Title
Geometric– and Learning–Based Perception and Control for Robotic Systems

Author
Fahandezhsaadi, Saman

Publication Date
2021

Peer reviewed|Thesis/dissertation

Geometric– and Learning–Based Perception and Control for Robotic Systems

by

Saman Fahandezhsaadi

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Masayoshi Tomizuka, Chair
Professor Francesco Borrelli
Professor Laurent El Ghaoui

Fall 2021

Geometric– and Learning–Based Perception and Control for Robotic Systems

Abstract

Geometric– and Learning–Based Perception and Control for Robotic Systems

by

Saman Fahandezhsaadi

Doctor of Philosophy in Engineering – Mechanical Engineering

University of California, Berkeley

Professor Masayoshi Tomizuka, Chair

A reliable, accurate, and robust robotic system is highly dependent on perception, the ability of a robot to sense and interpret its environment. A variety of sensing technologies and methods can be integrated for perception purposes depending on a particular robotic setting and the surrounding. Uncertainty, environment variability, and limited sensing capabilities are factors that pose challenges to the perception task. This dissertation focuses on exploiting geometric and probabilistic characteristics as well as hidden structural properties of robotic systems and their surroundings to address some of these challenges.

A geometric state estimator is presented for an agent with the ability to measure single ranges to fixed points (anchors) in its environment. The state estimator is generic, and can be immediately applied to any robot with the range sensor. A greedy optimization algorithm is developed to select the best measurement in each time step. The selection algorithm is added to the extended Kalman filter, resulting in choosing the best measurement out of all the available range values. The effectiveness of the presented estimator algorithm is demonstrated through experimental setup for a flying robot.

The estimation accuracy is improved under the assumption that the ranging infrastructure is not perfect. A real–time restructure of the setup allows to enhance the localization accuracy of the ego agent. The estimator is incorporated into an adaptive algorithm. Using a mobile UWB ranging sensor, the mobile anchor moves to improve localization accuracy of the main robot. The algorithm reconstructs the range sensor network in real–time to minimize the covariance matrix in the extended Kalman filter. The presented algorithm is experimentally validated in a network of range sensors.

A probabilistic–based approach for pose estimation using point clouds is presented. The point registration algorithm is based on *directional statistics*, which estimates the rigid transformation (i.e. rotation matrix and translation vector) between two point cloud frames. The algorithm outputs the robot's pose estimation (location and orientation).

The framework transforms the point registration task on a unit sphere, and solves the problem in two steps of *correspondence* and *alignment*. In particular, a mixture model (as an example of directional statistics on unit sphere in $\mathbb{R}^3$) is adopted and the process of point registration has been carried out by the two phases of Expectation–Maximization algorithm. The method has been evaluated with point clouds from LiDAR sensors in an indoor environment.

A deep graph network is presented, to improve the robustness and accuracy of point registration. The framework models the point registration task based on the flexible architecture of Graph Network (GN) blocks. Three main modules–an encoder, a core, and a decoder–are responsible to perform both steps of correspondence and assignment in point matching process. The experiments and examination of the proposed model shows comparable results with other state–of–the–art geometric–or learning–based algorithm in terms of accuracy as well as robustness with regard to bad initial conditions and presence of outliers in data points. The flexibility and configurability of the framework allows to easily change, add, and/or combine various customized deep modules and mechanisms to the presented graph–based framework.

The last part of this dissertation, studies ReLU network architecture in the domain of control. The input/output domain and structure of the network and its proximity to explicit Model Predictive Control (eMPC). The mathematical equivalency of feedforward ReLU and piecewise affine function is presented, and we investigate the prospect of representing state feedback policy of eMPC as a ReLU DNN, and vice versa. A sampling based method has been developed to identify input–space regions in ReLU networks.

This dissertation is dedicated to my parents, Rezvan & Rahnama

Contents

Contents ii

List of Figures iv

List of Tables ix

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Outline and Contribution . . . 2
- 1.3 Outlook . . . 4

I Geometric State Estimation 6

2 Optimal Measurement Selection 7
- 2.1 Overview . . . 7
- 2.2 Modeling . . . 9
- 2.3 Estimator . . . 10
- 2.4 Anchor selection algorithm . . . 12
- 2.5 Experimental validation . . . 14
- 2.6 Conclusion . . . 19

3 Restructure of a ranging-based localization network 21
- 3.1 Overview . . . 21
- 3.2 System Model . . . 22
- 3.3 State estimator . . . 24
- 3.4 Mobile anchor algorithm . . . 26
- 3.5 Experimental validation . . . 30
- 3.6 Conclusion . . . 33

II Probabilistic Pose Estimation **35**

4 Point Registration with Directional Mixture Model **36**
4.1 Overview . . . 36
4.2 Methods . . . 38
4.3 Point Registration Algorithm . . . 42
4.4 Experimental Results . . . 47
4.5 Conclusion . . . 51

5 Deep Graph Network Point Registration **53**
5.1 Overview . . . 53
5.2 Related Work . . . 54
5.3 Problem Formulation . . . 56
5.4 Graph-based Point Registration . . . 60
5.5 Experimental validation . . . 63
5.6 Conclusion . . . 69

III Control and Deep Learning **70**

6 In Proximity of ReLU Architecture and eMPC **71**
6.1 Overview . . . 71
6.2 Preliminaries and Problem Formulation . . . 72
6.3 Explicit MPC and ReLU DNN . . . 79
6.4 Experimental validation . . . 81
6.5 Conclusion . . . 85

Bibliography **86**

List of Figures

2.1 A schematic of the proposed systems: an agent (here a quadcopter) operates in a space prepared with multiple radio anchors at known locations (indexed A1-A8) in the Figure. At any given instant in time, the vehicle can only measure a distance to one anchor. 8
2.2 Estimation using ultra-wideband ranging localization (optimal anchor selection). The left column of plots shows the position in $[\mathrm{m}]$, the middle column shows the velocity in $[\mathrm{m/s}]$, and the right column shows the attitude of the quadcopter in [deg.]. The experiment illustrates the comparison of true states (driven from motion caption system) against our estimator using the optimal anchor choice. As seen, quadcopter takes off and hovers for $2\mathrm{s}$, then starts moving along the y axis horizontally for $8sec.$, and finally lands at the origin. The yellow area around the mean values on the plots shows the square roots of the diagonals of EKF covariance matrix (i.e. one standard deviation). Note that the estimator does not compute the estimate in terms of yaw, pitch, and roll angles; the estimator output is simply transformed into this format as it is easier to parse in a figure. 15
2.3 The quadcopter used in our experiments as the mobile agent. The UWB ranging sensor (sees in this picture) works at the rate of approximately 60 Hz. 16
2.4 Anchor setup and quadcopter trajectory. The set of five anchors are all located at $z = 0$, and the quadcopter's motion is confined to a plane where $x = 0$, with the horizontal motion at a height of $2\mathrm{m}$. 18
2.5 Closed-loop control of the quadcopter using the presented optimal anchor selection algorithm. The plot represents the position of quadcopter. The blue, green, and red lines show the command position, the onboard position estimator (EKF), and the motion capture system output (ground truth), respectively. The yellow area around the estimation shows one standard deviation on position in each direction. 20

3.1 A schematic of the proposed systems: an agent (here a quadcopter) operates in a space prepared with combination of one mobile (indexed as M1) and four fixed radio anchors (indexed A1-A4). At the left, the ellipse representing the position uncertainty is extended in the direction which there is no anchors present. At the right, the mobile anchor moves to the area with no anchors and reduces the uncertainty around the agent. 23
3.2 Closed loop control of quadcopter using UWB ranging sensor with and without use of mobile anchor. The experiment illustrates the square roots of the diagonals of EKF covariance matrix (i.e. one standard deviation). As seen using mobile anchor, decreases uncertainty (position and velocity) dramatically in the x direction, since there are enough fixed anchors along y direction but nothing in the x direction (see Fig. 3.3). Note that the estimator does not compute the estimate in terms of yaw, pitch, and roll angles; the estimator output is simply transformed into this format as it is easier to parse in a figure. 30
3.3 Top view. Red dots represent the position of fixed anchors in both sets of experiments. The black dot is the fifth fixed anchor in the first and the initial position of mobile anchor in the second set of experiments. The set of four anchors are all located at $z = 0$. The blue path is the projection of the mobile anchor trajectory on xy plane. 31
3.4 Determinant of the inverse of the covariance matrix. The plot shows the increase of the determinant as expected when the mobile anchor moving in the direction of gradient ascent; that means the state estimate variance is decreasing. 33
3.5 Closed loop control of the quadcopter using one mobile anchor. The plot shows that the quadcopter is tracking a horizontal trajectory (dashed blue) along the y axis. Despite the fact that the method was developed for a fixed agent, the results show that it can be used for moving agents as well. 34

4.1 Two data samples drawn from Kent distribution. **Left:** Samples drawn from a single Kent PDF. Arrows show $\boldsymbol{\mu}$ the mean direction of samples as well as $\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2$ major and minor axes which depend on the orientation of samples. **Right:** Shows a mixture of two Kent distributions with different parameters. The pointcloud surface normals with different concentrations, can be modeled as a mixture of Kent distributions. 37
4.2 Example from the KITTI dataset [37]. Figures show two urban scenes pointcloud data and associated surface normals on a unit sphere. As seen, as the vehicle moves, the surface normals are also moving on the unit sphere which indicates the rotation between consecutive frames. 39
4.3 The process of preparing surface normals from pointcloud data. 1) pointcloud 2) estimating surface normals 3) outlier removal 4) *spherical* clustering. 44
4.4 Top: Example of staircase pointcloud with 20% outliers (blue points) and its associated surface normals. Bottom: Comparison of rotation and translation errors. 48

4.5 Top: Example of staircase pointcloud with 20% outliers (blue points) and its associated surface normals. Bottom: Comparison of rotation and translation errors. 50
4.6 An example of matching quality (i.e. frames alignment) in two consecutive frames from indoor scanning data set [96] using the proposed method. Purple: the model pointcloud. Green: the transformed pointcloud. 52

5.1 *Graph Network* (GN) Block. The building block of our proposed method is based on the GN block. It consist of nodes, edges, and graph–level attribute $\boldsymbol{u}$. Input/output of the block should also be in standard graph form explained in this work. An input graph attributes pass through a set of computational steps based on update/aggregation functions to result the output graph. 57
5.2 Graph Network Blocks. The proposed architecture for the task of point registration consists of three main graph network blocks: Encoder, Core, and Decoder. The Encoder embeds the first latent graph from graph representation of point clouds $\boldsymbol{P}_{\mathcal{S}}$ and $\boldsymbol{P}_{\mathcal{T}}$. The core block has been repeated M times to represent the message–passing processes in graph theory. M steps are constructed recurrently with shared parameters to also capture the temporal information which exists during iterative point cloud alignment. The Decoder extracts rigid transformation information from the last latent graphs. For more detail see Fig. 5.3 58
5.3 Diagram of the proposed deep graph network operating on a pair of point clouds. **Top Diagram** shows three main blocks of the model. The Core blocks does M –step message–passing with shared parameter in each $\mathbf{GN}_c^m$. **Bottom Diagram** shows the details of the model and the input/output in each part of the model. The first latent graphs $\boldsymbol{G}_{\mathcal{S}}^0$ $\boldsymbol{G}_{\mathcal{T}}^0$ carry the embeddings based on non–local mechanism introduced in Encoder section 5.4. The concatenation of both graphs enters the Core block when message–passing process aggregates information across both graphs, and also captures temporal relationships due to the use of shared functions and parameters in the structure of Core block. The Decoder simply extract rigid transformation which was encoded as the global entity in the graphs. At the end, the transformation performed on the source frame estimates the target frame (This has been used during the training phase for computing the loss function comparing it with the true transformation between frames). 59
5.4 Encoder GN Block. For implementing the non–local mean, the edge update function $\phi^e(\cdot)$ is a MLP with arguments as the dot product of positional data points (a scalar pairwise–interaction function) and a vector–valued non–pairwise function (also represented as a MLP). The edge–to–node aggregation function $\rho^{e\to v}(\cdot)$ is only summation (or mean in case of normalizing the sum). The rest are also MLPs which will be described in Experiential Evaluation section 5.5. . 60

5.5 Core GN Block. All the update functions $\phi^e(\cdot),\ \phi^v(\cdot), \phi^u(\cdot)$ in this block are MLPs with shared parameters across pair of point clouds and all nodes and edges. All the aggregation functions $\rho^{e\to v}(\cdot),\ \rho^{e\to u}(\cdot),\ \rho^{v\to u}(\cdot)$ are just normalized sum of the functions' input. The message-passing and recurrent property also is implemented in the structure of the Core block. 62
5.6 Decoder GN Block. This block only updates $\phi^e(\cdot),\ \phi^v(\cdot), \phi^u(\cdot)$ nodes, edges, and graph–level entities with no aggregation as the final step in the model. The decoder independently decodes the edge, node, and global attributes (does not compute relations, etc.), on each message-passing step. 63
5.7 Examples of input to the trained model and the output registered results. **Row 1 & 3:** Two point clouds (source and target), each containing 2048 points, are set as the input to the model. **Row 2 & 4:** The output point clouds have been aligned and translated, showing that the trained model correctly finds the rigid transformation between frames. 64
5.8 **Left to Right:** The progress in objects alignment (three objects: vase, airplane, and bottle) during M–step message passing process inside the core block 5.4. The training loss is computed for each processing step of the core block. The reason is to encourage the model to solve the problem in as few steps as possible, and force the model to learn the infinitesimal changes in transformation matrix. 66
5.9 Three examples of point matching in the presence of noise and outliers. 68

6.1 This example shows the level of complexity that a feedforward neural network (NN) can represent. The plot shows how just a 2-layer NN maps input-space $\boldsymbol{x} \in \mathbb{R}^2, (x_1,\ x_2)$ to the output-space $y \in \mathbb{R}$ with 7 ReLU activation units in each layer (total of 84 parameters). The network creates a complex continuous PWA function which can be a close approximation of a highly nonlinear function. . . 73
6.2 Illustration of L–layer ReLU DNN $f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_{L+1}}$. Depending on application and complexity of function to be approximated, DNN can have an arbitrary number of layers (i.e. depth) and activation units in each layer (i.e width). Note that DNNs are recognized just by the number of *hidden layers*. Also as seen, the output layer is just a linear transformation of last hidden layer without activation mapping. 74

6.3 A ReLU DNN subdivides the input-space into polytopes. In fact, each hidden layer divides the input-space from the previous layer h_{l-1}, and this recursively subdivides the input–space of the whole network for the deeper layers forward. Here we have a 2–layer ReLU net with input $x \in \mathbb{R}^2$ and four activation units in each layer. The left plot shows the pre-activation functions $f_1 = W_1 x + b_1$ that is equivalent to four hyperplanes in $\mathbb{R}^2$. Hidden units are activated in one side of their corresponding hyperplanes. The right plot shows both hyperplanes from the first and second layers in blue and red, respectively. The hyperplanes in the second layer, as seen in the plot, are not straight lines, but rather they are bent at the first layer boundaries (blue lines). When those hyperplanes pass through different regions partitioned by the first layer, they will be bent. Therefore we still have four activation boundaries for four units in layer 2, but they are not straight lines. In the right plot we can see all the regions that the network can partition on the input-space. Also it represents different affine functions over each polytope. 77

6.4 This illustration gives an entire perspective that this chapter tries to depict. ReLU DNNs represent PWA function on polyhedra which subdivide the input-space. Assuming that the input to the neural network is the parameter $x(t)$, the network can exactly act as an explicit state feedback policy. The dashed arrows indicate the needs for further study of methods -analytical or approximate- which can reconstruct the mathematical structures of each block from the other in a constructive manner. 79

6.5 The plots show a scalar PWA function and its decomposition to two convex functions, $f(x) = \gamma(x) - \eta(x)$. 82

6.6 The plots show the explicit solution to the mp-LP (6.14) using MPT3 toolbox [46]. **Top & center:** show plots of the optimizers $z_1^* = \gamma(x)$ and $z_2^* = -\eta(x)$ which both are functions of the parameter x. the solution exactly results the PWA function (6.13). **bottom:** The plot depicts the optimal value $J^*(x)$ which is a function of parameter x and also is convex and PWA as we know from the theory of mp-LP (corollary 11.5 in [15]). 84

List of Tables

2.1 Comparison of estimation error from experiments 17

3.1 Comparison of estimation error from experiments 32

4.1 Comparison of PR Methods 51

5.1 Comparison of Deep point registration Methods, in [deg.] & [m] 65
5.2 Accuracy results in the presence of noise & outliers 67
5.3 Comparison of Inference time in millisecond 69

Acknowledgments

First and foremost, I have the utmost respect and gratitude for my advisor Professor Masayoshi Tomizuka, for his dedication, enthusiasm, and his vast knowledge that helps me during my PhD study. I was fortunate to have him as my mentor, and he was a source of inspiration and support throughout my PhD research.

I also would like to sincerely thank Professor Francesco Borrelli and Professor Laurent El Ghaoui for their help and support and being members of my dissertation committee as well as my qualification exam committee. I was fortunate to conduct research as an undergraduate student in Professor Borrelli's MPC Lab, and learn about model predictive control which has impacted my research during my graduate studies. I also had the opportunity to learn about convex optimization as a crucial tool in my research from Professor El Ghaoui.

I'd like to thank Professor Mark W Mueller for his collaboration and guidance during the beginning of my PhD. Also, I'd like to give special thanks to my collaborators Di Wang and Yiyang Zhou.

Chapter 1

Introduction

This dissertation presents multiple of closely connected approaches in geometric–, probabilistic–, and learning–based perception and control of robotic systems in various environments equipped with range sensors, LiDAR sensors, or any other 3D scanning technologies. The theoretical development in connection with each framework has been equipped with experimental results that confirms the performance improvement in each application.

1.1 Motivation

A localization technology should be reliable and robust in different situations like in bad weather or in indoor areas with smoke or dust. Also any sensing technology for robots has to work in area with limited accessibility or no pre–installed or destroyed infrastructure in situations like building inspection or rescue mission in disaster area. It also should be flexible in multi–robot cooperation settings. And finally it should be low cost. But since there are physical limitations in the hardware level in these technologies, it is still possible to improve the performance by developing reliable algorithms in the software level. Audio- or radio-beacon systems are very reliable, flexible and low cost compared to other sensing technologies like vision-based or laser-based systems.

The localization accuracy of range-based robots is sensitive to the ranging sensor layout, therefore it is impossible to attain reasonable accuracy in the presence of the constraints (e.g. limited line–of–sight, crowded indoor settings, dynamic environment with moving objects) in the environment. without careful planning, which may be difficult in harsh conditions.

Light Detection and Ranging (LiDAR) as well as 3D scanning sensor technologies share similar characteristics with ranging sensor, though the former produces extremely rich data from the environment comparing with the latter. While the beacon infrastructure of range sensors is missing in LiDAR sensors, localization frameworks such as SLAM and keypoint matching [28, 24, 14] on robots equipped with LiDAR incorporate *keypoints* and *landmarks* as virtual beacons/anchors in the scenes during the process of localization and

mapping. In addition, for robotic systems using the 3D LiDAR technology the problem of point matching (or point registration) is an initial key step towards an accurate and reliable pose estimation. Therefor the second part of this dissertation focuses on the problem of point registration as an essential part of robot localization and state estimation.

In addition to the probabilistic view, with the advances in deep neural networks in the domain of supervised learning, recently a new deep learning–based view of point registration problem has been emerged. This view adds new mathematical tools and empirical modules in tackling the persistent problems of point cloud registration to the existing body of work. Classical methods (mostly geometric–and hybrid with probabilistic interpretation of corresponding points, as the first step in solving the problem) dealing with point registration are subject to certain limitations such as converge to a local optimum near the initialization which makes the problem highly sensitive to the choice of initial condition. Also, the computation speed (number of iteration to convergence) and accuracy of these methods suffer from the sensitivity to the amount of outliers, noise, and repetitive geometry in the scene. The use of deep neural network in the point clouds' domain has its own challenges due to the fact that point clouds are unordered and unstructured [141, 140, 121, 97, 53, 69]. Nevertheless in recent years a lot of progress has been made on performing various tasks (e.g. object classification and detection, semantic segmentation, localization) [102, 101, 143]. And still it remains an active area of research due to unresolved obstacles, and since point clouds provide a very rich information of surrounding which is a decisive part for the success of complex robotic systems such as autonomous driving.

