$ is defined as $w_S(i)$:

$$w_S(i) = \alpha w_i^{k,k-N} + (1-\alpha) w_i^{k,t}, \tag{2.28}$$

where $w_i^{k,k-N}$ is computed by setting last keyframe P_{k-N} as the target cloud in Algorithm 2.2, and $w_i^{k,t}$ is computed by setting the current frame P_t as target cloud. In the experiments, α is empirically set as:

$$\alpha = \begin{cases} 1 & \textit{if} \quad t = k \\ 0.5N/(N+t-k) & \textit{otherwise} \end{cases} \tag{2.29}$$

As shown in Figure 2.13, if the keyframe is the current frame, the static weights are initialized with $w_i^{k,k-N}$. With passage of time, more influence from current frame is considered where the initialization term $w_i^{k,k-N}$ and the update term $w_i^{k,t}$ are complementary to each other .

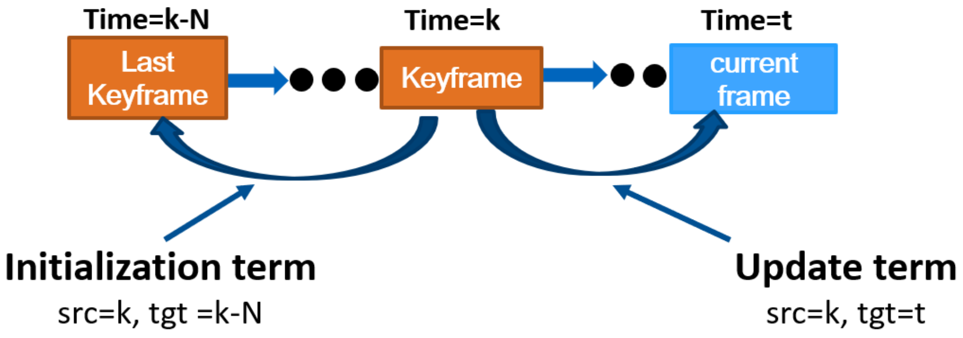


FIGURE 2.13: Initialization and update setting of static weighting for keyframe k.

The update term $w_i^{k,t}$ is used for two reasons. i) Previously static object can start moving after the keyframe has been defined, where previously static points can convert to dynamic points. If only the initialization term is used, the new dynamic points can only be detected by the time of the new keyframe $k+N$. This would result in drift problem in the time interval $[k+1, k+N-1]$. ii) Due to occlusion of foreground objects, the visible part of the environment always changes. Newly occluded part from the keyframe cannot find correct correspondences in the new frame P_t, therefore newly occluded part should be avoided in the transformation estimation process. Occluded points usually have large distance to their falsely found correspondences, thus by using $w_i^{k,t}$, they can be efficiently downweighted.

The initialization term is important as well as the update term. It is estimated by comparing the keyframe with last keyframe. If only the update term is used, dynamic objects with small velocity cannot be distinguished effectively. For instance, for the time step $k+1$, dynamic points with small

Algorithm 2.2 Static weighting for depth edge points

```
Require: - a source cloud $P_{src}$ and edge point set $B_{src}$
    - a target cloud $P_{tgt}$
    - corresponding point index for source cloud edge points $\{c(i)\}_{i \in B_{src}}$
    - current transformation estimate $\hat{\mathbf{T}}_{src}^{tgt}$
Ensure: - static weights: $w_i^{src,tgt}$ for $i \in B_{src}$
    for $i \in B_{src}$ do
        Calculate distance $d_i$ between warped source point $\hat{\mathbf{T}}_{src}^{tgt} P_{src}(i)$ and its
        corresponding point $P_{tgt}(c(i))$
    end for
    Calculate variance $\sigma_D$ of $\{d_i\}_{i \in B_{src}}$ (eq. (2.27))
    for $i \in B_{src}$ do
        Estimate static weight $w_i^{src,tgt}$ (eq. (2.26))
    end for
```

velocity is not moved too far within one frame (usually 30 [ms]), the distance between correspondences might land in the "small noise" range of static points. On the contrary, the initialization term is calculated with a relative larger time difference N, thus the d_i for small velocity objects is larger and more distinguishable from static points.

In the third row of Figure 2.11, some examples of estimated static weights are illustrated. It shows that our method can effectively downweight dynamic points for different cases, including one person moving, two persons moving and part of one person moving.

Motion estimation using IAICP

The motion estimation is performed using Intensity Assisted Iterative Closest Point (IAICP) method (Chapter 2.1). IAICP utilizes intensity information of each point in the correspondence matching and weighting stages. For dynamic environment, the static weight is combined into original IAICP to reduce the influence of dynamic object on the transformation estimation.

Given a source frame $< P_{src}, I_{src} >$ and a target frame $< P_{tgt}, I_{tgt} >$, IAICP estimates the relative transformation that aligns the source cloud P_{src} to the target cloud P_{tgt}. IAICP is an iterative method, where the transformation matrix $\mathbf{T}^*$ is usually initialized with identity matrix or with a motion prediction. In the experiments, $\mathbf{T}^*$ is initialized with first order motion prediction. Then the optimal transformation value $\mathbf{T}^*$ is searched iteratively, where the kth iteration can be summarized as:

- Search for each depth edge point $P_{src}(i)_{i \in B_{src}}$ of the source cloud a point in the target cloud as correspondence. For $P_{src}(i)$, the index of the corresponding point index in the target point cloud P_{tgt} is denoted as $c(i) \in B_{tgt}$, where

$$c(i) = \underset{j}{\operatorname{argmin}} \|\mathbf{T}^* P_{src}(i) - P_{tgt}(j)\|. \tag{2.30}$$

- Compute the optimal incremental transformation $\mathbf{T}_k$ that minimizes the sum of weighted Euclidean distance between the established correspondences:

$$\mathbf{T}_k = \underset{\mathbf{T}}{\operatorname{argmin}} \sum_i W(i) \|\mathbf{T}\mathbf{T}^* P_{src}(i) - P_{tgt}(c(i))\|, \tag{2.31}$$

  where $W(i)$ is a weighting term that indicates the quality of correspondence. The equation is usually solved with a closed-form solution [22] such as Singular Value Decomposition [4]. For efficient computation, not all edge points are used to compute eq. (2.31), instead we randomly select 120 depth edge points for each ICP iteration.

- Update $\mathbf{T}^*$ as: $\mathbf{T}^* \leftarrow \mathbf{T}_k\mathbf{T}^*$.

In the following, the computation of weighting term $W(i)$ and correspondence matching term $c(i)$ will be explained.
Correspondence weighting: In practice, not every established correspondence is determined correctly. The outliers badly influences the transformation estimation. To compensate outliers, a weighting term $W(i)$ for corresponding pair $< P_{src}(i), P_{tgt}(c(i)) >$, is estimated. The weighting is based on an intensity term $w_I(i)$ (Equation 2.9), a geometric term $w_G(i)$ (Equation 2.13) and also the static weighting term $w_S(i)$ (Equation 2.28):

$$W(i) = w_I(i) w_G(i) w_S(i). \tag{2.32}$$

Using the proposed weighting terms, outlying correspondences with large intensity difference or having large geometric distance are intuitively downweighted. Furthermore, the static weight is responsible to downweight the influence of dynamic object.
Correspondence matching: Taking advantage of the organized point cloud, the search of matching point is performed in the image coordinate. By warping the source frame point $P_{src}(i)$ with the current estimate of $\mathbf{T}^*$, the image coordinate of the warped point is $\mathbf{x}' = \pi(\mathbf{T}^* P_{src}(i))$. Then target depth edge points $P_{src}(j)_{j\in B_t}$ in the neighbourhood of $\mathbf{x}'$ are considered as candidate of correspondence for $P_{src}(i)$, where neighbourhood $N(\mathbf{x}')$ is defined as a square around image coordinate $\mathbf{x}'$.

Having a query pair $< P_{src}(i), P_{tgt}(j) >$ to check, a score function for this pair is given as:

$$\begin{aligned} s(i,j) &= w_I(I_{src}(ind^{-1}(i)) - I_{tgt}(ind^{-1}(j))) \\ &\quad w_G^{(\mu=0)}(\|\mathbf{T}^* P_{src}(i) - P_{tgt}(j)\|). \end{aligned} \tag{2.33}$$

$w_I(\cdot)$ (Equation (2.9)) and $w_G^{(\mu=0)}(\cdot)$ (Equation (2.17)) are weighting functions derived from last ICP's iteration, where $w_G^{(\mu=0)}$ sets the mean value to zero, because more closer point is more probable to be true corresponding point.

Then the correspondence is taken as the point that maximizes the score function:

$$c(i) = \underset{j \in (B_{tgt} \cap N(\mathbf{x}'))}{\operatorname{argmax}} s(i,j). \tag{2.34}$$

2.2.4 Integration to SLAM

A pure visual odometry system suffers from the drift problem, because current absolute pose is obtained by accumulating previous ego-motion estimates, which also accumulates the estimation errors. To compensate the drift problem, the visual odometry method is integrated into a pose graph based SLAM system [35]. In the pose graph, consecutive keyframes are connected with a pose constraint, that is from the visual odometry method. In addition to that, if a keyframe detects previously seen environment, new constraints are added to previous keyframes, thus the accumulated drift can be corrected using graph optimization considering all constraints. (refer to [35] for details about pose graph optimization.) In the following, the procedure of loop closure detection will be presented.

As a new keyframe P_k is set, loop closures of P_k with 10 randomly selected previous keyframe P_r are checked. A loop closure is detected between P_k and P_r, when three conditions are fulfilled.

- **Geometric proximity**: The estimated camera positions of the two keyframes should not be too far away from each other:

$$\|\mathrm{transl}(\hat{\mathbf{T}}_k^r)\| < \tau_{distance} \tag{2.35}$$

  where transl$(\mathbf{T})$ extracts the translation vector from the transformation matrix $\mathbf{T}$, and the threshold $\tau_{distance}$ is set to 1.5m in the experiments. This is because distanced keyframes have lower probability of viewing same part of the environment.

- **Common visible part**: The two keyframes should have common visible environment in view. A point from P_k, $P_k(i)$ is also possibly visible in P_r, if the warped point is still inside the image border:

$$\pi(\hat{\mathbf{T}}_k^r P_k(i)) \in \{[0, width - 1] \times [0, height - 1]\}. \tag{2.36}$$

  For efficient checking, 100 edge points are randomly selected from P_k. If less than 30% of points are visible, no loop closure between P_k and P_r is defined.

- **Forward-backward consistency check**: If the previous two conditions are satisfied, the pair-wise registration is then performed using above described visual odometry method. The registration is performed twice by setting P_r as source frame in one time and as target frame in the other time. Two registration results $\hat{\mathbf{T}}_k^r$ and $\hat{\mathbf{T}}_r^k$ are compared for forward-backward consistency. The consistency check can be passed if:

$$\begin{aligned} \|\text{transl}(\hat{\mathbf{T}}_k^r \hat{\mathbf{T}}_r^k))\| &< \tau_{distanceDiff}, \\ \|\text{rotation}(\hat{\mathbf{T}}_k^r \hat{\mathbf{T}}_r^k))\| &< \tau_{angleDiff}, \end{aligned} \tag{2.37}$$

where thresholds are set as $\tau_{distanceDiff} = 0.02m$ and $\tau_{angleDiff} = 3°$.

If the keyframe pair $< P_k, P_r >$ fulfills all three conditions, a new relative pose constraint $\hat{\mathbf{T}}_k^r$ between them is added into the pose graph, which means detection of a new loop closure.

2.2.5 Experimental results

The experiments are conducted with TUM RGB-D dataset [115]. Many previous papers [57, 58, 74, 38, 135] evaluated their methods on this dataset and achieved good results, however the sequences containing dynamic objects were not often used for evaluation. In these sequences, people move in the environment, while the camera also moves with different patterns (static, xyz, rpy and halfsphere). These sequences are challenging due to the large proportion of dynamic parts in the observation, where in extreme case more than half of the image is occupied with dynamic object. Figure 2.11 shows some example frames taken from the "walking" sequences. To handle the high dynamic environment, previous methods require non real-time method to segment dynamic part [116, 131] or suffer from large drift [61].

For testing the effectiveness of proposed method, the "sitting", "walking" sequences from the TUM dataset are used. In "sitting" sequences, people move their body parts while sitting on chairs, therefore they are considered as low dynamic sequences. In "walking" sequences, people walk around a table, creating large proportion of dynamic parts in the observed images. Therefore "walking" sequences are considered as high dynamic sequences.

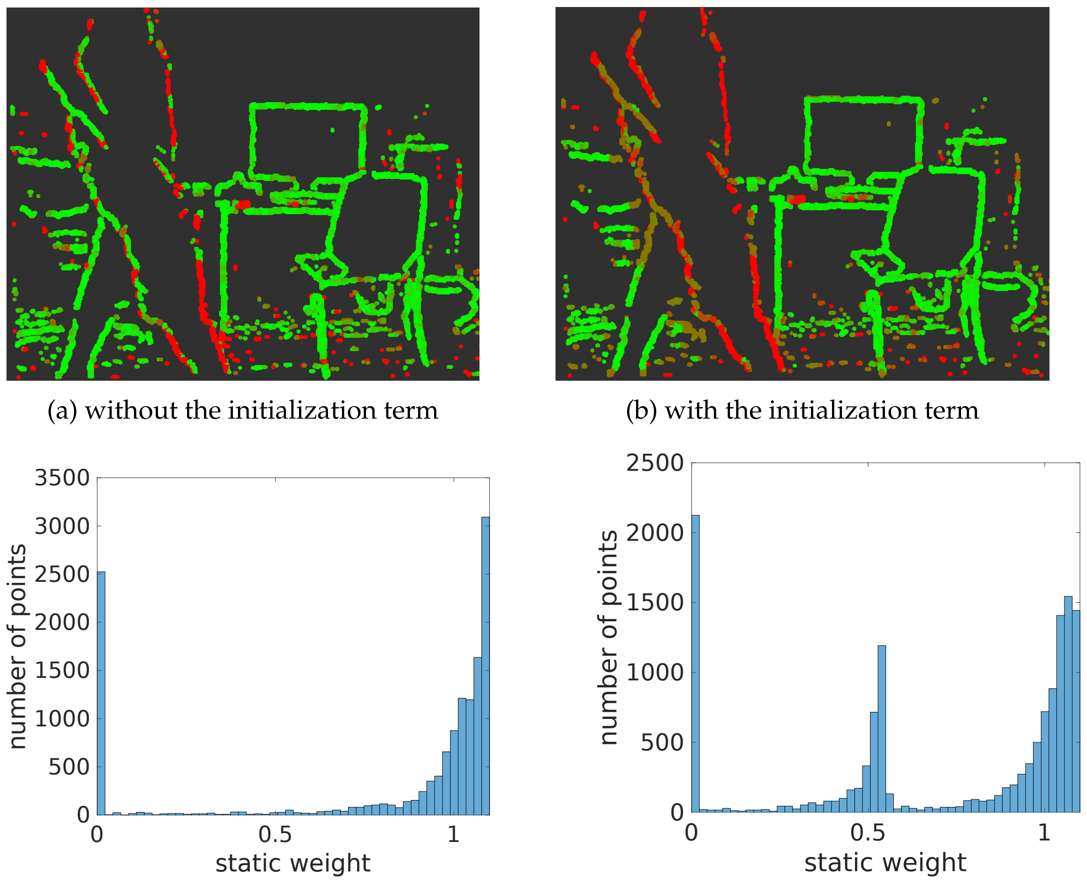
In the following, the visual odometry method and SLAM system are evaluated and compared with previous methods [58, 61, 116]. All the experiments are performed on a desktop computer with Intel Core i7-4790K CPU (4GHz) and 16 GB RAM. The visual odometry method only uses one CPU core, and for SLAM system, an additional CPU core is used for loop closure detection and graph optimization.

Evaluation of visual odometry method

For the evaluation of visual odometry, Relative Pose Error (RPE) metric is used. The following will firstly investigate the effectiveness of the proposed static weighting strategy and then compare the proposed method with previous methods.

Effect of static weighting: To verify the effectiveness of the proposed static weighting strategy, the visual odometry method is tested both with and without the static weight term $w_S(I)$ in the IAICP part (eq. (2.32)). The comparison is shown in Table 2.6, where "Depth edge + IAICP" means our method without static weighting. In "Depth edge + RANSAC + IAICP", a RANSAC based outlier rejection procedure is used, where the RANSAC procedure uses 100 iterations and has a outlier threshold of 1.5cm. The static weighting term improves the visual odometry result in most of the sequences and works better than the RANSAC based outlier rejection method. The average improvement in terms of translational drift for low-dynamic sequences is 8%, and for high-dynamic sequences, the average improvement is 52%. This verifies that our static weighting strategy effectively reduces the influence of dynamic objects, especially for high-dynamic environments.

Effect of static weight initialization: The static weight initialization with



(a) without the initialization term (b) with the initialization term

(c) without the initialization term (d) with the initialization term

FIGURE 2.14: Effectiveness of static points weighting: Static weight of 401th frame (keyframe) in "fr3/walking_xyz" sequence at time step $t = 402$. (a)(b) show the visualization of static weights and (c)(d) show the histogram of static weights.

previous keyframe is important as explained above. To verify the importance of the initialization, experiments are also performed by setting α in eq. (2.28) to zero. An example case is illustrated in Figure 2.14(a)(b), where a

TABLE 2.6: Visual odometry results: translational drift and rotational drift on TUM RGB-D dataset

sequences		RMSE of translational drift [m/s]					RMSE of rotational drift [°/s]				
		DVO	BaMVO	Depth edge + IAICP	Depth edge + RANSAC + IAICP	Our method	DVO	BaMVO	Depth edge +IAICP	Depth edge + RANSAC + IAICP	Our method
static	fr2/desk	0.0296	0.0299	0.0174	**0.0170**	0.0173	1.3920	1.1167	0.7325	**0.7145**	0.7266
	fr3/long-office	0.0231	0.0332	0.0200	0.0193	**0.0168**	1.5689	2.1583	0.9001	1.0683	**0.8012**
low dynamic	fr2/desk-person	0.0354	0.0352	0.0245	0.0189	**0.0173**	1.5368	1.2159	1.0389	0.8310	**0.8213**
	fr3/sitting-static	**0.0157**	0.0248	0.0198	0.0210	0.0231	**0.6084**	0.6977	0.5823	0.6220	0.7228
	fr3/sitting-xyz	0.0453	0.0482	0.0256	0.0254	**0.0219**	1.4980	1.3885	0.9152	0.9791	**0.8466**
	fr3/sitting-rpy	0.1735	0.1872	0.1058	0.1076	**0.0843**	6.0164	5.9834	**5.2157**	10.4392	5.6258
	fr3/sitting-halfsphere	0.1005	0.0589	0.0624	0.0583	**0.0389**	4.6490	2.8804	2.5247	2.7427	**1.8836**
high dynamic	fr3/walking-static	0.3818	0.1339	0.1192	0.0496	**0.0327**	6.3502	2.0833	2.9475	1.3791	**0.8085**
	fr3/walking-xyz	0.4360	0.2326	0.1802	0.1482	**0.0651**	7.6669	4.3911	3.4778	3.8904	**1.6442**
	fr3/walking-rpy	0.4038	0.3584	0.2855	0.3031	**0.2252**	7.0662	6.3398	**5.5704**	11.4640	5.6902
	fr3/walking-halfsphere	0.2628	0.1738	0.2016	0.0799	**0.0527**	5.2179	4.2863	4.5076	4.5912	**2.4048**

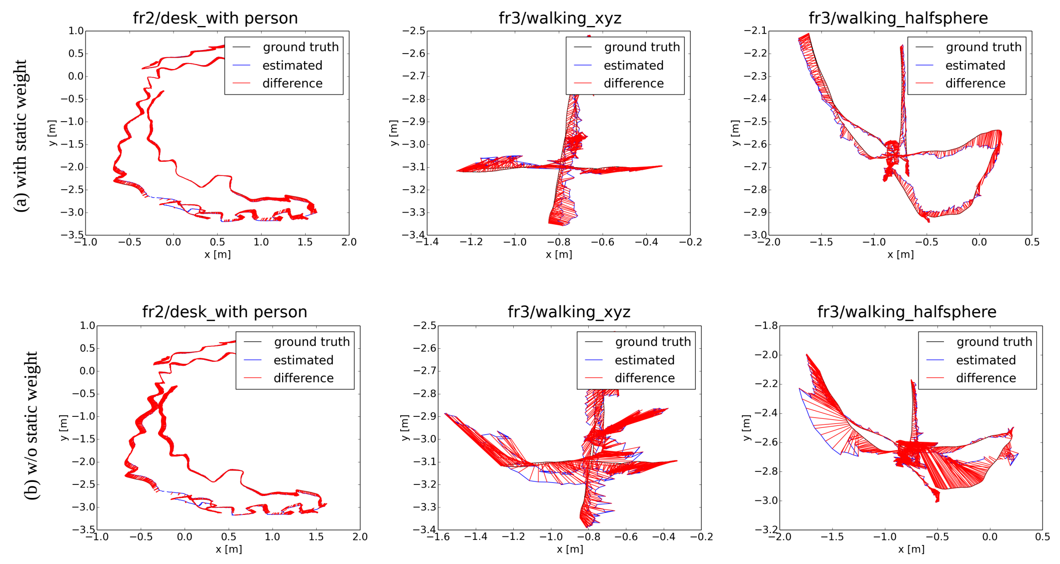


FIGURE 2.15: Examples of estimated trajectories from the SLAM system. (a) Estimated trajectories with the proposed static weighting term. (b) Estimated trajectories without the proposed static weighting term.

person is walking to the right. In this example, static weights are estimated for the keyframe P_k ($k = 401$). At time step $t = 402$, the person's movement is not that large between consecutive frames, because the velocity of the person is not that large. Therefore if the weight initialization is not applied, then some parts (especially the left part of the right leg) of the human body are considered as a static part (green). If the initial value is considered, which are obtained by comparing the keyframe with last keyframe ($t = 396$), then the human body is more distinguished as a dynamic object by leveraging a larger distance during the 5 frames. The effectiveness is also clearly visible in the histograms (Figure 2.14(c)(d)) of the static weights.

Comparison with previous methods: The proposed visual odoemtry method is compared with Dense Visual Odometry (DVO) [58] method and model-based dense-visual-odometry (BaMVO) [61] method. DVO is a state-of-the-art RGB-D visual odometry method for static environment, which can only handle small amount dynamic objects. BaMVO is specially designed to handle dynamic environment. The comparison results are shown in Table 2.6, our result outperforms in almost all dynamic sequences. Even for the static environment sequences, the proposed visual odometry method still outperformed DVO, which takes the static environment assumption. For the highly dynamic sequences, our method outperforms significantly. The proposed method improves the visual odometry performance by 74.6% compared to DVO, and by 58.2% compared to BaMVO.

The sources of improvements are twofold: firstly by using sparse foreground depth edge points, correct correspondences can be efficiently found, where a higher correct ratio results in a more accurate transformation estimation; secondly, using these correspondences, the proposed static weighting strategy effectively reduces the influence of dynamic objects. Compared

to the proposed method, DVO takes the static environment assumption in their problem formulation and cannot perform normally in highly dynamic sequences. In BaMVO, static weights are calculated based on depth difference, where points on the same image coordinate are simply approximated as correspondence. This approximation might cause the aperture problem for parallel dynamic motion to the image plane.
Computation time: The proposed visual odometry method using static points weighting is performed on VGA image resolution (640 $\times$ 480) and requires only one CPU thread. The average computation time per frame is 22ms. In comparison, DVO requires 32ms per frame (320 $\times$ 240 resolution, i7-2600 CPU with 3.40GHz) and BaMVO requires 42ms per frame (320 $\times$ 240 resolution, Intel i7 CPU with 3.3GHz). The computation time of the proposed method is less because no dense operation is needed as in DVO and BaMVO, both static weighting and transformation estimation are only performed on sparse depth edge points. The real-time performance makes the proposed method suitable for on-line applications.

Evaluation of SLAM system

Finally, the performance of the integrated SLAM system, which includes loop closure detection and map optimization. is evaluated. For evaluating SLAM system, Absolute Trajectory Error (ATE) [115] metric is used. The estimated trajectories are compared to ground truth, and some examples are shown in Figure 2.15. In the first row of Figure 2.15, the trajectories are estimated with the proposed weighting term, and in the second row of Figure 2.15, the trajectories are estimated without the proposed weighting term. It is notable that for the low-dynamic sequence "fr2desk_with_person" the improvement with static weighting is small, and for high-dynamic sequences the trajectory error is reduced greatly.

The integrated SLAM system is compared to a non real-time method [116], which is a recent state-of-the-art RGB-D SLAM method for dynamic environment. In [116], dense dynamic object segmentation is performed for each frame, which takes half second per frame. The authors segment dynamic objects from each frame, and directly use the segmented frames as input for DVO-SLAM system [57]. The comparison is shown in Table 2.7. The first column shows the sequence name from the TUM Dataset, where both low-dynamic "sitting" sequences and high-dynamic "walking" sequences are used for comparison. Our SLAM system works better in most of the sequences. The improvement for low-dynamic sequences is 15.2%, and the improvement for high-dynamic sequences is more notable with 24.7%.

The average computation time for the whole SLAM system takes ca. 45ms per frame, including visual odometry estimation, loop closure detection and pose graph optimization. Compared to this, the method from [116] cannot be applied for real-time application, since their segmentation procedure alone already takes half second per frame.

TABLE 2.7: SLAM results: RMSE of Absolute Trajectory Error [m]

sequence	Motion Remvoal+ DVO SLAM [116]		Our SLAM system	
	RMSE	standard deviation	RMSE	standard deviation
fr3/walking_halfsphere	0.1252	0.0903	**0.0489**	**0.7266**
fr3/walking_rpy	**0.1333**	**0.0839**	0.1791	0.1161
fr3/walking_static	0.0656	0.0536	**0.0261**	**0.0122**
fr3/walking_xyz	0.0932	0.0534	**0.0601**	**0.0330**
fr3/sitting_halfsphere	0.0470	0.0249	**0.0432**	**0.0246**
fr3/sitting_xyz	0.0482	0.0282	**0.0397**	**0.0206**
fr2/desk_with_person	0.0596	0.0239	**0.0484**	**0.0237**

2.2.6 Summary and Conclusion

In this section, a real-time RGB-D visual odometry method is proposed, which can handle highly dynamic environment such as the "walking" sequences from TUM Dataset [115]. The method uses foreground depth edge point to compute pair-wise point cloud registration. A robust static weighting strategy is proposed based on depth edge correspondences distance. Fusing the static weighting strategy into the intensity assisted ICP (Chapter 2.1), the visual odometry system handles dynamic environment robustly. Furthermore, loop closure detection and map optimization are integrated, resulting a real-time SLAM system suitable for dynamic environment. The accuracy and computation efficiency are tested on the dynamic sequences from TUM Dataset [115]. Compared to state-of-the-art real-time method [61], in terms of translational drift per second, the proposed method improves the visual odometry accuracy by 58% in challenging "walking" sequences. The performance of the SLAM system is also proven using the TUM Dataset, which shows better performance than recent non real-time method [116].

The proposed method highly depends on the quality of foreground depth edges. Therefore, the proposed method is limited to geometry rich environments. For a scene with only large plane structures, the proposed method might fail due to the the lack of foreground depth edges. In the future work, it is worth to investigate how to efficiently propagate the sparsely estimated static weights to the entire image, such that denser information can be used for registration. Nevertheless, the proposed method fulfills the requirement for the object scanning applications, where the scanned objects contains rich foreground depth edges. In the next section, object scanning application using the proposed SLAM system will be shown.

2.3 Application for object modeling

The previous sections introduced camera localization methods for an unknown environment. Using these techniques, the camera can be robustly localized with respect to the first frame's camera coordinate. This camera localization approach builds the foundation for object modelling application, where the object will be captured from different viewpoints and merged into

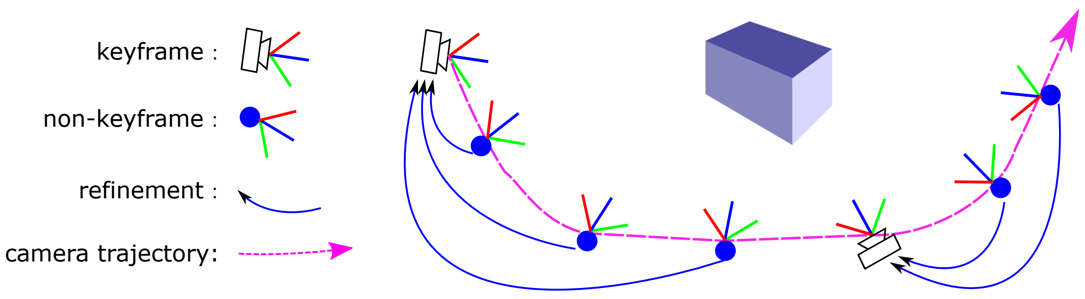


FIGURE 2.16: Object scanning scenario. The keyframes are refined with following non-keyframes.

a whole model. In the following, the object modelling pipeline will be described and some modelling results will be shown.

Object modeling pipeline

Figure 2.16 shows the object scanning scenario. The camera moves around the object and capture RGB-D image pairs at a frequency of 30 Hz, where the RGB-D pairs can be transformed to organized point cloud format (Equation 2.1). The camera's ego-motion is constantly updated as the camera moves and if the camera has been moved for more than 15cm or rotated for more than 20°, a new keyframe is selected.