1.2 Outline and Contribution

This dissertation has been organized in the following order. In chapter 2 a geometric–based state estimator is derived for an agent with the ability to measure single ranges to fixed points (anchors) in its environment. The state estimator makes no agent–specific assumptions and it is generic, and can be immediately applied to any rigid body agent with the radio–beacon range sensor. As the considered system can only make a single range measurement at a time, a greedy optimization algorithm has been presented for selecting the best measurement in each time step. The selection algorithm is added to the extended Kalman filter which minimizes the variance matrix in the beginning of the correction step of the Kalman filter, resulting in choosing the best measurement out of all the available range values at each time. An important property of the partial derivative of the measurement model has been exploited in the process of computing the best anchor to range. Experiments in an indoor testbed using an externally controlled multicopter demonstrate the efficacy of the proposed algorithm, specifically showing an improvement over a naïve strategy of a fixed sequence of measurements. In separate experiments, the algorithm is also used in feedback control, to control the position of the multicopter. Other anchor selection methods like [130, 57] only work in a collaborating manner where using information from multiples of robots, or completely ignore the geometric property of ranging

sensor technologies. Also they mostly rely on probabilistic properties of Kalman filter like Fisher information which are not as reliable as the deterministic geometric property that is exploited in this presented work.

In chapter 3, the localization accuracy (and eventually the state estimation accuracy) is further improved for robot agents by introducing a real–time restructure of range sensor setup. In addition to the optimal algorithmic scheme introduced in chapter one, the framework in this chapter provides a reliable state estimation with accurate localization characteristics in the absent of a structured range sensor setting. The method improves the localization accuracy of robotic systems operating in a range–based localization network, and it is favorable especially when the robots operate in harsh environments where the access to a robust and reliable localization system is limited. Although it is only designed to work with as few as two moving robots, the proposed method can be extended to a multi–agent cooperative positioning framework using the ranging sensor technologies. A state estimator is used for a six degree of freedom object using inertial sensors as well as an Ultra–wideband (UWB) range measurement sensor. The estimator is incorporated into an adaptive algorithm, improving the localization quality of an agent by using a mobile UWB ranging sensor, where the mobile anchor moves to improve localization accuracy of the main robot. The algorithm reconstructs the range sensor network in real–time to minimize the covariance matrix in the extended Kalman filter. The proposed algorithm is experimentally validated in a network of range sensors consisting of one mobile and four fixed anchors. The main agent which should be localized and the mobile anchor are both quadcopter. The algorithm also show similar result when using multiple mobile anchors in restructuring the network of range sensors. Although the range measurement based localization is deployed in various multi–agent positioning and wireless sensor networks [90, 106, 27, 132, 105], the idea of mobile anchors (i.e. the range sensors that adapt their locations in such a way to improve the position accuracy of the ego agent) is missing in all of the previous works. The new approach proposed in chapter 3 addresses this limitation that can be incorporated naturally in any previous ranging–based perception framework.

Chapter 4 presents a probabilistic–based approach for pose estimation using point clouds. The point registration algorithm is based on *directional statistics* [122, 93, 75, 67], which estimates the rigid transformation (i.e. rotation matrix and translation vector) between two point cloud frames. The final result will be the estimation of robot's pose (its location and orientation) that is equipped with a LiDAR sensor capable of scanning the 3D environment. The method improves the robustness of point registration and consequently the robot localization in the presence of outliers which always occurs due to occlusion, dynamic objects, and sensor errors. The framework transforms the point registration task on a unit sphere, and solves the problem in two steps of *correspondence* and *alignment*. In particular, a mixture model (as an example of directional statistics on unit sphere in $\mathbb{R}^3$) is adopted and the process of point registration has been carried out by the two phases of Expectation–Maximization algorithm. Other methods mostly use the positional information of point clouds, but the presented method utilizes both position as well as surface normals of each data point for pose estimation. The method has been

evaluated with point clouds from LiDAR sensors in an indoor environment. Previous methods of point registration like ICP or GICP are sensitive to outliers and initialization of the algorithm, both of these disadvantages have been resolved and the results show improvements in those fronts.

Chapter 5 presents a novel deep neural network for pose estimation which improves the shortcomings in previously introduced models for the task of point registration. To remedy prior limitations, a deep point registration method consists of blocks of graph structure has been introduced. The graph–based architectures have demonstrated *relational inductive biases* [7, 21, 98, 144], a key property in deep learning algorithms that leads to combinatorial generalization. The framework models the point registration task based on the flexible architecture of Graph Network (GN) blocks. Three main modules–an encoder, a core, and a decoder–are responsible to perform both steps of correspondence and assignment in point matching process. The experiments and examination of the proposed model shows comparable results with other state–of–the–art geometric–or learning–based algorithm in terms of accuracy as well as robustness with regard to bad initial conditions and presence of outliers in data points. The flexibility and configurability of the framework allows to easily change, add, and/or combine various customized deep modules and mechanisms to the presented graph–based framework.

Parallel to the previous chapter concerning with the study of mathematical structure of graph neural network, the last part of this dissertation investigates specific deep neural architectures in a different task domain. Chapter 6 examines the input/output domain and structure of famous Multi–Layer Perceptron (or MLPs) and its link to Model Predictive Control (MPC). As the first step, feedforward rectifier (ReLU) MLPs and their relation to the piecewise affine (PWA) functions as an essential linking mathematical entity has been studied. Afterward, this chapter investigates the prospect of representing explicit state feedback policy of model predictive control (eMPC) as a ReLU DNN, and vice versa. The complexity and architecture of the DNN has been discussed by introducing bounds on domain partitioning of such neural networks. A sampling based method has been developed for input–space identification of ReLU networks. The method ultimately approximates a PWA function over polyhedral regions as an equivalent for the networks. Inverse multiparametric linear or quadratic programs (mp–LP or mp–QP) are another connection in reconstruction of constraints and cost function given a PWA function that has been investigated as a potential way of representing eMPC solution by means of PWA functions and eventually ReLU networks.

1.3 Outlook

The goal of this dissertation is to highlight the benefits of carefully exploiting geometric and probabilistic characteristics as well as hidden structural properties of robotic systems and their surroundings in the field of perception and control.

The geometric characteristics can be utilized to further improve the existing algorithms

for localization and state estimation. The probabilistic properties such as directional statistics in this dissertation was employed to extract probabilistic aspects of the perception modules to enhance the quality of pose estimation. In addition, the use of LiDAR–based sensors provides a rich set of information from the environment that anticipates the incorporation of efficient deep learning–based approaches to cope with the huge amount of processing data at each time.

The experimental parts of this dissertation demonstrate the theoretical implication of incorporating those characteristics to improve the accuracy and robustness of state estimation in robotic systems. The experiments and discussions also illuminate the challenges that needs to be overcome in the future, so that the combination of all the presented theatrical findings can be deployed successfully on a wide range of robots and dynamical systems.

Part I

Geometric State Estimation

Chapter 2

Optimal Measurement Selection

2.1 Overview

Low cost, flexible, and reliable localization technology is a key enabling technology for robotics. One of the main current limitations for the deployment of autonomous agents is the agents' ability to reliably and accurately determine their position. Various different sensing technologies exist for localization, ranging from purely self-contained on the agent to globally distributed satellite navigation.

Different technologies represent a variety of different trade-offs, with varying requirements of computational power, electric power, precision, reliability, and accuracy. Perhaps the most widely used localization technology relies on satellites (e.g. GPS or Galileo) – these systems work reliably and accurately when the agent has a clear line-of-sight to the satellites (typically, outdoors, and far from tall structures), some example robotic systems are [23, 134, 51, 6, 52]. However, the reliance on a clear line of sight to the satellites is also the main drawback, leading to very poor (or nonexistent) localization in the presence of tall structures (e.g. in cities) or indoors.

In research laboratories, a popular in-door alternative is to use optical motion capture equipment, which can yield extremely precise measurements (errors on order of millimetres) at high rates, but only over small volumes (on the order of $100m^3$). Such systems provide extremely rich data, but they are expensive, fragile, and are very constrained – for examples of robotic systems relying on motion capture are [50, 77, 73].

Another paradigm relies exclusively on sensors on the agent itself, for example cameras or laser range finders. Here, the agent may fuse the measurements with other sensors, to simultaneously build a map of its environment, and localize within it (the SLAM problem) [36, 88, 112, 20]. Such systems are attractive, since they are self-contained, but also require substantial computational power, heavy sensors/cameras – this leads to large, heavy, complex, and energy-hungry agents. Such systems may also be fragile, especially in environments with visual changes (e.g. smoke, changing light conditions and shadows, etc.).

Alternatively, radio- or audio-beacons can be installed indoor, to build an indoor analogue of GPS, such as in [99, 62, 114, 100, 84]. Such systems may be created out of relatively low-cost components, provide high-quality localization over large areas, and do not impose particularly large restrictions on the individual agents using the localization system. However, they do require the installation of infrastructure, and as such are less flexible than vision-based systems. Ultra-wideband (UWB) radio ranging is an example of such localization system. In [57, 31], this type of technology (e.g. radio beacon) is used to solve the SLAM problem based on the robot cooperation with sensor networks.

Utilizing the UWB as range sensor for localization purposes is well-developed in the literature. Range-only SLAM is a precise way of localizing wireless sensor networks (WSN) node positions [120]. In [30], the UWB ranging sensor is used for the range-only SLAM approach. In [130], a method is described based on the Fisher information matrix of the Kalman filter which improves the target tracking accuracy of the wireless network. The method is related to our proposed method in this chapter which selects sensors for future measurements.

A schematic of UWB ranging system is given in Fig. 2.1. Furthermore, such systems typically use active measurements, wherein a measurement involves the transmission of a radio/audio message from the localization infrastructure to/from the agent. Since these communications use the same frequency band, this imposes a constraint on the number of simultaneous measurements.

In this chapter, a flexible state estimator, and an algorithm for selecting the optimal localization measurement will be presented, so that an agent may maximize its localization quality. A first-principles model is developed for an autonomous agent localizing by measuring distances to fixed (known) locations in the world, which is incorporated in a Kalman filter for six degrees-of-freedom (6DOF) state estimation. The results are validated in a series of experiments, where the estimator is deployed on a low-cost quadcopter system.

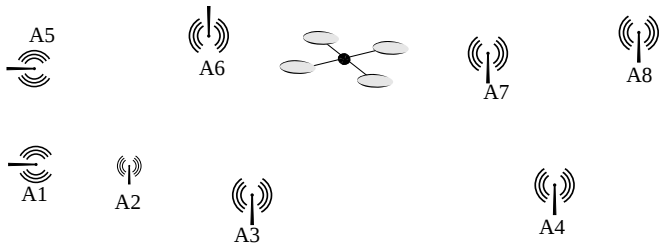

Figure 2.1: A schematic of the proposed systems: an agent (here a quadcopter) operates in a space prepared with multiple radio anchors at known locations (indexed A1-A8) in the Figure. At any given instant in time, the vehicle can only measure a distance to one anchor.

2.2 Modeling

We consider the problem of estimating the state of an agent, modelled as a generic six degree of freedom rigid body, equipped with an inertial measurement (accelerometer and rate gyroscope), and with the ability to measure the distance to any of a fixed set of points in its environment. The body's degrees of freedom are three in translation, and three in rotation; yielding a six-dimensional state vector to estimate. The goal is to have as general as possible a model, so that the resulting estimator may be applied to a variety of types of agents without modification.

Equations of motion

The convention here is of using bold-face symbols for vector/matrix quantities, and regular font for scalars. Specifically, the position of the object is denoted as $\boldsymbol{x}$, expressed in a coordinate system fixed with respect to the ground. The object's velocity and acceleration are given respectively by $\boldsymbol{v}$ and $\boldsymbol{a}$, again expressed in the ground. The orientation of the object is encoded with the rotation matrix $\boldsymbol{R}$, and the angular velocity is given by $\boldsymbol{\omega}$. The rotation matrix is defined so that multiplication by $\boldsymbol{R}$ is equivalent to a coordinate transformation to the inertial frame, from the body-fixed frame. The time derivatives of these quantities are given as

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x} = \boldsymbol{v} \tag{2.1}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{v} = \boldsymbol{a} \tag{2.2}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{R} = \boldsymbol{R}\boldsymbol{S}(\boldsymbol{\omega}) \tag{2.3}$$

where $\boldsymbol{S}(\boldsymbol{\omega})$ is the skew-symmetric matrix version of the cross product, so that $\boldsymbol{S}(\boldsymbol{x})\boldsymbol{y} = \boldsymbol{x} \times \boldsymbol{y}$. Note we do not use the derivative of the angular velocity, as the angular velocity can be reliably estimated directly from the rate gyroscope outputs.

Inertial measurements

The agent's inertial measurement unit outputs accelerometer and rate gyroscope measurements, $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$, respectively. Both sensors are assumed to be well-calibrated, specifically having no scale errors nor bias offset.

The rate gyroscope measures the angular velocity, corrupted by an additive noise $\boldsymbol{\nu}_\gamma$, and the measurement is modeled as

$$\boldsymbol{\gamma} = \boldsymbol{\omega} + \boldsymbol{\nu}_\gamma \tag{2.4}$$

The accelerometer measures the 'proper acceleration' of the vehicle, in the vehicle's body-fixed coordinate system, as given by:

$$\boldsymbol{\alpha}_m = \boldsymbol{R}^{-1}\left(\boldsymbol{a} - \boldsymbol{g}\right) + \boldsymbol{\nu}_\alpha \tag{2.5}$$

where we again assume corruption by an additive noise $\boldsymbol{\nu}_\alpha$, and $\boldsymbol{g}$ is the gravitational acceleration vector, constant in the earth-fixed frame – see [66] for a good tutorial. A typical 'z-up' coordinate system would have $\boldsymbol{g} = (0, 0, -9.81)\,\mathrm{m/s^2}$.

Range measurement system

At discrete times, the agent is able to measure the distance from itself to one of a set of fixed positions in the world (here, called 'anchors'). The anchors are at known positions $\boldsymbol{p}_i$ in the world, and a measurement ρ_i to anchor i is modelled as

$$\rho_i = \|\boldsymbol{x} - \boldsymbol{p}_i\| + \nu_\rho \tag{2.6}$$

where $\|\cdot\|$ is the Euclidean norm, and ν_ρ is a scalar, additive noise. Note that this assumes that the radio antenna is located at the same point as the inertial measurement unit; this assumption can be relaxed easily however. Furthermore, no dependency is assumed on the orientation, though this has been shown to be a potentially important effect [65].

A single distance measurement between the agent and an anchor consists of a set of four radio messages, which allow the agent and the anchor to determine the distance between them by measuring the time-of-flight of the radio signal (see [84] for a similar scheme). The agent can communicate only with one anchor at a time, meaning that only a single range measurement can be taken at any instant in time.

2.3 Estimator

The goal of the estimator is to estimate the 12-element state of the rigid body agent, using measurements from the inertial measurement unit and the range measurements. We create a "kinematic" state estimator for a generic 6DOF object, making specifically no assumptions on the forces or torques acting on the system. This yields a flexible estimator, that may be readily applied to a variety of rigid bodies; the flexibility comes at the cost of some precision (if we had an accurate model of the forces/torques acting on the agent, this information could be used to improve the estimator performance).

The estimator is based on the Extended Kalman Filter (EKF) [118], specifically using the technique of [83] to encode an attitude in the state with correct-to-first-order statistics. It is worth mentioning that no trilateration method has been used to calculate the position of the agent from a set of range measurements. The estimator only relies on a single measurement at each step of EKF.

Although the 6DOF agent has twelve states, the estimator's stochastic state $\boldsymbol{\xi}$ is 9 dimensional:

$$\hat{\boldsymbol{\xi}} = \left(\hat{\boldsymbol{x}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{\delta}}\right) \tag{2.7}$$

with the hat denoting estimated quantities, and where $\boldsymbol{\delta}$ is an "attitude error" measure, assumed to be small. The estimator uses a redundant attitude representation, with a 'reference attitude' $\boldsymbol{R}_{\text{ref}}$ and the attitude error $\hat{\boldsymbol{\delta}}$ combined yielding the estimator's attitude estimate $\hat{\boldsymbol{R}}$ which is

$$\hat{\boldsymbol{R}} = \boldsymbol{R}_{\text{ref}}\left(\boldsymbol{I} + \boldsymbol{S}(\hat{\boldsymbol{\delta}})\right) \tag{2.8}$$

with $\boldsymbol{I}$ the identity matrix. This representation allows for a singularity-free attitude estimation using only a three-dimensional representation of the attitude error. This is achieved by enforcing the requirement that $\boldsymbol{\delta}$ is zero after each Kalman filtering step – a complete discussion of this approach is given in [83].

The estimator does not include the angular velocity $\boldsymbol{\omega}$ as a state, and instead uses the measurement from the rate gyroscope directly. This is justified by the high-quality measurements from modern rate gyroscopes, and is a standard approach in e.g. attitude estimation for satellites (see, e.g. [76]). Not only is this conceptually simpler than encoding additional states, it substantially reduces the computational complexity of the resulting state estimator (since the computational complexity of a Kalman filter scales approximately like the number of states cubed).

In the prediction stage of the extended Kalman filter, the output of the accelerometer and rate gyroscope are used, so that the acceleration is given by:

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{v} = \boldsymbol{R}\boldsymbol{\alpha}_m + \boldsymbol{g} - \boldsymbol{R}\boldsymbol{\nu}_\alpha \tag{2.9}$$

The orientation differential equation is rewritten in terms of $\boldsymbol{\delta}$, with specifically

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\delta} = \boldsymbol{\gamma} - \boldsymbol{\nu}_\gamma \tag{2.10}$$

It is assumed that the sensor noise terms $\boldsymbol{\nu}_\alpha$ and $\boldsymbol{\nu}_\gamma$ are zero-mean, and spatially and temporally independent, so that they can be straight-forwardly modelled as process noise in the Kalman filter formulation.

The output from the ranging radios is used for the estimator's measurement update step. Specifically, given a measured distance ρ_i (with the subscript i indicating the choice of anchor to which the range was measured), the measurement equation (2.6) can be linearized straight-forwardly to apply the Extended Kalman filter formulation.

Since the measurement model is the distance of the agent from the anchor, the partial derivative with respect to the estimator state has an interesting property:

$$\boldsymbol{H}_i := \frac{\partial \rho_i}{\partial \boldsymbol{\xi}} = \left(\frac{\partial \rho_i}{\partial \boldsymbol{x}}, \frac{\partial \rho_i}{\partial \boldsymbol{v}}, \frac{\partial \rho_i}{\partial \boldsymbol{\delta}} \right) \tag{2.11}$$

$$\frac{\partial \rho_i}{\partial \boldsymbol{x}} = \frac{\boldsymbol{x} - \boldsymbol{p}_i}{\|\boldsymbol{x} - \boldsymbol{p}_i\|} =: \boldsymbol{e}_i \tag{2.12}$$

$$\frac{\partial \rho_i}{\partial \boldsymbol{v}} = \frac{\partial \rho_i}{\partial \boldsymbol{\delta}} = \mathbf{0} \tag{2.13}$$

This means that the measurement sensitivity is the unit vector in the direction of the agent from the anchor $\boldsymbol{e}_i$ – a very intuitive property that will be exploited in the next section to determine which anchor i should be used for the measurement.

The Kalman filter also computes an estimated covariance matrix, $\boldsymbol{\Sigma}$ relying on the partial derivatives and using the approach of [83]. The matrix may be partitioned into blocks, as below

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{\boldsymbol{xx}} & \boldsymbol{\Sigma}_{\boldsymbol{xv}} & \boldsymbol{\Sigma}_{\boldsymbol{x\delta}} \\ \boldsymbol{\Sigma}_{\boldsymbol{vx}} & \boldsymbol{\Sigma}_{\boldsymbol{vv}} & \boldsymbol{\Sigma}_{\boldsymbol{v\delta}} \\ \boldsymbol{\Sigma}_{\boldsymbol{\delta x}} & \boldsymbol{\Sigma}_{\boldsymbol{\delta v}} & \boldsymbol{\Sigma}_{\boldsymbol{\delta\delta}} \end{bmatrix} \in \mathbb{R}^{9\times 9} \tag{2.14}$$

with e.g. $\boldsymbol{\Sigma}_{\boldsymbol{x\delta}}$ the 3×3 cross-covariance between the position and attitude states.

2.4 Anchor selection algorithm

As the agent is only capable of measuring the distance to a single anchor at any given time, there is the freedom to choose which anchor. A simple algorithm to use is to proceed sequentially through the list of anchors, consistently following some pre-determined ordering. Here, instead, we describe a computationally efficient and greedy selection algorithm that maximizes the information gain from the anchors at each time step. This is done, specifically, by choosing to get that measurement which produces the largest decrease in the estimator's variance, using the matrix trace as measure of size. This is closely related to the information matrix (Fisher information), which for a Gaussian case is the same as the inverse of covariance matrix, and provides the measure of information about the state present in the observations [85]. This means by minimizing the covariance matrix, the maximum of available information in the measurements can be extracted.

Given a measurement ρ_i from anchor i, the Kalman filter's covariance is updated at each time instant according to the standard Kalman filter equations:

$$\boldsymbol{K}_i = \boldsymbol{\Sigma}\boldsymbol{H_i}^\top (\boldsymbol{H_i}\boldsymbol{\Sigma}\boldsymbol{H_i}^\top + r)^{-1} \tag{2.15}$$

$$\boldsymbol{\Sigma}_i^+ = (I - \boldsymbol{K}_i\boldsymbol{H_i})\,\boldsymbol{\Sigma} \tag{2.16}$$

where Σ_i^+ is the updated covariance matrix after a measurement update, $\boldsymbol{K}_i$ is the EKF gain matrix, and $\boldsymbol{H_i}$ is the measurement matrix as computed in (2.11). By substituting (2.15) in (2.16), we define the change in covariance due to a measurement update from anchor i as $\Delta\boldsymbol{\Sigma}_i$

$$\Delta\boldsymbol{\Sigma}_i = \boldsymbol{\Sigma}_i^+ - \boldsymbol{\Sigma} = -\boldsymbol{\Sigma}\boldsymbol{H}_i^\top(\boldsymbol{H}_i\boldsymbol{\Sigma}\boldsymbol{H}_i^\top + r)^{-1}\boldsymbol{H}_i\boldsymbol{\Sigma} \tag{2.17}$$

with r the (scalar) variance of the ranging noise ν_ρ. Since the measurements are scalar, the matrix inverse is simply an ordinary division. Also, the matrix H_i is sparse, therefore (2.17) can be decomposed as

$$\Delta\boldsymbol{\Sigma}_i = \frac{-1}{\boldsymbol{e}_i^\top\boldsymbol{\Sigma}_{\boldsymbol{xx}}\boldsymbol{e}_i + r}\begin{bmatrix}\boldsymbol{\Sigma}_{\boldsymbol{xx}}\boldsymbol{e}_i\\ \boldsymbol{\Sigma}_{\boldsymbol{xv}}\boldsymbol{e}_i\\ \boldsymbol{\Sigma}_{\boldsymbol{x\delta}}\boldsymbol{e}_i\end{bmatrix}\begin{bmatrix}\boldsymbol{\Sigma}_{\boldsymbol{xx}}\boldsymbol{e}_i\\ \boldsymbol{\Sigma}_{\boldsymbol{xv}}\boldsymbol{e}_i\\ \boldsymbol{\Sigma}_{\boldsymbol{x\delta}}\boldsymbol{e}_i\end{bmatrix}^\top \tag{2.18}$$

consisting of the projection of the variance onto the unit vector to the anchor i.

By comparing this change in covariance, different potential measurements (to anchors at different locations) at any given time can be compared. Notable is the intuitive form that this change takes, which will be exploited to generate an easily computed metric, below.

Minimizing trace of covariance matrix

The performance metric we minimize is the trace of the covariance matrix after the measurement, that is $\mathrm{tr}\left(\boldsymbol{\Sigma}_i^+\right)$. The trace of covariance matrix is the mean of the norm of the error squared of state estimator (i.e. $\mathrm{tr}\left(\boldsymbol{\Sigma}_i^+\right) = \mathbb{E}||\boldsymbol{\xi} - \hat{\boldsymbol{\xi}}||^2$). This metric perfectly makes sense, since the goal is to reduce the estimation error as much as possible [17].

From the definition of $\Delta\boldsymbol{\Sigma}_i$ in (2.18), and the linearity of the trace operator, it follows that this is equivalent to maximizing $\mathrm{tr}\left(\Delta\boldsymbol{\Sigma}_i\right)$ (i.e. by maximizing the difference between covariance of Kalman filter before and after measurement, the algorithm chooses the anchor which reduces the covariance trace the most).

Starting from equation (2.17), using the cyclic property of matrix traces (e.g. $\mathrm{tr}(\boldsymbol{AB}) = \mathrm{tr}(\boldsymbol{BA})$), we simplify to get

$$\mathrm{tr}(\Delta\boldsymbol{\Sigma}_i) = -\frac{\boldsymbol{H}_i\boldsymbol{\Sigma}\boldsymbol{\Sigma}\boldsymbol{H}_i^\top}{\boldsymbol{H}_i\boldsymbol{\Sigma}\boldsymbol{H}_i^\top + r} \tag{2.19}$$

This can be simplified further by exploiting the sparsity of $\boldsymbol{H}_i$, to yield

$$\mathrm{tr}(\Delta\boldsymbol{\Sigma}_i) = -\frac{\|\boldsymbol{\Sigma}_{\boldsymbol{xx}}\boldsymbol{e}_i\|^2 + \|\boldsymbol{\Sigma}_{\boldsymbol{xv}}\boldsymbol{e}_i\|^2 + \|\boldsymbol{\Sigma}_{\boldsymbol{x\delta}}\boldsymbol{e}_i\|^2}{\boldsymbol{e}_i^\top\boldsymbol{\Sigma}_{\boldsymbol{xx}}\boldsymbol{e}_i + r} \tag{2.20}$$

At a given instant in time, the system computes $\Delta\boldsymbol{\Sigma}_i$ for each anchor i in the network, and then selects the maximizing anchor. This is a simple computation, requiring only

a small number of multiplications to compute, and therefore easily implemented on computationally limited hardware. (i.e. Using (2.20) with the necessary number of fixed points in the space (usually 5 or 6), the computation cost is significantly small considering relatively powerful microcontroller used in today's robotic systems).

2.5 Experimental validation

The approach is validated in experiment, where a quadcopter is used as the autonomous agent. A first set of experiments is an ensemble, showing the performance of the algorithm for the quadcopter under external control, so that the motion is repeatable. These experiments compare performance when using the greedy optimization to that when using a fixed, sequential measurement sequence. The second experiment demonstrates closed-loop control of the quadcopter, using the resulting state estimate for feedback control.

Experimental setup

The proposed algorithm was tested on a Crazyflie 2.0 quadcopter (shown in Fig. 2.3), with approximate mass of $30\mathrm{g}$, and a scale of approximately $105\mathrm{mm}$. The quadcopter is equipped with an STM32F4 microcontroller, uses an Invensense MPU9250 inertial measurement unit, and a Decawave DW1000 module for the ultra-wideband ranging measurements. The anchors shared the same computational and sensing hardware. All computations for the state estimation (including the measurement selection) were performed on the microcontroller. Measurements from the accelerometer and rate gyroscope were taken at $500\mathrm{Hz}$, and range measurements were taken at approximately $60\mathrm{Hz}$. The estimator performance is quantified by using a ceiling-mounted motion capture system, whose measurements are taken as ground truth.

The anchor arrangement as well as the quadcopter's commanded trajectory can be seen in Fig. 2.4. In general at least four anchors is needed in order to estimate a unique position in the space (otherwise the estimate of position may be arbitrarily rotated, reflected, or even translated). But in real-time implementation, it is a good practice to have more than four anchors for redundancy. Since the estimator uses only one range measurement at a time from one anchor, increasing the number of anchors will not affect the results theoretically, though this should be investigated in real-time experiment. Notable is that three anchors are placed in very close proximity to one another, so that their measurements convey very similar information. This was chosen so as to highlight the effect of the greedy optimization algorithm that was proposed in this chapter.

Estimation only

In the first set of experiments, the state estimate is not used for control, and instead an offboard controller is used. This offboard controller uses measurements from the motion

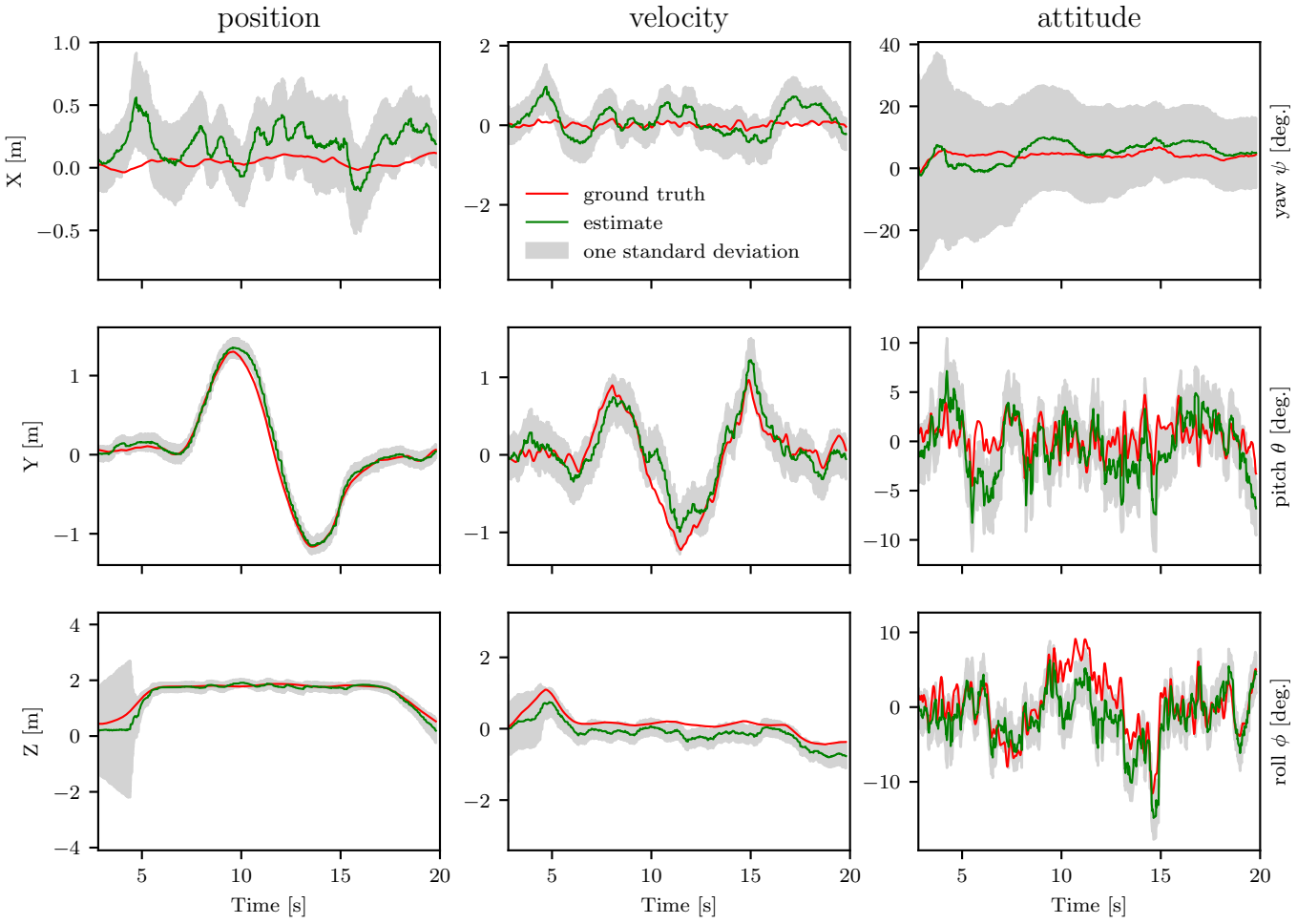

Figure 2.2: Estimation using ultra-wideband ranging localization (optimal anchor selection). The left column of plots shows the position in $[\mathrm{m}]$, the middle column shows the velocity in $[\mathrm{m/s}]$, and the right column shows the attitude of the quadcopter in [deg.]. The experiment illustrates the comparison of true states (driven from motion caption system) against our estimator using the optimal anchor choice. As seen, quadcopter takes off and hovers for $2\mathrm{s}$, then starts moving along the y axis horizontally for $8sec.$, and finally lands at the origin. The yellow area around the mean values on the plots shows the square roots of the diagonals of EKF covariance matrix (i.e. one standard deviation). Note that the estimator does not compute the estimate in terms of yaw, pitch, and roll angles; the estimator output is simply transformed into this format as it is easier to parse in a figure.

Figure 2.3: The quadcopter used in our experiments as the mobile agent. The UWB ranging sensor (sees in this picture) works at the rate of approximately 60 Hz.

capture system, and consists of a set of cascaded controllers similar to that of [73]. The use of the offboard controller resulted in repeatable experiments, where the quadcopter moved along very similar trajectories for each flight (which therefore yields a fair comparison). For all experiments, the estimator of Section 2.3 runs exclusively on the agent's microcontroller.

The estimator outputs were examined and compared with two different EKF settings: when the quadcopter sequentially ranges to anchors (the naïve approach), and when the quadcopter uses the optimization of Section 2.4. The reference trajectory path is also shown in Fig. 2.4.

The experiment was repeated ten times for each algorithm, and the resulting root mean square error (RMSE) for each trial are shown in Table 2.1. Due to the system's stochastic nature, the best run using the naïve sequential selection approach is better than the worst run using the optimization algorithm. Nonetheless, a clear improvement is observed on average, with an RMSE reduction of approximately 11% in position and velocity, and 17% in attitude when using the optimal measurement selection algorithm.

Fig. 2.2 shows experimental data from a representative trial using the optimal anchor selection algorithm. The graph shows that the estimated state is close to the ground truth

	Root Mean-Square Error (RMSE)					
trials	Position [m]		Velocity [m/s]		Attitude [deg.]	
	Sequential ranging	Optimal anchor selection	Sequential ranging	Optimal anchor selection	Sequential ranging	Optimal anchor selection
1	0.286	0.274	0.638	0.536	8.1	5.3
2	0.278	0.267	0.440	0.432	6.9	4.8
3	0.312	0.291	0.635	0.585	8.9	5.7
4	0.236	0.245	0.415	0.474	7.9	7.5
5	0.489	0.363	0.875	0.731	7.6	7.9
6	0.347	0.273	0.589	0.525	6.7	8.6
7	0.310	0.263	0.585	0.489	8.7	6.7
8	0.436	0.383	0.821	0.690	14.0	7.6
9	0.228	0.262	0.436	0.426	6.4	6.4
10	0.422	0.329	0.866	0.593	8.3	8.2
Avg.	0.334	0.295	0.630	0.548	8.4	6.9
diff.	-11.7%		-13.0%		-17.5%	

Table 2.1: Comparison of estimation error from experiments

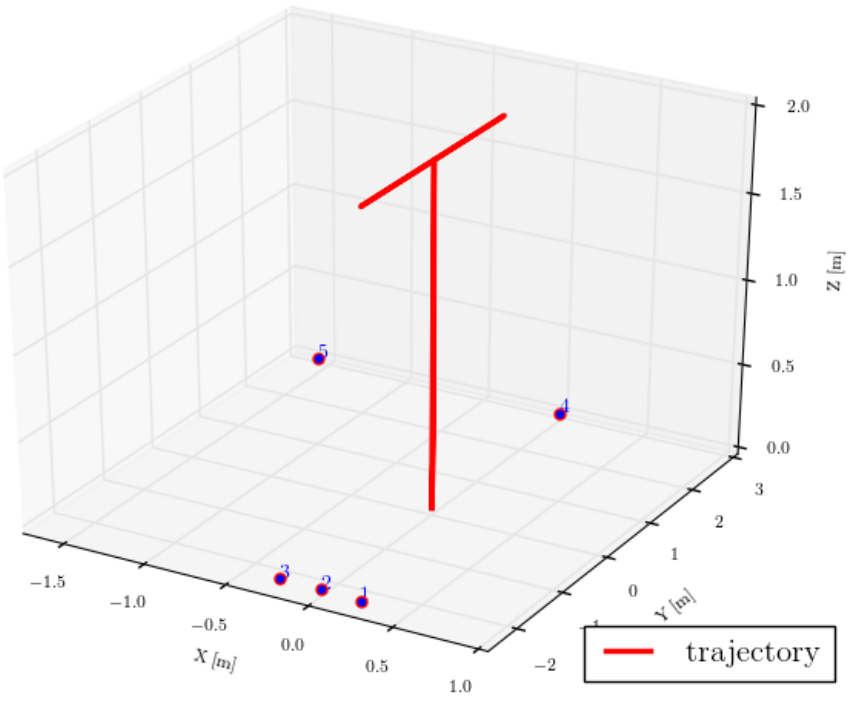

Figure 2.4: Anchor setup and quadcopter trajectory. The set of five anchors are all located at $z = 0$, and the quadcopter's motion is confined to a plane where $x = 0$, with the horizontal motion at a height of $2\mathrm{m}$.

data from the motion capture system, with specifically also the estimator variance being a reliable indication of estimate quality. The results are closely related to our conclusion on maximizing the most informative direction in the space. The extreme anchor arrangement, gives very little information in the x direction to the estimator, and this creates large uncertainty on that direction. Notable also is the large initial uncertainty in attitude for the rotation about gravity. This is because this state is unobservable without horizontal motion, and the uncertainty rapidly decreases when the vehicle executes the side-to-side motion.

Closed-loop tracking experiment

The second experiment demonstrates that the estimator performs sufficiently well to allow the quadcopter to fly without the use of the external system. The result of the experiment in position can be seen in Fig. 2.5. As seen in the graph, the quadcopter tracks the command in y and z directions well, but performs more poorly in the x direction. This is due to the larger uncertainty in this direction, stemming from the anchor layout.

2.6 Conclusion

This chapter presented a computationally low-cost algorithm for estimating the six degree of freedom state of a rigid body equipped with an inertial measurement unit and operating inside a system of range-measuring radios. Specific attention was paid to the optimal selection of a ranging measurement, where the resulting approach is based on greedily minimizing the trace of the estimator covariance matrix. Although increasing the rate of measurement can potentially improve the estimator performance, the hardware limitation is always an issue. Also, even with higher measurement rate, the proposed algorithm can be used to improve the performance further more. The resulting algorithm can be deployed on low-cost, computationally constrained devices. The estimator has the advantage that it makes no specific assumptions about the agent's motion, and could thus be immediately applied to a large variety of agents.

A series of flight experiments were performed to demonstrate the efficacy of the estimator as implemented on a low-cost quadcopter. Over twenty repeated experiments with the quadcopter externally controlled, a substantial improvement due to the selection of the anchor using the greedy optimization approach compared to using a naïve sequential ranging algorithm was shown. Further experiments showed that the estimator performs sufficiently well to be used for closed-loop control as well, even for fast, unstable systems such as quadcopters.

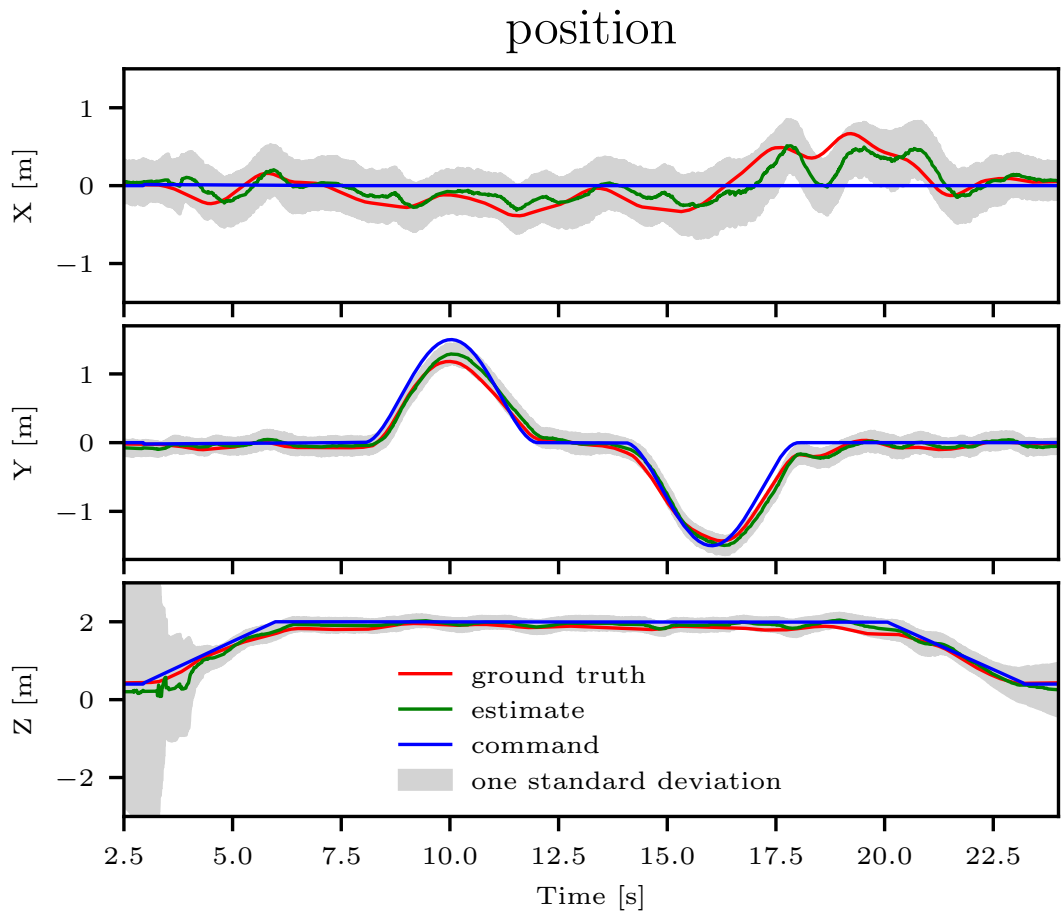

Figure 2.5: Closed-loop control of the quadcopter using the presented optimal anchor selection algorithm. The plot represents the position of quadcopter. The blue, green, and red lines show the command position, the onboard position estimator (EKF), and the motion capture system output (ground truth), respectively. The yellow area around the estimation shows one standard deviation on position in each direction.