Since the RGB-D camera contains uncertain measurement [89], the captured surface point cloud is noisy, which is undesired for the generated object model. Given the fact that the same part of object surface is captured multiple times with the RGB-D video, we decide to smooth all the keyframes using the whole video sequence. With the estimated camera pose of one keyframe, the point clouds of its following non-keyframes can be transformed into one common global coordinate. These point clouds are then be merged to reduce noise.

The refinement merges the frames at depth image level. A point $P_t(i)$ in a non-keyframe t will be firstly transformed and projected to the keyframe's image coordinate, $(u_i, v_i)^T = \pi(\mathbf{T}_t^k P_t(i))$. Then the depth value of the projected coordinate will be refined using a weighting factor $\lambda_t(i)$. Algorithm 2.3 describes the refinement process in detail.

To merge the i-th point into keyframe, we consider its measurement uncertainty based on its traveled distance d from the keyframe:

$$d = ||P_t(i) - \mathbf{T}_t^k P_t(i)||,$$

and the weighting factor for it is:

$$\lambda = min(0.016/d, 20.0), \tag{2.38}$$

which means less traveled point has a lower uncertainty and will be therefore assigned with a higher weighting factor.

As the visual odometry method runs, loop closure detection and map graph optimization is also performed, resulting in a global map graph, where

Algorithm 2.3 Keyframe refinement

```
Require: - Keyframe $P_k$ as organized point cloud.
  - Weighting factors $W$ with $W(i)$ for the i-th point $P_k(i)$
  - Following $T$ non-keyframes $\{P_t\}_{t=1,...,T}$
Ensure: - Refined keyframe $P_k$
  - initialize weighting factors:
  for $i = 1 : N$ do
    $W(i) = 100$
  end for
  for $t = 1 : T$ do
    for $i = 1 : N$ do
      - transformation to keyframe coordinate: $P_t(i)' = \mathbf{T}_t^k P_t(i)$
      - projection to image coordinate: $\mathbf{x} = \mathbf{T}_t^k P_t(i)$
      - get point index $j$ corresponding to $\mathbf{x}$: $j = ind(\mathbf{x})$
      - calculate update weighting factor of the point to be merged $\lambda$ (Equation 2.38).
      - merge the depth value and update weighting factor:
      $depth(P_k(j)) \leftarrow \frac{depth(P_k(j))W(j)+depth(P_t(i)')\lambda(j)}{W(j)+\lambda(j)}$
      $W(j) \leftarrow W(j) + \lambda(j)$
    end for
  end for
```

each keyframe is connected to several other keyframes. A connected pair indicates that the same part of the scene is seen twice from different viewpoints, and the model of that part of scene can be refined by integrating the keyframes. Therefore, as a final step, each keyframe will be refined by connecting keyframes, using the same procedure described above.

Modelling results

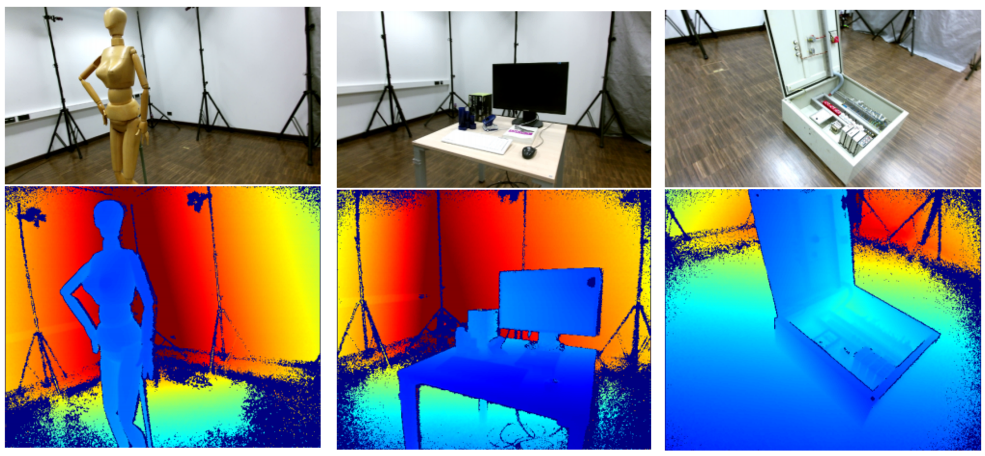


FIGURE 2.17: [CoRBS benchmark [133]. Top row shows the raw color images and the second row shows the color coded depth images.

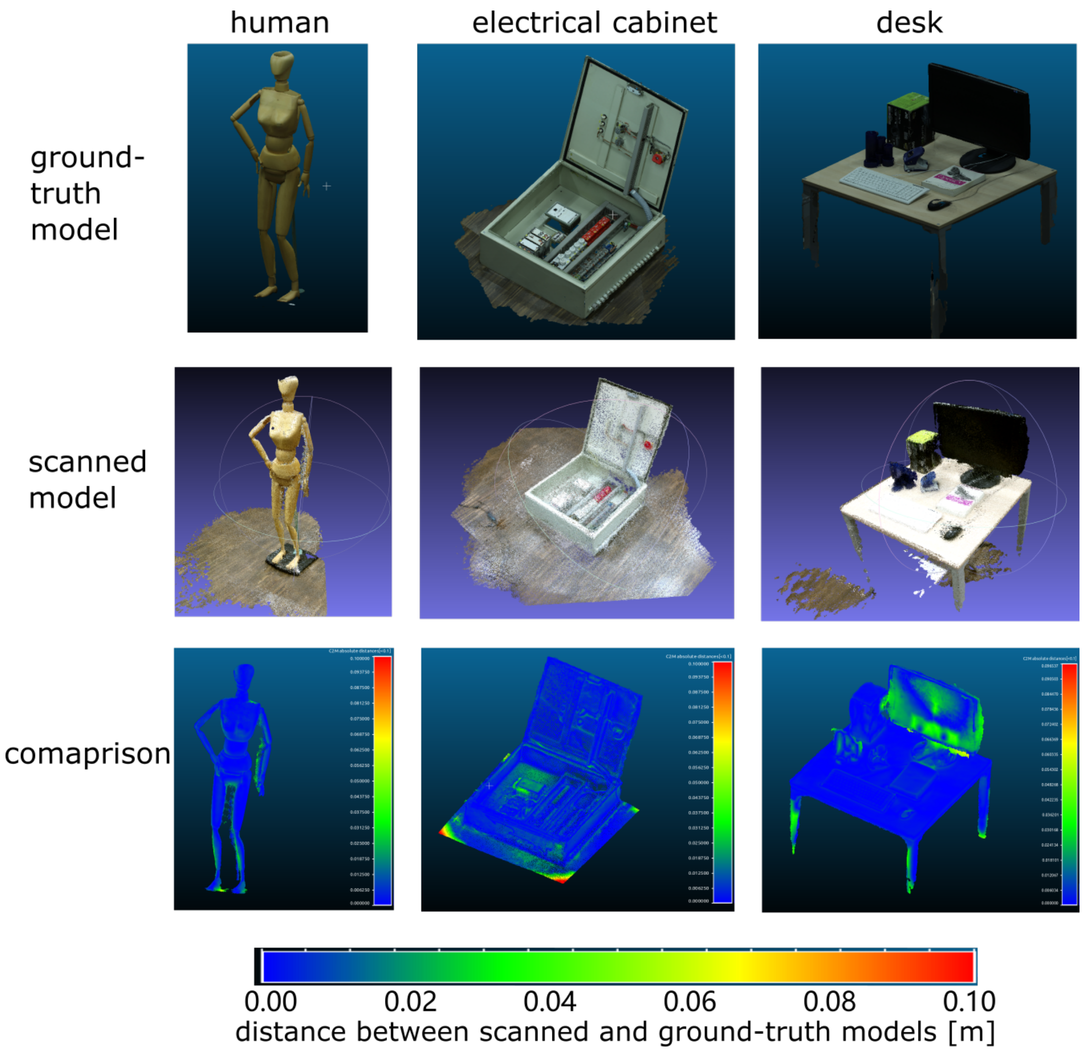


FIGURE 2.18: Object scanning results on CoRBS dataset. First row: ground-truth object model provided by the dataset. Second row: scanned objects using the proposed camera localization methods. Third row: the distance between scanned models and ground-truth models, where blue means close and red means more distanced.

To test the proposed object modeling pipeline, the CoRBS (Comprehensive RGB-D Benchmark for SLAM) benchmark dataset [133] is used. The dataset contains object scanning sequences with ground truth trajectories and groud-truth object models, where the RGB-D sequences are captured with a Kinect v2 camera (Figure 2.17).

We use the proposed pipeline to estimate camera trajectories and smooth the keyframes' point cloud measurement. Then the combined point cloud is compared to the ground truth object model and the distance between each point and ground truth object model's surface is used as the evaluation metric. As seen in Figure 2.18, the generated object models are very close to the ground-truth models, where the average distance to ground truth model is around 5 mm. The quantitative results are shown in Table 2.8.

TABLE 2.8: Distance between scanned points and ground-truth object model [m]

human		electrical cabinet		desk	
mean	median	mean	median	mean	median
0.0047	0.0030	0.0075	0.0054	0.0060	0.0024

Summary

This section described an object scanning pipeline using the earlier proposed camera localization methods. The accurate camera ego-motion estimation ensures successful scanning process. In addition, a surface smoothing procedure is proposed to reduce the noisy measurement of the keyframes. To clarify, the generated object models are not perfect as models designed using CAD (Computer-aided design), but it provides flexibility to quickly generate models for unknown objects, which will become more often for an intelligent robotic learning scenario.

2.4 Conclusion and future works

Conclusion

For the purpose of object scanning, this chapter presented camera localization methods using RGB-D camera. In the first section, the fast and robust visual odometry estimation method based on intensity assisted ICP (IAICP) is presented. By contributing in the selection, matching and weighting stages, IAICP improves the conventional ICP significantly. In the second section, an extension to IAICP is proposed, which can handle highly dynamic environment such as the "walking" sequences from TUM Dataset [115]. The method uses foreground depth edge point to compute pair-wise point cloud registration. A robust static weighting strategy is proposed to down-weight dynamic points in the framework of IAICP. Furthermore, loop closure detection and map optimization are integrated, resulting a real-time SLAM system suitable for dynamic environment. The proposed framework in the first two sections was used to construct a object scanning pipeline in the third section.

Discussion and Future works

The proposed visual odomery method is fast and accurate with a single core CPU. In comparison, previous methods have to rely on dense pixel-wise optimization, which require either GPU computation [38] or sophisticated code optimization and downsampled inputs to achieve real-time performance [57]. In comparison, the fast performance with 78 Hz of the proposed visual odometry method is gained from the choice of semi-dense salient points of the source frame and random sampling at each iteration. By using the illumination component in all stages in the ICP framework, the performance is significantly improved compared to the conventional ICP method [8]. The choice of semi-dense feature point was also the key factor for successful static point weighting. The foreground depth edge points are distinctive enough for the static weighting process. Furthermore, compared to sparse SIFT like keypoints, the semi-dense key-points are more evenly distributed the environment, such that it is more robust against highly textured dynamic objects which attracts too much keypoints on the dynamic object.

Despite the success of the proposed method, it has also its limitations. The proposed method highly depends on the quality of foreground depth edges. Therefore, the current method is limited to geometry rich environments. For a scene with only large plane structures, the proposed method might fail due to the the lack of foreground depth edges. Therefore, in the future work, it is worth to investigate how to improve the keypoint detection module in different ways. First, we can investigate how to efficiently propagate the sparsely estimated static weights to the entire image, such that denser information can be used for registration. For example, the static weights could be combined with an optical flow estimation process to propagate the staticness information to the whole image [138].

The second way for an improved keypoint detection module is to replace manually designed keypoint detection method with machine learning methods. With the thrive of deep learning research in the past few years, many different works have shown the ability to detect robust, distinctive and repetitive keypoints. Previous works have shown success in both 2D RGB image [21, 19] and in 3D point cloud [79, 5, 73]. Using pose alignment losses, these deep learning frameworks are trained to extract keypoints which contribute the most to robust motion estimation. A further benefit of deep learning based method is that some of these methods could be trained even in a self-supervised way [73, 21], without the need of labeled ground-truth data. However, current deep learning methods are designed for mono-modal input, either for RGB data alone or for point cloud alone. A new research direction could be to follow the similar idea in this chapter, i.e. to combine the RGB and point cloud information for training a method that can detect robust key-points in RGB-D domain.

Apart from the keypoint detection part, using deep learning based methods, another future research direction could be how to filter out dynamic objects using a trained model. Two possible options could be investigated. The first option is to detect static keypoints from semantic information, e.g. [79] is trained on autonomous driving scenario, such that the keypoints will be never occur on statistically moving objects, such as cars and pedestrians. The second option is to train a correspondence matching network [109], such a network takes the feature descriptor and keypoint coordinates as input, and can predict whether they are from static background or dynamic objects.

The object modelling pipeline could be also investigated more in the future. To fulfill the object modelling requirement, we have simply concatenated the point cloud taken from different viewpoint into a common coordinate system and performed voxelization and smoothing operation. However, this is far from generating a high quality object model. In future work, deep learning based method could be also used for object model generation.

A key technology from recent research, called differentiable rendering [77, 55] can be used for the object modelling task. This technology can render a mesh model, while keeping the gradient values at each operation, such that it can be used in the back-propagation process of a deep learning framework. Using differentiable rendering, the object mesh model, camera poses, lighting conditions etc. can be all viewed as trainable parameters. Then all parameters can be optimized in a single framework, using image reconstruction loss between different viewpoints.

Chapter 3

Hand pose estimation using depth image

This chapter includes the content of following publications: **Article:**

- *Shile Li*, Jan Wöhlke* and Dongheui Lee. "Model-based hand pose estimation for generalized hand shape with spatial transformer network." European Conference on Computer Vision (ECCV), Extended Abstract Presentation in 4th International Workshop on Observing and Understanding Hands in Action (HANDS2018). 2018. *equal contribution*

Other contributors:

- Jan Wöhlke (Master student)
- Dongheui Lee (Thesis supervisor)

Author contributions

- **SL** and JW developed the method and wrote the code jointly, where **SL** is more responsible for designing the method and implementing the appearance normalization layer and JW is more responsible for implementation of the variable kinematic layer. **SL** wrote the main body of the article. **SL**, JW, DL analyzed the results and revised the article.

3.1 Introduction

This chapter introduces a deep learning based hand pose estimation method. The proposed method uses depth image as input and regresses the hand joints' Cartesian coordinate.

Motivation

Hand pose estimation is an important precondition for many tasks in fields such as human computer interaction or augmented reality. Therefore, the 3D hand pose estimation problem has attracted many researchers' interest in the last ten years [24, 118, 146]. The availability of cheap commercial depth cameras has especially increased the interest in 3D hand pose estimation, because depth provides additional geometry information compared to RGB image and furthermore depth image has similar data structure as RGB image, such that mature deep learning elements for images (Convolutional Neural Networks) can be directly applied. However, the hand pose estimation task remains challenging due to several reasons: the kinematic complexity of the hand, which results in a large number of DoFs, self-occlusions, different viewpoints, and shape variations across different persons.

Hand pose estimation approaches can be divided into three categories: 1) the **generative**, model-driven approaches that fit a hand model to the image observations by minimizing a cost function [59, 111, 113, 54], 2) the **discriminative**, data-driven approaches that directly predict the 3D joint locations from the images [15, 20, 34, 31, 37, 80, 91, 90, 84], and 3) the **hybrid** approaches that combine discriminative and generative elements [92, 123, 127, 150].

Discriminative methods play an important role because they are needed to initialize generative tracking methods and to recover in the case of tracking failure. State-of-the-art discriminative methods use deep learning components such as 2D [15, 34, 37, 80, 91, 92, 90, 123, 143, 150] or 3D [20, 31, 84] Convolutional Neural Networks (CNN) that also might incorporate residual modules [15, 37, 84, 90]. Relying on a large annotated training dataset, the discriminative methods either directly regress joint locations [15, 20, 31] or output a probability density map for each joint [123, 34, 84]. Most discriminative approaches do not explicitly consider the kinematics and physical constraints of the human hand. As a result, implausible hand pose estimates can occur in the regression results. For example, the physical limits of the finger joint angles can be violated. To ensure physically plausible results, hybrid models that incorporate a generative component can be used. For example, Zhou et al. [150] integrate a generative forward kinematics hand model into their deep learning approach and impose physical constraint losses on the estimated hand parameters. However, in [150], the palm shape and bone lengths are fixed to a specific user so that the approach cannot generalize towards new hand shapes.

Since the emergence of large annotated datasets [147], state-of-the-art hand pose estimation methods have been mostly based on discriminative learning [146]. In 2016, a hybrid approach has embedded a kinematic layer into the deep learning structure in such a way that the pose estimates obey the physical constraints of human hand kinematics [150]. However, the existing approach relies on a single person's hand shape parameters, which are fixed constants. Therefore, the existing hybrid method has problems to generalize to new, unseen hands. In this chapter, a extended method to [150] is proposed, in which:

- the kinematic layer is extended to make the hand shape parameters adaptable. In this way, the learnt network can generalize towards arbitrary hand shapes.
- Furthermore, the Spatial Transformer Network (STN) [51] is also applied, such that the performance of a regression task can be also improved by 6%.

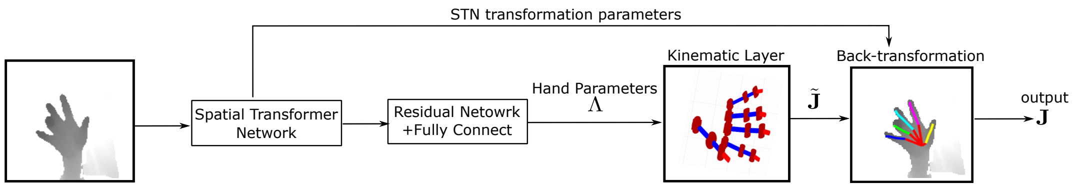

STN transformation parameters
Spatial Transformer Network
Residual Netowrk +Fully Connect
Hand Parameters Λ
Kinematic Layer
$\tilde{\mathbf{J}}$
Back-transformation
output $\mathbf{J}$

FIGURE 3.1: Overview of the proposed method.

The effectiveness and limitations of the proposed approach are evaluated on the Hands 2017 challenge dataset [147].

Spatial Transforer Networks

Hinton et al. [45] propose a transforming autoencoder as a generative model that models 2D affine transformations. The generative model learns to generate a transformed image of the input image, where the target pose is defined during the training. Zimmermann et al. [152] estimate the hand pose in a normalized coordinate system. The global transformation parameters are regressed separately. Jaderberg et al. [51] introduce a dynamic mechanism, the Spatial Transformer Network, that is trained end-to-end with the rest of the network without changing the loss function. A localization network regresses the transformation parameters from the input image, which are then used by the grid generator to transform a regular grid into a sampling grid. This sampling grid is applied to the input image to obtain the warped output image. In this thesis, considering the goal of regressing 3D poses, a Spatial Transformer Network for rigid transformation is applied.

3.2 Method

Overview

An overview of the proposed method is shown in Figure 3.1. The pre-processed depth input images are appearance-normalized using Spatial Transformer Network. Then, hand parameters $\mathbf{\Lambda}$ are estimated using Residual Network module, which are fed into a differentiable kinematic layer that maps $\mathbf{\Lambda}$ to joint locations. Finally, the joint locations are back-transformed into the initial coordinate system.

Pre-processing

First, a coarse 3D bounding box containing the hand is determined from the joint location ground truth annotation $\mathbf{J}^{gt}$ (or the bounding boxes are provided by the dataset), where the depth pixels inside the bounding box are converted to 3D points. Then the 3D points are demeaned by the 3D center of mass. Finally, the 3D points are back-projected to a depth image and resized to 128×128 pixels. For smoothing, 3×3 median filtering is applied. Afterwards, the depth values are normalized to the range $[-1, 1]$.

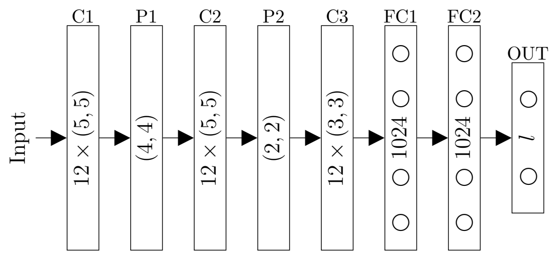
During training, online data augmentation is applied. The images are scaled, rotated, and translated by random factors drawn from the following distributions:

- **scaling:** normal $\mathcal{N}(1.0, 0.075)$ within range $[0.75, 1.25]$
- **rotation:** uniform $\mathcal{U}(-180°, 180°)$
- **translation:** normal $\mathcal{N}(0, 4)$ mm within range $[-15, 15]$ mm for x-, y-, and z-direction individually

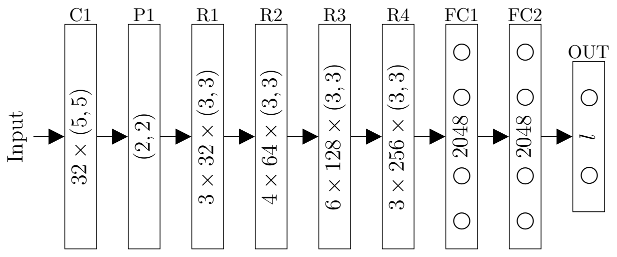
Network architectures

Due to its simplicity and computational efficiency, a basic CNN architecture is used for the Spatial Transformer Network. The detailed structure is illustrated in Figure 3.2(a).

To extract complex image features for the sake of higher pose estimate accuracy, residual network is used to regress the hand parameters out of the appearance normalized image. The detailed structure is illustrated in Figure 3.2(b).



(a) CNN architecture used for Spatial Transformer Network



(b) Residual network architecture used for regressing hand parameters

FIGURE 3.2: Network architectures. **C:** convolutional layer with number of feature maps and kernel size in brackets, **FC:** fully-connected layer with number of units, **OUT:** linear output layer with l output units, **P:** max pooling with kernel size in brackets, **R:** residual module with the number of bottleneck blocks, the number of feature maps (bottleneck layer), and kernel size of bottleneck layers.

Spatial Transformer Network

We apply a Spatial Transformer Network (STN) [51] to the input image. The STN estimates a 2x3 matrix

$$\mathbf{T}_{STN} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \tag{3.1}$$

to transform the images to a similar distribution of appearance. In order to preserve the validity of the hand kinematic layer, the spatial transformation parameters are constrained to rigid motion that contains rotation, scaling and translation, where $a_{11} = a_{22}$ and $a_{12} = -a_{21}$. This is achieved by estimating the scaling factor s, sine and cosine value of the rotation angle α separately:

$$\mathbf{T}_{STN} = s \begin{bmatrix} cos_\alpha & -sin_\alpha & a_{13} \\ sin_\alpha & cos_\alpha & a_{23} \end{bmatrix}, \tag{3.2}$$

To ensure that sine and cosine values are valid for the same α value, a constraint is added into the loss function:

$$L_{stn} = |1 - cos_\alpha^2 + sin_\alpha^2| \tag{3.3}$$

During training, with the estimated STN parameters, the corresponding ground truth data are also transformed accordingly for correct loss computation. For inference, the estimated joint positions can be also back-transformed in a straight forward way.

Kinematic layer for arbitrary hand shapes

The kinematic hand model layer implements the forward kinematics of the hand and therefore represents a mapping from hand parameters $\mathbf{\Lambda}$ to 3D joint locations $\mathbf{J}$. It is parameter free and differentiable to allow back-propagation.

The inputs $\mathbf{\Lambda}$ to the kinematic layer divide into four groups (Figure 3.3) listed below.

- 6D global pose and orientation of the middle finger MCP (Metacarpophalangeal) joint, which we define as the hand base $\mathbf{b}$ (6D)
- finger base positions in the local hand coordinate, $\mathbf{v}_i$, $(i =_1^5)$ (15D)
- 15 finger bone lengths $r_{i,n}$, $(i =_1^5, n =_1^3)$, where i indicates the index of the five fingers and n indicates the index of the three links per finger (15D)
- 25 finger joint angles $\theta_{i,n}$, $(i =_1^5, n =_1^5)$, where each finger has 5 DoFs, where i indicates the index of the five fingers and n indicates the index of the five DoFs per finger (25D)

The residual network in front of the kinematic layer regresses the 61 hand parameters $\mathbf{\Lambda}$. The 3D joint locations $\mathbf{J}$ are calculated using the hand parameters $\mathbf{\Lambda}$ by a cascade of transformation matrices. Each joint is considered to be the origin of its own local coordinate system. In order to back-transform these local coordinates to global world coordinates, kinematic transformations must be applied. For example, the 3D joint location of the thumb's ($i = 1$) tip's position $\tilde{\mathbf{j}}_{1,TIP}$ is

$$\tilde{\mathbf{j}}_{i,TIP} = \mathbf{T}_{\text{BASE}}\left(\mathbf{b}\right)\mathbf{T}_{\text{VEC},1}\left(\mathbf{v}_1\right)\prod_{n=1}^{5}\mathbf{T}_{\text{DH},1,n}\left(\theta_{1,n}, r_{1,n}\right)\left(0\ 0\ 0\ 1\right)^T, \tag{3.4}$$

The kinematics of the fingers are modeled using the DH (Denavit Hartenberg) convention, where $\mathbf{T}_{\text{DH},n,i}(\theta, r)$ is a joint specific transformation estimator, which is predefined according to hand's structure. It takes the joint angle and bone length as input and outputs a transformation matrix. $\mathbf{T}_{\text{Base},n,i}(\mathbf{b})$ extracts global transformation matrix out of the global pose parameters. For the convenience of notation, $\mathbf{T}_{VEC}(\mathbf{v})$ converts translation parameter to a homogeneous transformation matrix.

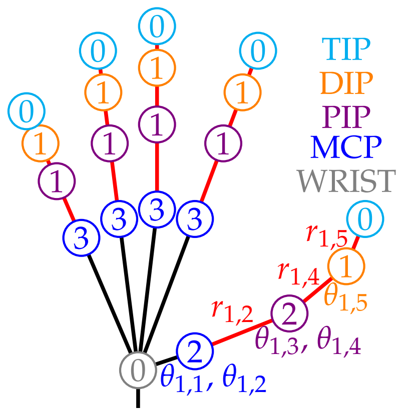


FIGURE 3.3: Kinematic hand modeling: Number of DoFs associated with the individual joints are indicated in the circles. Finger angles $\theta_{i,n}$ and bone lengths $r_{i,n}$ exemplary are indicated for the thumb.

The procedure of chaining transformation matrices is computationally efficient, as many intermediate results can be re-used for the coordinate transformation of other joints along the same finger. For training using backpropagation, the gradients of the layer are automatically calculated in TensorFlow [1], using its support of auto-differentiation.

Training losses

To learn the correct hand pose, L2 losses are enforced on the final pose $\mathbf{J}^{est}$ in Cartesian coordinate, as well as on the hand shape parameters $\mathbf{b}^{est}$ and $\mathbf{v}^{est}$, where *est* stands for estimated value and *gt* stands for ground-truth value.

$$\begin{aligned} L_{pose} &= \|\mathbf{J}^{est} - \mathbf{J}^{gt}\|^2 \\ L_{shape} &= \|\mathbf{b}^{est} - \mathbf{b}^{gt}\|^2 + \|\mathbf{v}^{est} - \mathbf{v}^{gt}\|^2 \end{aligned} \tag{3.5}$$

In order to force the finger joint angles $\boldsymbol{\theta}$ to stay within physically valid limits, physical constraint losses on the estimated joint angles $\boldsymbol{\theta}^{est}$, is are used. The physical constraint loss term which penalizes violations of upper and lower joint angle limits $\boldsymbol{\theta}_{up}$ and $\boldsymbol{\theta}_{low}$ is

$$\begin{aligned} L_{\text{constr}} = \sum_{\text{batch}} (|| \min\left(\left(\boldsymbol{\theta}^{\text{est}} - \boldsymbol{\theta}_{low}\right), 0\right) ||^2 \\ + || \max\left(\left(\boldsymbol{\theta}^{\text{est}} - \boldsymbol{\theta}_{up}\right), 0\right) ||^2) \end{aligned}. \tag{3.6}$$

3.3 Experimental results

The proposed method is evaluated on the Hands 2017 challenge dataset [147]. It is currently (at the time of writing this thesis, Aug. 2020) the largest and most diverse real-captured dataset available. Its training set contains 957032 depth images of five different hands. Therefore, it is suited for learning to regress hand parameters for various hand shapes. The test set consists of 295510 depth images of ten different hand shapes, of which five are the same as in the training set and five are entirely new.

The proposed pipeline is implemented using TensorFlow [1]. The networks are trained on a PC with an AMD FX-4300/Intel Core i7-860 CPU and an Nvidia GeForce GTX1060 6GB GPU. For training, the Adam optimizer is used with a learning rate of 1×10^{-4}. The batch size is 32.

As evaluation metric, the average per joint error is used:

$$e_{\text{joint}} = \frac{1}{N_{\text{images}}} \frac{1}{N_{\text{joints}}} \sum_{\text{images}} \sum_{\text{joints}} ||\mathbf{j}^{\text{est}}_{i,k} - \mathbf{j}^{\text{gt}}_{i,k}.|| \tag{3.7}$$

In addition to that, the joint angle limit violation is also used to evaluate the physical plausibility of the estimation result:

$$e_{\text{violation}} = \frac{1}{N_{\text{images}}} \frac{1}{N_{\text{dofs}}} \sum_{\text{images}} \sum_{\text{dofs}} \max(\boldsymbol{\theta}^{\text{est}}_{i,k} - \boldsymbol{\theta}^{\text{upper}}_{k}, 0) + \max(\boldsymbol{\theta}^{\text{lower}}_{k} - \boldsymbol{\theta}^{\text{est}}_{i,k}, 0) \tag{3.8}$$

Table 3.1 compares the pose estimation result of different methods. Three different errors are given: 1) the average across the complete test set (*Avg test*), 2) the average across the test set images of hand shapes seen during training (*Seen test*), and 3) the average across the test set images of unseen hand shapes (*Unseen test*). The Kinematic version achieves a good accuracy of

TABLE 3.1: Average per joint error of different models

Approach	Avg test [mm]	Seen test [mm]	Unseen test[mm]
Direct Regression	10.97	8.98	12.62
Kin	12.98	10.71	14.86
STN+Kin	12.12	9.93	13.95

12.98 mm, indicating that the kinematic layer can generalize to different hand shapes successfully. Adding the STN, the error reduces to 12.12 mm, showing

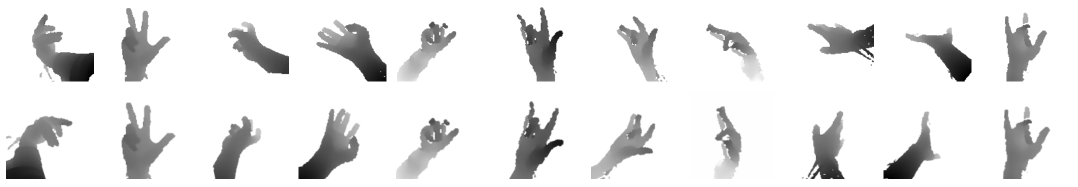


FIGURE 3.4: Appearance normalizing effect of STN: First row: input images. Second row: transformed image after STN

that the STN can be also applied to improve regression tasks. The STN shows an appearance normalizing effect on the hand pose estimation task (Figure 3.4), where the STN tends to rotate the hand such that the hand points to the upper-right direction. The direct regression of hand poses using a Residual Net achieves even lower error of 10.97 mm. However, by fitting the result of direct regression using inverse kinematic, there are 8.32% of estimated joint angles that violate physical limit of human hand (Table 3.2). Using the the proposed approach with kinematic constraints, the joint angle limit violation is reduced to 0.057%.

TABLE 3.2: Joint angle constraint violation with and without kinematic hand model layer

Approach	Violated joint limits	Avg violation in case of violation	Avg violation in total
Direct Regression	8.32%	35.57°	5.77°
Our Approach	0.057%	0.51°	0.00°

3.4 Summary

This chapter introduced a depth based hand pose estimation method. This method incorporates an embedded differentiable kinematic layer into the deep learning networks. Apart from joint angles, the proposed kinematic layer also takes hand shape parameters as input, thus it generalizes on different hand shapes. Experiments on public benchmark shows the pose estimation accuracy is proven to be accurate as 12mm. Furthermore, by using kinematic layer, the number of physically implausible results is reduced. We have also shown that applying appearance normalization using Spatial Transformer Network, the pose estimation accuracy can be further improved.

3.5 Discussion and future works

This chapter applied differentiable kinematic layer for hand pose estimation. The proposed differentiable kinematic layer improved the limitation of previous work by making the hand shape also trainable parameters, thus the proposed method could generalize to different subjects' hand. The major benefit of using a kinematic layer is that the output poses are constrained by physical limitation of human hand. The joint angle limitation violation of the direct regression method could be effectively eliminated. However, there is a trade-off between the robustness and average accuracy, by considering the joint limit, the accuracy of hand pose in Euclidean space is worse. Therefore, in practice, one might needs need to consider the application scenario to choose which type of method to use. For example, for entertainment application with Virtual Reality, the hand shape obeying joint limit is obviously more important, since a hand with physical implausible pose will harm the user experience a lot. Another example is computer vision based learning by demonstration, where the robot has to imitate human motion precisely to perform certain task, such as grasping a cup. In this scenario, precise Euclidean position of finger tips is important to let the the robot achieve the grasping task [103], because inaccurate finger tip pose will result in failure of imitating the task.

To reduce the variation of appearance in the input data, this chapter has applied Spatial Transformer Network (STN). The reduced variation eases the complexity requirement of the subsequent networks. Indeed, the experiment results have shown the hand pose accuracy is improved by using STN. However, there are two shortages of using STN in practice. Firstly, it introduces additional computation overhead, which consumes more memory and computation time. Secondly, the training process for our task requires careful tuning of hyper-parameters, because the network easily gets diverged. Possible reason is that the rigidity constraint on the network output might cause instability for the training process. In future work, it is worth to investigate other rotation representation, such as predicting the quaternion representation.

Due to the hardware limitations, this work has used a shallow ResNet structure with 18 layers that is trained from scratch. In future work, it is worth to try a more complex deep neural network, e.g. ResNet-50, to see whether there could be an improvement. Furthermore, we could use pre-trained weights on ImageNet's RGB images as starting point for training. However, the ImageNet has a different domain (RGB) than our data (depth), in future work, it is worth to investigate on domain transfer strategies to see whether it is possible to still benefit using the pre-trained weights from RGB images, e.g. using a domain classifier [26]

This work uses a differentiable kinematic layer using DH convention. Our extension to previous work makes the hand size and bone lengths as variables, such that it can adapt to different subjects. However, this work still cannot control the complete hand's appearance, e.g. the finger thickness, shape of the muscle tissue, skin colors etc. Since the end of this work, a more complex hand model, MANO [106], was made differentiable and used for deep learning based method in several works [40, 10]. The MANO is a deformable hand model, which can create near-realistic 3D mesh model of human hand. In the future work, the MANO model could substitute the simple kinematic layer in various hand pose related researches.

In this work, Residual Networks and CNN structures, which are usually used for 2D RGB images, are used for the depth image, because depth image shares the similar structure with the RGB image. However, the depth camera actually provides 3D geometry information of the environment. Using same method, which was designed for RGB image, could limit the performance. Therefore, in the next chapter, deep learning method using point cloud will be presented.

Chapter 4

Hand pose estimation using point cloud

The presented method in the last chapter uses depth image as input, which provides the conveniences to use the well developed convolutional neural networks or residual networks. However, methods using 2D images as input cannot fully utilize 3D spatial information in the depth image. Furthermore, the appearance of the depth image is dependent on the camera parameters, such that the trained model using one camera's image cannot generalize well to another camera's image. On the other hand, 3D data is more "direct" and "distinctive" than depth image because the appearance of 3D data is unique and invariant to camera intrinsic parameters. For the purpose of hand pose estimation, this chapter presents a 3D data based method using unordered point cloud as input format. This chapter includes the content of the following publication:

- *Shile Li and Dongheui Lee. "Point-to-pose voting based hand pose estimation using residual permutation equivariant layer." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2019.*

4.1 Motivation

Hand pose estimation plays an important role in human-robot interaction tasks, such as gesture recognition and learning grasping capability by human demonstration. Recently, methods using 3D data as input have shown the outperformance over depth image based methods [146]. One way to use 3D input data is to convert 2D depth image to volumetric representation, such as 3D voxels [84] [20], where occupied 3D voxel is set to 1 and voxels with empty space is set to 0. Using the voxelized data brings the convenience to directly use 3-dimensional CNN learning structure. However, the voxelization requires large amount of memory to represent the input and output data, which prevents the deployment of a very deep structure.

Another way to use 3D input data is to use unordered point cloud as input [33, 30, 16]. Recently, PointNet, a deep learning structure for point cloud, has shown its success in different tasks. The PointNet estimates point-wise features for individual points and extract global feature from individual points using a max-pooling layer, such that the network is invariant to the order of points. Ge et al. use PointNet [99, 101] as backbone to estimate hand

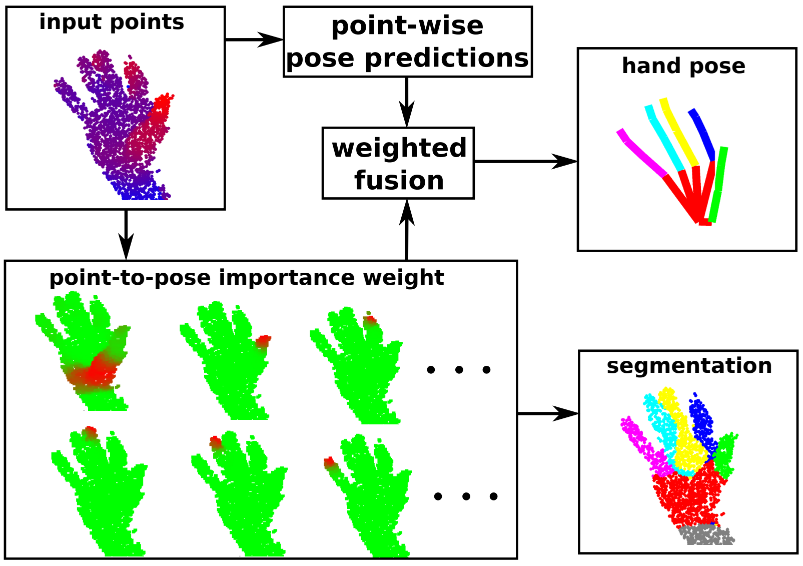


FIGURE 4.1: The proposed method takes point cloud as input. Then each point predicts the hand pose and its importance weights for different pose dimensions. The final pose is obtained through weighted fusion from each point's pose prediction. Using the importance weight, the hand can be clearly segmented into different parts, although no segmentation ground-truth was used during training.

pose from point cloud [33]. However, tedious pre-processing steps such as surface normal estimation and k-nearest-neighours search are required for [33]. Moreover, the final max-pooling layer in the PointNet neglects many informations that might be crucial for pose estimation.

In this chapter, a more flexible learning structure for unordered point sets, the permutation equivariant layer (PEL) [105, 148], is explored. The PEL is a deep learning structure that can be applied for unordered points. In PEL, point-wise features are computed, where each point's feature does not only depend on its own input, but also the global maximum value. Using PEL as the basic element, a residual network version of PEL is proposed to construct a deep network for hand pose estimation task. Moreover, a point-to-pose voting scheme is proposed to merge point-wise features, which eliminates the use of max-pooling layer to extract global feature, thus avoiding the loss of information. Furthermore, the generated point-to-pose importance weights can be also used for the hand segmentation task (Fig. 4.1), where clear segmentation result can be obtained even without the segmentation ground-truth. The proposed method is evaulated on Hands2017 Challenge dataset and NYU dataset, where state-of-the-art performance is shown. It achieved the lowest pose error on the Hands2017 Challenge dataset before November 2018. The contributions of this chapter are:

- A novel deep learning based hand pose estimation method for unordered point cloud is proposed. Using Permuation Equivariant Layer as the basic element, a residual network version of PEL is used to solve the hand pose estimation task. Compared to PointNet [101] based methods, the proposed method doesn't require tedious steps such as normal estimation, nearest neighbour estimation.

- A point-to-pose voting scheme is proposed to merge the information from point-wise local features, which also generates weakly-supervised segmentation results without the need of segmentation ground-truth.
- The proposed method is evaluated on Hands2017 Challenge dataset and NYU dataset, where state-of-the-art performance is shown.

4.2 Related works

A lot of research about hand pose estimation has been done in the last decade, which can be categorized to generative, discriminative and hybrid methods. Generative methods rely on a hand model and an optimization method to fit the hand model to the observations [106, 122, 102, 93]. Discriminative methods use learning data to learn a mapping between observation and the hand pose [91, 123, 84, 20, 16, 90, 111, 120]. Hybrid methods use a combination of the generative and discriminative methods [92, 111, 143]. The proposed method in this chapter is a learning based method thus falls into the second category.

4.2.1 Deep learning for hand pose estimation

With the success of deep learning methods for 2D computer vision, depth image based deep learning methods also showed good performance in hand pose estimation task. Tompson et al. use 2D CNN to predict heatmaps of each joint and then rely on PSO optimization to estimate the hand pose [123]. Oberweger et al. [91] uses 2D CNN to directly regress the hand pose out of the image features, where a bottleneck layer was used to force the predicted pose obey certain prior distribution. In a later work, Oberweger and Lepetit [90] replaced CNN to a more sophisticated learning structure, ResidualNet50, to improve the performance of feature extraction. Zhou et al. [150] regress a set of hand joint angles and feed the joint angles into an embedded kinematic layer to obtain the final pose. Ye et al. [142] use a hierarchical mixture density network to handle the multi-modal distribution of occluded hand joints.

Recently, 3D deep learning has been also applied for the hand pose estimation task. Moon et al. use 88^3 voxels to represent hand's 3D geometry and use 3D CNN to estimate hand pose [84]. Their method achieved very accurate result, however, 3D voxelization of the input and output data requires large memory size, such that their method only runs at 3.5 FPS. Ge et al. [33, 30] use 1024 3D points as input, and rely on PointNet [101] structure to regress the hand pose. Their method achieved satisfying performance, but tedious pre-processing steps are required, which includes oriented bounding box (OBB) calculation, surface normal estimation and k-nearest-neighbours search for all points. Chen et al. improves Ge's method by using a spatial transformer network to replace the OBB and furthermore added a auxiliary hand segmentation task to improve the performance [16]. Their method can

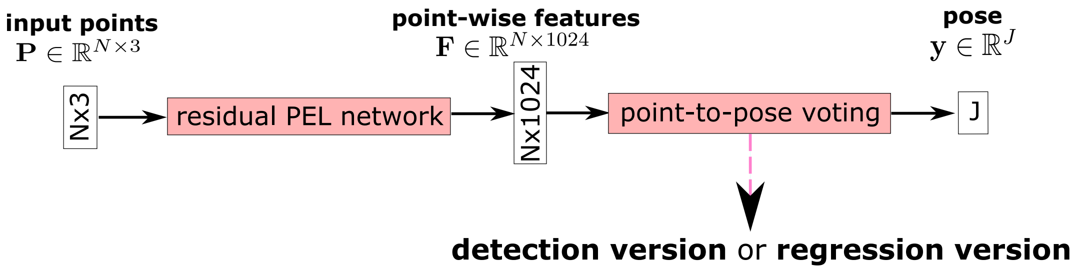

input points
$\mathbf{P} \in \mathbb{R}^{N\times 3}$
Nx3
residual PEL network
point-wise features
$\mathbf{F} \in \mathbb{R}^{N\times 1024}$
Nx1024
point-to-pose voting
pose
$\mathbf{y} \in \mathbb{R}^{J}$
J
detection version or regression version

FIGURE 4.2: Overview of the proposed method.

be trained end-to-end without OBB, but the segmentation ground-truth data require a extra pre-computation step from the pose data.

4.2.2 3D Deep learning

Since 3D data cannot be directly fed into a conventional 2D CNN, some methods project the 3D data onto different views to obtain multiple depth images and perform CNN on all images [34, 100] [42, 144]. Another way to process the 3D data is to use volumetric representation and process the data with 3D CNN [32, 137, 82, 84]. These methods can capture the feature of input data more effective, but they require large memory size. Qi et al. developed PointNet to handle unordered point cloud [99]. The PointNet estimates point-wise local features and obtains global features with a max-pooling layer. Later on, PointNet++ extends PointNet by hierarchically upsampling the local features into higher levels [101].

Other recent methods taking 3D points as input include point-wise CNN [47], Deep kd-Networks [63], Self-Organizing Net [72] and Dynamic Graph CNN [132]. Despite their good performance for different tasks, they all require extra steps to estimate k-nearest neighbours or construct kd-tree, which are not required in our proposed residual PEL network.

4.3 Method

The overview of the proposed method is illustrated in Fig. 4.2. The network takes N 3D points $\mathbf{P} \in \mathbb{R}^{N\times 3}$ with arbitrary order as input, and regresses the vectorized 3D hand pose $\mathbf{y} \in \mathbb{R}^{J}$ in the end, where $J = 3 \times \#joints$. To estimate the hand pose, the residual permutation equivariant layers (PEL) (Fig. 4.4) first extract features from each point. Using the point-wise local features, point-to-pose voting is applied to estimate the final pose output, where two versions for point-to-pose voting are developed, which are the detection version and the regression version.

4.3.1 Pre-processing with view normalization

For pre-processing, first, the depth pixels in the hand region are converted to 3D points. The next step is to create a 3D bounding box for the hand points to obtain normalized coordinate of these points. A usual pre-processing method will simply create a bounding box aligned with the camera coordinate system (Fig 4.3a). However, because of self-occlusion of the hand, this will result in different set of observation points for the exact same pose label, which creates one-to-many mapping of the input-output pairs.

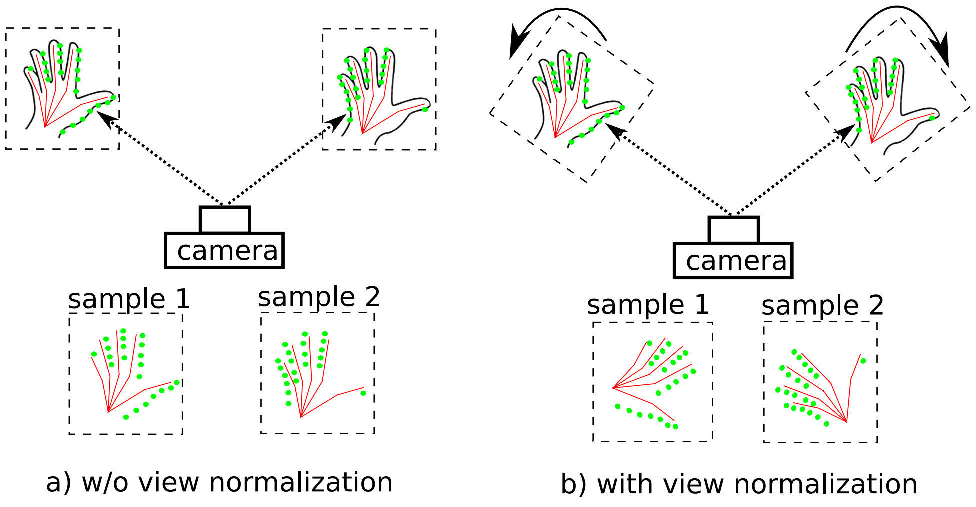


FIGURE 4.3: View normalization as pre-processing step. Red skeletons indicate ground-truth pose, green points indicate observed points of the camera. a) The same hand pose results in different observations due to different view directions, thus the resulted training samples will contain one-to-many mappings. b) With view normalization, the different observations will also have different pose labels, thus the input-output pairs will have a one-to-one mapping.

To maintain the one-to-one mapping relation of the input-output pairs, we propose to use view normalization to align the bounding box's z-axis $[0, 0, 1]^T$ with the view direction towards the hand centroid point $\mathbf{c} \in \mathbb{R}^3$. The alignment is performed by rotating the hand points with a rotation matrix $\mathbf{R}_{cam}$:

$$\begin{aligned} \alpha_y &= atan2(\mathbf{c}_x, \mathbf{c}_z), \\ \widetilde{\mathbf{c}} &= \mathbf{R}_y(-\alpha_y) \cdot \mathbf{c}, \\ \alpha_x &= atan2(\widetilde{\mathbf{c}}_y, \widetilde{\mathbf{c}}_z), \\ \mathbf{R}_{cam} &= \mathbf{R}_y(-\alpha_y) \cdot \boldsymbol{R}_x(\alpha_x). \end{aligned} \tag{4.1}$$

After rotating the observation points and ground truth pose with $\mathbf{R}_{cam}$, the hand is rotated such that it appears right in front of the camera. As illustrated in Fig. 4.3b), the one-to-many mapping problem is then avoided.

4.3.2 Residual Permutation Equivariant Layers

The feature extraction module in our method is called Residual Permutation Equivariant Layers. The basic element is the permutation equviariant layer (PEL), which follows the design from [105]. A PEL takes a set of unordered points as input and computes separate features for each individual input point.

Assuming that the input for a PEL is $\mathbf{x} \in \mathbb{R}^{N \times K_{in}}$ and the output is $\mathbf{x}' \in \mathbb{R}^{N \times K_{out}}$, where N is the number of points and K_{in}, K_{out} are the size of input and output feature dimensions. The output $\mathbf{x}'$ of the PEL is:

$$\mathbf{x}' = \sigma(\mathbf{1}_N \boldsymbol{\beta}^T + (\mathbf{x}\,\mathrm{diag}(\boldsymbol{\lambda}) + \mathbf{1}_N \mathbf{x}_{max}^T\,\mathrm{diag}(\gamma))\mathbf{W}), \tag{4.2}$$

where $\lambda \in \mathbb{R}^{K_{in}}, \gamma \in \mathbb{R}^{K_{in}}$ are weighting terms for the point's own feature and the global maximum value respectively, and $\mathbf{x}_{max} \in \mathbb{R}^{K_{in}}$ is a vector representing maximum values for each column of $\mathbf{x}$. $\mathbf{W} \in \mathbb{R}^{K_{in} \times K_{out}}$ is the weight term and $\boldsymbol{\beta} \in \mathbb{R}^{K_{out}}$ is the bias term and $\mathbf{1}_N \in \mathbb{R}^N$ is a vector full of ones. Furthermore, an activation function $\sigma(\cdot)$ is applied to provide non-linearity, where a sigmoid function is used later for the proposed hand pose estimation method.

This layer is invariant to input order because the output value of each individual point only depends on its own input feature and the global maximum values in each feature dimension, whereas the global maximum values are also invariant to the order of input points.[1] In this way, each point's local feature is not only computed based on its own input feature, each point also exchanges information with other points through the weighted summation of $\mathbf{x}_{max}$.

For the practical side, four elements need to be trained, which are $\boldsymbol{\beta}$, $\boldsymbol{\lambda}$, γ and $\mathbf{W}$. In total, the number of parameters needed for one layer is $K_{out} + (K_{out} + 2)K_{in}$, which is only slightly more than a fully-connected layer, thus it is feasible for training in practice.

In order to extract very complex features, we construct a residual network with 39 PEL layers. As illustrated in Fig. 4.4, we use three residual blocks, whereas each residual block consists of 13 PELs and four short-cut connections.

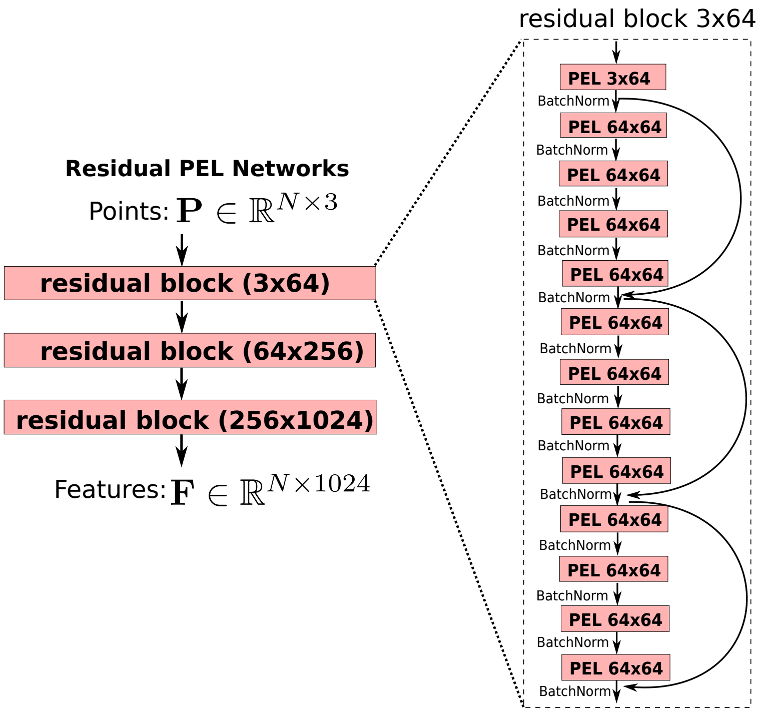
4.3.3 Point-to-pose voting

With the residual PEL module, features $\mathbf{F}$ of points are computed, where each row of $\mathbf{F}$ represents local feature for one point. Using these local point-wise features, the hand pose $\mathbf{y} \in \mathbb{R}^J$ will be estimated using a point-to-pose voting scheme. Two versions for point-to-pose voting are explored, which are the detection based version and the regression based version. The performance of these two versions will be compared in the experiment section.

Detection version

In the detection version (Fig. 4.5), probability distributions of each pose dimension is firstly detected and the pose is then integrated from the distributions. We use two separate fully connected modules to estimate two matrices: an importance term $\mathbf{G} \in \mathbb{R}^{N \times J}$ and a distribution term $\hat{\mathbf{D}} \in \mathbb{R}^{N \times J \times B}$. An element of importance matrix $\mathbf{G}_{nj}$ represents the confidence level of nth input point to predict the jth output pose dimension. In other words, each of the

[1]The detailed proof of the invariance for PEL can be found in [105].



residual block 3x64
Residual PEL Networks
Points: $\mathbf{P} \in \mathbb{R}^{N\times 3}$
residual block (3x64)
residual block (64x256)
residual block (256x1024)
Features: $\mathbf{F} \in \mathbb{R}^{N\times 1024}$
PEL 3x64
PEL 64x64
BatchNorm

FIGURE 4.4: Residual network of permutation equivariant layer

N points predicts J B-dimensional distributions and J corresponding importance weights. Notice that the final layer of the two fully connected modules are sigmoid functions, such that all elements of $\mathbf{G}$ and $\hat{\mathbf{D}}$ are in the range of $[0, 1]$.

$\hat{\mathbf{D}}$ represents the output pose distributions, where each point makes its own predictions to J output dimensions. Each of the output pose dimension is represented as discrete distribution using a B bins, representing the value range in $[-r, +r]$ with the resolution per bin $\Delta d = 2r/B$. For the jth dimension of the output pose $\mathbf{y}_j$, the corresponding bin index for itself is then:

$$index_j^{gt} = \lceil (\mathbf{y}_j^{gt} + r)/\Delta d \rceil,$$

and the ground truth distribution is defined as:

$$\mathbf{D}_{jb}^{gt} = \begin{cases} 1, & \text{if } b \in [index_j^{gt} - 1, index_j^{gt} + 1] \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

whereas the three bins around ground truth pose are set to one and all other bins are set to zero.

The final distribution for the J-dimensional output $\mathbf{D} \in \mathbb{R}^{J\times B}$ is then obtained by merging the predictions of all N points:

$$\mathbf{D}_{jb} = \frac{\sum_{n=1}^{N} (\mathbf{G}_{nj} \hat{\mathbf{D}}_{njb})}{\sum_{n=1}^{N} \mathbf{G}_{nj}}. \tag{4.4}$$

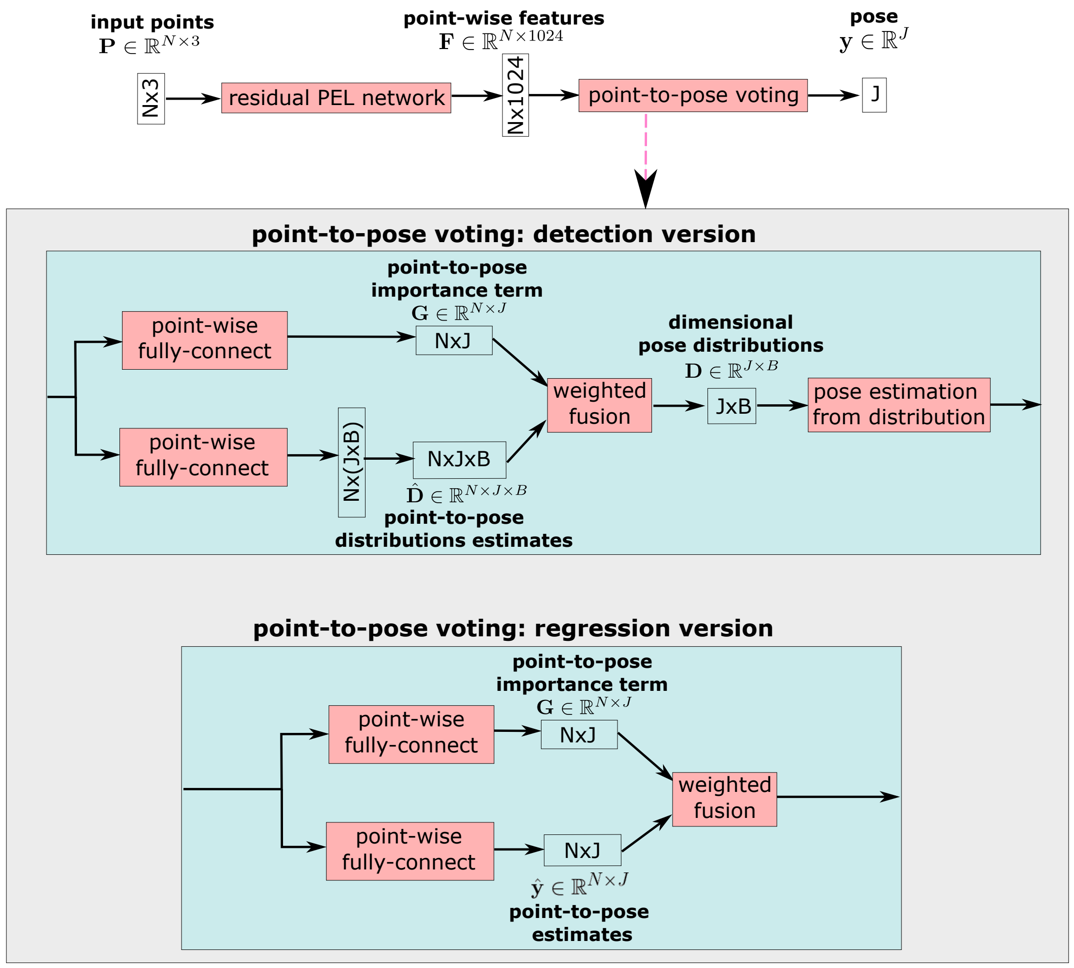

input points
$\mathbf{P} \in \mathbb{R}^{N\times3}$
Nx3
residual PEL network
point-wise features
$\mathbf{F} \in \mathbb{R}^{N\times1024}$
Nx1024
point-to-pose voting
pose
$\mathbf{y} \in \mathbb{R}^{J}$
J
point-to-pose voting: detection version
point-wise fully-connect
point-to-pose importance term
$\mathbf{G} \in \mathbb{R}^{N\times J}$
NxJ
point-wise fully-connect
Nx(JxB)
NxJxB
$\hat{\mathbf{D}} \in \mathbb{R}^{N\times J\times B}$
point-to-pose distributions estimates
weighted fusion
dimensional pose distributions
$\mathbf{D} \in \mathbb{R}^{J\times B}$
JxB
pose estimation from distribution
point-to-pose voting: regression version
point-wise fully-connect
point-to-pose importance term
$\mathbf{G} \in \mathbb{R}^{N\times J}$
NxJ
point-wise fully-connect
NxJ
$\hat{\mathbf{y}} \in \mathbb{R}^{N\times J}$
point-to-pose estimates
weighted fusion

FIGURE 4.5: Two different versions for point-to-pose voting.