Chapter 3

Restructure of a ranging-based localization network

3.1 Overview

An accurate, robust, and accessible localization system is crucial for robot operation in the case of, for example, emergency services, building fault detection in disaster areas, or rescue missions. We assume that in these situations the pre-installed infrastructure is destroyed or limited (i.e. no access to GPS signal), and therefore a local and stand-alone alternative localization system is vital. Also, the harsh environment of these situation requires to use reliable sensory devices. For example, the sensors should function in smoke, dust, and in foggy or heavy rain conditions where optical systems cannot be used reliably. An infrastructure (e.g. radio/audio beacons) for the local positioning system is important to be placed in the environment such that it maximizes the localization accuracy, but it is most likely impossible due to limited accessibility of such environment. The proposed method in this chapter will address above issues by using a reliable range measurement sensor with mobile infrastructure setting.

The recently popular ultra-wideband radio ranging is a flexible, relatively low-cost, and reliable localization technology mostly for but not limited to indoor environments – see e.g. [109, 29, 110, 39, 74]. The infrastructure components (i.e radio-beacon) of the system are easy to setup in the environment. The radio-beacon (here called 'anchor') communicates via radio messages with an agent which is equipped with UWB sensor. The result is a range measurement, the distance between the agent and each individual anchors.

The UWB sensors are used in a wide range of localization methods. For example, in conjunction with the SLAM (simultaneous localization and mapping) problem, a particular UWB transmitter-receiver configuration on an agent is used in [30]. The authors explain that it is crucial to use UWB ranging specially in emergency situation, when other technologies like camera- or laser-based sensors fail to operate in such harsh environment. The UWB technology also has medical applications. For example, authors in [19] developed

an algorithm to improve the localization accuracy of surgical devices using UWB sensor in highly reflective and dense indoor environments such as operating rooms where multipath and no-line-of-sight conditions are an issue. In [64], authors developed a Kalman filter estimator using fusion of inertial measurement unit with UWB ranging sensor for localization in crowded and dynamic environments like imaging rooms, where technical devices like C-arm imaging systems or operating tables are moving.

The range measurement based localization is also used in multi-agent cooperative positioning and wireless sensor network [90, 106, 27, 132, 105]. The main idea is to localize several agents (or wireless sensor nodes) with the capability of exchanging their information (i.e. relative distance, position, orientation, etc). Our proposed method considers using several fixed UWB anchors as well as mobile anchors, the ones that adapt their positions in order to improve the position accuracy of the agent. In other words, the cooperation occurs between an agent and a set of mobile anchors. Since the localization accuracy is sensitive to the anchor arrangement, it is hard to achieve reasonable accuracy with restricted anchor's arrangement without careful planning, which may be difficult in harsh conditions. In this chapter, the position of mobile anchors is assumed to be known (i.e. they have access to a separate, independent localization technology), whereas the agents rely only on the mobile and fixed anchors for localization, since they don't have access to any localization infrastructure. Future work will look at generalizing this by removing the separate localization technology which mobile anchors currently rely on. A schematic of our proposed system is shown in Fig. 3.1

In this chapter, we use a state estimator with no assumption on force-torque, and we develop an adaptive localization algorithm on top of the estimator for moving a known ranging measurement point (i.e. a mobile anchor), so that an agent may minimizes its position uncertainty. The state estimation consists of a general six degrees of freedom (6DOF) model with measurement model as agent's distance to known positions in the world. The adaptive algorithm maximizes the agent's localization quality in the least-square sense. The results are validated in a series of experiments, where the estimator is implemented on a quadcopter system.

3.2 System Model

We consider model of a generic 6 degree of freedom (3 in translation and 3 in rotation) rigid body for the purpose of estimating the states of an agent. The agent is equipped with inertial measurement unit (accelerometer and rate gyroscope) and a range measurement sensor that allows to measure the distance to any of fixed or mobile sets of anchors (with known location) in its environment. In the following subsections we will briefly describe the underlying models of the estimator.

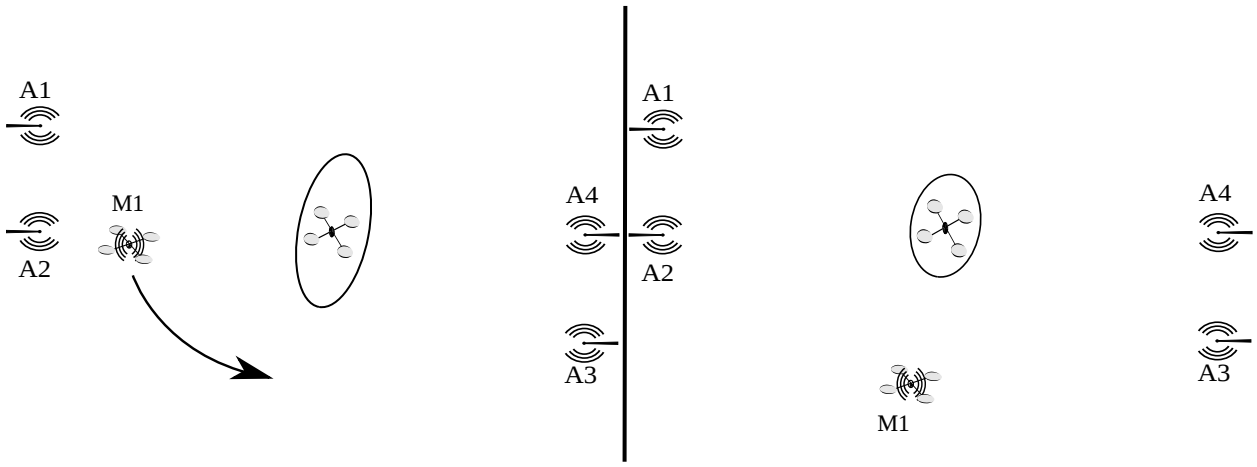

Figure 3.1: A schematic of the proposed systems: an agent (here a quadcopter) operates in a space prepared with combination of one mobile (indexed as M1) and four fixed radio anchors (indexed A1-A4). At the left, the ellipse representing the position uncertainty is extended in the direction which there is no anchors present. At the right, the mobile anchor moves to the area with no anchors and reduces the uncertainty around the agent.

Equations of motion

We will use the convention of using bold-face symbols for vector/matrix quantities, and regular font for scalars. In the model, the position of the rigid body (agent) is denoted as $\boldsymbol{x}$, its velocity as $\boldsymbol{v}$, and acceleration as $\boldsymbol{a}$, all expressed in the inertial frame fixed to the ground. The rotation matrix and the angular velocity are given respectively by $\boldsymbol{R}$ and $\boldsymbol{\omega}$, where the rotation matrix represents the orientation of the agent. The multiplication by the rotation matrix will result a coordination transformation from the body-fixed frame to the inertial frame. The time derivatives of these quantities are given as

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x} = \boldsymbol{v} \tag{3.1}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{v} = \boldsymbol{a} \tag{3.2}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{R} = \boldsymbol{R}\boldsymbol{S}(\boldsymbol{\omega}) \tag{3.3}$$

note that $\boldsymbol{S}(\boldsymbol{\omega})$ is the skew-symmetric matrix version of cross product, where $\boldsymbol{S}(\boldsymbol{x})\boldsymbol{y} = \boldsymbol{x} \times \boldsymbol{y}$.

Inertial measurements

The inertial measurement unit outputs the accelerometer and rate gyroscope measurements, $\boldsymbol{\alpha}$ and γ. The accelerometer measures the 'proper acceleration' in the body-fixed frame which we assume it is corrupted by additive noise $\boldsymbol{\nu}_\alpha$.

$$\boldsymbol{\alpha}_m = \boldsymbol{R}^{-1}\left(\boldsymbol{a} - \boldsymbol{g}\right) + \boldsymbol{\nu}_\alpha \tag{3.4}$$

The gravitational acceleration is in the inertial frame fixed to the ground, and has magnitude $\boldsymbol{g} = (0, 0, -9.81)\,\mathrm{m/s^2}$. The rate gyroscope measures angular velocity of the agent in the body-fixed frame. The measurement is modeled as

$$\boldsymbol{\gamma} = \boldsymbol{\omega} + \boldsymbol{\nu}_\gamma \tag{3.5}$$

here the the measurement is corrupted by $\boldsymbol{\nu}_\gamma$. Both $\boldsymbol{\nu}_\alpha$ and $\boldsymbol{\nu}_\gamma$ are assumed to be zero mean, based on the fact that the sensors are well calibrated and scale/bias-free.

UWB Range measurement system

The UWB radio mounted on the agent communicates with other radios in the environment. At each time instant, the agent measures the distance from its position at $\boldsymbol{x}$ to one of fixed or mobile position at $\boldsymbol{p}_i$ (anchor's position) in the world. This measurement ρ_i to anchor i is modelled as the Euclidean norm corrupted with additive scalar noise ν_ρ with zero mean.

$$\rho_i = \|\boldsymbol{x} - \boldsymbol{p}_i\| + \nu_\rho \tag{3.6}$$

The UWB radio uses two-way ranging time-of-flight based algorithm to calculate the distance (see [84] for two-way ranging scheme). The agent can communicate only with one anchor at a time, meaning that only a single range measurement can be taken at any instant in time.

3.3 State estimator

For the state estimation, an extended Kalman filter (EKF) [118] presented in this section estimates the 12-element state of the agent consist of position, attitude and their derivatives. The kinematic model makes the Kalman filter an estimator for a generic 6DOF rigid body with no assumption on forces or torques acting on the agent. The EKF uses the technique of [83] to encode an attitude in the state with correct-to-first-order statistics.

The estimator does not include the angular velocity as a state, instead uses the output of rate gyroscope measurement; assuming that the modern sensors output high-quality measurements which is a standard approach in attitude estimation for satellites [76]. Thus, the estimator's stochastic state $\boldsymbol{\xi}$ is a 9 dimensional vector:

$$\hat{\boldsymbol{\xi}} = \left(\hat{\boldsymbol{x}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{\delta}}\right) \tag{3.7}$$

with the hat denoting estimated quantities, and where $\hat{\boldsymbol{\delta}}$ represents attitude error measure, assumed to be small. The estimator uses a redundant attitude representation, with a 'reference attitude' $\boldsymbol{R}_{\text{ref}}$ and the attitude error $\hat{\boldsymbol{\delta}}$ combined yielding the estimator's attitude estimate $\hat{\boldsymbol{R}}$

$$\hat{\boldsymbol{R}} = \boldsymbol{R}_{\text{ref}} \left(\boldsymbol{I} + \boldsymbol{S}(\hat{\boldsymbol{\delta}}) \right) \tag{3.8}$$

with $\boldsymbol{I}$ the identity matrix. This representation allows for a singularity-free attitude estimation using only a three-dimensional representation of the attitude error – a complete discussion of this approach is given in [83].

The EKF uses the output of accelerometer and rate gyroscope for the prediction step, so the state differential is given by

$$\frac{\mathrm{d}}{\mathrm{d}t}\hat{\boldsymbol{v}} = \boldsymbol{R}\boldsymbol{\alpha}_m + \boldsymbol{g} - \boldsymbol{R}\boldsymbol{\nu}_\alpha \tag{3.9}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\hat{\boldsymbol{\delta}} = \boldsymbol{\gamma} - \boldsymbol{\nu}_\gamma \tag{3.10}$$

In the measurement update step, the estimator uses the output of the UWB ranging radios. The linearization of measurement equation (2.6) is as follows

$$\boldsymbol{H}_i := \frac{\partial \rho_i}{\partial \boldsymbol{\xi}} = \left(\frac{\partial \rho_i}{\partial \boldsymbol{x}}, \frac{\partial \rho_i}{\partial \boldsymbol{v}}, \frac{\partial \rho_i}{\partial \boldsymbol{\delta}} \right) \tag{3.11}$$

$$\frac{\partial \rho_i}{\partial \boldsymbol{x}} = \frac{\boldsymbol{x} - \boldsymbol{p}_i}{\|\boldsymbol{x} - \boldsymbol{p}_i\|} =: \boldsymbol{e}_i \tag{3.12}$$

$$\frac{\partial \rho_i}{\partial \boldsymbol{v}} = \frac{\partial \rho_i}{\partial \boldsymbol{\delta}} = \boldsymbol{0} \tag{3.13}$$

Note that the measurement sensitivity with respect to the agent's position $\boldsymbol{e}_i$ is the unit vector in the direction of anchor i from the agent.

The estimate covariance matrix $\boldsymbol{\Sigma}$ computed by EKF relies on the partial derivatives of linearization process and using the approach of [83]. This is the partitioned matrix as below

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{\boldsymbol{xx}} & \boldsymbol{\Sigma}_{\boldsymbol{xv}} & \boldsymbol{\Sigma}_{\boldsymbol{x\delta}} \\ \boldsymbol{\Sigma}_{\boldsymbol{xv}}^{\boldsymbol{T}} & \boldsymbol{\Sigma}_{\boldsymbol{vv}} & \boldsymbol{\Sigma}_{\boldsymbol{v\delta}} \\ \boldsymbol{\Sigma}_{\boldsymbol{x\delta}}^{\boldsymbol{T}} & \boldsymbol{\Sigma}_{\boldsymbol{v\delta}}^{\boldsymbol{T}} & \boldsymbol{\Sigma}_{\boldsymbol{\delta\delta}} \end{bmatrix} \in \mathbb{R}^{9\times 9} \tag{3.14}$$

with e.g. $\boldsymbol{\Sigma}_{\boldsymbol{x\delta}}$ the 3×3 cross-covariance between the position and attitude states. The very intuitive property of measurement sensitivity will be used in the following section to determine how the anchors can move in order to maximize the localization quality of the agent.

3.4 Mobile anchor algorithm

The measurement model as described in (3.12) is sensitive to the location of the anchors $\boldsymbol{p}_i$. This means, it is possible to affect the variance of the state estimates by moving the anchors. This allows to create an optimization problem in order to move the anchors in the direction which minimizes the estimation error of the agent's position. The covariance matrix is used as the metric for estimation quality. In the following subsections we will discuss this approach in detail.

Least squares approach

The least-square approach is used as an easy-to-analyze approximation of the EKF used to estimate the agent's location. This is motivated by the least squares interpretation of the Kalman filter.

All the derivations in this section are assumed to be for a single mobile anchor with $N-1$ fixed anchors. It is simple to generalize the derivation for multiple mobile anchors, since the desired moving direction of each anchor decouples. All the quantities without number subscription are introduced for the single mobile anchor.

Notable is that we have two sets of decision variables in this section. The position estimate of the agent $\boldsymbol{x}$ is the decision variable for the least squares problem. The mobile anchor's position $\boldsymbol{p}$ is the decision variable for minimizing the variance of the agent's position estimate. We are using these two variables throughout this section.

We consider again the ranging measurement model in (2.6). Note, here we look at a batched version of this equation (i.e. the equation is concatenation of scalar measurements at each discrete time step for N anchors, fixed or mobile anchors), which is an approximation of EKF

$$\boldsymbol{\rho} = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{p}) + \boldsymbol{\nu}_\rho \tag{3.15}$$

$$\begin{bmatrix} \rho \\ \rho_1 \\ \vdots \\ \rho_{N-1} \end{bmatrix} = \begin{bmatrix} \|\boldsymbol{x} - \boldsymbol{p}\| \\ \|\boldsymbol{x} - \boldsymbol{p}_1\| \\ \vdots \\ \|\boldsymbol{x} - \boldsymbol{p}_{N-1}\| \end{bmatrix} + \begin{bmatrix} \nu_\rho \\ \nu_{\rho_1} \\ \vdots \\ \nu_{\rho_{N-1}} \end{bmatrix},$$

where $\boldsymbol{\rho}$ is the measurement vector, $\boldsymbol{\nu}_\rho$ is the additive noise vector with zero mean, and $\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{p})$ is the nonlinear measurement model (vector-valued equation) mapping the position of an agent in $\mathbb{R}^3$ to the batch of anchors' measurements in $\mathbb{R}^N$. The partial derivative of the nonlinear model with respect to the agent's position will result a $N \times 3$ Jacobian

matrix

$$\boldsymbol{H}(\hat{\boldsymbol{x}}, \boldsymbol{p}) = \frac{\partial \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{p})}{\partial \boldsymbol{x}} = \begin{bmatrix} \boldsymbol{e}^\top \\ \boldsymbol{e}_1^\top \\ \vdots \\ \boldsymbol{e}_{N-1}^\top \end{bmatrix} \in \mathbb{R}^{N \times 3} \tag{3.16}$$

where ith row of the matrix is the unit vector pointing from the agent to anchor i. Assuming noise in (3.15) to be zero mean with isotropic covariance $\mathrm{Var}(\boldsymbol{\nu}_\rho) = q\boldsymbol{I}$, the position estimate can be found by minimizing the $2-$norm squared of noise

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{\nu}_\rho\|^2$$

As stated before, for the nonlinear measurement model this will be done iteratively by linearizing the model at each time step, and moving in the direction of the gradient descent.