And the final pose $\mathbf{y}$ is estimated with integration over the distribution:

$$\mathbf{y}_j = \frac{\sum_{b=1}^{B}(b-0.5)\mathbf{D}_{jb}}{\sum_{b=1}^{B}\mathbf{D}_{jb}}, \tag{4.5}$$

where $b - 0.5$ represents the bin center position.

Regression version

In the regression version (Fig. 4.5), each point will directly predict the pose without the intermediate distribution detection. Similarly to the detection version, two separate fully connected modules are used to estimate the importance term $\mathbf{G} \in \mathbb{R}^{N\times J}$ and the point-to-pose estimates $\hat{\mathbf{y}} \in \mathbb{R}^{N\times J}$. Then the final pose output is merged as the weighted average over all points' predictions:

$$\mathbf{y}_j = \frac{\sum_{n=1}^{N}(\mathbf{G}_{nj}\hat{\mathbf{y}}_{nj})}{\sum_{n=1}^{N}\mathbf{G}_{nj}}. \tag{4.6}$$

4.3.4 Segmentation using importance term

The importance term $\mathbf{G} \in \mathbb{R}^{N \times J}$ is estimated automatically without the ground-truth information. However, it still provides vital information of each point's importance to the pose output. Therefore, the obtained importance term can be also used for the hand segmentation task based on the most contributed pose dimension. For the n-th point having the importance terms $\mathbf{g} = \mathbf{G}_n$, the point's most contributed pose dimension is:

$$j_{max} = \underset{j}{\operatorname{argmax}}\, \mathbf{g}_j,$$

where the pose dimension j_{max} can be categorized to a specific hand part. In this work, we categorized the J pose dimensions to palm, thumb, index, ring and pinky fingers.

4.3.5 Training Loss

The only training loss for the detection version is the logarithm loss of the pose distributions:

$$\begin{aligned} L_{det} = -\sum_{j=1}^{J}\sum_{b=1}^{B} \mathbf{D}_{jb}^{gt} log(\mathbf{D}_{jb} + \epsilon) \\ +(1 - \mathbf{D}_{jb}^{gt})(1 - log(\mathbf{D}_{jb} + \epsilon)), \end{aligned} \tag{4.7}$$

where $\epsilon = 10^{-7}$ is a small offset to avoid feeding zero to the logarithm operator.

The only training loss used for the regression version is the L2 loss between predicted pose and ground-truth pose:

$$L_{reg} = \frac{1}{2}\sum_{j=1}^{J}(\mathbf{y}_j^{gt} - \mathbf{y}_j)^2. \tag{4.8}$$

Notice that for both detection and regression versions, the importance term $\mathbf{G} \in \mathbb{R}^{N \times J}$ is estimated automatically without the ground-truth information.

4.4 Experimental results

The proposed hand pose estimation method is evaluated on the Hands2017 Challenge dataset [147] and the NYU [123] dataset. The Hands2017 Challenge is composed from parts of the Big Hand 2.2M dataset [145] and the First-person Hand Action Dataset (FHAD) [28], it is currently the largest dataset available. Its training set contains 957032 depth images of five different hands. The test set consists of 295510 depth images of ten different hand shapes, of which five are the same as in the training set and five are entirely new. The NYU dataset contains 72757 training images of a single subject's hand and 8252 test images that include a second hand shape besides the one from the training set. The NYU dataset provides depth images from three different views, we trained the method with two opitons, one uses only frontal view data and the other one uses all three views. For testing, we test only using the frontal view.

The proposed method is implemented using TensorFlow [1]. The networks are trained on a PC with an AMD FX-4300/Intel Core i7-860 CPU and an nVidia GeForce GTX1060 6GB GPU. We train 100 epochs for the NYU dataset and train only 20 epochs for the Hands 2017 challenge dataset since the challenge dataset has a large size. For both datasets, the first 50% of the epochs are trained with smaller number of points ($N = 256$) to boost the training speed. The remaining epochs are trained with a point size of $N = 512$. The Adam optimizer is used for training with an initial learning rate of 10^{-3} and the learning rate is decreased to 10^{-4} for the last 10% of the epochs. For the detection version, we set $r = 15mm$ and $B = 60$. Online augmentation was performed with random translation in all three dimensions within $[-15, 15]mm$, random scaling within $[0.85, 1.15]$ and random rotation around z-axis within $[-\pi, \pi]$.

4.4.1 Self-comparison

In this subsection, we perform self-comparison to show the effects of different components in our method. The detailed comparison can be found in Table 4.1 and Table 4.2.

View normalization. To validate the necessity of view normalization, we trained our method using both view normalized data and original data for the detection version. It is evident from Table 1 that view normalization decreases the pose estimation error by about 1.5 mm for the Hands2017 Challenge dataset.

Detection vs. regression. Yuan et.al. indicates that detection based methods work in general better than regression-based methods [146], therefore we implemented both detection-based (ours/distribution) and regression-based (ours/regression) variations. As seen from in Table 4.1 and Table 4.2, in both datasets, both variations show similar performance, where regression-based variation slightly outperforms the detection-based counterpart. Possible reasons for this can be quantization effect of the binary distribution and the simplification of 1-dimensional heat vectors compared to 2D or 3D heat maps used in previous works. However, the 1D heat vector representation is much more efficient than the 3D heatmap representation. For the heat vectors, we need $B \times J$ values to represent the pose output, whereas 3D heatmaps require $J \times B^3$ values [84]. In future work, it is worth to investigate more different loss types and heat map representations.

Number of points. Taking advantage of the PEL structure and voting-based scheme, our method is very flexible to the input point cloud size. Although the network was trained with 512 points, arbitrary number of points can be used at the testing stage. For an online application, this property can

	detection	detection w/o view normalization	regression
256 points	11.34	13.14	11.21
512 points	10.23	11.93	10.11
1024 points	9.93	11.67	9.82
2048 points	9.93	11.69	9.87

TABLE 4.1: Self-comparison result on Hands2017Challenge dataset: mean joint error [mm]

	detection/ single view	regression/ single view	regression/ three views
256 points	9.82	9.45	9.05
512 points	9.33	9.06	8.49
1024 points	9.25	8.99	8.35
2048 points	9.32	9.08	8.35

TABLE 4.2: Self-comparison result on NYU dataset: : mean joint error [mm]

be beneficial to choose an arbitrary number of points based on the computational resources available. As seen from in Table 4.1 and 4.2, different number of points were tested for both datasets. Our method can achieve good performance with only 256 points, the mean joint error only increased by 0.11 mm compared to 512 points. In general, more points provides better performance, but it doesn't improve any more after 1024 points. Therefore, we choose 1024 points for testing to compare our method with other state-of-the-art methods.

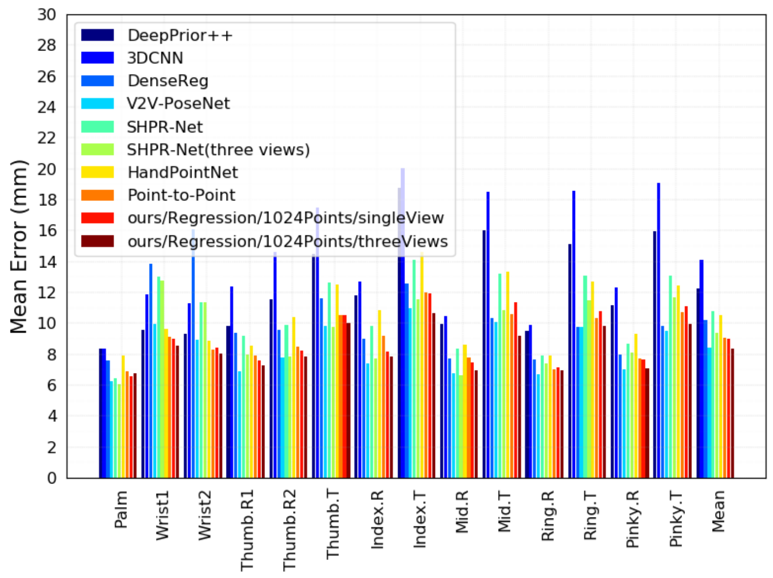
Comparison of different network structures. We also performed self-comparison at the initial exploration stage to verify the advantage of ResPEL network. From Table 4.3, the incremental improvements of residual version and voting scheme can be shown. 4PELs is a shallow network with four PELs with (64,256,512,1024) hidden neurons and Maxpooling means the voting scheme is replaced by a max-pooling layer and 4 consequent fully-connected layers to regress the hand pose. By replacing the shallow network with residual version (ResPEL+Maxpooling), the pose error is reduced significantly. Further improvement can be achieved by using our proposed voting scheme (ResPEL+Voting).

	4PELs+ Maxpooling	ResPEL+ Maxpooling	ResPEL+ Voting
mean joint error [mm]	16.60	11.02	8.35

TABLE 4.3: Self-comparison of different network structures on NYU dataset (three views)

4.4.2 Comparison to state-of-the-art methods

Hands2017 Challenge dataset. Since the ground-truth data for the testing set publicly available, some previous papers divide the training set on their own to create their own testing set. Therefore, for fair comparison, we only compare to those methods, who have also tested on the official testing website (status: November 2018). In Table 4.4, we compare our method with five other top performing methods on the Hands2017 Challenge dataset, which include both methods using 3D input data and methods using 2D depth image. RCN-3D [146], THU VCLab [15] and Vanora [146] use depth image as input data. V2V-PoseNet [84] uses voxel representation for both input data and output heatmaps. Oasis [33] also uses 3D point cloud as input and their method is constructed based on PointNet [99]. Three different errors are used for comparison: 1) the average across the complete test set (*avg test*), 2) the average across the test set of seen subjects' hand during training (*seen test*), and 3) the average across the test set images of unseen subjects' hand (*unseen test*). Currently, our method achieves the lowest overall mean joint error on the test dataset of 9.82 mm. For seen subjects' hand and unseen subjects' hand, the mean joint errors are 7.15 mm and 12.04 mm respectively, which shows the generalizability of the proposed method even without regularization on the parameters. In comparison to other 3D data based methods, our method is slightly better than V2V-PoseNet, whereas V2V-PoseNet requires 10 good GPUs to run realtime and our method requires only one moderate GPU. Compared to oasis, which also uses 1024 3D points as input, our method is 1.48 mm better, where oasis requires more input information like surface normal and k-nearest neighbours.



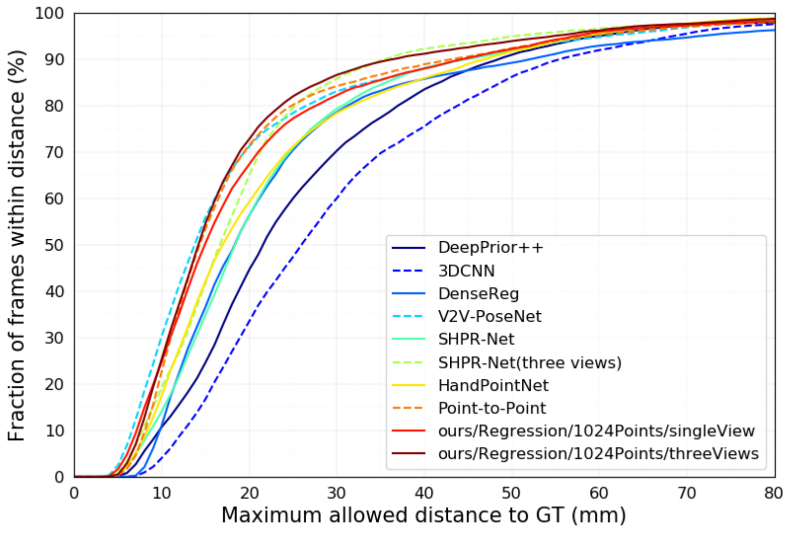


FIGURE 4.6: Comparison with state-of-the-arts on NYU [123] dataset. Top: mean errors of different joints. Down: proportion of correct frames based on different error thresholds.

NYU dataset. For the NYU dataset, we only compared to recent state-of-the-art methods after 2017. For testing the performance, only the frontal view was used. Following previous works [91][123][30], only 14 joints out of 36 joints provided were used for evaluation. For a fair comparison, we only compared to the methods trained solely on the NYU dataset without additional data. The compared methods include depth image based methods (DeepPrior++ [90], DenseReg [20]), 3D voxel based methods (3DCNN [32], V2V-PoseNet [84]) and point cloud based methods (SHPR-Net [16], HandPointNet [33], Point-to-Point [30]). The comparison is shown in Figure 4.6, where our method performs comparably good as V2V-PoseNet [84] and Point-to-Point [30], and outperforms all other methods. A closer comparison of the mean joint error value can be found in Table 4.5, where the our method

method	avg test	seen test	unseen test
Ours/regression	9.82	7.15	12.04
Ours/detection	9.93	7.18	12.22
V2V-PoseNet [84]	9.95	6.97	12.43
RCN-3D [146]	9.97	7.55	12.00
oasis [33]	11.30	8.86	13.33
THU VCLab [15]	11.70	9.15	13.83
Vanora [146]	11.91	9.55	13.89

TABLE 4.4: Comparison of our method with state-of-the-art methods on the Hands2017Challenge dataset

method	mean joint error (mm)
Ours/regression/singleView	8.99
Ours/regression/threeViews	8.35
Ours/detection	9.25
DeepPrior++ [90]	12.23
3DCNN	14.11
DenseReg [20]	10.21
V2V-PoseNet [84]	8.42
SHPR-Net [16]	10.77
SHPR-Net (three views) [16]	9.37
HandPointNet [33]	10.54
Point-to-Point [30]	9.04

TABLE 4.5: Comparison of our method with state-of-the-art methods on the NYU dataset

trained with single view is the second best, and our method trained with three views outperforms all recent state-of-the-art methods.

4.4.3 Segmentation using importance term

Besides showing the quantitative results relying on the ground-truth data, we also show some qualitative result of the segmentation using the automatically inferred importance term. As seen from Figure 4.7, the segmentation result is shown alongside the original point cloud. The samples are taken from the Hands2017 Challenge dataset. Both samples with all visible fingers and samples with different levels of self-occlusion are shown. In all cases, the fingers are clearly segmented with each other, even the fingers are twisted together. The points has no contribution to any joint has very small importance values and they are classified as background. As Figure 4.7 shows, the arm and the background points are clearly segmented in gray. Notice that the segmentation result is obtained without the ground-truth data for segmentation. This leads to a future research question about whether we can perform this method on hand-object interaction cases, where the influence of the object can be automatically removed.

	our method			V2V-PoseNet [84]	Hand3D [20]	P2P-Regression [30]
GPU	GTX1060			Titan X	Titan X	Titan Xp
time		detection	regression	285.7 ms	33.3 ms	23.9 ms
	256 points	3.5 ms	2.9 ms			
	512 points	6.9 ms	5.5 ms			
	1024 points	12.5 ms	10.7 ms			

TABLE 4.6: Comparison of runtime and hardware

4.4.4 Runtime and model size

Compared to depth image based methods, our method requires more computation time and memory storage, this limits our training to use only 512 points (batch size=32) on our hardware setting. To store the learned models, the proposed method takes 38 MB for the regression version and 44 MB for the detection version. Compared to 420MB for a 3D CNN based method [20], our model size is much smaller. For the testing stage, the runtime of our method is 12.5 ms and 10.7 ms per frame for the detection and regression version respectively, where 1024 points are used as input. When less input points is used, the runtime can be further reduced with a small performance loss. Table 4.6 shows a comparison of runtime to other state-of-the-art 3D methods [84, 20, 30]. Although the other methods all used a more powerful GPU than ours, our method require the least processing time.

4.5 Summary

This chapter proposes a novel neural network architecture, ResidualPEL, for hand pose estimation using unordered point cloud as input. The proposed method is invariant to input point order and can handle different numbers of points. Compared to previous 3D voxel based methods, our method requires less memory size. And compared to PointNet based methods, our method does not require surface normal and K-nearest-neighours information. A voting-based scheme was proposed to merge information from individual points to pose output, where the resulting importance term can be also used to segment the hand into different parts. The performance is evaluated on two datasets, where the proposed method outperforms the state-of-the-art methods on both datasets. In future work, the proposed ResidualPEL and voting scheme can be also applied to similar problem such as human pose estimation and object pose estimation.

4.6 Discussion and future works

This work has shown two strategies to predict the pose in the final layer, the "direct regression" strategy and "detection" from heatmap strategy. Initially, we have expected that the "detection" version would outperform, since it is usually the case in previous work [146]. However, the "regression" version slightly outperforms in the experiments. We suspect that the simplified 1D heat vector representation could be the cause of slightly worse performance

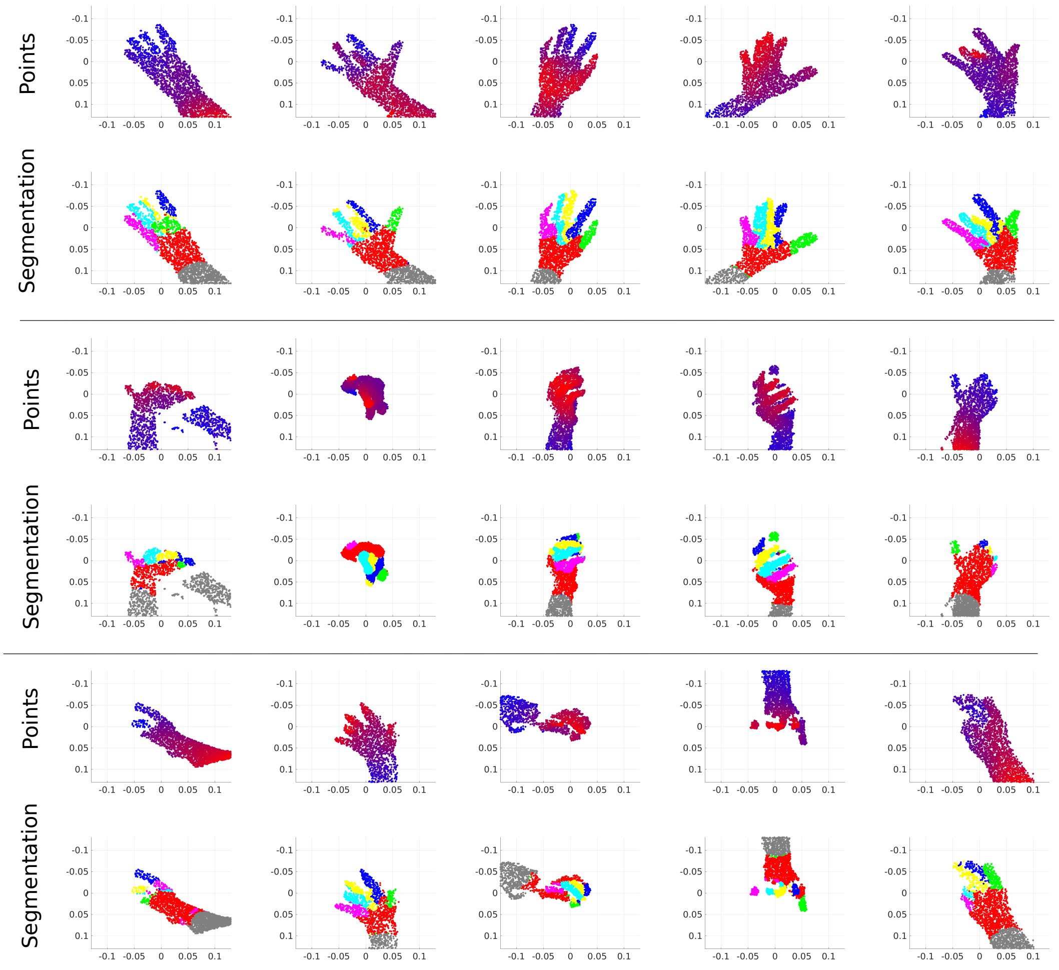


FIGURE 4.7: Segmentation results based on importance weights (best viewed in color). Points: input point cloud, color indicates depth value, blue points are more distanced and red points are more closer to the camera. Segmentation: each part of the hand is indicated with an different color, palm (red), thumb (green), index (blue), middle (yellow), ring (cyan), pinky (pink) and irrelevant points with low importance weight for all parts (gray).

for "detection" strategy. However, the 1D representation is much more efficient than the 3D counterparts. Moreover, the heatmap representation also provides an additional benefit, where it can represent distribution with multiple local maxima intuitively. This can be quite useful to predict occluded joints' position as a probability distribution instead of a single average value. Therefore, in the future work, it is worth to investigate on how to improve the performance on the "detection" strategy, e.g. to add an additional network component after the "detection" layer, to see whether there could be potential improvement.

This works proposed to use a voting layer to aggregate point-wise prediction based on their importance weights. The importance weights provides promising semi-supervised segmentation results, which is significant on its own. We didn't investigate too much for the segmentation because the main focus of this work was hand pose estimation, and the segmentation cannot be evaluated properly due to the lack of ground-truth. In the future work, we could investigate the segmentation performance on other datasets which also provide segmentation ground-truth.

The proposed method is very flexible on the size of input point cloud, it could be trained and tested using different number of input points, which is not possible with structured input data, such as RGB image, depth image or 3D voxels. This provides a benefit for online application, where the application can choose the number of input points based on current computation load adaptively. The experiment has shown that with increasing number of input points, the pose estimation accuracy also improves. However, when the input size surpass 1024, the improvement is marginal. We are currently not sure about the exact reason of the saturated performance with more points. Our guess is that the complexity of the hand appearance could already perfectly represented by 1024 points, such that more points just provide redundant information.

The proposed network structure of this work can be also potentially migrated to other similar tasks. Probably the most similar task would be human pose estimation, where both hand and human body are articulated objects with many DoFs. Apart from that, the proposed ResPEL and voting layer can be potentially used for other point cloud based tasks, such as object detection and object pose estimation.

This chapter and the previous chapter used depth image and point cloud as input respectively. We have chosen to use them because they provide 3D geometry information, which is less ambiguous than RGB images. However, many real testing scenarios can only provide RGB image as input, such as mobile phone camera or robot camera without depth sensors. In the next chapter, we will explore how to train a pipeline using both 3D and RGB data and test only using RGB data, such that the testing performance can be improved.

Chapter 5

Hand pose estimation using multiple modalities

This chapter includes the content of the following publication:
Article:

- *Linlin Yang*,* ***Shile Li*****, Dongheui Lee and Angela Yao. "Aligning Latent Spaces for 3D Hand Pose Estimation." Proceedings of the IEEE International Conference on Computer Vision (ICCV). 2019. *equal contribution*

Other contributors:

- Linlin Yang (PhD student)
- Dongheui Lee (Thesis supervisor)
- Angela Yao (Supervisor of Linlin Yang)

Author contributions

- LY and **SL** organized the study. LY and SL developed the method and wrote the code jointly, where LY is more responsible for developing the idea and **SL** is more responsible for implementing the framework and experiments. LY and **SL** wrote the main body of the article. **SL**, LY, DL and AY analyzed the results and revised the article.

This chapter presents a hand pose estimation method using multi-modality data as input. In particular, the developed method is flexible such that it can be trained using multi-modality data (RGB and point cloud) and can be deployed using only RGB data.

5.1 Motivation

Hand pose estimation plays an important role in areas such as human activity analysis, human computer interaction, and robotics. Depth-based 3D hand pose estimation methods are now highly accurate [127, 76, 141] largely due to advancements from deep learning. Despite depth sensors being more common, high-quality depth maps can still only be captured with limiting sensing range, thereby limiting the environments in which depth-based methods can be deployed. Furthermore, simple RGB cameras, as well as existing

RGB footage are still far more ubiquitous than depth cameras and depth data. As such, there is still a need for accurate RGB-based 3D hand pose estimation methods, especially from monocular viewpoints. Therefore, it is essential to have a hand pose estimation method using RGB image.

Unfortunately, RGB-based hand pose methods, in particular monocular ones, suffer from geometric ambiguities, which is not present in depth based method. To tackle this problem, previous works for monocular RGB image have relied on large amounts of training data [152, 85]. However, gains from purely increasing dataset size tend to saturate, because it is very difficult to obtain accurate ground truth pose for RGB images. Annotating 3D hand joint positions accurately is a difficult task and there is often conflicted annotation between human annotators [118]. While several methods have been developed to generate synthetized images [85], there still exists a large domain gap between synthesized and real-world data, limiting the usability of synthetic data.

Even though accurate ground truth for RGB data is hard to collect, there exists plenty of unlabelled RGB-D hand data which can be leveraged together with labelled depth maps. Cai *et. al.* [11] first proposed the use of labelled depth maps as regularization to boost RGB-based methods. Yang *et. al.* [139] introduced a disentangled representation so that viewpoint can be used as a weak label. Inspired by these works, we aim to leverage multiple modalities as weak labels in the training process to enhance the performance of RGB-based hand pose estimation.

In the proposed method, different modalities of hand data (*e.g.* RGB images, depth maps, point clouds, 3D poses, heat maps and segmentation masks) are considered, where RGB-based hand pose estimation is formulated as a cross-modal inference problem. In particular, a multi-modal variational autoencoder (VAE) is used to encode and decode these modalities. VAEs are an attractive class of deep generative models which can be learned on large-scale, high-dimensional datasets. They have been shown to capture highly complex relationships across multiple modalities [119, 126, 136] and have also been applied to RGB-based pose estimation in the past [112, 139]. However, previous works [112, 139] learn a single shared latent space and as a result must compromise on pose reconstruction accuracy.

In this chapter, instead of using a shared latent space among modalities, multiple latent spaces from individual modalities will be aligned. More specifically, we derive different objectives for three diverse modalities, namely 3D poses, point clouds, and heat maps, and explored different ways to aligning their associated hand latent spaces. While such a solution may appear more complicated than learning one shared latent space directly, this learning scheme through alignment offers more flexibility in working with non-corresponding data and also weak supervision. The resulting latent representation allows for estimating highly accurate hand poses and synthesizing realistic-looking point clouds of the hand surface, all from monocular RGB images.

Furthermore, we explore non-conventional inputs such as point clouds and heat maps for learning the latent hand space and show how they can be leveraged for improving the accuracy of an RGB-based hand pose estimation system. A side product of our framework is that we can synthesize realistic-looking point clouds of the hand from RGB images.

Experiments on two publicly available benchmarks show that the proposed method makes full use of auxiliary modalities during training and improves the accuracy of RGB pose estimates for testing. The resulting pose accuracy surpass state-of-the-art methods on monocular RGB-based hand pose estimation, with a 19% improvement on the challenging RHD dataset [152]

5.2 Related Works

One way to categorize hand pose estimation approaches is according to either generative or discriminative methods. Generative methods employ a hand model and use optimization to fit the hand model to the observations [102, 93, 122]. They usually require a good initialization; otherwise they are susceptible to getting stuck in local minima. Discriminative methods learn a direct mapping from visual observations to hand poses [123, 139, 76, 91, 152, 11]. Thanks to large-scale annotated datasets [152, 147, 123], deep learning-based discriminative methods have shown very strong performance in the hand pose estimation task.

In particular, works using depth or 3D data as input are the most accurate. Oberweger *et. al.* [91] use 2D CNNs to regress the hand pose from depth images, using a bottleneck layer to regularize the pose prediction to a certain prior distribution, which is initialized by principal component analysis. Moon *et. al.* [84] use 3D voxels as input and regress the hand pose with a 3D CNN. More recent works [76, 33] apply 3D point clouds as input and can estimate very accurate hand poses.

3D data is not always available either at training or at testing stage. Some recent works have started to explore the use of monocular RGB data. For example, Zimmermann *et. al.* [152] regress heatmaps for each hand keypoint using RGB images and then regress the 3D hand pose from these heatmaps with fully-connected layers. Mueller *et. al.* [85] follow a similar approach, but obtain the final 3D hand pose by using a kinematic skeleton model to fit the probability distribution of predicted heat maps.

More recent monocular RGB-based methods leverage depth information for training [11, 112], even though testing is done exclusively with RGB images. The proposed method also falls into this line of work. Cai *et. al.* [11] propose an additional decoder to render depth maps from corresponding poses to regularize the learning of an RGB-based pose estimation system. This architecture is essentially two independent networks with a shared hand pose layer. This shared layer however cannot leverage data without pose annotations. Spurr *et. al.* [112] propose a VAE-based method that learns a shared latent space for hand poses from both RGB and depth images. However, its alternating training strategy from the different modalities ignores the availability of corresponding data and leads to a slow convergence speed.

5.3 Strategies for mutli-modality input data

Our goal is to capture relationships between different modalities so that it is possible to obtain information of target modalities given observations of some other modalities, such that training can be performed for multi-modal data and testing can be performed using unimodal data. In the following, for the purpose of single modal testing, 4 different training strategies (S1-S4) will be explored to handle multi-modal data.

The input modality is denoted as $\mathbf{x}$, whereas the target output modality is denoted as $\mathbf{y}$. Any additional corresponding modality is denoted as $\mathbf{w}_1, \mathbf{w}_2, ...$, for the sake of simplicity, we firstly only consider a single additional modality $\mathbf{w}$. In the case of hand pose estimation, $\mathbf{x}$ is the RGB image, $\mathbf{y}$ is the 3D hand pose and $\mathbf{w}$ is the corresponding point cloud data.

5.3.1 S1: Baseline with cross-modal VAE

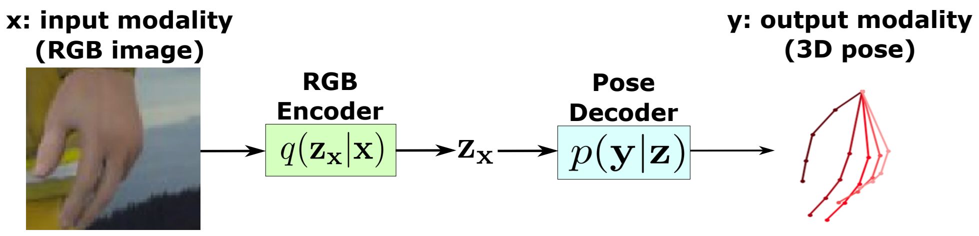


FIGURE 5.1: Strategy1:Baseline with cross-modal VAE

The simplest baseline is to ignore the availability of $\mathbf{w}$ and construct a cross-modal VAE framework to regress (Figure 5.1). Given data sample $\mathbf{x}$ from the input modality, the system aims to estimate its corresponding target value $\mathbf{y}$ in a target modality. It comprises of an encoder part and a decoder part:

$$Enc: \ \mathbf{z} \sim Enc(\mathbf{x}) = q(\mathbf{z}|\mathbf{x}), \ \ Dec: \ \mathbf{y} \sim Dec(\mathbf{z}) = p(\mathbf{y}|\mathbf{z})$$

The VAE reconstructs $\mathbf{y}$ to be as close as possible to the input $\mathbf{x}$, at the same time a prior is enforce on the latent space distribution $p(\mathbf{z})$, where an Gaussian prior is used. Therefore, the total loss function for the VAE is:

$$\begin{aligned} L_{S1} &= L_{recons}^{pose} + \beta L_{prior}, \\ L_{recons}^{pose} &= -E_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{y}|\mathbf{z})], \\ L_{prior} &= D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})), \end{aligned} \tag{5.1}$$

where D_{KL} is the is the Kullback-Leibler divergence and β is a weighting factor to tune the importance between reconstruction accuracy and prior distribution. This baseline simply encodes the RGB image to the latent space and reconstruct 3D pose out of it, where available point cloud information is totally ignored.

5.3.2 S2: Mutli-task learning

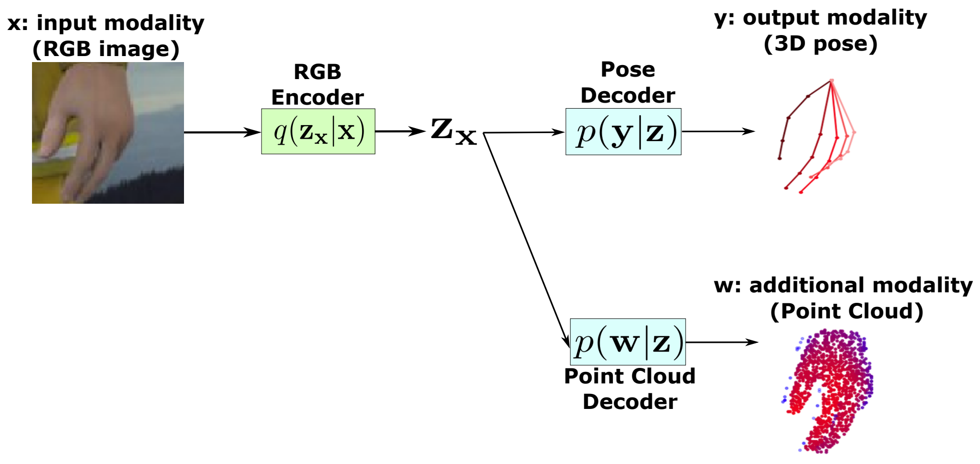


FIGURE 5.2: Strategy2: Mutli-task learning

The second strategy is to take the additional modality as a multi-task learning target (Figure 5.2) Using the latent variable $\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})$, both 3D pose $\mathbf{y}$ and point cloud $\mathbf{w}$ will be reconstructed. The total loss then becomes:

$$L_{S2} = L^{pose}_{recons} + L^{cloud}_{recons} + \beta L_{prior}, \tag{5.2}$$

where details of point cloud reconstruction loss will be introduced later (Equation 5.4.2). Compare to the baseline (Figure 5.1), the latent space contains richer information with reconstructing point cloud using the shared latent space.

5.3.3 S3: Alignment to multi-modal latent space

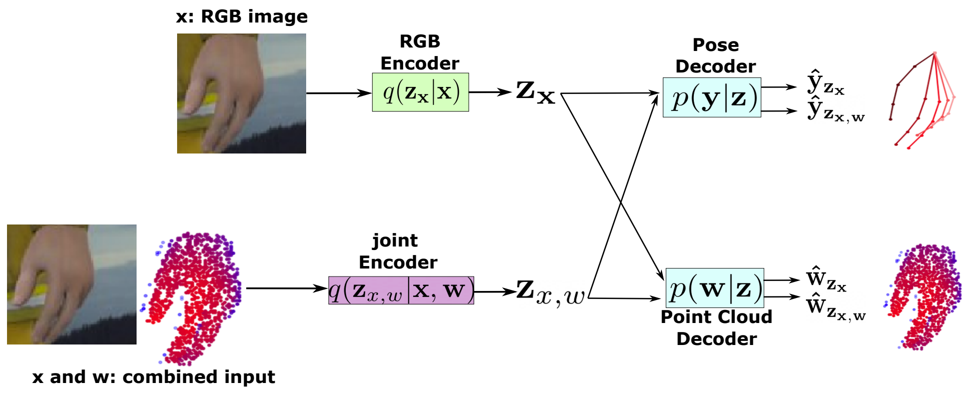

x: RGB image
RGB
Encoder
$q(\mathbf{z_x}|\mathbf{x})$
$\mathbf{z_x}$
Pose
Decoder
$p(\mathbf{y}|\mathbf{z})$
$\hat{\mathbf{y}}_{\mathbf{z_x}}$
$\hat{\mathbf{y}}_{\mathbf{z_{x,w}}}$
joint
Encoder
$q(\mathbf{z}_{x,w}|\mathbf{x},\mathbf{w})$
$\mathbf{z}_{x,w}$
$p(\mathbf{w}|\mathbf{z})$
$\hat{\mathbf{w}}_{\mathbf{z_x}}$
$\hat{\mathbf{w}}_{\mathbf{z_{x,w}}}$
Point Cloud
Decoder
x and w: combined input

FIGURE 5.3: Strategy3: Alignment to multi-modal latent space

We consider to learn a multi-modal latent space using concatenated input $(\mathbf{x}, \mathbf{w})$, obtaining a multi-modal latent variable: $\mathbf{z_{x,w}} \sim q(\mathbf{z_{x,w}}|\mathbf{x}, \mathbf{w})$ and then reconstruct the 3D pose out of it: $\mathbf{y_{x,w}} \sim p(\mathbf{y_{x,w}}|\mathbf{z_{x,w}})$. The multi-modal latent space contains richer information than using unimodal input and thus can produce more accurate pose result. However, for testing, only $\mathbf{x}$ is available, which hinders the usability of the trained model. Therefore, the third strategy (Figure 5.3) proposes to train a unimodal cross-VAE simultaneously and then align the two latent spaces by reducing the KL divergence loss between them:

$$L_{align} = D_{KL}(q(\mathbf{z}_{\mathbf{x},\mathbf{w}}|\mathbf{x},\mathbf{w})||q(\mathbf{z}_{\mathbf{x}}|\mathbf{x})) \tag{5.3}$$

Then the total loss becomes:

$$\begin{aligned} L_{S3} = & L^{pose}_{recons}(\mathbf{y}_{\mathbf{x}}) + L^{pose}_{recons}(\mathbf{w}_{\mathbf{x},\mathbf{w}}) \\ & + L^{cloud}_{recons}(\mathbf{w}_{\mathbf{x}}) + L^{cloud}_{recons}(\mathbf{w}_{\mathbf{x},\mathbf{w}}) \\ & + \beta' L_{align} \end{aligned} \tag{5.4}$$

This strategy forces the unimodal latent distribution $p(\mathbf{z}_{\mathbf{x}})$ to be as close as possible to the multi-modal latent distribution $p(\mathbf{z}_{\mathbf{x},\mathbf{w}})$, thus improves the quality of the unimodal path. However, this strategy suffers from two limitations. Firstly, as the number of additional modalities increases, the combined input $\mathbf{x}, \mathbf{w}$ becomes difficult to learn. Secondly, it cannot leverage on the data pair $(\mathbf{w}, \mathbf{y})$, which is a pity because a lot of large, accurate dataset only contains depth image data. To overcome these limitations, the fourth strategy uses the Gaussian product as an alternative form of alignment.

5.3.4 S4: Alignment using Gaussian product

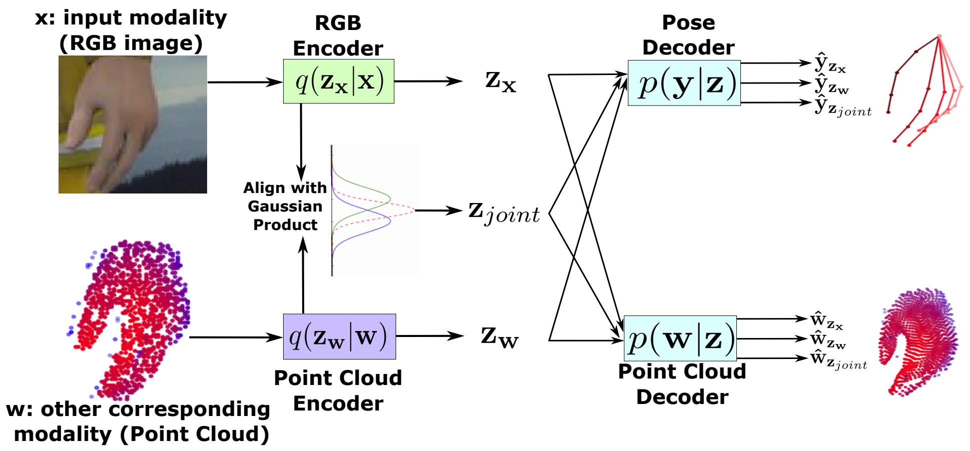

x: input modality (RGB image)
RGB Encoder
$q(\mathbf{z}_\mathbf{x}|\mathbf{x})$
$\mathbf{z}_\mathbf{x}$
Pose Decoder
$p(\mathbf{y}|\mathbf{z})$
$\hat{\mathbf{y}}_{\mathbf{z}_\mathbf{x}}$
$\hat{\mathbf{y}}_{\mathbf{z}_\mathbf{w}}$
$\hat{\mathbf{y}}_{\mathbf{z}_{joint}}$
Align with Gaussian Product
$\mathbf{z}_{joint}$
$q(\mathbf{z}_\mathbf{w}|\mathbf{w})$
$\mathbf{z}_\mathbf{w}$
$p(\mathbf{w}|\mathbf{z})$
$\hat{\mathbf{w}}_{\mathbf{z}_\mathbf{x}}$
$\hat{\mathbf{w}}_{\mathbf{z}_\mathbf{w}}$
$\hat{\mathbf{w}}_{\mathbf{z}_{joint}}$
Point Cloud Encoder
Point Cloud Decoder
w: other corresponding modality (Point Cloud)

FIGURE 5.4: Strategy4: Alignment using Gaussian product

It was proven in [136] that the joint posterior is proportional to the product of individual posteriors, *i.e.* $q(\mathbf{z}|\mathbf{x},\mathbf{w}) \propto p(\mathbf{z})q(\mathbf{z}|\mathbf{x})q(\mathbf{z}|\mathbf{w})$. To that end, we can estimate the joint latent representation from unimodal latent representations (Figure 5.4), in which each input modality is encoded separately to $\mathbf{z}_\mathbf{x}$ and $\mathbf{z}_\mathbf{w}$, and then the joint latent space is constructed using Gaussian product

$$\mathbf{z}_{joint} = \text{GProd}(\mathbf{z}_\mathbf{x}, \mathbf{z}_\mathbf{w}),$$

where the product of two Gaussian experts is also Gaussian with mean μ and variance σ. Suppose that $q(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mu_1, \sigma_1)$ and $q(\mathbf{z}|\mathbf{w}_1) = \mathcal{N}(\mu_2, \sigma_2)$. The product of the two is also Gaussian with mean μ and variance σ where

$$\mu = (\mu_1 T_1 + \mu_2 T_2)/(T_1 + T_2), \qquad \text{and}$$
$$\sigma = 1/(T_1 + T_2), \quad \text{where } T_1 = 1/\sigma_1, T_2 = 1/\sigma_2.$$

Note that all operations in GProd$(\cdot)$ are element-wise. In this way, we can build a connection between $\mathbf{z}_{\text{joint}}$ and $\mathbf{z}_\mathbf{x}, \mathbf{z}_\mathbf{w}$, forcing them all into one joint latent space.

These three latent variables will then use shared decoders to reconstruct $\mathbf{y}$ and $\mathbf{w}$, resulting in the loss function:

$$\begin{aligned} L_{S4} = {} & L^{pose}_{recons}(\mathbf{y}_{\mathbf{z}_\mathbf{x}}) + L^{pose}_{recons}(\mathbf{y}_{\mathbf{z}_\mathbf{w}}) + L^{pose}_{recons}(\mathbf{y}_{\mathbf{z}_{\mathbf{x},\mathbf{w}}}) \\ & + L^{cloud}_{recons}(\mathbf{w}_{\mathbf{z}_\mathbf{x}}) + L^{cloud}_{recons}(\mathbf{w}_{\mathbf{z}_\mathbf{w}}) + L^{cloud}_{recons}(\mathbf{w}_{\mathbf{z}_{\mathbf{x},\mathbf{w}}}) \\ & + \beta D_{KL}(q(\mathbf{z}|\mathbf{x})||\mathcal{N}(0,1)), \end{aligned} \tag{5.5}$$

This strategy is more flexible than Strategy 3, because the encoders of different modalities can be trained individually, even using data from different datasets, while for S3, the encoder for multi-modal input must be trained on the complete $(\mathbf{x}, \mathbf{w})$ pairs.

Notation for hand pose estimation

In the context of RGB-based hand pose estimation, $\mathbf{x}$ represents RGB images and $\mathbf{y}$ represents 3D hand poses. Other modalities like heatmaps, depth maps, point clouds and segmentation masks can be used as $\mathbf{w}$ during training to improve the learning of the latent space and thereby leading to more accurate hand pose estimates from RGB inputs. In this paper, we use point clouds (C) and heat maps (H) as additional modalities $\mathbf{w}$ to improve the cross modal inference of RGB (R) to 3D poses (P). In the following, we use the format "A2B" to represent the estimation of target modality "B" from input modality "A" during training. For example, R2CHP represents the estimation of point clouds, heat maps and 3D poses from RGB input. Note that unless indicated otherwise, the test settings use RGB images as the input modality and 3D hand poses as the output modality.

5.4 Details of implementation

5.4.1 Data pre-processing and augmentation

From the RGB image, the region containing hand is cropped from ground truth masks and resized to 256×256. The corresponding region in the depth image is converted to point clouds using the provided camera intrinsic parameters. For each training step, a different set of 256 points are randomly sampled as training input.

Viewpoint correction. After cropping the hand from the RGB image, the center of the hand in the image moves from some arbitrary coordinates to the center of the image. As such, the 3D hand pose and associated point cloud must be rotated such that the viewing angle towards the hand aligns with the optical axis. As indicated in Chapter 4.3.1, this correction is necessary to remove the many-to-one observation-pose pairings. We follow the approach given in Chapter 4.3.1 with slight modification. Assuming that the crop's center coordinates on the original image is $[u_c, v_c]$, the rotation matrix $\mathbf{R}_{vc} \in \mathbb{R}^{3\times3}$ to correct for the viewing angle can be obtained as follows:

$$\begin{aligned} \alpha_y &= \text{atan2}(u_c - o_x, f), \\ \widetilde{\mathbf{c}} &= \mathbf{R}_y(-\alpha_y) \cdot [u_c - o_x, v_c - o_y, f]^T, \\ \alpha_x &= atan2(\widetilde{\mathbf{c}}_2, \widetilde{\mathbf{c}}_3), \\ \mathbf{R}_{vc} &= \mathbf{R}_y(-\alpha_y) \cdot \boldsymbol{R}_x(\alpha_x), \end{aligned} \tag{5.6}$$

whereas f is the camera focal length, o_x, o_y are the camera center coordinates, $\widetilde{\mathbf{c}}$ is an intermediate result and $\alpha_{x,y}$s are rotation angles around corresponding axis. After viewpoint correction, the 3D poses and point clouds are subtracted with the hand's centroid point and normalized to a canonical size.

Data augmentation was performed online during training. The images are scaled randomly between $[1, 1.2]$, translated $[-20, 20]$ pixels and rotated $[-\pi, \pi]$ around the camera view axis. Furthermore, the hue of the image is randomly adjusted in the range of [-0.1, +0.1]. The point clouds are rotated randomly around the camera view axis and the 3D pose labels are also rotated accordingly.

5.4.2 Encoder and decoder modules

The proposed method in strategy 4 (S4) is highly flexible and can integrate many different modalities to construct a common latent space. In total, the following modules are learned for S4, encoders for RGB images and point clouds, decoders for 3D hand poses, point clouds and heat maps of the 2D hand key points on the RGB image. We choose to convert the 2.5D depth information as 3D point clouds instead of standard depth maps, due to its superior performance in hand pose estimation, as shown in previous works [76, 16, 30] and Chapter 4. Heat maps are chosen as a third modality for decoding to encourage convergence of the RGB encoder, since the heat maps are closely related to activation areas on the RGB images.

For encoding RGB images, Resnet-18 from [41] and two additional fully connected layers are used to predict the mean and variance vector of the latent variable. For encoding point clouds, the ResPEL network [76] is employed, which is an learning architecture that takes unordered point cloud as input. While we use same number of PEL layers as in Chapter 4, the number of hidden units are reduced by half to ease the computational load.

To decode the heatmaps, the decoder architecture of the DC-GAN [104] is used. The loss function used for the heatmaps is the L2 loss function of pixel-wise difference between prediction and ground-truth:

$$L_{\text{heat}} = \sum_{j=1}^{J} ||\hat{H}_j - H_j||, \tag{5.7}$$

whereas H_j is the ground-truth heatmap for the j-th hand keypoint and $\hat{H}_j$ is the prediction. For decoding point clouds, we follow the FoldingNet architecture [141] and try to reconstruct a point cloud representing the visible surface of the hand. To learn the decoder, we use two different loss terms based on the Chamfer distance and Earth Mover's distance (EMD). The Chamfer distance is the sum of the Euclidean distance between points from one set and its closest point in the other set and vice versa:

$$L_{\text{Chamfer}} = \frac{1}{|P|} \sum_{p \in P} \min_{\hat{p} \in \hat{P}} ||\hat{p} - p|| + \frac{1}{|\hat{P}|} \sum_{\hat{p} \in \hat{P}} \min_{p \in P} ||\hat{p} - p||. \tag{5.8}$$

For the Earth Mover's distance, one-to-one bijective correspondences are established between two point clouds, and the Euclidean distances between them are summed:

$$L_{\text{EMD}} = \min_{\phi : P \to \hat{P}} \frac{1}{|P|} \sum_{p \in P} ||p - \phi(p)||, \tag{5.9}$$

In both Eq. 5.8 and 5.9, $\hat{P}, P \in \mathbb{R}^3$ represent the predicted point clouds and the ground truth point clouds respectively and the number of points in both clouds are 256.

The point cloud reconstruction loss is then the sum: $\mathrm{L}^{cloud}_{recons} = L_{\text{Chamfer}} + L_{\text{EMD}}$.

The decoder for 3D pose consists of 4 fully-connected layers with 128 hidden units for each layer. To learn the pose decoder, we use an L2 loss:

$$\mathcal{L}^{pose}_{recons} = ||\hat{y} - y||, \tag{5.10}$$

where $\hat{y}, y$ are the predicted and the ground truth hand poses describing the 3D locations of 21 keypoints.

5.5 Experimental results

In the experiments, the dimensionality of latent variable $\mathbf{z}$ is set to 64, λ_{heat} to 0.01, λ_{cloud} to 1 for all cases and β' to 1 for Equation 5.4. The proposed method is implemented with Tensorflow. For training, an Adam optimizer is used with an initial learning rate of 10^{-4} and a batch size of 32. The learning rate is lowered by a factor of 10 two times after convergence. The value of β is annealed from 10^{-5} to 10^{-3}.

5.5.1 Datasets and evaluation metrics

The proposed method is evaluated on two publicly available datasets: the Rendered Hand Pose Dataset (RHD) [152] and the Stereo Hand Pose Tracking Benchmark (STB) [149].

RHD is a synthesized dataset of rendered hand images with 320×320 resolution from 20 characters performing 39 actions. It is composed of 41238 samples for training and 2728 samples for testing. For each RGB image, a corresponding depth map, segmentation mask, and 3D hand pose are provided. The dataset is highly challenging because of the diverse visual scenery, illumination, and noise.

STB contains videos of a single person's left hand in front of six different real-world backgrounds. The dataset provides stereo images and RGB-D pairs with 640×480 resolution and 3D hand pose annotations. Each of the 12 sequences in the dataset contains 1500 frames. To make the 3D pose annotations consistent for RHD, we follow [152, 11] and modify the palm joint in STB to the wrist point. Similar to [152, 11, 112, 139], we use 10 sequences for training and the other 2 for testing.

To evaluate the accuracy of the estimated hand poses, two common metrics are used: mean end-point-error (EPE) and area under the curve (AUC) on the percentage of correct keypoints (PCK) curve. EPE is measured as the average Euclidean distance between predicted and ground-truth hand joints, whereas AUC represents the percentage of predicted keypoints that fall within certain error thresholds compared with ground-truth poses. To compare with the state-of-the-art methods in a fair way, we follow the similar condition used in [112, 49, 11, 139] to assume that the global hand scale and the hand root position are known in the experimental evaluations, where the middle finger's base position is set as the root of the hand.

5.5.2 Qualitative results

Using the flexible design, the networks are trained by exploiting all the available modalities and tested using only limited modalities. Some qualitative examples of poses and point clouds decoded from the $\mathbf{z}_{rgb}$ are shown in Figure 5.5. The 3D poses and point clouds can be successfully reconstructed from the same latent variable $\mathbf{z}$. The reconstructed point clouds' surfaces are smoother than the original inputs, since the inputs are sub-sampled from raw sensor data, while the reconstructed point clouds hold some structured properties from the FoldingNet decoder.

We further evaluate the ability of trained model to synthesize hand poses and point clouds. From two RGB images of the hand, corresponding latent variables $\mathbf{z}_{1,2}$ are estimated, and then multiple intermediate points are sampled by linearly interpolating between these two. Corresponding 3D hand poses and point clouds are then reconstructed using the learned decoders as shown in Figure. 5.6. The learned latent space reconstructs a smooth and realistic transition between different poses, with changes in both global rotations and local finger configurations.

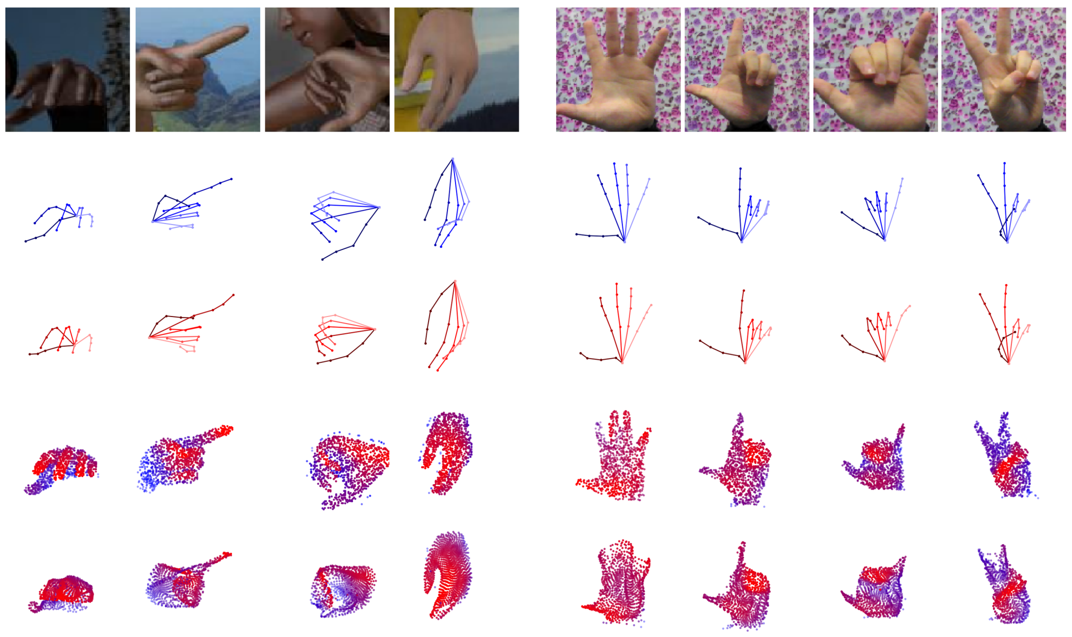


FIGURE 5.5: 3D pose estimation and point cloud reconstruction for RHD (left) and STB (right) dataset. From top to bottom: RGB images, ground-truth poses in blue, estimated poses from $\mathbf{z}_{\text{rgb}}$ in red, ground-truth point clouds, reconstructed point clouds from $\mathbf{z}_{\text{rgb}}$. The color for point clouds decodes the depth information, closer points are more red and further points are more blue. Note that the ground-truth point clouds are not used for inference, they are shown here only for the comparison purpose.

5.5.3 RGB 3D Hand Pose Estimation

Note that even though our network is trained with multiple modalities, the results provided here are based only in monocular RGB inputs.

Training Strategy. We first compare different training strategies (S) in Table 5.1:

- S1. Baseline method to only use RGB-pose pairs for training.
- S2. Training with multiple decoders, where the latent variables $\mathbf{z}_{\text{rgb}}$ reconstruct more modalities (heatmaps and point clouds) besides poses.
- S3. Training with an additional encoder for point clouds, where the different latent variables are aligned with Equation 5.4.
- S4. The alignment method in S3 is changed to the Gaussian product.

The comparison results with AUC metric are shown in Figure 5.7. Comparing S1 to the other strategies, we observe that the baseline performance can be improved by training with increasing number of additional encoders or decoders. Comparing S4 to S3, the alignment with the Gaussian product outperforms the intuitive KL-divergence alignment method by capturing a better joint posterior of different input modalities.

Furthermore, we emphasize the necessity of viewpoint correction. We applied both view corrected and uncorrected data for training the baseline

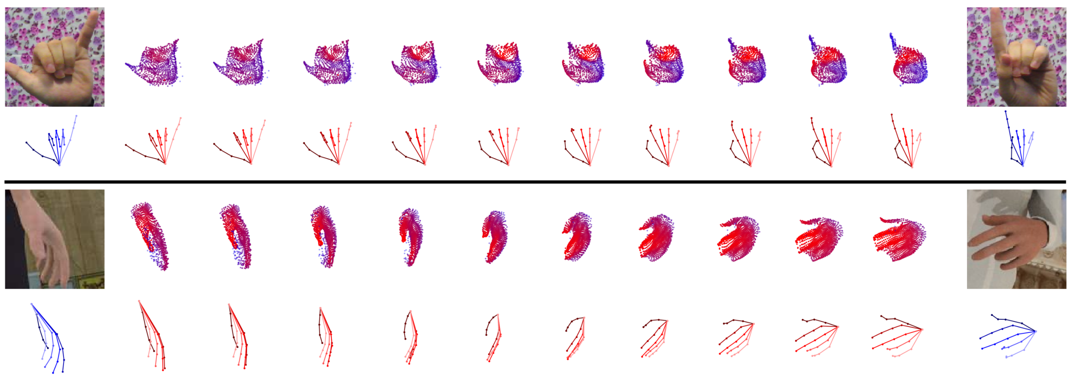


FIGURE 5.6: Latent space interpolation. Two examples of reconstructing point clouds and hand poses from the latent space. The most left and most right column are RGB images and their corresponding ground-truth poses. Other columns are generated point clouds and poses when interpolating linearly on the latent space.