Statistical properties

Since we use the linearized version of the equation, the mean of the estimated position is zero (i.e. unbiased estimator) but the variance is

$$\begin{aligned} \mathrm{Var}(\hat{\boldsymbol{x}}) &\approx ((\boldsymbol{H}^\top \boldsymbol{H})^{-1} \boldsymbol{H}^\top) q\boldsymbol{I} ((\boldsymbol{H}^\top \boldsymbol{H})^{-1} \boldsymbol{H}^\top)^\top \\ &= q(\boldsymbol{H}^\top \boldsymbol{H})^{-1} \end{aligned} \tag{3.17}$$

For brevity, we use $\boldsymbol{H}$ instead of $\boldsymbol{H}(\hat{\boldsymbol{x}}, \boldsymbol{p})$ from now on. Our goal is to minimize the effect of noise on the state estimates (i.e. to minimize the variance of the state estimates by varying the anchor location). Specifically, we choose to minimize the largest eigenvalue of the covariance matrix, which is equivalent to maximizing the smallest eigenvalue of its inverse:

$$\max_{\boldsymbol{p}} \min_{i} \lambda_i(\boldsymbol{H}^\top \boldsymbol{H}) \tag{3.18}$$

By expanding matrix $\boldsymbol{H}^\top \boldsymbol{H}$, the max-min problem can be converted to a simple form. Substituting (3.16) in matrix $\boldsymbol{H}$, the result is the sum of N rank-one matrices

$$\begin{aligned} \boldsymbol{H}^\top \boldsymbol{H} &= \begin{bmatrix} \boldsymbol{e} & \boldsymbol{e}_1 & \dots & \boldsymbol{e}_{N-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{e}^\top \\ \boldsymbol{e}_1^\top \\ \vdots \\ \boldsymbol{e}_{N-1}^\top \end{bmatrix} \\ &= \boldsymbol{e}\boldsymbol{e}^\top + \sum_{i=1}^{N-1} \boldsymbol{e}_i \boldsymbol{e}_i^\top \end{aligned} \tag{3.19}$$

Since all $\boldsymbol{e}_i$ and $\boldsymbol{e}$ are unit vectors (the direction to the anchors), the trace of $\boldsymbol{H}^\top \boldsymbol{H}$ is always constant and equal to the number of anchors:

$$\operatorname{tr}(\boldsymbol{H}^\top \boldsymbol{H}) = \operatorname{tr}(\boldsymbol{e}^\top \boldsymbol{e}) + \sum_{i=1}^{N-1} \operatorname{tr}(\boldsymbol{e}_i^\top \boldsymbol{e}_i) = N \tag{3.20}$$

The trace of a matrix is also equal to the sum of its eigenvalues; that means the sum of eigenvalues of the covariance matrix in this problem is constant.

$$\operatorname{tr}(\boldsymbol{H}^\top \boldsymbol{H}) = \sum_{i=1}^{3} \lambda_i = N \tag{3.21}$$

Using this fact, the max-min optimization problem can be transformed to

$$\begin{aligned} \max_{\boldsymbol{\lambda}, t} \quad & t \\ \text{subject to} \quad & \mathbf{1}^\top \boldsymbol{\lambda} = N \\ & t \leq \lambda_i \ \ \forall i \\ & 0 \leq \lambda_i \ \ \forall i \end{aligned} \tag{3.22}$$

where t is a slack variable, and $\boldsymbol{\lambda}$ is a vector representing the eigenvalues of matrix $\boldsymbol{H}^\top \boldsymbol{H}$. This problem can be solved by introducing its dual problem. Since (3.21) shows that the sum of eigenvalues is constant, we can conclude that (3.22) is equivalent to the following problem

$$\max_{\boldsymbol{p}} \ \det(\boldsymbol{H}^\top \boldsymbol{H}) = \max_{i} \ \prod_{i=1}^{3} \lambda_i \tag{3.23}$$

It can be shown that for both problems the maximum occurs, when all the eigenvalues are equal. Specifically, (3.18) has interpretation for robust design (it minimizes the worst variance direction) as opposed to (3.23) which minimizes the volume of ellipsoid associated with the covariance matrix [17]. This shows that this method satisfies both design approaches (i.e. minimizing the largest variance direction will result in minimizing the ellipsoid volume).

Mobile anchor

Since the matrix consists of the mobile anchor's position, its determinant depends on the position of the mobile anchor at each time step.

$$\begin{aligned} \boldsymbol{H}^\top \boldsymbol{H} = & \frac{(\hat{\boldsymbol{x}} - \boldsymbol{p})(\hat{\boldsymbol{x}} - \boldsymbol{p})^\top}{\|\hat{\boldsymbol{x}} - \boldsymbol{p}\|} \\ & + \sum_{i=1}^{N-1} \frac{(\hat{\boldsymbol{x}} - \boldsymbol{p}_i)(\hat{\boldsymbol{x}} - \boldsymbol{p}_i)^\top}{\|\hat{\boldsymbol{x}} - \boldsymbol{p}_i\|} \in \mathbb{R}^{3\times 3} \end{aligned}$$

By taking partial derivative with respect to the mobile anchor's position $\boldsymbol{p} = (x, y, z)$ we can find a direction that the mobile anchor can move in order to minimize the determinant of the covariance matrix

$$\frac{\partial \det(\boldsymbol{H}^\top \boldsymbol{H})}{\partial \boldsymbol{p}} = \begin{bmatrix} \frac{\mathrm{d}}{\mathrm{d}x}\det(\boldsymbol{H}^\top \boldsymbol{H}) \\ \frac{\mathrm{d}}{\mathrm{d}y}\det(\boldsymbol{H}^\top \boldsymbol{H}) \\ \frac{\mathrm{d}}{\mathrm{d}z}\det(\boldsymbol{H}^\top \boldsymbol{H}) \end{bmatrix} \tag{3.24}$$

We rewrite the inverse of the covariance matrix in a compact form as

$$\boldsymbol{H}^\top \boldsymbol{H} = \begin{bmatrix} \boldsymbol{m}_1 & \boldsymbol{m}_2 & \boldsymbol{m}_3 \\ \boldsymbol{m}_2 & \boldsymbol{m}_4 & \boldsymbol{m}_5 \\ \boldsymbol{m}_3 & \boldsymbol{m}_5 & \boldsymbol{m}_6 \end{bmatrix} \tag{3.25}$$

where $\boldsymbol{m}_i$ are functions of mobile anchor's position. As an example, $\boldsymbol{m}_1$ and $\boldsymbol{m}_6$ have the form (note that $\hat{x}_1$, $\hat{x}_2$, $\hat{x}_3$ are the three first states of the estimator representing the agent's position):

$$\boldsymbol{m}_2 = \frac{(\hat{x}_1 - x)(\hat{x}_2 - y)}{\|\hat{\boldsymbol{x}} - \boldsymbol{p}\|^2}, \quad \boldsymbol{m}_6 = \frac{(\hat{x}_3 - z)^2}{\|\hat{\boldsymbol{x}} - \boldsymbol{p}\|^2}$$

Representing the inverse of the covariance matrix in the compact form of (3.25) allows to derive the determinant of the matrix and its partial derivative in terms of functions $\boldsymbol{m}_i$, and then substitute back the mobile anchor's position. At the end, (3.24) will be a set of three closed form equations in terms of the mobile anchor's position as well as the agent's position, which can be easily implemented.

We choose the control action as the velocity command for the mobile anchor which is the Jacobian of the determinant of the covariance matrix multiplying by a gain value $s > 0$ as follows

$$\boldsymbol{v}_{cmd} = s\frac{\partial \det(\boldsymbol{H}^\top \boldsymbol{H})}{\partial \boldsymbol{p}}$$

This is exactly the same as moving on the direction of gradient ascent iteratively until the mobile anchor reaches to the top (max point of determinant) and stays there. For sequence of steps the mobile anchors move as follows

$$\boldsymbol{p}(k+1) = \boldsymbol{p}(k) + (\Delta t)\boldsymbol{v}_{cmd} \tag{3.26}$$

where k is time steps and Δt is the discrete time interval. In the following section we will present the real-time experimental results of this approach.

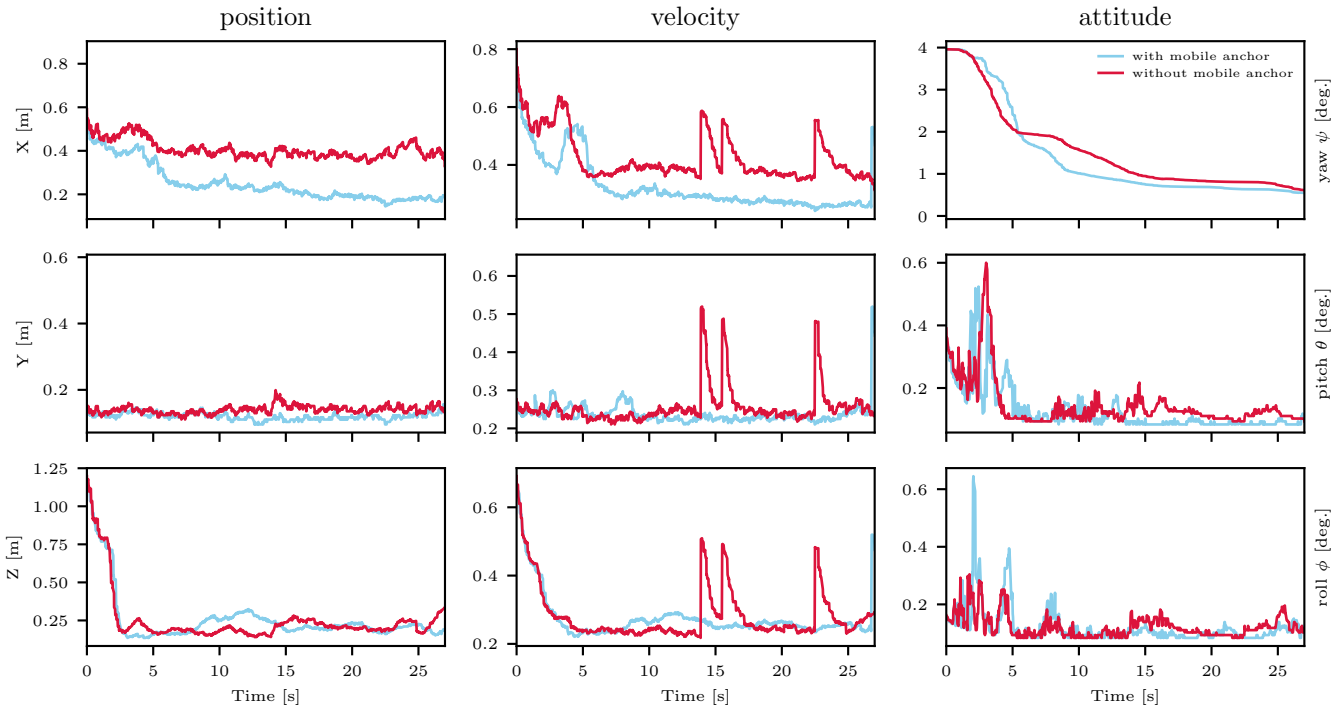

Figure 3.2: Closed loop control of quadcopter using UWB ranging sensor with and without use of mobile anchor. The experiment illustrates the square roots of the diagonals of EKF covariance matrix (i.e. one standard deviation). As seen using mobile anchor, decreases uncertainty (position and velocity) dramatically in the x direction, since there are enough fixed anchors along y direction but nothing in the x direction (see Fig. 3.3). Note that the estimator does not compute the estimate in terms of yaw, pitch, and roll angles; the estimator output is simply transformed into this format as it is easier to parse in a figure.

3.5 Experimental validation

The approach is validated in experiment, where two quadcopters are used, one as autonomous agent and one as mobile anchor. A first set of experiments just uses the fixed anchors as the only UWB measurement setting. The second experiment uses one mobile anchor in addition to the fixed anchors. Results will be presented by comparing the agent's variance computed by the onboard EKF for the two different experiment settings.

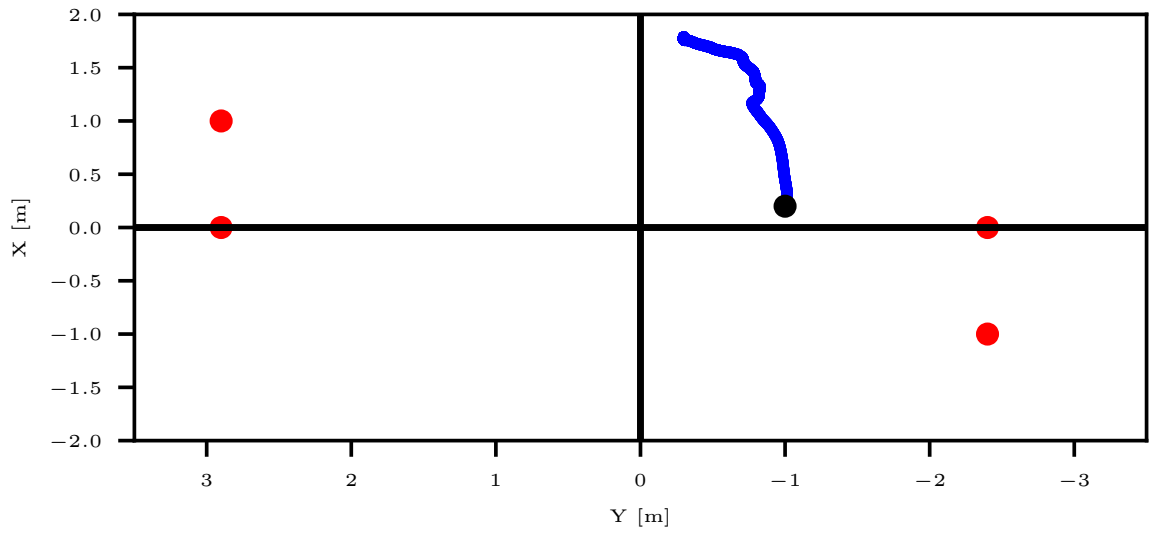

Figure 3.3: Top view. Red dots represent the position of fixed anchors in both sets of experiments. The black dot is the fifth fixed anchor in the first and the initial position of mobile anchor in the second set of experiments. The set of four anchors are all located at $z = 0$. The blue path is the projection of the mobile anchor trajectory on xy plane.

Experimental setup

The testbed we used to examine our algorithm is a Crazyflie 2.0 quadcopter, with approximate mass of $30\mathrm{g}$, and motor-to-motor distance of $105\mathrm{mm}$. We used two quadcopters as the agent and the mobile anchor in our experiments. The quadcopter is equipped with an STM32F4 microcontroller, an Invensense MPU9250 inertial measurement unit, and a Decawave DW1000 radio module for the ultra-wideband ranging measurements. Beside the mobile anchor which shared the same hardware, all the fixed anchors also have the same computational and sensing hardware. The state estimation (EKF) for the agent was performed on the microcontroller, but the mobile anchor used motion capture system for its own localization, and was commanded by the trajectory according to (3.26) computed off-board on a computer. Measurements from the accelerometer and rate gyroscope were taken at $500\mathrm{Hz}$, and range measurements were taken at approximately $80\mathrm{Hz}$. The effectiveness of our approach on the estimator performance is quantified by using the motion capture system, whose measurements are taken as ground truth.

The top view of the anchor arrangement can be seen in Fig. 3.3. Notable is that the anchors are placed such that their measurements carry sufficient information in the y direction, but not very much in the x direction. This was chosen so as to highlight the

Root mean-square error (RMSE)			
	Average		difference %
	4 fixed, 1 mobile	5 fixed anchors	
position [m]	0.156	0.182	-14.3%
velocity [m/s]	0.374	0.409	-8.6%
attitude [deg.]	5.140	5.315	-3.3%

Table 3.1: Comparison of estimation error from experiments

importance of mobile anchor and our proposed algorithm.

Experiment with mobile anchor

Two sets of similar experiments were conducted to verify the advantage of using mobile anchor against only fixed anchors to improve the localization accuracy. In the first set of experiments, we used five fixed anchors placed on the ground as pictured in Fig. 3.3. We ran this test several times while the quadcopter hovers at the same position at origin and the EKF described in Section 2.3 runs on the quacopter's microcontroller. In the second set of experiments, we used four fixed anchors in addition to one mobile anchor. The mobile anchor starts from its initial position (black dot in Fig. 3.3) and moves according to the algorithm described in Section 3.4.

Table 3.1 compares the average root mean square error (RMSE) on state estimates of the quadcopter (position, velocity, attitude) from several trials for both sets of experiments. As seen the average RMSE is improved by about 14%, 9%, and 3% for position, velocity and attitude respectively. The results show larger improvement on position rather than other states. This represent the fact that the proposed algorithm have direct impact on the position estimate quality but not on velocity and attitude. Fig. 3.4 shows the determinant of inverse covariance matrix in the proposed algorithm $\det(\boldsymbol{H}^T\boldsymbol{H})^{-1}$. This verifies that the mobile anchor has moved to decrease the position uncertainty around the agent.

Fig. 3.2 shows the experimental data of a trial of both experiments, which is the comparison of one standard deviation of state estimates. Comparing the monotonic decrease in the uncertainty of position in x direction (blue line) with no improvement in y shows that our intuition about anchor's arrangement was correct; absence of any anchor along the axis of symmetry between the fixed anchors causes insufficient available information along that axis, but when the mobile anchor starts to move towards the axis the uncertainty also decreases accordingly.

As the agent navigates in the environment to reach the target position, the localization network (anchors) is capable to reconstruct itself in real time accordingly. Although the proposed algorithm for the mobile anchor was derived assuming that the agent's position is fixed (i.e. no dynamic model was introduced), the algorithm works reliably even for the

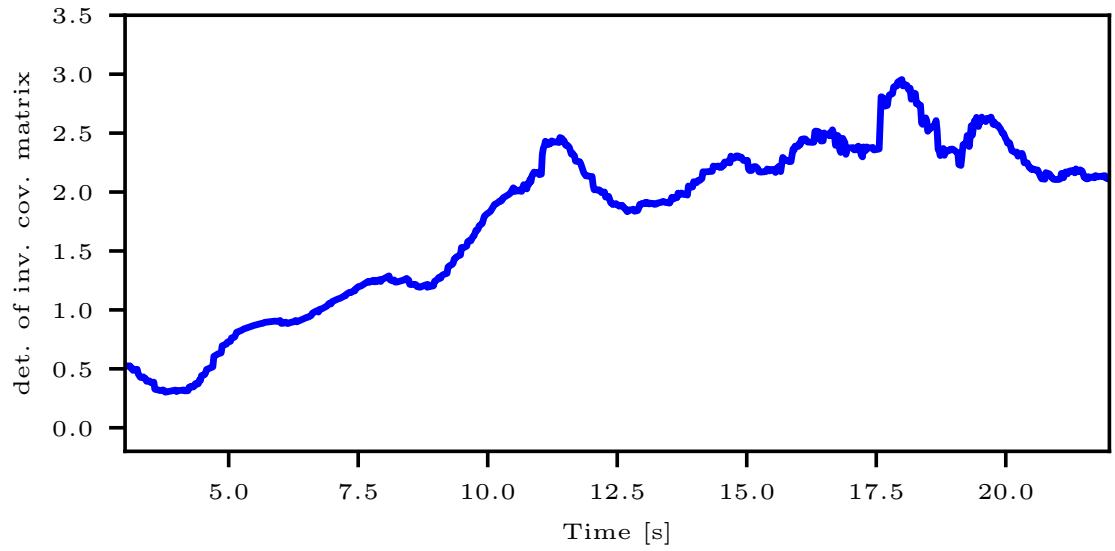

Figure 3.4: Determinant of the inverse of the covariance matrix. The plot shows the increase of the determinant as expected when the mobile anchor moving in the direction of gradient ascent; that means the state estimate variance is decreasing.

moving agent. As an example, Fig. 3.5 shows an experiment when the quadcopter tracks a trajectory, and the mobile anchor adapts its position based on the proposed approach reducing the uncertainty of the quadcopter's position estimate.

3.6 Conclusion

This chapter presented a EKF estimator for estimating a 6DOF states of an object using accelerometer, rate gyroscope, and UWB ranging sensor. Specifically, the chapter was focused on developing a method to use UWB mobile anchor and improving the localization quality of an agent. This was done by minimizing the determinant of the covariance matrix. The minimization will result a set of closed form equations representing a direction that the mobile anchor moves in that direction at each time instant.

By using an UWB mobile anchor and the presented method, the quadcopter was able to reliably fly (hovering and tracking a trajectory) while the fixed anchors was set in specific locations to cause a large uncertainty in one direction which makes the flight very hard. The experimental results presented in this chapter have shown a improvement on the position estimate (about 14%), which had also an indirect impact on the improvement of the velocity and attitude estimates. In addition, the result of a trajectory tracking scenario

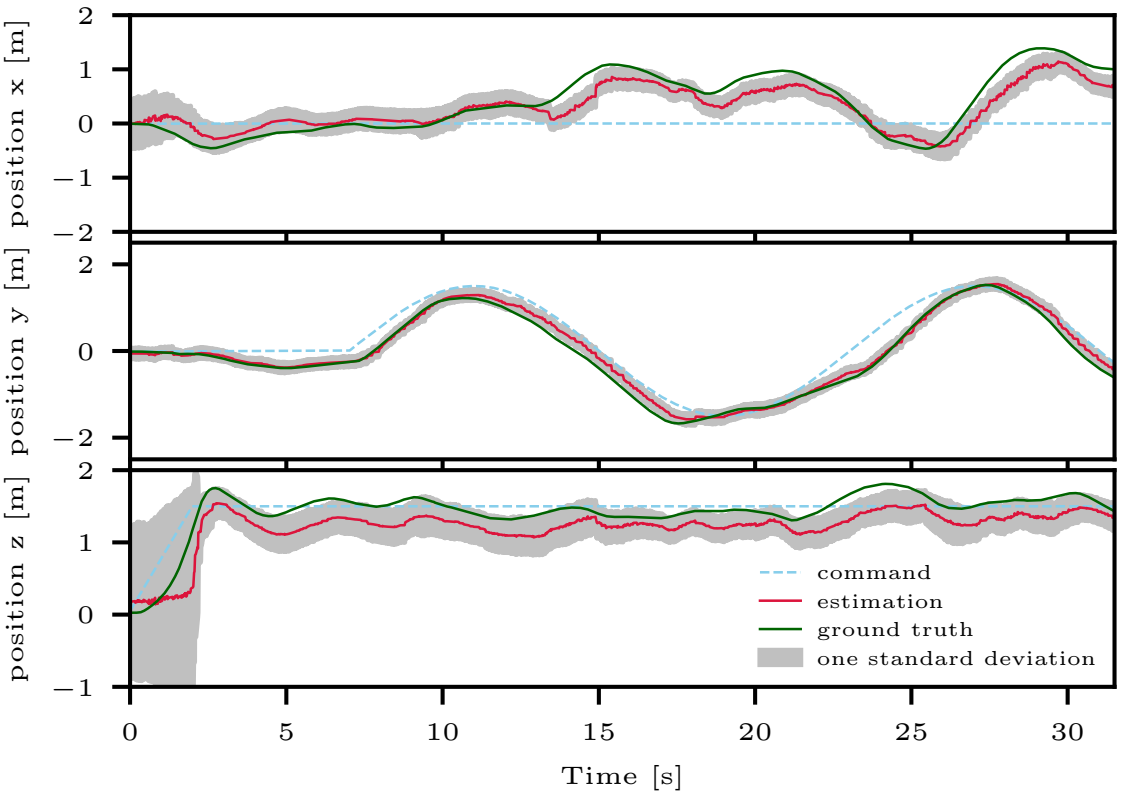

Figure 3.5: Closed loop control of the quadcopter using one mobile anchor. The plot shows that the quadcopter is tracking a horizontal trajectory (dashed blue) along the y axis. Despite the fact that the method was developed for a fixed agent, the results show that it can be used for moving agents as well.

showed that the proposed method can be applied for the moving agents, and will result to reducing the overall uncertainty of the state estimates.