Strategy	Encoder	Decoder	Mean EPE [mm]
S1	R	P	16.61
S2	R	H+P	16.10
	R	C+P	15.91
	R	C+H+P	15.49
S3	R+C	C+H+P	14.93
S4	R+C	C+H+P	**13.14**

TABLE 5.1: Comparison of different training strategies on the RHD dataset. The mean EPE values are obtained from monocular RGB images. (R: RGB, C: point cloud, P: pose, H: heatmap). Poses estimated from monocular RGB images can be improved by increasing number of different encoders and decoders during training.

strategy “R2P” (S1). The difference can be seen from Figure 5.7, where the view corrected data clearly improves the AUC metric.

Comparison to state-of-the-art. In Table 5.2, we compare the EPE of our method with VAE-based methods [112, 139] which are most related to our method as well as other state-of-the-art [152, 49]. On both datasets, the proposed method achieves the best results, including an impressive 1.61mm or 19% improvement on the STB dataset.

We also compare the PCK curve of our approach with other state-of-the-art methods [112, 139, 152, 49, 85, 95] in Figure 5.8 and Figure 5.9. For both datasets, our method achieves the highest AUC value on the 3D PCK. We marginally outperform the state-of-the-art [49, 11] on the STB dataset, whereas on the RHD dataset, we surpass all reported methods to date [152, 139, 11, 112] with a significant margin. Note that the STB dataset contains much less variation in hand poses and backgrounds than the RHD dataset and that performance by state-of-the-art methods on STB has become saturated. As such, there is little room for improvement on STB, whereas the benefits of our method is more visible on the RHD dataset.

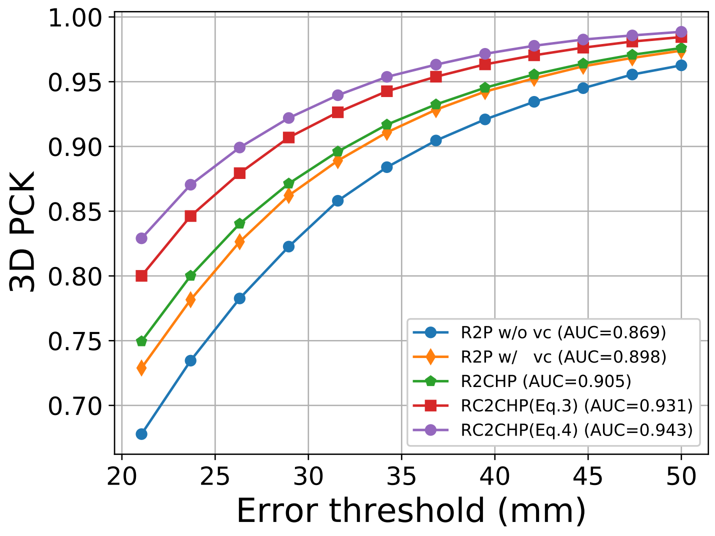


FIGURE 5.7: Comparisons of 3D PCK results of different strategies on RHD dataset. "vc" stands for "view correction"

	Method	RHD	STB
VAE-based	Spurr *et. al.* [112]	19.73	8.56
	Yang *et. al.* [139]	19.95	8.66
	Ours	**13.14**	**7.05**
Others	Z&B [152]	30.42	8.68
	Iqbal *et. al.* [49]	13.41	\

TABLE 5.2: Comparison to state-of-the-art on the RHD and STB with mean EPE [mm]. Ours refers to S4 in Table 5.1 (RC2CHP).

Weakly-supervised learning. Thanks to flexibility of the proposed method, (surface) point clouds can be also used as "weak" labels for unlabelled data to aid the training process. We tested our method under a weakly-supervised setting on the RHD dataset, where we sample the first m% samples as labelled data (including RGB, point clouds and 3D poses) and the rest as unlabelled data (including RGB, point clouds) by discarding 3D pose labels. We compare the supervised setting with the weakly-supervised setting for the "RC2CHP" networks (S4 in Table 5.1). In the supervised training setting, the networks are trained with only m% samples, In the weakly-supervised setting, besides fully supervised training on m% data, we also train the "RC2C" sub-parts with the rest (100-m)% samples simultaneously. The percentage of labelled data is varied from 5% to 100% to compare the mean EPE between supervised and weakly-supervised settings. From Figure 5.10 we can see that our method makes full usage of additional unlabelled information, where the improvement is up to 6%.

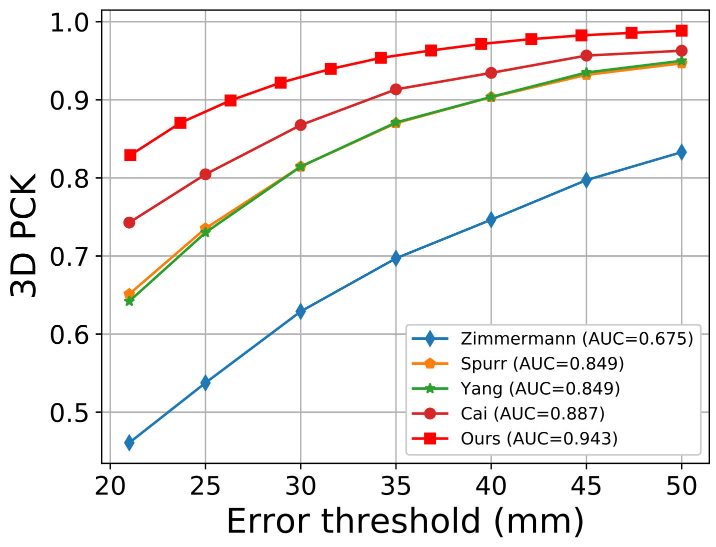


FIGURE 5.8: AUC: Comparison to state-of-the-art methods on the RHD dataset. Ours refers to S4 in Table 1 (RC2CHP).

5.6 Summary

In this chapter, for the purpose of hand pose estimation, we have explored how to utilize multi-modal data to boost the performance of unimodal testing, in which we show different ways of aligning associated latent spaces of different input modalities with a joint one. The developed final strategy (S4) is highly flexible, it can exploit different modalities as prior knowledge to improve the performance of RGB-based hand pose estimation as well as leverage weakly labelled data. Experiments on two publicly available datasets demonstrate that the alignment approach using Gaussian product (S4) outperform previous state-of-the-art methods. Moreover, the model size and runtime of the desgined architecture is kept the same as other VAE-based hand estimation methods at test time.

5.7 Discussion and future works

In this chapter, we have investigated on several strategies for multi-modal training and unimodal testing. The final strategy (S4) uses Gaussian product for the latent vectors, showing the best performance. The S4 is highly flexible to be trained using arbitrary numbers of input modalities and can be tested using only a subset or a single input modality. For training S4, we utilize corresponding data pairs to align the latent space. With the aid of corresponding data, S4 can converge much faster and get better joint representations. If S4 uses only non-corresponding data in a semi-/weakly-supervised setting, this deteriorates the quality of the latent alignment, where pose accuracy is reduced by 2mm. Corresponding data is generally required for joint objective training. This is not too much of a restriction in practice because there are several existing multi-modal datasets for hand pose estimation (e.g. NYU,

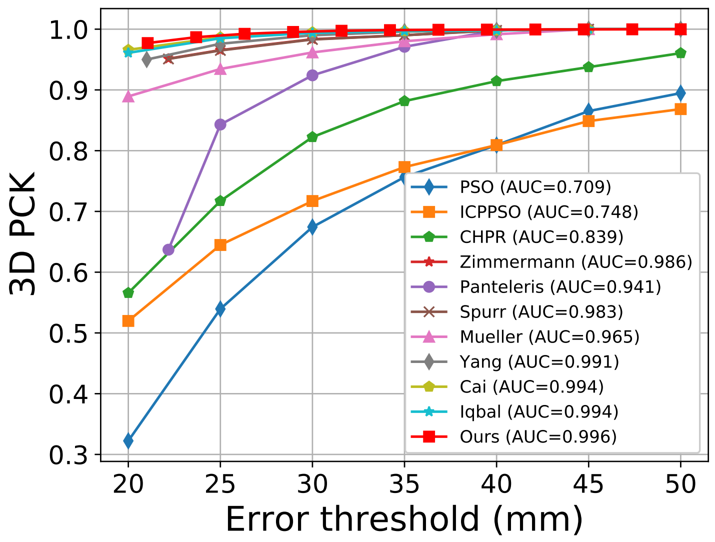


FIGURE 5.9: AUC: Comparison to state-of-the-art methods on the STB dataset. Ours refers to S4 in Table 1 (RC2CHP).

SynthHands, HandDB) and synthetic multimodal data is easy to obtain. If corresponding data is limited, different encoders can be pre-trained on large amounts of non-corresponding data separately and then aligned using limited amount of corresponding data. The semi-/weakly-supervised setting only needs small number of RGBD-pose pairs to align the latent space and the majority of training relies only on RGB-depth pairs (by setting the weighting of the pose estimation loss to zero), which can be easily recorded with any RGBD sensor. Compared to our baseline S1 which is fully supervised, we are more accurate with only 25% of labelled RGBD-pose pairs.

The proposed method focused on multimodal training and unimodal testing scenarios and achieved good performance. However, if multimodal input data is also available for testing cases, there could be more ways to utilize multi-modal data. Taking the example of merging RGB and point cloud, different ways of for merging exist. E.g., for object pose estimation, Wang et. al. [129] firstly compute pixel-wise features for RGB image and point-wise features for 3D point cloud, then use camera projection model to associate each RGB pixel with a 3D point and concatenate the features from different modalities for further processing. Another example from object detection: in a region proposal network, Ku et. al. [66] merges anchors generated from different modalities to generate more reliable 3D object proposals. In future works, these multimodal fusion ideas could be also applied for our hand pose estimation tasks.

Another future work direction is to further boost the performance of weakly supervised learning or eventually achieve fully self-supervised learning for hand pose estimation. Although the proposed S4 enables weakly supervised training, a certain amount of labeled data is still required. The labelling of hand pose data is a complex process and it always requires human intervention in the end to ensure accurate labelling. Very recent works start to use the differentiable rendering technique [77, 55] to perform self-supervised

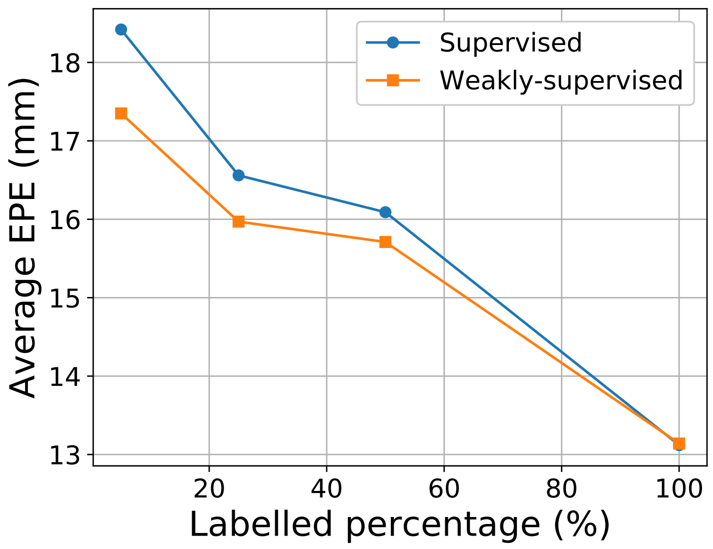


FIGURE 5.10: Mean EPE of our model on the weakly-supervised setting. The proposed method makes full use of unlabelled data, as the weakly-supervised setting performs almost as well as the supervised one.

learning. The main idea is to estimate a hand mesh model using input image and render a reconstructed input image. Then the image reconstruction loss can be used for guiding self-supervised learning. This removes the requirement of labelled input hand images. This could be a very interesting future research direction, since unlimited data can be produced for self-supervised learning, thus the performance of trained model will not be limited to the size of the dataset anymore, but will be more upper bounded by the network complexity.

In this chapter, despite the usage of multi-modal data, the method presented in this chapter only considers clean hand pose estimation, where hand-object interaction cases can not be properly handled. However, in the programming by demonstration application, the detailed hand pose observation is especially required when the hand is interacting with other objects. The inclusion of object will make the hand pose estimation task much harder, because the object creates severe occlusion to the hand. In the next chapter, the hand object interaction cases will be investigated.

Chapter 6

Hand pose estimation for hand-object interaction cases

This chapter includes the content of the following publication:
Article:

- *Shile Li*, Haojie Wang* and Dongheui Lee, Hand Pose Estimation for Hand-Object Interaction Cases using Augmented Autoencoder, in Proc. IEEE International Conference on Robotics and Automation (ICRA), 2020. *equal contribution*

Other contributors:

- Haojie Wang (Master student)
- Dongheui Lee (Thesis supervisor)

Author contributions

- **SL** and HW developed the method and wrote the code jointly, where **SL** is more responsible for designing the method and experiment setup and HW is more responsible for implementation. **SL** and HW wrote the main body of the article. **SL**, HW, DL analyzed the results and revised the article.

Hand pose estimation with objects is challenging due to object occlusion and the lack of large annotated datasets. To tackle these issues, we propose an Augmented Autoencoder based deep learning method using augmented clean hand data. Our method takes 3D point cloud of a hand with an augmented object as input and encodes the input to latent representation of the hand. From the latent representation, our method decodes 3D hand pose and we propose to use an auxiliary point cloud decoder to assist the formation of the latent space. Through quantitative and qualitative evaluation on both synthetic dataset and real captured data containing objects, we demonstrate promising performance for hand pose estimation with objects, even using only a small number of annotated hand-object samples.

6.1 Motivation

Hand pose estimation plays an important role in many human-robot interaction tasks, such as teleoperation, virtual/augmented reality and robot imitation learning [53, 97, 3, 12]. These applications require real-time and accurate hand pose estimation in 3D space. Recently, deep learning based methods have made significant progress in this area, which can be categorized to depth-based approaches [146, 75, 33, 34, 31, 127, 128, 91] and RGB-based approaches[10, 95, 140]. Despite the success of these methods, they rarely concern the hand-object interaction cases. These methods typically fail in manipulation tasks because of the occlusions caused by the grasped object.

Several previous works took object occlusion problems for hand pose estimation task into consideration. The majority are tracking based approaches [67, 39, 94]. The robust performance of these methods relies on tracking algorithms to exploit the temporal constraints between consecutive frames in input sequence. However, a good initialization is required for the first frame, and sometimes tracking drift happens. Other conventional methods [94, 6, 125] resort to multi-camera setups to reduce the influence of object occlusions from multiple viewpoints. However, it is expensive and complex to set up a synchronous and calibrated system with multiple sensors.

Currently, hand pose estimation for hand-object interaction cases is limited by existing available datasets. Public large-scale datasets with reliable 3D ground-truth annotations are lacking due to the complexity of annotating 3D hand pose. Although some large-scale datasets, like Hands2017 Challenge [147], have accurate 3D pose annotations, they are entirely composed from clean hand samples. Therefore, it is worth considering how to utilize existing clean hand datasets for hand-object cases.

In this following, a novel deep learning framework will be presented, it uses Augmented Autoencoder to tackle hand-object interaction problem in hand pose estimation tasks. The proposed method takes 3D occluded hand point cloud as input, which is obtained by a random data augmentation process from clean hand samples. The encoder extracts point-wise features and fuses them to a latent vector. Addressing the problem of object occlusion in hand-object interaction cases, an auxiliary decoder is used to reconstruct the clean hand point cloud from the latent vector (Figure 6.1), and another decoder estimates simultaneously the 3D hand pose from the same latent vector.

6.2 Related works

In the following, hand pose estimation works on both clean hand and hand-object interaction cases will be firstly reviewed. Then the backbone of the proposed framework, Augmented Autoencoder and FoldingNet will be briefly introduced.

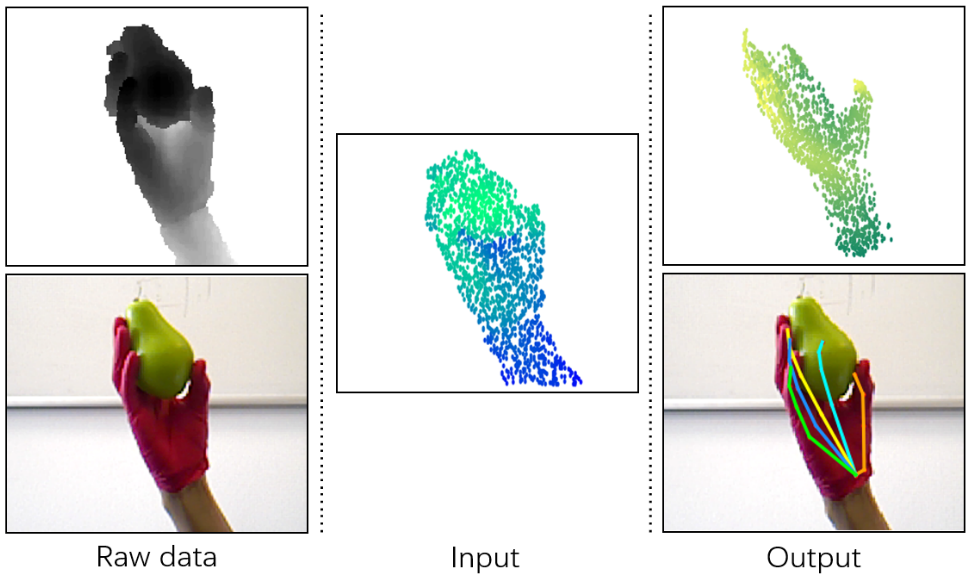


FIGURE 6.1: The raw data are captured from a RGB-D camera. Only the depth image is used to acquire the input point cloud. The RGB image is only used for visualization. For the output, besides the predicted pose, a clean hand is simultaneously reconstructed. (Brightness in point cloud indicates depth, where darker denotes larger distance to the camera.)

Clean hand pose estimation

In the past few years, a lot of 2D deep learning based methods for clean hand pose estimation has been proposed [128, 91, 127, 152, 112, 139]. In particular, 2D depth image based methods demonstrate robust performance. Oberweger et. al. [91] use 2D CNN to estimate the hand pose from the image features, where they introduce a bottleneck layer to force the predicted pose obey certain prior distribution. Wan et. al. [128] estimate hand pose with a proposed pose parameterization strategy, which decomposes the pose parameters into a set of per-pixel estimations, i.e. 2D/3D heat maps and unit 3D directional vector fields, to leverage the 2D and 3D properties of the input depth map.

Recently, 3D deep learning methods gain more attention due to the abundant information in the input data [75, 33, 101, 34, 31]. Ge et. al. [33] present a PointNet [101] based approach that directly takes point cloud as input to regress 3D hand joint locations. In order to handle variations of hand global orientations, they introduce the oriented bounding box (OBB) to normalize the hand point clouds. Li et. al. [75] propose a point-to-pose voting based residual permutation equivariant network for hand pose estimation task. Without the need of complex preprocessing steps, their method takes unordered 3D point cloud as input to compute point-wise features and through weighted fusion to obtain final hand pose estimates. Despite their good performance on hand pose estimation, they commonly ignore the crucial hand-object interaction cases.

Hand pose estimation with object interaction

There are some previous works that have taken the problem of object occlusion in hand pose estimation task into account [142, 27, 81, 121]. The work by Tekin et. al. [121] has impressive success of 3D hand pose estimation jointly with other parallel tasks. Their method takes a sequence of frames as input and outputs per-frame 3D hand and object pose predictions along with the estimates of object and action categories for the entire sequence, whereas it relies too much on a frame sequence rather than a single image. Gao et. al. [27] propose an object-aware method to estimate 3D hand pose from a single RGB image, where they rely on a deep structure to infer the category of the grasped object shape under the assumption that objects of a similar category are grasped in a similar way. Boukhayma et. al. [10] propose to use extracted hand parameters to control a mesh deformation hand model MANO [106] and project it into image domain to train the network. A similar model based method by Hasson et. al. [40] uses a contact loss to describe the spatial state of hand and object when a hand manipulates object, i.e. using a repulsion loss to penalize interpenetration and an attraction loss to encourage the hand to be in contact with the object. Nevertheless, these methods all require complex annotation process and could not fully utilize existing annotated clean hand datasets for hand-object interaction cases.

Augmented autoencoder and 3D shape reconstruction

Augmented Autoencoder is the backbone of our method, which is firstly proposed by Sundermeyer et. al. [117] in their real-time RGB-based pipeline for object detection and 6D pose estimation. In order to remove the effects of object occlusions and background clutters, they use an augmentation process to generate input data, which superimposes artificial occlusions and clutters to the clean data. Their work demonstrates that this training procedure is able to enforce the invariance of the encoded latent variable against a variety of different input augmentations. Encouraged by the idea of augmentation invariance, we apply a random augmentation process on clean hand samples of existing datasets to generate "noisy" input, and then recover corresponding clean hand samples with an auxiliary 3D shape reconstruction decoder.
3D shape reconstruction using deep learning has made a lot of advancement in recent years [18, 36, 141, 25]. Yang et. al. [141] propose a folding-based network, FoldingNet, which deforms a canonical 2D grid onto the underlying 3D target surface of a point cloud with two consecutive folding operations. For network complexity, FoldingNet consumes only about 7% parameters of a fully-connected layer based neural network to reconstruct a 3D target. Their method achieves low reconstruction errors even for targets with delicate structures. Therefore, the FoldingNet is chosen for the clean hand reconstruction decoder.

A critical challenge in 3D shape reconstruction is to evaluate the predicted point cloud. The loss function should be not only computationally efficient

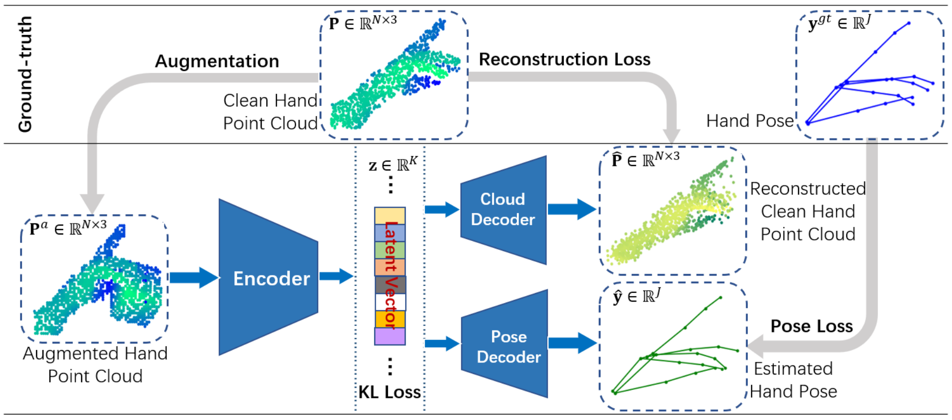

Ground-truth
Augmentation
$\mathbf{P} \in \mathbb{R}^{N\times3}$
Clean Hand Point Cloud
Reconstruction Loss
$\mathbf{y}^{gt} \in \mathbb{R}^{J}$
Hand Pose
$\mathbf{P}^{a} \in \mathbb{R}^{N\times3}$
Augmented Hand Point Cloud
Encoder
$\mathbf{z} \in \mathbb{R}^{K}$
Latent Vector
KL Loss
Cloud Decoder
Pose Decoder
$\hat{\mathbf{P}} \in \mathbb{R}^{N\times3}$
Reconstructed Clean Hand Point Cloud
$\hat{\mathbf{y}} \in \mathbb{R}^{J}$
Pose Loss
Estimated Hand Pose

FIGURE 6.2: Overview of the proposed method. The input of our network is occluded hand point cloud, which is obtained by a random augmentation process from clean hand point cloud. The encoder encodes the input hand to a latent vector. The obtained latent vector is then used to reconstruct clean hand point cloud by the auxiliary Cloud Decoder and predict 3D hand pose by the Pose Decoder. There are three losses in our VAE based framework, which are the KL loss, reconstruction loss and pose loss. (Brightness in point cloud indicates depth value.)

but also differentiable with respect to point coordinates. The Chamfer Distance (CD) and the Earth Mover's Distance (EMD) [107] are chosen to compare the reconstructed clean hand point cloud with ground-truth.

6.3 Augmented autoencoder for hand object interaction cases

The overview of the proposed method is illustrated in Figure 6.2 (left). The framework is based on the structure of Variational Autoencoder (VAE) [62]. The encoder takes 3D occluded hand point cloud as input, which is obtained by an augmentation process from clean hand and random objects. The encoder extracts point-wise features and fuses them to a latent vector, which is the latent representation of the input hand. Then, the acquired latent vector is used to reconstruct clean hand point cloud by the auxiliary Cloud Decoder and predict 3D hand pose by the Pose Decoder.

6.3.1 Data augmentation

The motivation behind the Augmented Autoencoder based hand pose estimation framework is to control what the latent vector encodes and which properties to be ignored. To take advantages of current large-scale clean hand dataset, a random augmentation process is applied by superimposing random objects from the ShapeNet [14] on clean hands to simulate hand-object interaction scenarios in reality. Simultaneously, the clean hand point cloud also serves as the ground-truth for reconstructed points by the auxiliary Cloud Decoder. Through this approach, the latent representation will be invariant to object occlusions when a hand is interacting with an object.

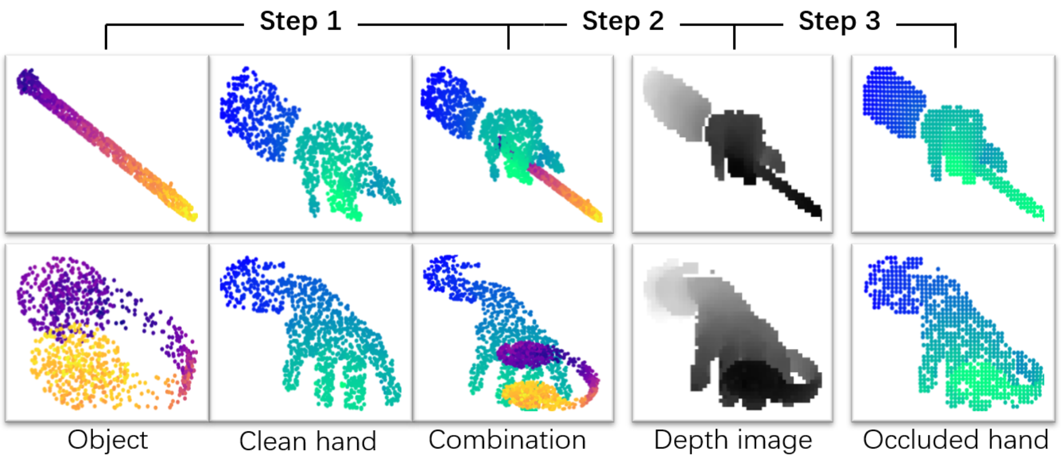


FIGURE 6.3: Data augmentation process. Step 1: combine hand point cloud and object; Step 2: project combined point cloud to depth image; Step 3: convert depth image to occluded hand point cloud. (Brightness in point cloud indicates depth, i.e. darker denotes further.)

The random augmentation process is shown in Figure 6.3. In step 1, a randomly selected object from ShapeNet is superimposed on a clean hand point cloud sample with random rotation, scaling and translation. Step 2 renders the combined point cloud to a depth image. Finally, step 3 converts the depth image back to point cloud format, which simulates the real hand object interaction cases.