Part II

Probabilistic Pose Estimation

Chapter 4

Point Registration with Directional Mixture Model

4.1 Overview

The problem of point registration or matching plays a crucial role in many engineering applications and scientific disciplines from robot navigation, odometry and autonomous vehicles to graphics, object modeling, and medical imaging. In all of these applications, it is important to acquire a very accurate point registration, despite the sensors errors and limitations. The goal of point registration is to use 3D dataset observation and try to find the best rigid transformation hypothesis that maps one frame to the next. The rigid transformation $(\boldsymbol{R}, \boldsymbol{t}) \in SE(3)$ consists of a rotation matrix $\boldsymbol{R} \in SO(3)$ and a translation vector $\boldsymbol{t} \in \mathbb{R}^3$; and a 3D point $\boldsymbol{x} \in \mathbb{R}^3$ can be accordingly transformed rigidly as $\boldsymbol{T}(\boldsymbol{x}) = \boldsymbol{R}\boldsymbol{x} + \boldsymbol{t}$.

Recently, the point registration algorithms have received extensive focus from academia and industries in autonomous driving, since the rapidly developed 3D sensing technology endows the intelligent vehicle the capacity of accurate mapping and localization. Superiority of 3D LiDAR which can reliably employed during night, bad weather (e.g. rain, snow), and in terms of computing complexity, makes it a suitable choice of 3D perception for intelligent vehicles. However, the limited sensor's coverage, intrinsic sensor errors, and outliers caused by occlusion or unpredictable traffic participators in the real traffic scenarios are formidable challenges which require to be addressed when trying to deploy the point registration algorithm in intelligent vehicles.

A standard approach for point registration problem is the Iterative Closest Point (ICP) algorithm [11] which solves the problem in two iterative steps: finding the correspondence points and then minimizing the sum of squared residuals to find the best transformation. The ICP method is sensitive to the outliers and the solution is heavily influenced by the initialization. There are several variants of ICP method that try to improve the original method in different regards: ICPp2pl, ICPpl2pl, GICP [70, 115, 113]. In all of these methods

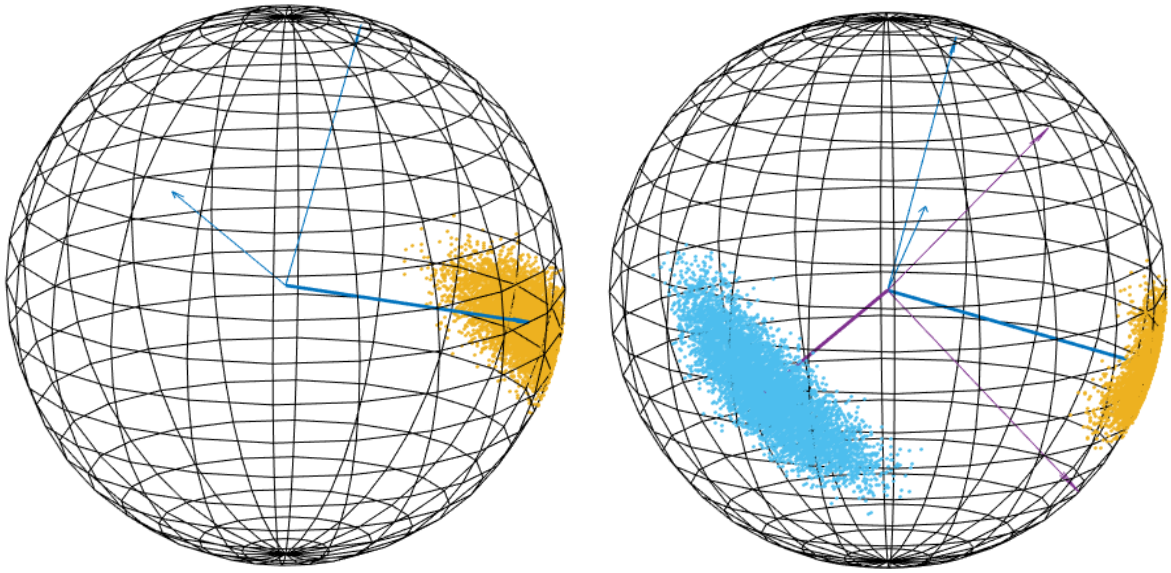

Figure 4.1: Two data samples drawn from Kent distribution. **Left:** Samples drawn from a single Kent PDF. Arrows show $\boldsymbol{\mu}$ the mean direction of samples as well as $\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2$ major and minor axes which depend on the orientation of samples. **Right:** Shows a mixture of two Kent distributions with different parameters. The pointcloud surface normals with different concentrations, can be modeled as a mixture of Kent distributions.

both rotation matrix and translation vector are found together iteratively. In addition, the correspondence phase is hard-assignment or one-to-one.

Our proposed method solves the problem of point registration in a probabilistic fashion by incorporating the orientational information from the pointcloud. The method is mainly motivated by the fact that it can robustify the registration to noise and outliers by incorporating surface normals. In particular, the surface normals are assumed to be drawn from a directional distribution called Kent distribution. Directional statistic is a suitable way to represent pointcloud features like surface normals, since they represent a directional pattern in the pointcloud and also they are invariant with respect to translation. In this approach the estimation of rigid transformation can be decoupled into two steps: first, the rotation matrix between two frames can be estimated using the surface normals, and second, the translation vector can be found using the positional information of the pointcloud. The decoupling of rotation matrix from translation vector is also addressed in [125].

Many previous works use the probabilistic approach (i.e. soft-assignment) for the correspondence phase [104, 72, 86]. In contrast to hard-assignment or one-to-one correspondence in classical ICP methods, in this method the points are associated to each other in two frames according to a probability distribution. [86] formulated the point registration

problem as a maximum likelihood (ML) estimation problem by using Gaussian mixture model (GMM). [33] also use GMM to describe and register multiple pointclouds jointly that are assumed to be drawn from an underlying distribution. There are other variants of GMM method with different noise and outlier conditions [54, 48].

Recently, [78] and [12] proposed point registration with Von Mises–Fisher distribution which is also a directional distribution defined on a unit sphere for 3D points. The method is a hybrid method that combines both directional and positional information into a hybrid of Gaussian and Von Mises distributions. Although the method improves the registration with regard to outliers, its isotropic assumption is not realistic in the real world applications. In contrast, Kent distribution accounts for anisotropic (i.e. the ovalness of surface normals on a unit sphere) dataset as seen in Fig. 4.1. This will further address the problem of outliers in dataset. Based on the proposed statistical framework, the iterations of finding correspondences and updating transformations are now considered as a type of Expectation-Maximization (EM) procedure.

The rest of this chapter is organized as follows; first we start by preliminaries and introducing nomenclatures, including the definition of Kent distribution. Then we introduce the mixture model that describes the probabilistic relationship between points in two frames. Then, we detail the steps in the EM algorithm in the proposed method. We present the results from the experiments on ETH dataset, and finally we make the concluding remarks and discuss the future directions.

4.2 Methods

Preliminaries

Throughout the chapter we use the following notations for pointclouds:

- M, N– number of points in the *model* and *observed* pointclouds, respectively,
- $\mathbf{Y} = \{\boldsymbol{y}_i\}_{m=1,\dots,M},\ \boldsymbol{y}_i \in \mathbb{R}^3$– the *model* pointcloud,
- $\mathbf{X} = \{\boldsymbol{x}_i\}_{n=1,\dots,N},\ \boldsymbol{x}_i \in \mathbb{R}^3$– the *observed* pointcloud,
- $\tilde{\mathbf{Y}} = \{\tilde{\boldsymbol{y}}_m \in \mathbb{R}^3 : \|\tilde{\boldsymbol{y}}_i\|_2 = 1\}_{i=1,\dots,M}$– the surface normals of *model* data,
- $\tilde{\mathbf{X}} = \{\tilde{\boldsymbol{x}}_n \in \mathbb{R}^3 : \|\tilde{\boldsymbol{x}}_i\|_2 = 1\}_{i=1,\dots,N}$– the surface normals of *observed* data,
- $\boldsymbol{\mu}_y$ the positional mean of the *model* pointcloud,
- $\boldsymbol{\mu}_x$ the positional mean of the *observed* pointcloud,
- $\boldsymbol{z}_n$, $1 \leq n \leq N$, the hidden random variable in the mixture model.

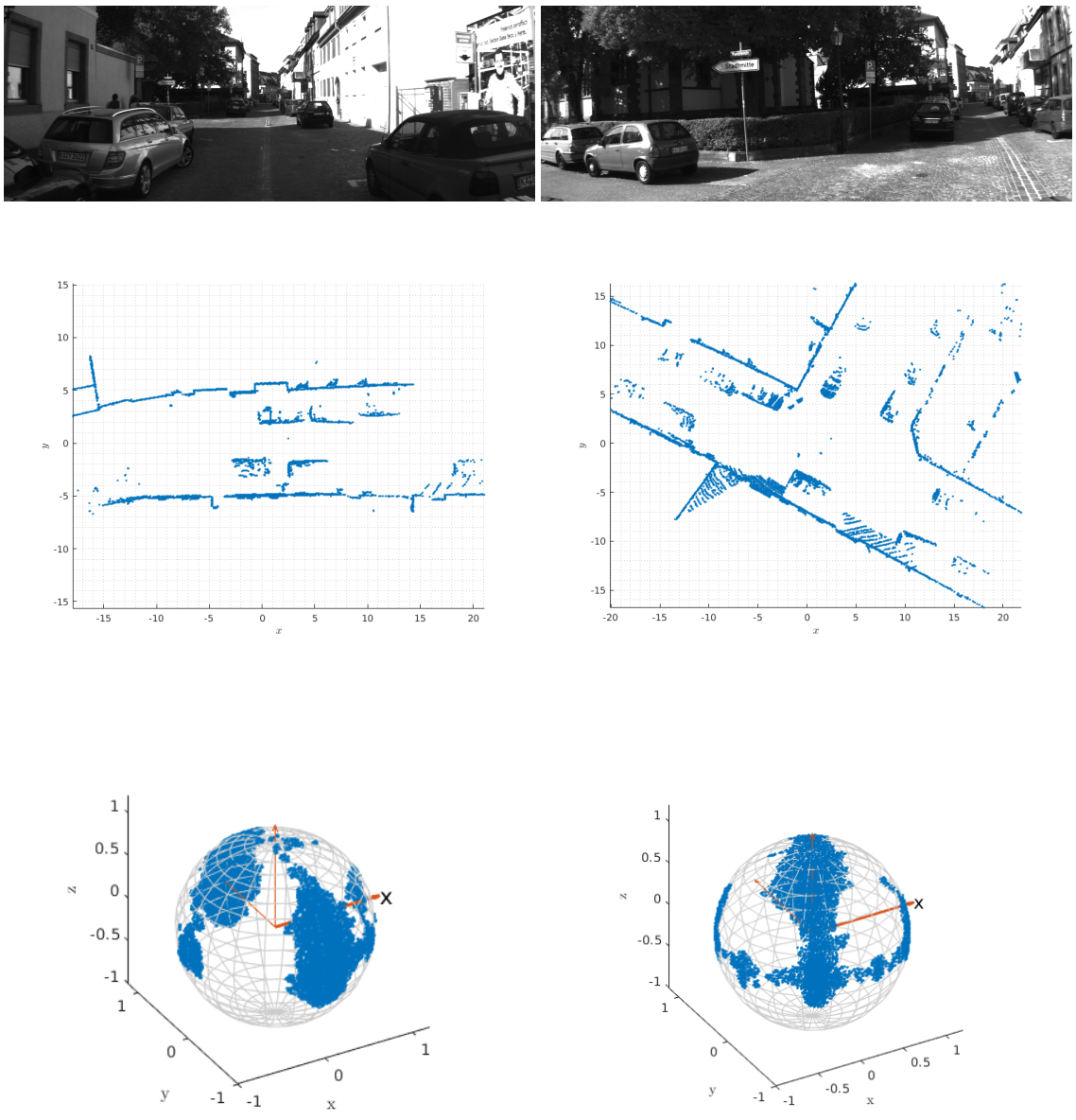

Figure 4.2: Example from the KITTI dataset [37]. Figures show two urban scenes pointcloud data and associated surface normals on a unit sphere. As seen, as the vehicle moves, the surface normals are also moving on the unit sphere which indicates the rotation between consecutive frames.

Kent distribution is the spherical analogous of the bivariate Gaussian distribution which was introduced by [59]. Since the distribution is defined over the set of unit vector on a sphere, it is suitable for representing surface normals of data points in a pointcloud. The distribution can be represented with 5 parameters $(\kappa, \beta, \boldsymbol{\mu}, \boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2)$ as[1],

$$\text{FB}_5(\tilde{\boldsymbol{x}}) = c(\kappa,\beta)^{-1} \exp\left\{\kappa \boldsymbol{\mu}^T \tilde{\boldsymbol{x}} + \beta\left[(\boldsymbol{\gamma}_1^T \tilde{\boldsymbol{x}})^2 - (\boldsymbol{\gamma}_2^T \tilde{\boldsymbol{x}})^2\right]\right\},$$

where $\kappa \geq 0$ is the *concentration*, $\beta \geq 0$ describes *ovalness*, $\boldsymbol{\mu} \in \mathbb{R}^3$ is the *mean direction*, and $\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2 \in \mathbb{R}^3$ are the *major axis* and *minor axis* of the orientation of the distribution, respectively. Together, $\boldsymbol{\Gamma} = (\boldsymbol{\mu}, \boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2)$ creates an orthogonal matrix representing the total orientation of data on the sphere. $c(\kappa, \beta)$ is the normalization term which can be expressed, in general, with an infinite sequence of the Modified Bessel functions, but it can be simplified under the assumption of $2\beta < \kappa$ and large κ as $c(\kappa,\beta) = \frac{2\pi e^{\kappa}}{(\kappa^2 - 4\beta^2)^{1/2}}$.

Problem formulation

The point registration problem can be modeled as a mixture of Kent distributions. The distribution which describes the probability of an observed data point corresponding to a model data point has the form [12],

$$\begin{aligned}&\text{FB}_5(\tilde{\boldsymbol{x}}_n | \boldsymbol{z}_n = \tilde{\boldsymbol{y}}_m) = \\ &c(\kappa,\beta)^{-1} \exp\left\{\kappa \tilde{\boldsymbol{y}}_m^T \tilde{\boldsymbol{x}}_n + \beta\left[(\boldsymbol{\gamma}_1^T \tilde{\boldsymbol{x}}_n)^2 - (\boldsymbol{\gamma}_2^T \tilde{\boldsymbol{x}}_n)^2\right]\right\},\end{aligned} \tag{4.1}$$

where $\tilde{\boldsymbol{y}}_m$ are substituted as the *mean directions*, and consequently the marginal distribution of an observed data point $\tilde{\boldsymbol{x}}_n$ can be computed as,

$$p(\tilde{\boldsymbol{x}}_n) = \sum_{m=1}^{M} \pi_m \text{FB}_5(\tilde{\boldsymbol{x}}_n | \boldsymbol{z}_n = \tilde{\boldsymbol{y}}_m) + \pi_0 p_0(\tilde{\boldsymbol{x}}_n). \tag{4.2}$$

In general, the membership probabilities are part of parameter estimation in mixture models, but here they are assumed to be constant and identical. We also add a uniform distribution which accounts for the noise/outlier in the data points as $p_0(\tilde{\boldsymbol{x}}_n) = \frac{1}{N}$. π_0 can be chosen empirically based on the approximate amount of noise/outlier in the dataset. Fig. 4.2 shows an example of two urban scenes and the estimated surface normals which represents a mixture of Kent distributions. Normals rotate on the sphere in consecutive frames and are invariant to the translation along the trajectory of the vehicle.

By assuming that all surface normals of the observed data points $\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_N$ are independent, the likelihood function is

$$L(\boldsymbol{\Theta}) = p(\tilde{\boldsymbol{X}} | \tilde{\boldsymbol{Y}}; \boldsymbol{\Theta}) = \prod_{n=1}^{N} p(\tilde{\boldsymbol{x}}_n), \tag{4.3}$$

[1] In the original paper the distribution was called Fisher-Bingham with 5 parameters, therefore we use the same notation $\text{FB}_5(.)$ in this chapter as well.

and the log-likelihood function can be computed as

$$\begin{aligned}\log L(\boldsymbol{\Theta}) &= \sum_{n=1}^{N} \log p(\tilde{\boldsymbol{x}}_n) \\ &= \sum_{n=1}^{N} \log \sum_{m=1}^{M} \left(\pi_m \mathrm{FB}_5(\tilde{\boldsymbol{x}}_n | \boldsymbol{z}_n = \tilde{\boldsymbol{y}}_m) + \pi_0 p_0(\tilde{\boldsymbol{x}}_n)\right),\end{aligned} \tag{4.4}$$

where the parameters are $\boldsymbol{\Theta} = (\kappa, \beta, \boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \boldsymbol{R})$. The matrix $\boldsymbol{R}$ rotates the surface normals of the *observed* pointcloud $\tilde{\boldsymbol{X}}$ to the surface normals of the model pointcloud $\tilde{\boldsymbol{Y}}$, and has been expressed implicitly in (4.1) as $\tilde{\boldsymbol{y}}_m = \boldsymbol{R}\tilde{\boldsymbol{x}}_n$. Rotating surface normals does not change other parameters in the distribution as stated in the following lemma.

Lemma 1 *If $\boldsymbol{R}$ is an orthogonal matrix, and $\tilde{\boldsymbol{x}} \sim FB_5(\kappa, \beta, \boldsymbol{\mu}, \boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2)$, then, $\tilde{\boldsymbol{y}} \sim FB_5(\kappa, \beta, \boldsymbol{R}\boldsymbol{\mu}, \boldsymbol{R}\boldsymbol{\gamma}_1, \boldsymbol{R}\boldsymbol{\gamma}_2)$ when $\tilde{\boldsymbol{y}} = \boldsymbol{R}\tilde{\boldsymbol{x}}$.*

Proof 1 *From the definition of Kent distribution we know that we always have $\tilde{\boldsymbol{x}}^* \sim FB_5(\kappa, \beta, \boldsymbol{I})$ when $\tilde{\boldsymbol{x}}^* = \boldsymbol{\Gamma}^T\tilde{\boldsymbol{x}}$. Then if $\tilde{\boldsymbol{x}}^* = (\boldsymbol{R}\boldsymbol{\Gamma})^T\tilde{\boldsymbol{y}}$, therefore, $\tilde{\boldsymbol{x}}^* = \boldsymbol{\Gamma}^T\boldsymbol{R}^T\tilde{\boldsymbol{y}}$, which means $\tilde{\boldsymbol{y}} = \boldsymbol{R}\tilde{\boldsymbol{x}}$.*

This is also intuitive that the orientation parameter does not change the compactness of the Kent distribution, since the rotation matrix is orthogonal, therefore κ and β are preserved. This helps us to estimate the change of the orientation of surface normals without considering any changes in the compactness and ovalness.

Log-likelihood function in (4.4) assumes a prior knowledge of point correspondence (i.e. the correspondence between points in two pointclouds). However, in practice we don't know those correspondences and we refer to the observed data as *incomplete*. Since the correspondence between surface normals in *observed* and *model* data is missing, the hidden variable $\boldsymbol{z}_n$ is introduced in EM algorithm to represent their probabilistic assignment. We describe an EM-like approach in the next section in order to iteratively solve for *incomplete-data* version of the maximum likelihood (ML) problem.

Algorithm 1 Main Algorithm

Input: observed and model pointclouds: $\boldsymbol{X}$, $\boldsymbol{Y}$.
Output: $\boldsymbol{R}$, $\boldsymbol{t}$

1: Surface normals estimation $\tilde{\boldsymbol{X}}$, $\tilde{\boldsymbol{Y}}$ according to Section 4.3
2: Divide $\tilde{\boldsymbol{Y}}$, $\tilde{\boldsymbol{X}}$ to K groups according to Section 4.3
3: Update $\boldsymbol{\Theta}_k$ in parallel for each group $k = 1, \ldots, K$ according to Algorithm 2.
4: Average $\boldsymbol{R}_k$ according to equation (4.14).
5: Update $\boldsymbol{t}$ according to equation (4.15).
6: Return $\boldsymbol{R}$, $\boldsymbol{t}$.

4.3 Point Registration Algorithm

In this section we describe the proposed approach for 3D point registration using Kent distribution. The method relies on decoupling of rotation from translation by estimating rotation based on surface normals and translation from the positional data points. Algorithms 1 and 2 outline the steps of the method.

Surface Normals Estimation

Several methods exist to estimate the surface normals associated with each point in the pointcloud [61]. In this chapter we compute the normals using principal component analysis (PCA) method. To improve the performance of the proposed method, we preprocess the surface normals with: a) orientation consistency and b) outliers removal. The orientations of surface normals computed by PCA are ambiguous (i.e. it is impossible to solve for the sign of normal vectors). To ensure we have consistent surface normals in the algorithm, we use a defined viewpoint $\boldsymbol{v}_p$ to make their orientation consistent [107]. All normals $\boldsymbol{n}_i$ of points $\boldsymbol{x}_i$ with the same directions should satisfy $\boldsymbol{n}_i \cdot (\boldsymbol{v}_p - \boldsymbol{x}_i) > 0$. We also use a modified version of outlier removal method for the surface normals. We utilize *cosine similarity* in order to remove normals that are far away from their *mean direction*. The method is detailed in [142] and [108].

Clustering with Spherical k-means

In the most of urban scene pointclouds, the majority of surface normals are estimated based on prominent surfaces such as buildings which generate the pattern of surface normals concentrated in specific areas on the sphere (for example see Fig. 4.2). Therefore we use spherical k-means approach [49] to cluster surface normals and use those normals from the same cluster in both observed and model data points. This helps to run the algorithm in parallel for the smaller size of data points in each cluster and finally take an average to find the overall rotation matrix in each step. It is worth noting that the spherical clustering is closely related to ML estimate of Von-Mises distribution which is the same as Kent distribution when $\beta = 0$ [5]. Fig. 4.3 depicts steps in order to prepare surface normals for the EM algorithm which will be explained in the following subsection.

EM-like Algorithm

A variant of EM approach is adopted here to estimate the parameter $\boldsymbol{\Theta}$. As stated in Section 4.2, since the point correspondence is not available (i.e. the observed data is viewed as being *incomplete*), log-likelihood function in (4.4) cannot be optimized directly.

Algorithm 2 EM-like Algorithm

Initialization: $\kappa, \beta, \boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \boldsymbol{R}$

1: **while** not converged **do**
2: *E-step:* Compute posterior probabilities τ_{mn} according to equation (4.6)
3: *M-step:*
4: Update $\boldsymbol{R}$ according to equation (4.12)
5: Compute parameters using moment estimates according to Section 4.3
6: **end while**
7: Return $\kappa, \beta, \boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \boldsymbol{R}$

The *complete-data log-likelihood* function is given by

$$\log L_c(\boldsymbol{\Theta}) = \sum_{n=1}^{N}\sum_{m=1}^{M} P(\boldsymbol{z}_n|\tilde{\boldsymbol{x}}_n)\Big(\log \pi_0 + \log p_0(\tilde{\boldsymbol{x}}_n) + \log \pi_m + \log \mathrm{FB}_5(\tilde{\boldsymbol{x}}_n|\boldsymbol{z}_n = \tilde{\boldsymbol{y}}_m)\Big), \tag{4.5}$$

where $P(\boldsymbol{z}_n|\tilde{\boldsymbol{x}}_n)$ is the posterior correspondence probability that links the observed data to the realization of hidden variables. The parameters will be found by minimizing the negative of the conditional expectation of $\log L_c(\boldsymbol{\Theta})$ iteratively in the following two steps.

E-step

The hidden correspondence-label $\boldsymbol{z}_n$ will be handled by the *E-step*, which is computing the posterior probability distribution τ_{mn}. This posterior distribution plays the role of *soft-assignment* between the model normals $\tilde{\boldsymbol{y}}_m$ and the observed normals $\tilde{\boldsymbol{x}}_n$, and can be computed as

$$\tau_{mn} = P(\boldsymbol{z}_n = \tilde{\boldsymbol{y}}_m|\tilde{\boldsymbol{x}}_n) = \frac{\pi_m \mathrm{FB}_5(\tilde{\boldsymbol{x}}_n|\boldsymbol{z}_n = \tilde{\boldsymbol{y}}_m)}{p(\tilde{\boldsymbol{x}}_n)}, \tag{4.6}$$

for $m = 1, \ldots, M$, $n = 1, \ldots, N$, and where the denominator is defined in (4.2). The posterior probability of assigning observed normal to an outlier is $\tau_{0n} = 1 - \sum_{m=1}^{M} \tau_{mn}$. By replacing τ_{mn} in (4.5) we have the function that needs to be maximized in the *M-step*,

$$Q(\boldsymbol{\Theta}) = \sum_{n=1}^{N}\sum_{m=1}^{M} \tau_{mn}\big(\log \pi_0 + \log p_0(\tilde{\boldsymbol{x}}_n) + \log \pi_m + \log \mathrm{FB}_5(\tilde{\boldsymbol{x}}_n|\boldsymbol{z}_n = \tilde{\boldsymbol{y}}_m)\big). \tag{4.7}$$

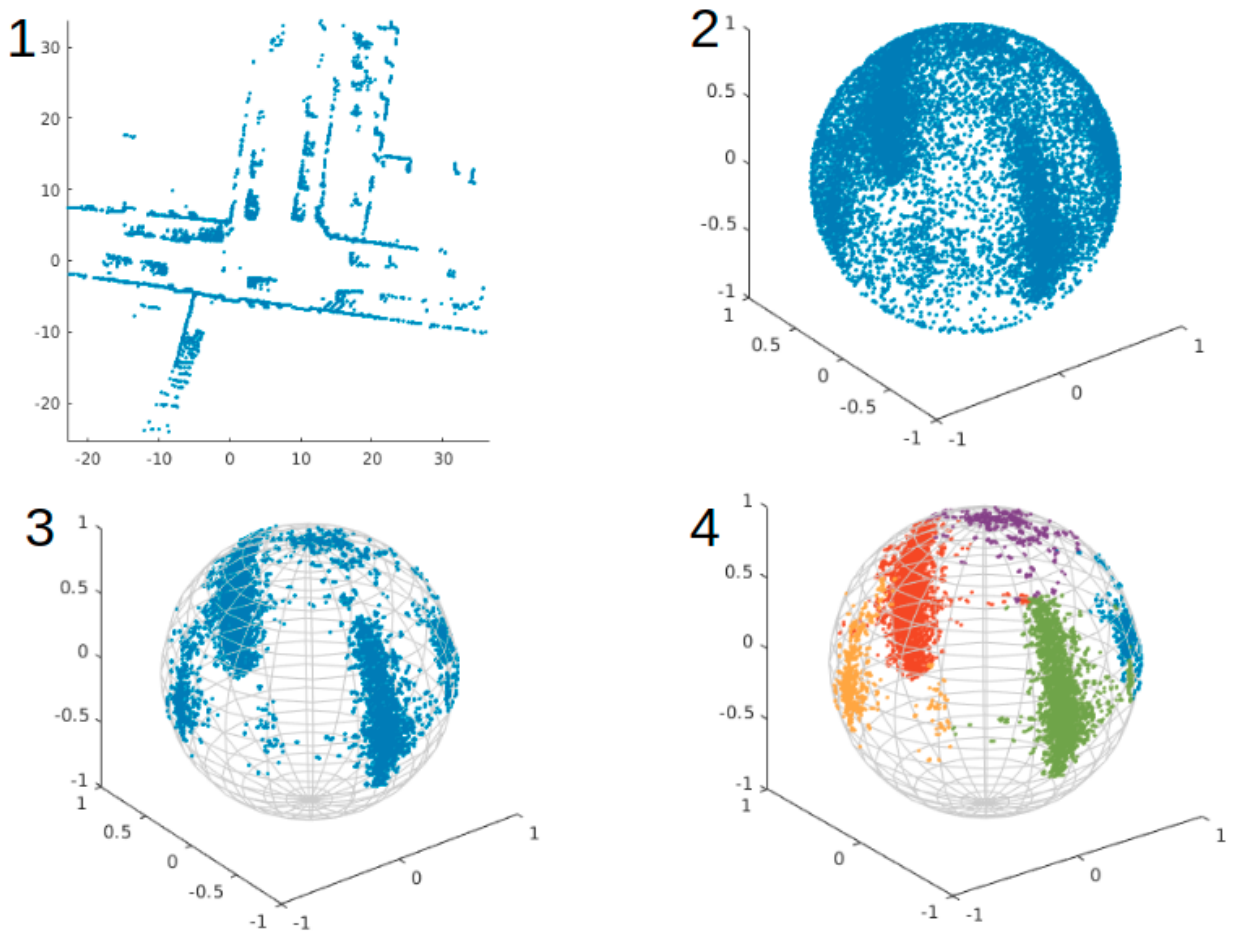

Figure 4.3: The process of preparing surface normals from pointcloud data. 1) pointcloud 2) estimating surface normals 3) outlier removal 4) *spherical* clustering.

M-step

This step requires to find the parameter $\mathbf{\Theta}$ of the distribution by maximizing the $Q(\mathbf{\Theta})$ function

$$\mathbf{\Theta}^* = arg\,max_{\mathbf{\Theta}}\, Q(\mathbf{\Theta}). \tag{4.8}$$

Although the optimization problem in (4.8) has closed form solution for some types of probability distributions, unfortunately due to the orthogonality constraints on Kent distribution parameters and rotation matrix, this constrained nonlinear problem is hard to solve. Therefore we solve the problem in two sub-steps: finding rotation matrix and then updating Kent distribution parameters.

Rotation Matrix Update:

Substituting in (4.7) we have

$$\begin{aligned} Q(\boldsymbol{\Theta}) = \sum_{m=1}^{M}\sum_{n=1}^{N}\tau_{mn}\Big(& \log\pi_0 + \log p_0(\tilde{\boldsymbol{x}}_n) \\ & + \log\pi_m + \log c(\kappa,\beta)^{-1} \\ & + \kappa\tilde{\boldsymbol{y}}_m^T\boldsymbol{R}\tilde{\boldsymbol{x}}_n + \beta\big[(\boldsymbol{\gamma}_1^T\boldsymbol{R}\tilde{\boldsymbol{x}}_n)^2 - (\boldsymbol{\gamma}_2^T\boldsymbol{R}\tilde{\boldsymbol{x}}_n)^2\big]\Big). \end{aligned} \tag{4.9}$$

The optimal rotation matrix can be found by maximizing Q-function with respect to the rotation matrix which belongs to the set of *special orthogonal* matrices $SO(3)$ defined as

$$SO(3) = \{\boldsymbol{R} \in \mathbb{R}^{3\times 3} | \boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I}, \det\boldsymbol{R} = 1\}. \tag{4.10}$$

After removing constant terms and also minimizing negative of Q-function, the constraint optimization problem can be written as,

$$\begin{aligned} & \boldsymbol{R}^* = arg\,min_{\boldsymbol{R}\in SO(3)} \; -Q(\boldsymbol{\Theta}) = arg\,min_{\boldsymbol{R}\in SO(3)} \\ & \sum_{m=1}^{M}\sum_{n=1}^{N}\tau_{mn}\big(-\kappa\tilde{\boldsymbol{y}}_m^T\boldsymbol{R}\tilde{\boldsymbol{x}}_n + \beta\big[(\boldsymbol{\gamma}_2^T\boldsymbol{R}\tilde{\boldsymbol{x}}_n)^2 - (\boldsymbol{\gamma}_1^T\boldsymbol{R}\tilde{\boldsymbol{x}}_n)^2\big]\big) \\ & = arg\,min_{\boldsymbol{R}\in SO(3)} -\kappa\sum_{m=1}^{M}\sum_{n=1}^{N}\tau_{mn}\tilde{\boldsymbol{y}}_m^T\boldsymbol{R}\tilde{\boldsymbol{x}}_n \\ & + \beta\sum_{m=1}^{M}\sum_{n=1}^{N}\tau_{mn}(\tilde{\boldsymbol{x}}_n^T\boldsymbol{R}^T(\boldsymbol{\gamma}_2\boldsymbol{\gamma}_2^T - \boldsymbol{\gamma}_1\boldsymbol{\gamma}_1^T)\boldsymbol{R}\tilde{\boldsymbol{x}}_n). \end{aligned} \tag{4.11}$$

Using matrix trace and its *cyclic property*, the first term above can be rewritten as

$$\begin{aligned} & \sum_{m=1}^{M}\sum_{n=1}^{N}\tau_{mn}\tilde{\boldsymbol{y}}_m^T\boldsymbol{R}\tilde{\boldsymbol{x}}_n = \sum_{m=1}^{M}\sum_{n=1}^{N}\mathrm{tr}(\tau_{mn}\tilde{\boldsymbol{y}}_m^T\boldsymbol{R}\tilde{\boldsymbol{x}}_n) = \\ & \sum_{m=1}^{M}\sum_{n=1}^{N}\mathrm{tr}(\boldsymbol{R}\tau_{mn}\tilde{\boldsymbol{x}}_n\tilde{\boldsymbol{y}}_m^T) = \mathrm{tr}(\boldsymbol{R}\sum_{m=1}^{M}\sum_{n=1}^{N}\tau_{mn}\tilde{\boldsymbol{x}}_n\tilde{\boldsymbol{y}}_m^T). \end{aligned}$$

The second term in (4.11) also can be simplified the same way,

$$\sum_{m=1}^{M}\sum_{n=1}^{N}\tau_{mn}\tilde{\boldsymbol{x}}_n^T\boldsymbol{R}^T(\boldsymbol{\gamma}_2\boldsymbol{\gamma}_2^T-\boldsymbol{\gamma}_1\boldsymbol{\gamma}_2^T)\boldsymbol{R}\tilde{\boldsymbol{x}}_n =$$
$$\sum_{m=1}^{M}\sum_{n=1}^{N}\mathrm{tr}\big(\tau_{mn}\tilde{\boldsymbol{x}}_n^T\boldsymbol{R}^T(\boldsymbol{\gamma}_2\boldsymbol{\gamma}_2^T-\boldsymbol{\gamma}_1\boldsymbol{\gamma}_1^T)\boldsymbol{R}\tilde{\boldsymbol{x}}_n\big) =$$
$$\sum_{m=1}^{M}\sum_{n=1}^{N}\mathrm{tr}\big(\boldsymbol{R}^T(\boldsymbol{\gamma}_2\boldsymbol{\gamma}_2^T-\boldsymbol{\gamma}_1\boldsymbol{\gamma}_1^T)\boldsymbol{R}\tau_{mn}\tilde{\boldsymbol{x}}_n\tilde{\boldsymbol{x}}_n^T\big) =$$
$$\mathrm{tr}\bigg(\boldsymbol{R}^T(\boldsymbol{\gamma}_2\boldsymbol{\gamma}_2^T-\boldsymbol{\gamma}_1\boldsymbol{\gamma}_1^T)\boldsymbol{R}\sum_{m=1}^{M}\sum_{n=1}^{N}\tau_{mn}\tilde{\boldsymbol{x}}_n\tilde{\boldsymbol{x}}_n^T\bigg).$$

By substituting the following matrices

$$\boldsymbol{A} = \boldsymbol{\gamma}_2\boldsymbol{\gamma}_2^T-\boldsymbol{\gamma}_1\boldsymbol{\gamma}_1^T$$
$$\boldsymbol{B} = \sum_{m=1}^{M}\sum_{n=1}^{N}\tau_{mn}\tilde{\boldsymbol{x}}_n\tilde{\boldsymbol{x}}_n^T$$
$$\boldsymbol{C} = \sum_{m=1}^{M}\sum_{n=1}^{N}\tau_{mn}\tilde{\boldsymbol{x}}_n\tilde{\boldsymbol{y}}_m^T,$$

and rearranging the terms, we can rewrite (4.11) in a compact matrix form as

$$\boldsymbol{R}^* = arg\,min_{\boldsymbol{R}\in SO(3)}\ \beta\mathrm{tr}(\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{R}\boldsymbol{B}) - \kappa\mathrm{tr}(\boldsymbol{R}\boldsymbol{C}). \tag{4.12}$$

Since the constraint in (4.10) is the set of *special orthogonal group* or the set of rotation matrices which are smooth, it is convenient to use Riemannian manifold optimization. We use the manifold optimization solver from the package in [16].

Distribution Parameters Update:

After updating the rotation matrix, we need to update distribution parameters as well. Recalling equation (4.9), we just retain terms that depend on the parameters and also substitute the optimal rotation matrix from previous step,

$$(\kappa^*,\beta^*,\boldsymbol{\gamma}_1^*,\boldsymbol{\gamma}_2^*) = arg\,min_{\boldsymbol{\gamma}_1^T\cdot\boldsymbol{\gamma}_2=0}\ \sum_{m,n=1}^{M,N}\tau_{mn}\bigg(-\log c(\kappa,\beta)^{-1}$$
$$-\kappa\tilde{\boldsymbol{y}}_m^T\boldsymbol{R}^*\tilde{\boldsymbol{x}}_n+\beta\big[(\boldsymbol{\gamma}_2^T\boldsymbol{R}^*\tilde{\boldsymbol{x}}_n)^2-(\boldsymbol{\gamma}_1^T\boldsymbol{R}^*\tilde{\boldsymbol{x}}_n)^2\big]\bigg). \tag{4.13}$$

There are some attempts to solve variant of this problem (in the context of mixture of Kent distributions) iteratively using BFGS quasi-Newton method with reparametrization of variables [12] or BSLM approach [87], but there is no guarantee to find a solution considering the nonconvex nature of the problem. A more convenient method to solve for parameters, as described in [92], is the *moment estimation* method. The method should be adapted for the mixture of Kent distribution, as it is only described before for the unimodal Kent distributions. Here we compute the *weighted sample moments* as follows,

$$\bar{\tilde{\boldsymbol{x}}} = \frac{\sum_{n=1}^{N} \mathcal{T}_n(\boldsymbol{R}^* \tilde{\boldsymbol{x}}_n)}{N_p}$$
$$\boldsymbol{S} = \frac{\sum_{n=1}^{N} \mathcal{T}_n(\boldsymbol{R}^* \tilde{\boldsymbol{x}}_n)(\boldsymbol{R}^* \tilde{\boldsymbol{x}}_n)^T}{N_p}$$

where $\mathcal{T}_n = \sum_{m=1}^{M} \tau_{mn}$ and $N_p = \sum_{m=1}^{M}\sum_{n=1}^{N} \tau_{mn}$. Using the above *weighted moments*, the rest of the method has been explained in [59], and we exclude the details here for brevity.

Using the moment estimate instead of ML estimate makes the EM algorithm to lose the theoretical convergence guarantees, however as stated in [59], the moment estimation is very close to ML estimate in case of small eccentricity $\frac{2\beta}{\kappa}$ or large κ, which is a usual case in practice.