6.3.2 Point cloud encoder

The Residual Permutation Equivariant Layer (PEL) (Chapter 4) is used as backbone to encode the input point cloud (Figure 6.4. The input, the occluded hand point cloud $\mathbf{P}^a \in \mathbb{R}^{N\times 3}$ is represented by N unoredered 3D points, it passes firstly through a residual PEL module, which consists of 3 residual PEL blocks. Then point-wise feature $\mathbf{F_1} \in \mathbb{R}^{N\times 1024}$ is computed for each individual input point, where each row of $\mathbf{F_1}$ represents the local feature for one point. The obtained $\mathbf{F_1}$ is imported to two separate point-wise fully-connected modules respectively, resulting in two separate terms, an importance term $\mathbf{G} \in \mathbb{R}^{N\times 256}$ and a new point-wise feature term $\mathbf{F_2} \in \mathbb{R}^{N\times 256}$, where the local feature dimension for each point is shrunk to 256. Each element of $\mathbf{G}$ indicates the weight for corresponding feature value in $\mathbf{F_2}$ and provides vital information of the importance of current feature value. Then, with a weighted fusion module, the information of both terms are fused to a global feature vector $\mathbf{F_3} \in \mathbb{R}^{256}$:

$$\mathbf{f}_i = \frac{\sum_{n=1}^{N}(\mathbf{G}_{ni}\mathbf{F_2}_{ni})}{\sum_{n=1}^{N}\mathbf{G}_{ni}}, \tag{6.1}$$

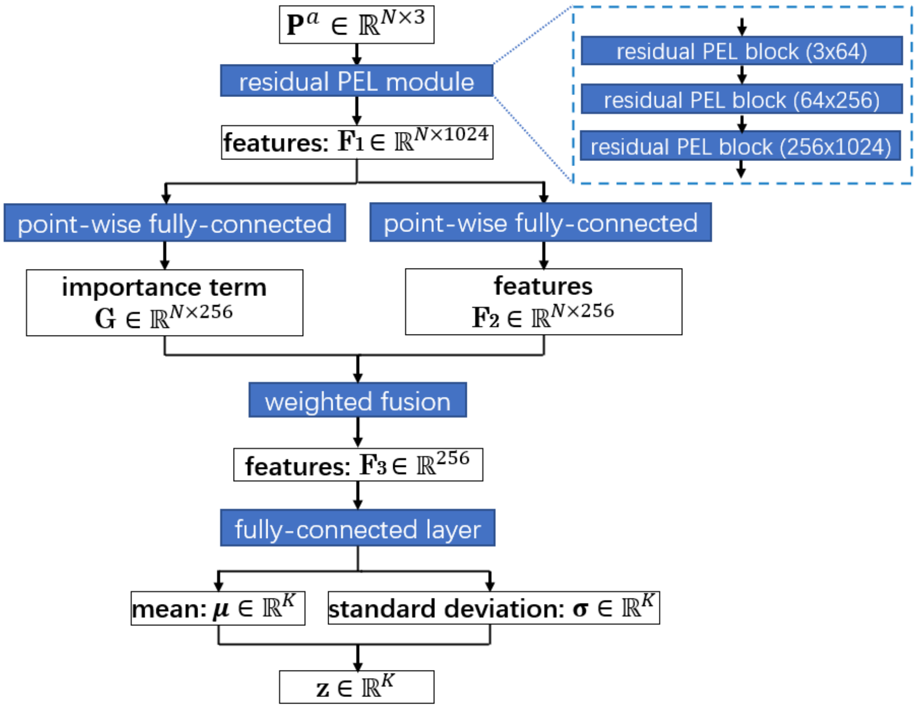

$\mathbf{P}^a \in \mathbb{R}^{N\times 3}$
residual PEL module
features: $\mathbf{F}_1 \in \mathbb{R}^{N\times 1024}$
residual PEL block (3x64)
residual PEL block (64x256)
residual PEL block (256x1024)
point-wise fully-connected
point-wise fully-connected
importance term $\mathbf{G} \in \mathbb{R}^{N\times 256}$
features $\mathbf{F}_2 \in \mathbb{R}^{N\times 256}$
weighted fusion
features: $\mathbf{F}_3 \in \mathbb{R}^{256}$
fully-connected layer
mean: $\boldsymbol{\mu} \in \mathbb{R}^K$
standard deviation: $\boldsymbol{\sigma} \in \mathbb{R}^K$
$\mathbf{z} \in \mathbb{R}^K$

FIGURE 6.4: Network structure of the point cloud encoder.

where $\mathbf{f}_i$ stands for the i-th dimension of $\mathbf{F}_3$.

In order to extract complex features, a network with 5 fully-connected layers is used to encode $\mathbf{F}_3$ to the final K-dimensional latent vector, which consists of a latent mean vector $\boldsymbol{\mu} \in \mathbb{R}^K$, and a latent standard deviation vector $\boldsymbol{\sigma} \in \mathbb{R}^K$. During training stage, a reparameterization process to sample from the distribution of the latent vector [62] is needed:

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon},$$

where $\boldsymbol{\epsilon} \in \mathbb{R}^K$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\odot$ denotes element-wise multiplication. The final latent vector $\mathbf{z} \in \mathbb{R}^K$ is Gaussian distributed and $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$.

6.3.3 Decoders and training loss

The obtained latent vector $\mathbf{z}$ from encoder is fed into the decoders. The clean hand reconstruction one, Cloud Decoder, is based on FoldingNet [141]. The Pose Decoder consists of multiple fully-connected layers.

Cloud Decoder is a FoldingNet [141] that transforms ("folds") 2d grid points of a square into 3D point cloud with two folding operations. In the folding operation, each grid point's coordinate is concatenated with the latent vector $\mathbf{z}$ and fed into a 4-layer perceptron to construct a more complex shape compared to the input. The final reconstructed points $\hat{\mathbf{P}}$ are evaluated by Chamfer Distance (CD) and Earth Mover's Distance (EMD) [107] with respect to the ground-truth clean hand point cloud $\mathbf{P} \in \mathbb{R}^{N\times 3}$. Note that the number of points in $\hat{\mathbf{P}}$ is required to be the same as $\mathbf{P}$.

The Chamfer Distance is defined as:

$$\mathcal{L}_{CD}\left(\hat{\mathbf{P}},\mathbf{P}\right)=\frac{1}{|\hat{\mathbf{P}}|}\sum_{\hat{p}\in\hat{\mathbf{P}}}\min_{p\in\mathbf{P}}\|\hat{p}-p\|+\frac{1}{|\mathbf{P}|}\sum_{p\in\mathbf{P}}\min_{\hat{p}\in\hat{\mathbf{P}}}\|\hat{p}-p\|, \tag{6.2}$$

where the CD algorithm finds for each point the nearest neighbor in the other point cloud and sums up the Euclidean distances.

The Earth Mover's Distance requires that $\hat{\mathbf{P}}$ and $\mathbf{P}$ contain the same number of points, i.e. $|\hat{\mathbf{P}}|=|\mathbf{P}|$, and it is defined as:

$$\mathcal{L}_{EMD}\left(\hat{\mathbf{P}},\mathbf{P}\right)=\frac{1}{|\mathbf{P}|}\min_{\phi:\mathbf{P}\to\hat{\mathbf{P}}}\sum_{p\in\mathbf{P}}\|p-\phi(p)\|, \tag{6.3}$$

where ϕ denotes one-to-one bijective correspondences from the ground-truth $\mathbf{P}$ to the predicted point set $\hat{\mathbf{P}}$. The Euclidean distances of all matched point pairs are then summed.

Both loss functions have their own intrinsic characteristics. For example, while EMD roughly captures the shape corresponding to the mean value of the hidden variable of the hand point cloud, CD tends to give a splashy shape that blurs the shape's geometric structure [25]. To make the Cloud Decoder more expressive, both loss functions are used during training time. Therefore, implicitly, the proposed method requires the reconstructed clean hand points have the same size N as the ground-truth.

For 3D hand pose prediction, Pose Decoder, which consists of 5 fully-connected layers, takes the reparameterized latent vector as input and outputs the vectorized 3D hand pose $\hat{\mathbf{y}}\in\mathbb{R}^J$, where $J=3\times\#joints$. The training loss between predicted hand pose $\hat{\mathbf{y}}$ and ground-truth pose $\mathbf{y}^{gt}\in\mathbb{R}^J$ is the $L2$ loss:

$$\mathcal{L}_{pose}=\frac{1}{2}\sum_{j=1}^{J}\left(\hat{\mathbf{y}}_j-\mathbf{y}_j^{gt}\right)^2. \tag{6.4}$$

As the proposed framework is based on VAE, a KL (Kullback–Leibler divergence) loss is essential to force the computed latent vector $\mathbf{z}$ given observed occluded data to be close to the centered isotropic multivariate Gaussian $\mathcal{N}(\mathbf{z};\mathbf{0},\mathbf{I})$ (Figure 6.2). The KL loss is defined as:

$$\mathcal{L}_{KL}=\frac{1}{2}\sum_{k=1}^{K}\left(\mu_k^2+\sigma_k^2-\log\left(\sigma_k^2\right)-1\right), \tag{6.5}$$

where K denotes the number of dimensions of the latent vector $\mathbf{z}$, μ_k is the k-th dimension of the latent mean $\boldsymbol{\mu}$ and σ_k denotes the k-th dimension of the latent standard deviation $\boldsymbol{\sigma}$.

The resulting total loss for our method is the summation of $\mathcal{L}_{CD}$, $\mathcal{L}_{EMD}$, $\mathcal{L}_{pose}$ and weighted $\mathcal{L}_{KL}$ terms:

$$\mathcal{L}_{total}=\mathcal{L}_{CD}+\mathcal{L}_{EMD}+\mathcal{L}_{pose}+\alpha\mathcal{L}_{KL}, \tag{6.6}$$

where α is the weight factor.

6.4 Experimental results

The networks are implemented using the TensorFlow framework with an ADAM optimizer. The learning rate is tapered down from 0.01 to 0.00001 during the course of training. For all experiments, we use an input and reconstruction point size of $N = 625$ for training, and $N = 900$ for testing. For the latent vector $\mathbf{z} \in \mathbb{R}^K$, the number of dimension is set to $K = 64$ and the KL Loss is weighted using a factor of $\alpha = 0.001$. Before the object augmentation process, each clean hand sample is randomly translated in all three dimensions within $[-15, 15]$ *mm*, randomly scaled within $[0.75, 1.25]$ and randomly rotated around z-axis within $[-\pi, \pi]$ *radian*. The trained model can be employed for real-time applications, since the network backbones, the ResPEL [75] and FoldingNet [141], are both real-time capable.

6.4.1 Datasets and evaluation metrics

Datasets

For training and evaluating the proposed network, Hands2017 Challenge dataset [147], SynthHands dataset [86] and EgoDexter dataset [86] are used. The Hands2017 Challenge is collected from parts of the BigHand2.2M [145] and the First-Person Hand Action (FHAD) [29]. The training set contains 957032 depth images, and the test set contains 295510 depth images. All samples in Hands2017 Challenge are clean hands, where the hands are not in contact with objects. The egocentric dataset SynthHands is a synthetic dataset created by posing a photorealistic hand model with real hand motion data. It captures multiple variations in natural hand motion, such as pose, skin color, shape, texture, background clutter as well as camera viewpoint. This dataset contains accurate annotated 92536 RGB-D images of clean hands and 91600 RGB-D images of hands interacting with objects, of which we use 69402 clean samples and 68700 interacting hand samples for training. Except the training samples, the rest 23134 clean samples serve as our clean test set and 22900 interacting samples as our interacting test set. The benchmark dataset EgoDexter consists of four real sequences with hand-object interactions (Rotunda, Desk, Kitchen, Fruits), which contain in total 1485 frames with 3D finger tip annotations. We compare the accuracy to the state-of-the-art method in [86] using this dataset. Notice that the Kitchen sequence is excluded from the experiment due to its many annotation errors, and use the other three sequences for evaluation.

For the random augmentation process for clean hand samples, object models from ShapeNetCore are used, which is a subset of the object repository ShapeNet [14] and covers 55 object categories with about 51300 unique 3D models. As preprocessing, we sample these 3D models to point clouds.

Evaluation metrics

We evaluate the performance qualitatively on real data for the trained model on Hands2017 Challenge, because it contains no annotated samples for hand-object interaction cases. For the SynthHands dataset, two standard metrics are used for evaluation. The first one is the mean joint error (*mm*), which measures the average Euclidean distance error for all joints across the whole test set. The second metric is correct frame proportion, which indicates the percentage of frames that have all joint errors within a certain threshold compared to the ground-truth. The correct frame proportion metric is challenging, since a single joint violation will cause an incorrect frame. For the EgoDexter dataset with only finger tip annotations, finger tip error is used for evaluation, which is the mean joint error for 3D finger tip positions.

6.4.2 Comparison to state-of-the-art

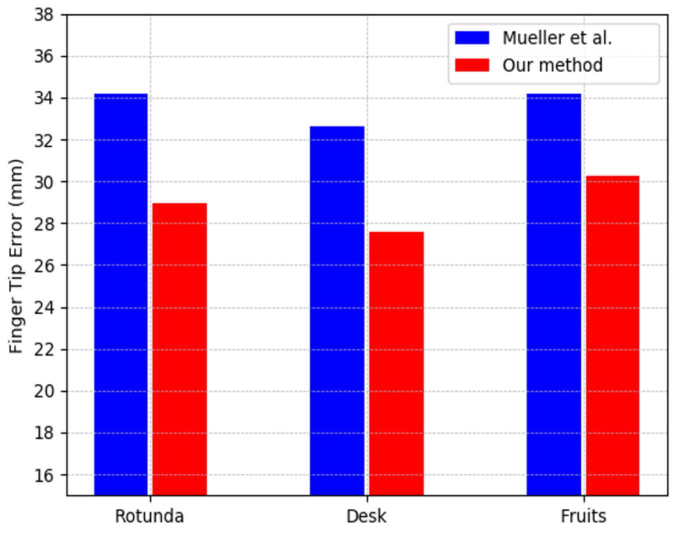


FIGURE 6.5: Comparison to state-of-the-art method on EgoDexter benchmark.

Since the EgoDexter dataset is only annotated on 3D finger tip positions, only finger tip error is used to compare the performance of our method with the kinematic pose tracking method proposed by Mueller et. al. [86]. We follow the same training dataset in their work, where all samples in SynthHands are used. As shown in Figure 6.5, our method outperforms the state-of-the-art method on the test sequences, achieving the average error of 28.70 *mm*. Note that the objects in EgoDexter are different from the objects in SynthHands training data. It shows the generalization ability of the proposed method to unknown objects.

6.4.3 Ablation study

In the first ablation experiment, we mix different proportions of interacting hand samples to training set to compare the effect on performance of different trained models.

Using the training samples from SynthHands, 4 different training datasets are chosen with varying percentages of hand-object interaction samples:

- Dataset A: 100% clean hand samples.
- Dataset B: 75% clean + 25% interacting hand samples.
- Dataset C: 50% clean + 50% interacting hand samples.
- Dataset D: 25% clean + 75% interacting hand samples.

Note that the interacting hand samples are not augmented during training time. Also, note that the performance of interacting hand is usually much worse than the clean hand samples due to occlusion.

The detailed comparison of mean joint errors on our both test sets can be found in Table 6.1. We can already obtain a reasonably good result on 100% clean hand Dataset A. Even if using only augmented hand samples from clean hand without any interacting hand samples, the error on interacting test set is 19.13 *mm*, which indicates the effectiveness of the augmentation strategy.

Furthermore, the best performance is achieved with training Dataset B, which contains 25% interacting hand samples. Compared to Dataset A, the mean joint error is decreased for 5 *mm* on interacting hand test set by mixing only a small proportion of real interacting hand samples in the training dataset. However, with the increasing proportion of interacting hand for training, the results become slightly worse, even on the interacting test set. The possible reason for this is that the decrease of clean hand proportion leads to less data augmentation, which means less random objects are seen for the training process, resulting in less generalizability on the unseen objects in the test set. Moreover, for the interacting training samples, hand reconstruction part were not trained since there is no available clean hand ground-truth to guide reconstruction, this leads to insufficient training of the reconstruction decoder and in turn influences the quality of the latent space. This experiment shows that, in practice, people can utilize large clean hand dataset and mix a small proportion of interacting hand samples, which are expensive to annotate, to achieve robust performance.

TABLE 6.1: Comparison of different training methods on SynthHands.

Training Dataset	**Error on Test Dataset (mm)**	
	clean hand	interacting hand
A	9.67	19.13
B	**9.63**	**14.16**
C	10.69	14.35
D	12.52	15.99

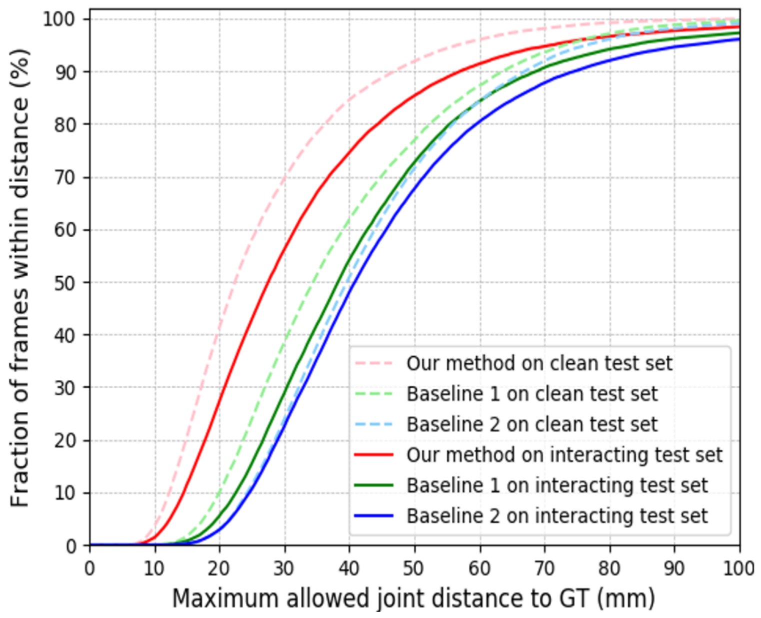


FIGURE 6.6: Proportion of correct frames with respect to different error thresholds.

In the second ablation study, the proposed method is compared to the following baseline methods to show the effects of the data augmentation and points reconstruction approaches:

- Baseline 1. Ours without object augmentation.
- Baseline 2. Ours without clean hand reconstruction.

Both baselines are trained using Dataset B. As seen in Figure 6.6 and 6.7, our method outperforms the two baselines on both clean hand test set and interacting hand test set. Table 6.2 shows that the results of baselines are worse even on clean hand test set. The possible reason for this is that the latent representation in baselines is implicitly correlated to the mixture of clean hands and interacting hands rather than clean hands alone in our Augmented Autoencoder based framework. By this result, we demonstrate the significant effects of the augmentation component and the reconstruction component in our method.

TABLE 6.2: Comparison with baselines on SynthHands.

Model	**Error on Test Dataset (mm)**	
	clean hand	interacting hand
Our method	**9.63**	**14.16**
Baseline 1	15.44	20.78
Baseline 2	19.60	23.46

6.4.4 Qualitative Results

For the SynthHands dataset, the qualitative comparison of our method with two baselines is shown in Figure 6.8 on the interacting test set.

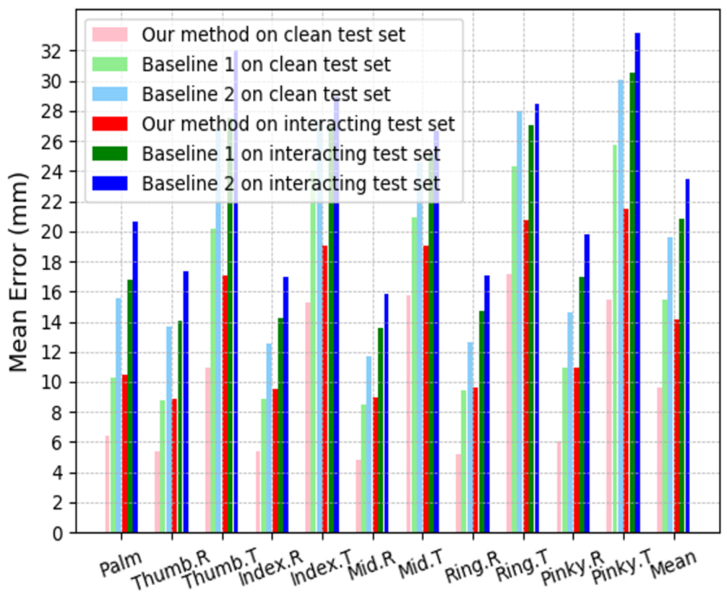


FIGURE 6.7: Comparison to baseline on SynthhandsTest: Mean errors of different joints.

For the Hands2017 Challenge dataset, as the training set and test set contain only clean hands, the model is trained only with augmented data. Figure 6.9 shows qualitative results on real captured interacting cases, where the hand interacts with different objects, such as ball, bucket, phone, paper box, which are not seen during training. Although the model is trained only with clean hand data on the Hands2017 Challenge dataset, the results shows good performance. Note that high quality point cloud reconstruction is not strictly required in our method. Figure 6.9 shows that occluded objects are roughly removed after reconstruction, indicating the importance of the Cloud Decoder for the formation of the latent space of the clean hand.

6.5 Summary

In this chapter, a novel deep learning framework using Augmented Autoencoder is presented to handle hand pose estimation tasks for hand-object interaction cases. The proposed method consumes 3D hand point cloud and predicts accurate 3D hand pose. The augmentation process and auxiliary clean hand reconstruction decoder implicitly force the latent representation of the pose only to be correlated to clean hand and the reconstructed clean hand despite the object occlusion in hand-object interaction cases. Furthermore, the proposed hand pose estimation training strategy is able to utilize existing clean hand datasets to tackle hand-object interaction cases. Quantitative and qualitative evaluation results show that the proposed framework is capable of achieving low joint errors on both clean hand input ($\sim 9\ mm$) and interacting hand input ($\sim 14\ mm$).

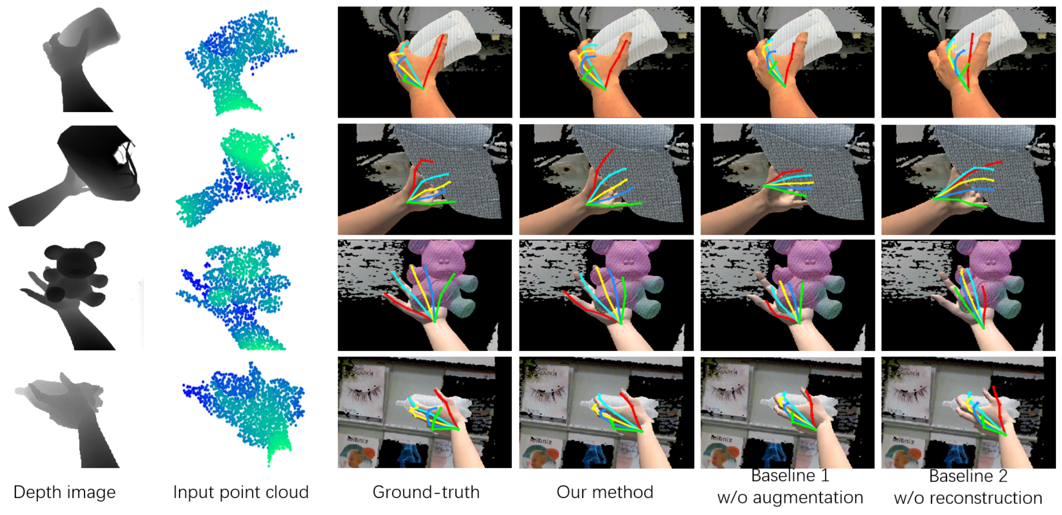


FIGURE 6.8: Qualitative results compared with baselines on SynthHands. (Brightness in point cloud indicates depth, i.e. darker denotes further.)

6.6 Discussion and future works

In the experiments, we have trained a model using synthetic dataset (SynthHands) and tested the trained model directly on real-captured dataset (EgoDexter). The average finger tip error on the real-captured data is by 9 mm worse than the performance on synthetic test data (28.7 mm vs. 19.5). This implies that there is a domain gap between the synthetic data and real-captured data. Since annotated synthetic data is easy to obtain, an interesting future research direction would be to use domain transfer method to close the domain gap between synthetic and real data. For example, domain adaptation method like CycleGAN [151] could be used in the future to firstly transform synthetic images to photo-realistic images, and then perform hand pose estimation tasks on transformed images. Another possibility would be to transform real-captured data to synthetic domain and then the model trained on synthetic data could be used directly.

This paper presents an augmentation method to superimpose artificial object with the "clean" hand samples, which shows success to utilize "clean" hand dataset to train a model for hand-object interaction cases. However, in the proposed augmentation method, the object is placed at a random location around the hand, which does not match the reality. E.g. the hand and the object could penetrate each other, or the object is not firmly grasped by the fingers. Despite of the simple and unrealistic augmentation process, it has showed success to test on hand-object samples. But to further improve the performance of the augmented autoencoder, the augmentation process should be more realistic in the future works. We can obtain realistic augmentation from synthetic data, where annotated hand-object samples with physically plausible grasping pose can be created. To synthetize physically plausible data, object shape and pose also play important roles. In this regard, object shape and pose annotation is also available for the synthetic data. Therefore, an interesting future research direction is to estimate hand pose

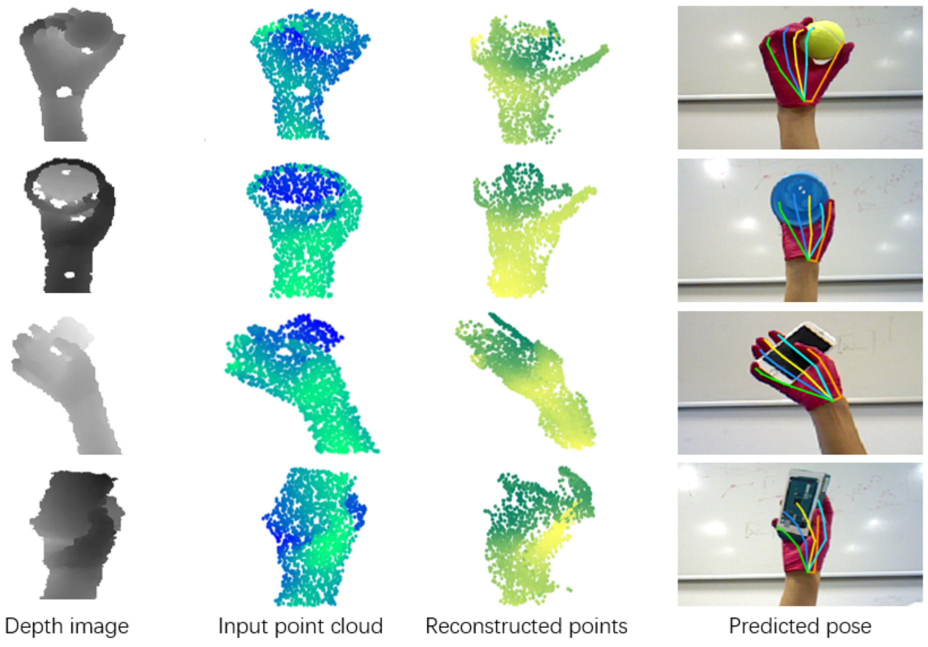


FIGURE 6.9: Qualitative results on real data. (Color in point cloud indicates depth, where darker indicates larger distance to the camera.)

and object pose simultaneously, where in a recent pioneer work [40], Hasson et.al. demonstrated promising result of joint hand pose estimation and object shape reconstruction.

Joint hand-object pose estimation is challenging because of the difficulty to annotate. Therefore, another trend to tackle this task is to rely on self-supervised learning to directly learn on real-captured data without annotations. Recently, the differentiable rendering techniques become popular, it is a novel field which allows the gradients of 3D objects to be calculated and propagated through images [56]. Using a differentaible renderer, the whole pipeline could be trained end-to-end in a self-supervised manner, where in our case, the parameters of hand and object will be extracted form input image to construct 3D model of hand and object. Then the 3D hand model and object model will be rendered to reconstruct the input image, where the image reconstruction loss will be used to guide the network parameters to be learned.

Chapter 7

Conclusion and future research direction

Motivated by the need of human hand motion observation, this thesis aimed to investigate the following aspects:

- How to localize the camera from image sequences, in order to generate object models?
- How to estimate human hand pose using different formats of sensor data, for a clean hand without object?
- How to utilize clean hand dataset and scanned object model to design hand pose estimation method for hand-object interaction cases?

Camera localization

The camera localization is achieved by proposed visual odometry and SLAM algorithms. Chapter 2.1 presents a novel visual odoemtry method using RGB-D image sequences, it is a fast and robust visual odometry estimation method based on intensity assisted ICP (IAICP). By contributing in the selection, matching and weighting stages, IAICP improves the conventional ICP significantly. Intelligent salient point selection is performed on the source frame thus drastically reduced the computation time. Correspondences are established by searching nearby points in the image coordinate. With weighting function devised from statistics, robustness against outlying correspondences is ensured. The proposed method was evaluated on the TUM Dataset both quantitatively and qualitatively. In terms of translational drift, it outperforms state-of-the-art methods in 11 out of the 14 tested video sequences. Furthermore, IAICP runs with an average frame rate of 78 Hz using a single CPU thread, thus it can be used on a mobile device with limiting computation power, which is a crucial factor for object scanning application. With changes of parameter settings, IAICP can even achieve 107 Hz by loosing ca. 12% precision of drift error. Experimental results showed that the proposed approach achieved overall better accuracy than approaches with GPU parallelization.

On top of the visual odometry method, Chapter 2.2 presented extensions to the IAICP method to allow existence of dynamic objects in the environment. The extended method uses foreground depth edge point to compute pair-wise point cloud registration. A robust static weighting strategy is proposed based on depth edge correspondences distance. Fusing the static weighting strategy into the IAICP, the visual odometry system can handle dynamic environment robustly. Furthermore, loop closure detection and map optimization are integrated, resulting a real-time SLAM system suitable for dynamic environment. The accuracy and computation efficiency are tested on the dynamic sequences from TUM Dataset. Compared to state-of-the-art real-time method [61], in terms of translational drift per second, the proposed method improves the visual odometry accuracy by 58% in challenging "walking" sequences. The performance of the SLAM system is also proven using the TUM Dataset, which shows better performance than recent non real-time method [116].

With the developed robust and fast camera localization system, object scanning can be performed using the camera localization results. Chapter 2.3 presented the object modelling pipeline and a surface smoothing procedure is proposed to reduce the noisy measurement of the keyframes. Some successful scanning results are shown using the CoRBS dataset [133].

The proposed visual camera localization methods in this thesis are based on the Iterative Closest Point framework, which is a traditional geometry based methods. In recent years, researchers are starting to combine deep learning methods into the geometry world in camera localization. For example, the correspondence matching algorithm used in this thesis is purely based on predefined distance metric, this could be replaced by a more sophisticated feature based method, where the features can be learned using a deep learning method. Moreover, this thesis can only filter out the depth edges of dynamic objects. To densely segment dynamic objects, a deep learning based semantic segmentation method can be used for this task. Furthermore, this thesis focused on camera localization in an unknown environment, whereas future research could investigate more on how to localize the camera in a predefined map.

Clean hand pose estimation

Chapter 3-5 presented deep learning based hand pose estimation methods for the clean hand cases, where the hand is not touching or occluded by other objects.

Firstly, Chapter 3 introduced a depth map based method. It incorporates an embedded differentiable kinematic layer into the deep learning networks. Apart from joint angles, the proposed kinematic layer also takes hand shape parameters as input, thus it generalizes on different hand shapes. Experiments have shown that by using kinematic layer, the number physically implausible results is reduced, furthermore we have also shown that applying appearance normalization using Spatial Transformer Network, the pose estimation accuracy can be further improved.

Chapter 4 proposed a novel neural network architecture, ResidualPEL, for hand pose estimation using unordered point cloud as input. The point cloud based method has shown superior performance compared to depth image based methods. The proposed method is invariant to input point order and can handle different numbers of points. Compared to previous 3D voxel based methods, it requires less memory size and compared to PointNet based methods, it does not require surface normal and K-nearest-neighours information. A voting-based scheme was proposed to merge information from individual points to pose output, where the resulting importance term can be also used to segment the hand into different parts. The performance is evaluated on two datasets, where the proposed method outperforms the state-of-the-art methods on both datasets.

Finally, Chapter 5 explored explored how to utilize multi-modal dataset in training to boost the performance of unimodal testing, in which different ways of latent space alignment are shown and evaluated. The developed Gaussian product based alignment strategy is highly flexible, it can exploit different modalities as prior knowledge to improve the performance of RGB-based hand pose estimation as well as leverage weakly labelled data to further boost pose estimation performance.

For hand pose estimation, several deep learning based methods were presented. Their success all rely on the availability of large, annotated dataset. In future work, to overcome the limitation of the size and quality of dataset, self-supervised learning method should be researched more. For example, many recent works start to use the differentiable hand model, MANO [106], to estimate a 3D hand mesh model from input image. Combining the MANO model and differentiable rendering pipeline, end-to-end self-supervised learning of hand pose can be a very interesting future research topic.

Hand pose estimation for hand-object interaction cases

Chapter 6 presented a novel deep learning framework using Augmented Autoencoder to handle hand pose estimation tasks for hand-object interaction cases. The proposed method consumes 3D hand point cloud and predicts accurate 3D hand pose. The augmentation process and auxiliary clean hand reconstruction decoder implicitly force the latent representation of the pose only to be correlated to clean hand and the reconstructed clean hand despite the object occlusion in hand-object interaction cases. Furthermore, the proposed hand pose estimation training strategy is able to utilize existing clean hand datasets to tackle hand-object interaction cases. Quantitative and qualitative evaluation results show that the proposed framework is capable of achieving low joint errors on both clean hand input ($\sim 9\ mm$) and interacting hand input ($\sim 14\ mm$).

Targeting at the limitation of dataset, augmentation technique was used to simulate hand object interaction cases. Although this method provides reasonably good results, however, the augmented samples are not realistic compared to real captured data, which can be a limiting factor on the performance. In future research, self-supervised learning methods can be also applied on hand object interaction cases, where object pose and hand pose can be jointly optimized relying on differentiable rendering and physical constraint between the hand and the object. Furthermore, the relation between object shape and hand grasp pose should be investigated more, since the object shape provides a vital prior information on the grasp type. Eor example, with the help of Mixture Density Networks, a prior distribution of grasp pose can be firstly obtained from the object information, and then the distribution can be refined with the observation of human hand.

Appendix A

Presented and Published Papers

- Shile Li and Dongheui Lee. "RGB-D SLAM in dynamic environments using static point weighting." IEEE Robotics and Automation Letters (RA-L) 2017
- Shile Li and Dongheui Lee. "Fast visual odometry using intensity-assisted iterative closest point." IEEE Robotics and Automation Letters (RA-L) 2016
- Shile Li*, Wöhlke Jan* and Dongheui Lee. "Model-based hand pose estimation for generalized hand shape with spatial transformer network." European Conference on Computer Vision (ECCV), Extended Abstract Presentation in 4th International Workshop on Observing and Understanding Hands in Action (HANDS2018) 2018. *equal contribution
- Shanxin Yuan, Guillermo Garcia-Hernando, Björn Stenger, Gyeongsik Moon, Ju Yong Chang, Kyoung Mu Lee, Pavlo Molchanov, Jan Kautz, Sina Honari, Liuhao Ge, Junsong Yuan, Xinghao Chen, Guijin Wang, Fan Yang, Kai Akiyama, Yang Wu, Qingfu Wan, Meysam Madadi, Sergio Escalera, Shile Li, Dongheui Lee, Iason Oikonomidis, Antonis Argyros, Tae-Kyun Kim. "Depth-based 3d hand pose estimation: From current achievements to future goals." IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2018.
- Shile Li and Dongheui Lee. "Point-to-pose voting based hand pose estimation using residual permutation equivariant layer." IEEE Computer Vision and Pattern Recognition (CVPR), 2019.
- Linlin Yang*, Shile Li*, Dongheui Lee and Angela Yao. "Aligning Latent Spaces for 3D Hand Pose Estimation." IEEE International Conference on Computer Vision (ICCV) 2019. *equal contribution
- Anil Armagan, Guillermo Garcia-Hernando, Seungryul Baek, Shreyas Hampali, Mahdi Rad, Zhaohui Zhang, Shipeng Xie, MingXiu Chen, Boshen Zhang, Fu Xiong, Yang Xiao, Zhiguo Cao, Junsong Yuan, Pengfei Ren, Weiting Huang, Haifeng Sun, Marek Hrúz, Jakub Kanis, Zdeněk Krňoul, Qingfu Wan, Shile Li, Linlin Yang, Dongheui Lee, Angela Yao, Weiguo Zhou, Sijia Mei, Yunhui Liu, Adrian Spurr, Umar Iqbal, Pavlo

Molchanov, Philippe Weinzaepfel, Romain Brégier, Gregory Rogez, Vincent Lepetit, Tae-Kyun Kim. "Measuring Generalisation to Unseen Viewpoints, Articulations, Shapes and Objects for 3D Hand Pose Estimation under Hand-Object Interaction." European Conference on Computer Vision (ECCV) 2020

- Shile Li*, Haojie Wang* and Dongheui Lee. "Hand Pose Estimation for Hand-Object Interaction Cases using Augmented Autoencoder." IEEE International Conference on Robotics and Automation (ICRA) 2020. *equal contribution

Bibliography

[1] Martín Abadi et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems". In: *arXiv:1603.04467* (2016).

[2] Dafni Antotsiou, Guillermo Garcia-Hernando, and Tae-Kyun Kim. "Task oriented hand motion retargeting for dexterous manipulation imitation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

[3] Brenna D Argall et al. "A survey of robot learning from demonstration". In: *Robotics and autonomous systems* 57.5 (2009), pp. 469–483.

[4] K Somani Arun, Thomas S Huang, and Steven D Blostein. "Least-squares fitting of two 3-D point sets". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 5 (1987), pp. 698–700.

[5] Xuyang Bai et al. "D3Feat: Joint Learning of Dense Detection and Description of 3D Local Features". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 6359–6367.

[6] Luca Ballan et al. "Motion capture of hands in action using discriminative salient points". In: *European Conference on Computer Vision*. 2012, pp. 640–653.

[7] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features". In: *European conference on computer vision*. Springer. 2006, pp. 404–417.

[8] Paul J Besl and Neil D McKay. "Method for registration of 3-D shapes". In: *Robotics-DL tentative*. International Society for Optics and Photonics. 1992, pp. 586–606.

[9] Laurie Bose and Arthur Richards. "Fast depth edge detection and edge based RGB-D SLAM". In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 1323–1330.

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

[11] Yujun Cai et al. "Weakly-supervised 3d hand pose estimation from monocular rgb images". In: *ECCV*. 2018.

[12] S. Calinon and D. Lee. "Learning Control". In: *Humanoid Robotics: a Reference*. Ed. by P. Vadakkepat and A. Goswami. Springer, 2019, pp. 1261–1312. DOI: `10.1007/978-94-007-6046-2_68`.

[13] Sylvain Calinon and Dongheui Lee. “Learning Control, Humanoid Robotics: a Reference, P. Vadakkepat and A. Goswami (eds.)” In: (2017).

[14] Angel X Chang et al. “Shapenet: An information-rich 3d model repository”. In: *arXiv preprint arXiv:1512.03012* (2015).

[15] Xinghao Chen et al. “Pose Guided Structured Region Ensemble Network for Cascaded Hand Pose Estimation”. In: *arXiv preprint: 1708.03416* (2017).

[16] Xinghao Chen et al. “SHPR-Net: Deep Semantic Hand Pose Regression from Point Clouds”. In: *IEEE Access* 6 (2018), pp. 43425–43439.

[17] Changhyun Choi, Alexander JB Trevor, and Henrik I Christensen. “RGB-D edge detection and edge-based registration”. In: *IEEE/RSJ International Conference onIntelligent Robots and Systems (IROS)*. 2013, pp. 1568–1575.

[18] Christopher B Choy et al. “3d-r2n2: A unified approach for single and multi-view 3d object reconstruction”. In: *European conference on computer vision*. Springer. 2016, pp. 628–644.

[19] Peter Hviid Christiansen et al. “Unsuperpoint: End-to-end unsupervised interest point detector and descriptor”. In: *arXiv preprint arXiv: 1907.04011* (2019).

[20] Xiaoming Deng et al. “Hand3d: Hand pose estimation using 3d neural network”. In: *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE* (2018).

[21] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. “Superpoint: Self-supervised interest point detection and description”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 224–236.

[22] David W Eggert, Adele Lorusso, and Robert B Fisher. “Estimating 3-D rigid body transformations: a comparison of four major algorithms”. In: *Machine Vision and Applications* 9.5-6 (1997), pp. 272–290.

[23] Felix Endres et al. “3-d mapping with an rgb-d camera”. In: *IEEE Transactions on Robotics* 30.1 (2014), pp. 177–187.

[24] Ali Erol et al. “Vision-based hand pose estimation: A review”. In: *Computer Vision and Image Understanding* 108.1-2 (2007), pp. 52–73.

[25] Haoqiang Fan, Hao Su, and Leonidas J Guibas. “A point set generation network for 3d object reconstruction from a single image”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 605–613.

[26] Yaroslav Ganin and Victor Lempitsky. “Unsupervised domain adaptation by backpropagation”. In: *International conference on machine learning*. PMLR. 2015, pp. 1180–1189.

[27] Yafei Gao et al. “Variational Object-Aware 3-D Hand Pose From a Single RGB Image”. In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 4239–4246.

[28] Guillermo Garcia-Hernando et al. "First-Person Hand Action Benchmark with RGB-D Videos and 3D Hand Pose Annotations". In: *arXiv: 1704.02463* (2017).

[29] Guillermo Garcia-Hernando et al. "First-person hand action benchmark with rgb-d videos and 3d hand pose annotations". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 409–419.

[30] Liuhao Ge, Zhou Ren, and Junsong Yuan. "Point-to-point regression pointnet for 3d hand pose estimation". In: *ECCV*. 2018, pp. 475–491.

[31] Liuhao Ge et al. "3d convolutional neural networks for efficient and robust hand pose estimation from single depth images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1991–2000.

[32] Liuhao Ge et al. "3D Convolutional Neural Networks for Efficient and Robust Hand Pose Estimation From Single Depth Images". In: *IEEE conference on computer vision and pattern recognition*. 2017, pp. 1991–2000.

[33] Liuhao Ge et al. "Hand PointNet: 3d hand pose estimation using point sets". In: *CVPR*. 2018.

[34] Liuhao Ge et al. "Robust 3d hand pose estimation in single depth images: from single-view cnn to multi-view cnns". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3593–3601.

[35] Giorgio Grisetti et al. "A tutorial on graph-based SLAM". In: *IEEE Intelligent Transportation Systems Magazine* 2.4 (2010), pp. 31–43.

[36] Thibault Groueix et al. "A papier-mâché approach to learning 3d surface generation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 216–224.

[37] Hengkai Guo et al. "Towards Good Practices for Deep 3D Hand Pose Estimation". In: *arXiv:1707.07248* (2017).

[38] Daniel Gutierrez-Gomez, Walterio Mayol-Cuevas, and J.J. Guerrero. "Inverse depth for accurate photometric and geometric error minimisation in RGB-D dense visual odometry". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 83–89.

[39] Henning Hamer et al. "Tracking a hand manipulating an object". In: *2009 12th International Conference on Computer Vision*. IEEE. 2009.

[40] Yana Hasson et al. "Learning joint reconstruction of hands and manipulated objects". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 11807–11816.

[41] Kaiming He et al. "Deep residual learning for image recognition". In: *CVPR*. 2016.

[42] Xinwei He et al. "Triplet-Center Loss for Multi-View 3D Object Retrieval". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[43] Peter Henry et al. "RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments". In: *Experimental robotics*. 2014, pp. 477–491.

[44] Peter Henry et al. "RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments". In: *The International Journal of Robotics Research* 31.5 (2012), pp. 647–663.

[45] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. "Transforming auto-encoders". In: *International Conference on Artificial Neural Networks*. Springer. 2011, pp. 44–51.

[46] Kai Hu and Dongheui Lee. "Prediction-based synchronized human motion imitation by a humanoid robot". In: *at-Automatisierungstechnik* 60.11 (2012), pp. 705–714.

[47] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. "Pointwise Convolutional Neural Networks". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[48] Peter J Huber. *Robust statistics*. Springer, 2011.

[49] Umar Iqbal et al. "Hand pose estimation via latent 2.5 d heatmap regression". In: *ECCV*. 2018.

[50] Shahram Izadi et al. "KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera". In: *Proceedings of the 24th annual ACM symposium on User interface software and technology*. 2011, pp. 559–568.

[51] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. "Spatial transformer networks". In: *Advances in neural information processing systems*. 2015, pp. 2017–2025.

[52] M. Jaimez and J. Gonzalez-Jimenez. "Fast Visual Odometry for 3-D Range Sensors". In: *IEEE Transactions on Robotics* PP.99 (2015), pp. 1–14.

[53] Youngkyoon Jang et al. "3d finger cape: Clicking action and position estimation under self-occlusions in egocentric viewpoint". In: *IEEE Transactions on Visualization and Computer Graphics* 21.4 (2015), pp. 501–510.

[54] David Joseph Tan et al. "Fits like a glove: Rapid and reliable hand shape personalization". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 5610–5619.

[55] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. "Neural 3d mesh renderer". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3907–3916.

[56] Hiroharu Kato et al. "Differentiable rendering: A survey". In: *arXiv preprint arXiv:2006.12057* (2020).

[57] Christian Kerl, Jurgen Sturm, and Daniel Cremers. "Dense visual slam for rgb-d cameras". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2100–2106.

[58] Christian Kerl, Jurgen Sturm, and Daniel Cremers. "Robust odometry estimation for RGB-D cameras". In: *2013 IEEE International Conference onRobotics and Automation (ICRA)*, pp. 3748–3754.

[59] Sameh Khamis et al. "Learning an efficient model of hand shape variation from depth images". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 2540–2548.

[60] Deok-Hwa Kim, Seung-Beom Han, and Jong-Hwan Kim. "Visual odometry algorithm using an RGB-D sensor and IMU in a highly dynamic environment". In: *Robot Intelligence Technology and Applications 3*. Springer, 2015, pp. 11–26.

[61] Deok-Hwa Kim and Jong-Hwan Kim. "Effective Background Model-Based RGB-D Dense Visual Odometry in a Dynamic Environment". In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1565–1573.

[62] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[63] Roman Klokov and Victor Lempitsky. "Escape from cells: Deep kd-networks for the recognition of 3d point cloud models". In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE. 2017, pp. 863–872.

[64] Sebastian Klose, Peter Heise, and Aaron Knoll. "Efficient compositional approaches for real-time robust direct visual odometry from RGB-D data". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1100–1106.

[65] Seongyong Koo, Dongheui Lee, and Dong-Soo Kwon. "Unsupervised object individuation from RGB-D image sequences". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, pp. 4450–4457.

[66] Jason Ku et al. "Joint 3d proposal generation and object detection from view aggregation". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 1–8.

[67] Nikolaos Kyriazis and Antonis Argyros. "Physically plausible 3d scene tracking: The single actor hypothesis". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 9–16.

[68] Kenneth L Lange, Roderick JA Little, and Jeremy MG Taylor. "Robust statistical modeling using the t distribution". In: *Journal of the American Statistical Association* 84.408 (1989), pp. 881–896.

[69] Dongheui Lee and Yoshihiko Nakamura. "Motion recognition and recovery from occluded monocular observations". In: *Robotics and Autonomous Systems* 62.6 (2014), pp. 818–832.

[70] Dongheui Lee and Christian Ott. “Incremental kinesthetic teaching of motion primitives using the motion refinement tube”. In: *Autonomous Robots* 31.2-3 (2011), pp. 115–131.

[71] Tung-Sing Leung and Gerard Medioni. “Visual navigation aid for the blind in dynamic environments”. In: *IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2014, pp. 565–572.

[72] Jiaxin Li, Ben M. Chen, and Gim Hee Lee. “SO-Net: Self-Organizing Network for Point Cloud Analysis”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[73] Jiaxin Li and Gim Hee Lee. “Usip: Unsupervised stable interest point detection from 3d point clouds”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 361–370.

[74] Shile Li and Dongheui Lee. “Fast Visual Odometry Using Intensity-Assisted Iterative Closest Point”. In: *IEEE Robotics and Automation Letters* 1.2 (2016), pp. 992–999.

[75] Shile Li and Dongheui Lee. “Point-to-pose voting based hand pose estimation using residual permutation equivariant layer”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 11927–11936.

[76] Shile Li and Dongheui Lee. “Point-to-Pose Voting based Hand Pose Estimation using Residual Permutation Equivariant Layer”. In: *CVPR*. 2019.

[77] Shichen Liu et al. “Soft rasterizer: A differentiable renderer for image-based 3d reasoning”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 7708–7717.

[78] David G Lowe. “Object recognition from local scale-invariant features”. In: *Proceedings of the seventh IEEE international conference on computer vision*. Vol. 2. Ieee. 1999, pp. 1150–1157.

[79] Weixin Lu et al. “Deepvcp: An end-to-end deep neural network for point cloud registration”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 12–21.

[80] Meysam Madadi et al. “End-to-end global to local cnn learning for hand pose recovery in depth data”. In: *arXiv:1705.09606* (2017).

[81] Meysam Madadi et al. “Occlusion aware hand pose recovery from sequences of depth images”. In: *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*. IEEE. 2017, pp. 230–237.

[82] Daniel Maturana and Sebastian Scherer. “Voxnet: A 3d convolutional neural network for real-time object recognition”. In: *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE. 2015, pp. 922–928.

[83] Maxime Meilland, Andrew Comport, Patrick Rives, et al. "A spherical robot-centered representation for urban navigation". In: *2010 IEEE/RSJ International Conference onIntelligent Robots and Systems (IROS)*, pp. 5196–5201.

[84] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. "V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map". In: *CVPR*. 2018, pp. 5079–5088.

[85] Franziska Mueller et al. "GANerated hands for real-time 3D hand tracking from monocular RGB". In: *CVPR*. 2018.

[86] Franziska Mueller et al. "Real-time hand tracking under occlusion from an egocentric rgb-d sensor". In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2017, pp. 1284–1293.

[87] Richard A Newcombe and Andrew J Davison. "Live dense reconstruction with a single moving camera". In: *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2010, pp. 1498–1505.

[88] Richard A Newcombe et al. "KinectFusion: Real-time dense surface mapping and tracking". In: *2011 10th IEEE international symposium on Mixed and augmented reality (ISMAR)*, pp. 127–136.

[89] Chuong V Nguyen, Shahram Izadi, and David Lovell. "Modeling kinect sensor noise for improved 3d reconstruction and tracking". In: *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*. IEEE, pp. 524–530.

[90] Markus Oberweger and Vincent Lepetit. "Deepprior++: Improving fast and accurate 3d hand pose estimation". In: *ICCV workshop*. Vol. 840. 2017, p. 2.

[91] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. "Hands Deep in Deep Learning for Hand Pose Estimation". In: *WACV*. 2015.

[92] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. "Training a feedback loop for hand pose estimation". In: *IEEE International Conference on Computer Vision*. 2015, pp. 3316–3324.

[93] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. "Efficient model-based 3d tracking of hand articulations using kinect." In: *BMVC*. 2011.

[94] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. "Full dof tracking of a hand interacting with an object by modeling occlusions and physical constraints". In: *2011 International Conference on Computer Vision*. IEEE. 2011, pp. 2088–2095.

[95] Paschalis Panteleris, Iason Oikonomidis, and Antonis Argyros. "Using a single rgb frame for real time 3d hand pose estimation in the wild". In: *WACV*. 2018.

[96] Affan Pervez et al. "Motion encoding with asynchronous trajectories of repetitive teleoperation tasks and its extension to human-agent shared teleoperation". In: *Autonomous Robots* 43.8 (2019), pp. 2055–2069.

[97] Thammathip Piumsomboon et al. "User-defined gestures for augmented reality". In: *IFIP Conference on Human-Computer Interaction*. Springer. 2013, pp. 282–299.

[98] Sai Manoj Prakhya et al. "Sparse Depth Odometry: 3D keypoint based pose estimation from dense depth data". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4216–4223.

[99] Charles R Qi et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE* 1.2 (2017), p. 4.

[100] Charles R Qi et al. "Volumetric and multi-view cnns for object classification on 3d data". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 5648–5656.

[101] Charles Ruizhongtai Qi et al. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space". In: *Advances in Neural Information Processing Systems*. 2017, pp. 5099–5108.

[102] Chen Qian et al. "Realtime and robust hand tracking from depth". In: *CVPR*. 2014.

[103] Zeju Qiu et al. "Hand Pose-based Task Learning from Visual Observations with Semantic Skill Extraction". In: *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. 2020.

[104] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).

[105] Siamak Ravanbakhsh, Jeff Schneider, and Barnabas Poczos. "Deep learning with sets and point clouds". In: *arXiv preprint arXiv:1611.04500* (2016).

[106] Javier Romero, Dimitrios Tzionas, and Michael J Black. "Embodied hands: Modeling and capturing hands and bodies together". In: *ACM Transactions on Graphics (TOG)* 36.6 (2017), p. 245.

[107] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. "The earth mover's distance as a metric for image retrieval". In: *International journal of computer vision* 40.2 (2000), pp. 99–121.

[108] Szymon Rusinkiewicz and Marc Levoy. "Efficient variants of the ICP algorithm". In: *Third International Conference on 3-D Digital Imaging and Modeling, 2001. Proceedings*. IEEE. 2001, pp. 145–152.

[109] Paul-Edouard Sarlin et al. "Superglue: Learning feature matching with graph neural networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 4938–4947.

[110] Matteo Saveriano and Dongheui Lee. “Distance based dynamical system modulation for reactive avoidance of moving obstacles”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5618–5623.

[111] Toby Sharp et al. “Accurate, robust, and flexible real-time hand tracking”. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM. 2015, pp. 3633–3642.

[112] Adrian Spurr et al. “Cross-modal deep variational hand pose estimation”. In: *CVPR*. 2018.

[113] Srinath Sridhar et al. “Fast and robust hand tracking using detection-guided optimization”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3213–3221.

[114] Jörg Stückler and Sven Behnke. “Multi-resolution surfel maps for efficient dense 3D modeling and tracking”. In: *Journal of Visual Communication and Image Representation* 25.1 (2014), pp. 137–147.

[115] Jürgen Sturm et al. “A benchmark for the evaluation of RGB-D SLAM systems”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 573–580.

[116] Yuxiang Sun, Ming Liu, and Max Q-H Meng. “Improving RGB-D SLAM in dynamic environments: A motion removal approach”. In: *Robotics and Autonomous Systems* 89 (2017), pp. 110–122.

[117] Martin Sundermeyer et al. “Implicit 3d orientation learning for 6d object detection from rgb images”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 699–715.

[118] James S Supancic et al. “Depth-based hand pose estimation: data, methods, and challenges”. In: *ICCV*. 2015.

[119] Masahiro Suzuki, Kotaro Nakayama, and Yutaka Matsuo. “Joint multimodal learning with deep generative models”. In: *arXiv preprint arXiv: 1611.01891* (2016).

[120] Danhang Tang et al. “Opening the black box: Hierarchical sampling optimization for estimating human hand pose”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 3325–3333.

[121] Bugra Tekin, Federica Bogo, and Marc Pollefeys. “H+ o: Unified egocentric recognition of 3d hand-object poses and interactions”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4511–4520.

[122] Anastasia Tkach et al. “Online generative model personalization for hand tracking”. In: *ACM Transactions on Graphics (TOG)* 36.6 (2017), p. 243.

[123] Jonathan Tompson et al. “Real-time continuous pose recovery of human hands using convolutional networks”. In: *ACM Transactions on Graphics (TOG)* 33.5 (2014), p. 169.

[124] Tommi Tykkälä, Cédric Audras, Andrew Comport, et al. “Direct iterative closest point for real-time visual odometry”. In: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 2050–2056.

[125] Dimitrios Tzionas et al. “Capturing hands in action using discriminative salient points and physics simulation”. In: *International Journal of Computer Vision* 118.2 (2016), pp. 172–193.

[126] Ramakrishna Vedantam et al. “Generative models of visually grounded imagination”. In: *arXiv preprint arXiv:1705.10762* (2017).

[127] Chengde Wan et al. “Crossing nets: Combining GANs and VAEs with a shared latent space for hand pose estimation”. In: *CVPR*. 2017.

[128] Chengde Wan et al. “Dense 3d regression for hand pose estimation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 5147–5156.

[129] Chen Wang et al. “Densefusion: 6d object pose estimation by iterative dense fusion”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3343–3352.

[130] Wei Wang and Darius Burschka. “Dense and Deformable Motion Extraction in Dynamic Scenes Based on Hierarchical MRF Optimization in RGB-D Images”. In: *2015 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1115–1122.

[131] Youbing Wang and Shoudong Huang. “Towards dense moving object segmentation based robust dense RGB-D SLAM in dynamic scenarios”. In: *13th International Conference on Control Automation Robotics & Vision (ICARCV)*. IEEE. 2014, pp. 1841–1846.

[132] Yue Wang et al. “Dynamic graph CNN for learning on point clouds”. In: *arXiv preprint arXiv:1801.07829* (2018).

[133] Oliver Wasenmüller, Marcel Meyer, and Didier Stricker. “CoRBS: Comprehensive RGB-D benchmark for SLAM using Kinect v2”. In: *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2016, pp. 1–7.

[134] Thomas Whelan et al. “Kintinuous: Spatially Extended KinectFusion”. In: ().

[135] Thomas Whelan et al. “Robust real-time visual odometry for dense RGB-D mapping”. In: *2013 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5724–5731.

[136] Mike Wu and Noah Goodman. “Multimodal generative models for scalable weakly-supervised learning”. In: *NIPS*. 2018.

[137] Zhirong Wu et al. “3d shapenets: A deep representation for volumetric shapes”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1912–1920.

[138] Jonas Wulff, Laura Sevilla-Lara, and Michael J Black. "Optical flow in mostly rigid scenes". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 4671–4680.

[139] Linlin Yang and Angela Yao. "Disentangling Latent Hands for Image Synthesis and Pose Estimation". In: *CVPR*. 2019.

[140] Linlin Yang et al. "Aligning latent spaces for 3d hand pose estimation". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 2335–2343.

[141] Yaoqing Yang et al. "Foldingnet: Point cloud auto-encoder via deep grid deformation". In: *CVPR*. 2018.

[142] Qi Ye and Tae-Kyun Kim. "Occlusion-aware hand pose estimation using hierarchical mixture density network". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 801–817.

[143] Qi Ye, Shanxin Yuan, and Tae-Kyun Kim. "Spatial attention deep net with partial PSO for hierarchical hybrid hand pose estimation". In: *European conference on computer vision*. Springer. 2016, pp. 346–361.

[144] Tan Yu, Jingjing Meng, and Junsong Yuan. "Multi-View Harmonized Bilinear Network for 3D Object Recognition". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[145] Shanxin Yuan et al. "Bighand2. 2m benchmark: Hand pose dataset and state of the art analysis". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 4866–4874.

[146] Shanxin Yuan et al. "Depth-based 3d hand pose estimation: From current achievements to future goals". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2636–2645.

[147] Shanxin Yuan et al. "The 2017 hands in the million challenge on 3d hand pose estimation". In: *arXiv preprint arXiv:1707.02237* (2017).

[148] Manzil Zaheer et al. "Deep sets". In: *Advances in Neural Information Processing Systems*. 2017, pp. 3391–3401.

[149] Jiawei Zhang et al. "A hand pose tracking benchmark from stereo matching". In: *ICIP*. 2017.

[150] Xingyi Zhou et al. "Model-based deep hand pose estimation". In: *Twenty-Fifth International Joint Conference on Artificial Intelligence*. AAAI Press. 2016, pp. 2421–2427.

[151] Jun-Yan Zhu et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2223–2232.

[152] Christian Zimmermann and Thomas Brox. "Learning to estimate 3d hand pose from single rgb images". In: *ICCV*. 2017.