Rotation averaging and Translation:

Finally we need to compute average of the rotations from different clusters and estimate the translation. Rotation averaging in Euclidean sense can be computed as,

$$\boldsymbol{R} = \overline{\boldsymbol{R}}\boldsymbol{U}\text{diag}(\frac{1}{\sqrt{\Lambda_1}}, \frac{1}{\sqrt{\Lambda_2}}, \frac{s}{\sqrt{\Lambda_3}})\boldsymbol{U}^T \tag{4.14}$$

where $\overline{\boldsymbol{R}} = \frac{\sum_i^N \boldsymbol{R}_i^*}{N}$, $N^2\boldsymbol{U}^T\boldsymbol{D}\boldsymbol{U} = N^2\overline{\boldsymbol{R}}^T\overline{\boldsymbol{R}}$, and $\boldsymbol{D} = \text{diag}(\Lambda_1, \Lambda_2, \Lambda_3)$. Also $s = 1$ if $\det(\overline{\boldsymbol{R}})$ is positive and $s = -1$ otherwise [79]. $\boldsymbol{R}_i^*$ are the optimal rotation matrices found by the algorithm for each cluster in the dataset. The translation vector also can easily be computed based on the positional mean of rotated *observed* data and *model* data,

$$\boldsymbol{t}^* = \boldsymbol{\mu}_y - \boldsymbol{R}^*\boldsymbol{\mu}_x. \tag{4.15}$$

This is valid, since it is the optimal translation in the sense of Euclidean norm which has been used in ICP-based methods as well.

4.4 Experimental Results

In this section the proposed method is tested and verified via 3D pointcloud registration on real indoor scanner pointclouds [96]. We pre-processed the pointcloud frames before

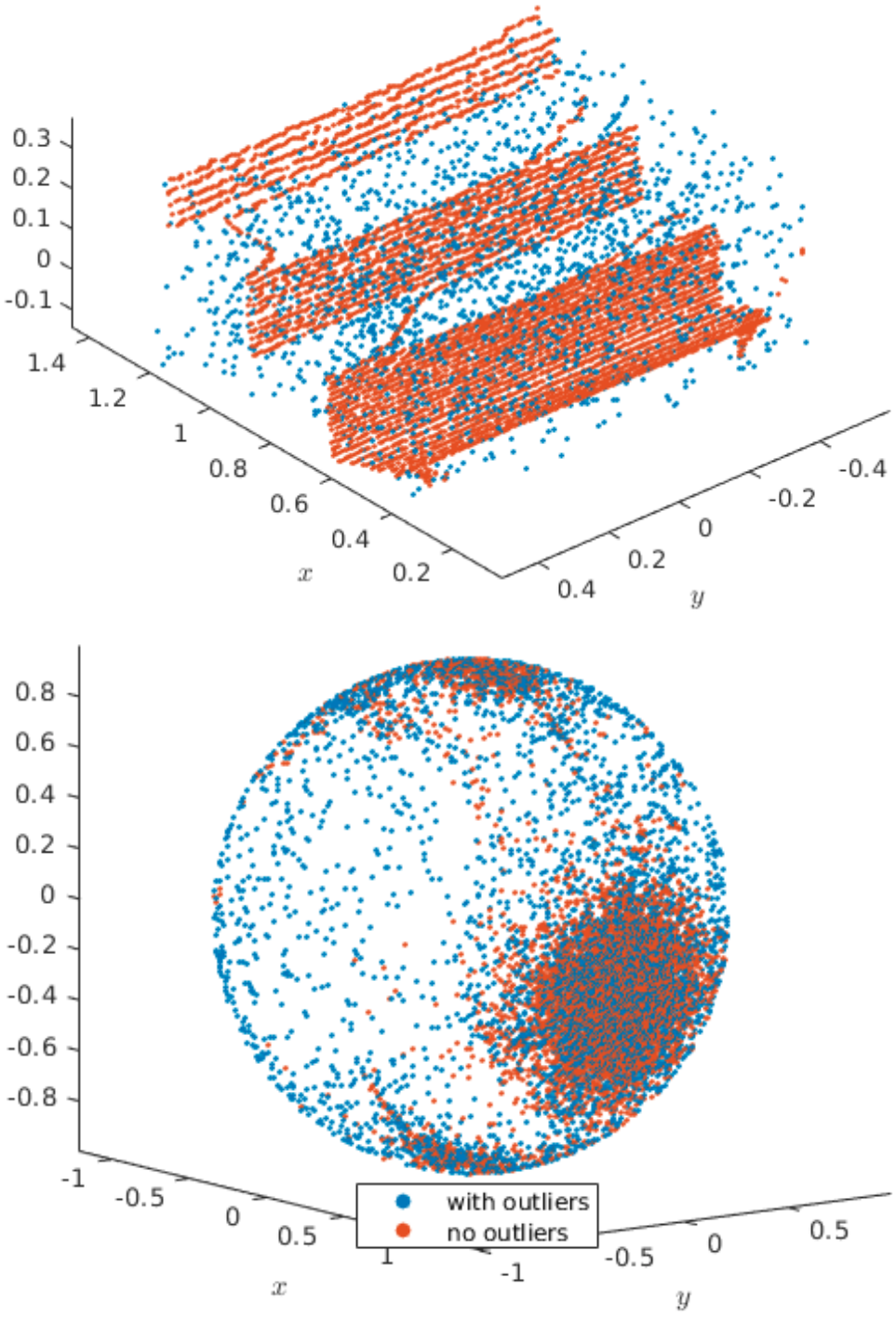

Figure 4.4: Top: Example of staircase pointcloud with 20% outliers (blue points) and its associated surface normals. Bottom: Comparison of rotation and translation errors.

applying the algorithm. In the first stage, frames were down sampled to 10% of the original size (about $10,000$ points) which was empirically chosen. In addition, all the processes on the surface normals described in Section 4.3 was performed. The proposed method was implemented in MATLAB. The algorithm was initialized with identity rotation matrix and zero translation vector. The distribution parameters were initialized in *E*-step based on the current frame. We empirically found best results with 15 neighborhood points for surface normals estimation. We adopt the error metric for comparison of estimated and true rotation matrices from [129],

$$e_{\widehat{\boldsymbol{R}}} = \cos^{-1}\left(\frac{\mathrm{tr}(\boldsymbol{R}^T\widehat{\boldsymbol{R}}) - 1}{2}\right)$$

where $\widehat{\boldsymbol{R}}$ and $\boldsymbol{R}$ are estimated and true rotation matrix, respectively. And for the translation error we have $l2$-norm of the difference of estimated and true translation $e_{\widehat{\boldsymbol{t}}} = ||\widehat{\boldsymbol{t}} - \boldsymbol{t}||_2$.

Outlier Robustness

We performed a set of point registration experiments comparing the robustness of the proposed method with ICP and NDT methods. Fig. 4.4 shows an example of frame with outlier points in blue. It can be seen that how the positional outliers impact and spread their associated surface normals on the unit sphere. The bottom plots illustrate the average of rotation and translation errors with different level of outliers injected into the pointclouds over $N_{trials} = 100$ trials. As seen, ICP and NDT methods have inconsistent and higher error with regard to outliers which makes them unreliable in practice. In contrast, the proposed method has relatively lower errors and consistent with the level of outliers.

Performance Evaluation

The results from comparison of different point registration methods are tabulated in table 4.1. The results show the average rotation and translation error on the sequence of same frames from the dataset. For NDT [123], the grid size is set as $2.0, 4.0, 2.0, 1.0$ meters with aligns the pointclouds in a finer-coarser-to-finer manner, and for CICP [32] the scale factor is $\sigma = 1.0$ meter with a decay rate of 0.96. Fig. 4.6 represents an example of the point matching of two consecutive frames. The green is the pointcloud that is transformed back into the model pointcloud based on the estimated rigid transformation found by the proposed method. As seen in the table 4.1, the proposed algorithm outperforms other methods with regard to translation error and generates satisfying rotation errors compared with the state-of-the-art NDT and GICP. It is reasonable to infer that the more information involved, the more accurate of the point registration algorithm. For instance, the ICP and CICP based on point-to-point distance produce the worse matching results. Likewise, GICP and NDT utlize more geometrical information like curvatures and more abstracted covariance matrices, which guarantees the better results.

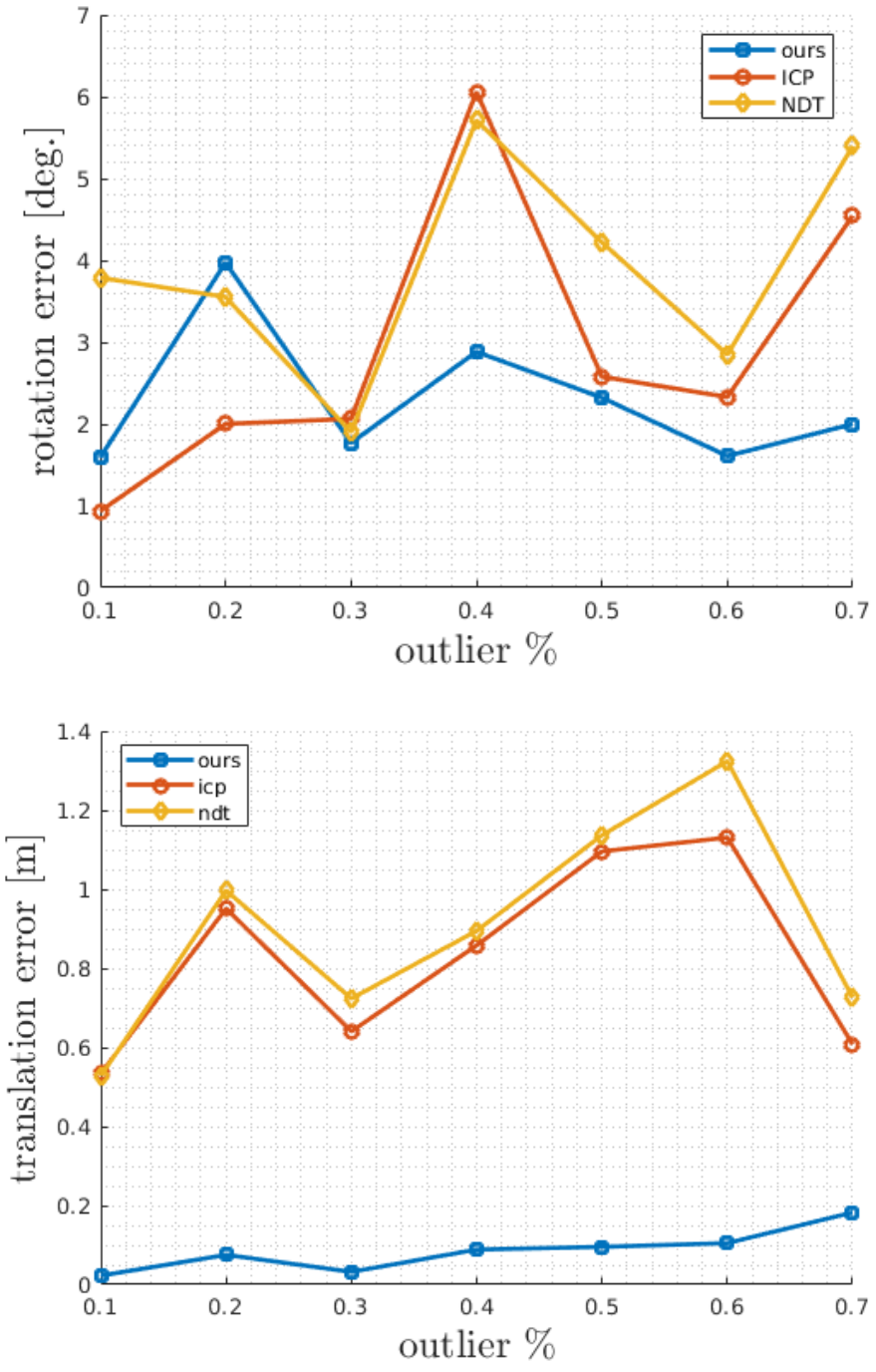

Figure 4.5: Top: Example of staircase pointcloud with 20% outliers (blue points) and its associated surface normals. Bottom: Comparison of rotation and translation errors.

Table 4.1: Comparison of PR Methods

Trials	ICP	PointNetLK	RPM-Net	Ours
angular error Avg. $e_{\widehat{\boldsymbol{R}}}$	1.27	0.847	0.305	0.74
translation error Avg. $e_{\widehat{\boldsymbol{t}}}$	0.403	0.0054	0.003	0.018

The proposed feature-based point registration method has two advantages over other methods: firstly only the directional information is utilized for rotation estimation and secondly the computationally expensive correspondence phase in ICP- or NDT-based methods is avoided, but the registration results are still competitive even with NDT or GICP. On the other hand, since the indoor pointcloud is free of most of missing data comparing with outdoor scenes, we expect our method would perform better in the presence of outliers, occluded, and missing data as presented in the previous experiment case.

Finally as stated in Section 4.3, although using *moment estimate* method makes the EM algorithm without convergence guarantee, since the algorithm utilizes the highly concentrated surface normals on the unit sphere (i.e. large κ) the moment estimate is closely related to ML estimates. In fact, the empirical results from our experiment show that (4.7) the Q-function is not decreasing after each iteration of the algorithm.

4.5 Conclusion

In this chapter, our proposed feature-based method for 3D point registration was presented and evaluated with examples of indoor pointclouds. The algorithm incorporates surface normals in the scene as directional information to be used within our cohesive and probabilistic framework which improves the registration accuracy comparing to ICP-based methods. The method utilizes the fact that the translation-invariant property of surface normals decouples the estimation of rotation from translation. Additionally, the proposed method provides a robust mechanism that rejects outliers in the correspondence phase in a probabilistic fashion. The computation time comparison is not analyzed in this chapter and one possible future direction can be reducing the computation time of the proposed algorithm.

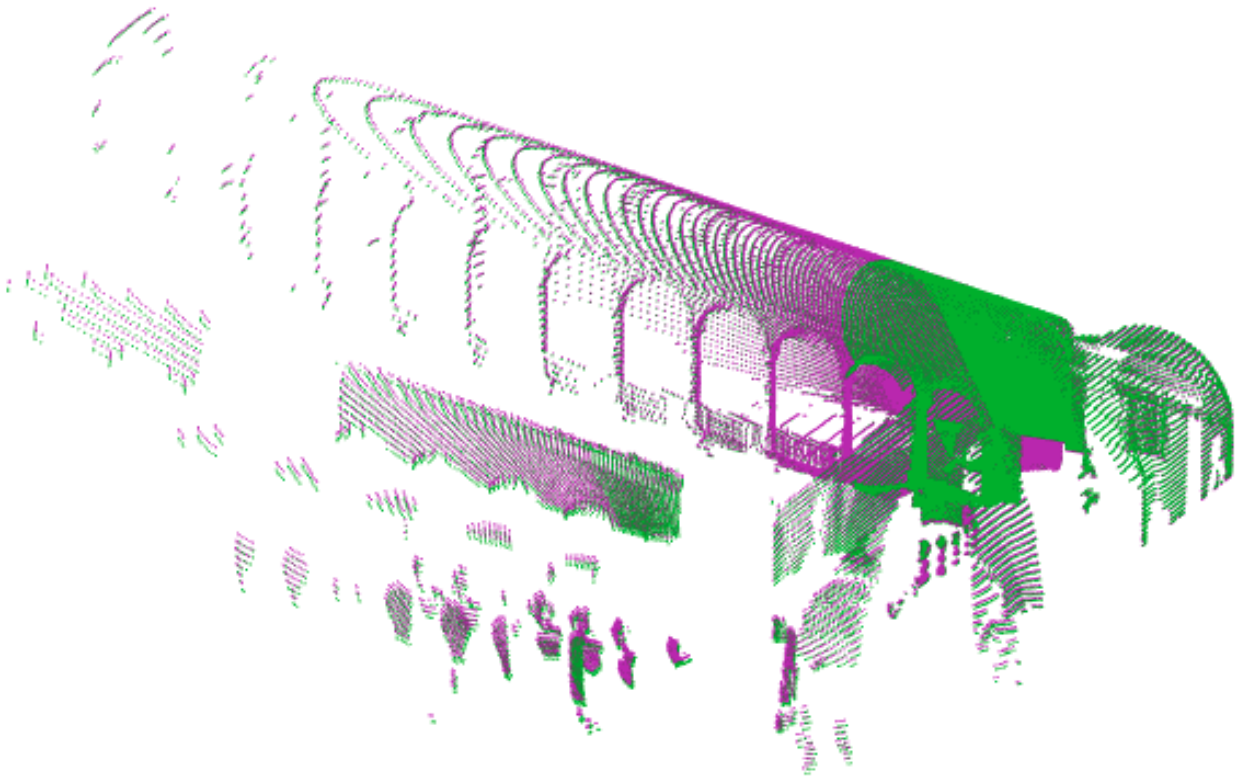

Figure 4.6: An example of matching quality (i.e. frames alignment) in two consecutive frames from indoor scanning data set [96] using the proposed method. Purple: the model pointcloud. Green: the transformed pointcloud.

Chapter 5

Deep Graph Network Point Registration

5.1 Overview

In recent years with the sensory advancement of robotic systems equipped with LiDAR (Light Detection and Ranging) technologies, deployment of fast and accurate algorithms is inevitable for processing and extracting information from 3D data points in real-time applications. A lot of progress has been made in most computer vision tasks from image classification and detection to semantic segmentation and reconstruction. But since 3D point clouds are unstructured and have permutation ambiguity, make them a difficult choice when using deep learning neural architectures.

The point registration or point cloud matching is a crucial part in many engineering applications and scientific disciplines such as robot navigation, autonomous driving, graphics, object modeling, and medical imaging. Introduction of Lidar sensors and other 3D range scanners offers an opportunity to obtain ample amount of sensor information from the environment in the form of 3D point clouds. It is important to acquire a very accurate point matching, despite the sensors errors and limitations enforced by each setting.

The point registration utilizes 3D point dataset of observations and tries to find the best rigid transformation hypothesis that maps one frame of the pointcloud to the other. The rigid transformation $(\boldsymbol{R}, \boldsymbol{t}) \in SE(3)$ consists of a rotation matrix $\boldsymbol{R} \in SO(3)$ and a translation vector $\boldsymbol{t} \in \mathbb{R}^3$; and a point $\boldsymbol{x} \in \mathbb{R}^3$ can be accordingly transformed using the following equation $\boldsymbol{T}(\boldsymbol{x}) = \boldsymbol{R}\boldsymbol{x} + \boldsymbol{t}$.

The point registration has recently received extensive attention in autonomous driving, since the rapidly developing 3d sensor technology enriches the intelligent vehicle with the capacity of accurate mapping and localization. Superiority of 3D LiDAR over other sensing technologies like camera which can be reliably employed in night and bad weather (e.g. rain, snow) makes it a suitable choice of 3D perception for intelligent vehicles. However, the limited sensor's coverage, intrinsic sensor errors, and outliers caused by occlusion or unpredictable traffic participators in the real traffic scenarios are burdensome

challenges which needs to be addressed when we deploy the point registration algorithms in autonomous driving.

In this work we introduce a flexible graph-based deep learning model for point registration on 3D point clouds. In particular inspired by the development of a new graph network formalism in [7], a graph-based framework has been introduced that consists of three main parts (i.e. *graph network blocks*). First the input to the model which is two point cloud frames are transformed into a graph representation. Then,

1. the graph representation of inputs will be encoded into a latent representation by a *graph network block*;
2. the latent representation is passing through the core graph network multiple times;
3. and finally the decoder decodes the latent graphs into the output graph. The global attribute of the output graphs represents the rigid transformation.

We train and test our model on *ModelNet40* dataset [135]. We also compare our results with the recent state-of-the-art method in the field of point registration. The proposed method is flexible and configurable in its structure, and it is possible to incorporate other modules into it, in order to improve the results.

The graph networks are far better in terms of inductive biases comparing with convolutional or fully connected layers. They are invariant to permutations of the input signal, which makes them a suitable choice when we work with unordered and unstructured form of data like point cloud in our case. Graph networks are flexible and configurable. It is easily possible to incorporate other mechanisms and structures within the graph net blocks. They are also flexible in terms of the size and shape of input and output, a property that is lacking in other deep learning algorithms. And most importantly related to the task of point registration, by using graph network the local geometry of each point as well as the whole structure and topology of point clouds in a larger range are considered in establishing correspondences between frames. This ultimately helps in an accurate alignment estimate of the two frames. This property will be reinforced later by the introduction of non-local mechanism that explicitly encodes the long-range dependencies in each frame.

5.2 Related Work

The problem of point registration can be divided into two main categories of geometric- and learning-based approaches.

Geometric Point Registration

The classical methods of point registration are based on geometry of the point clouds such as coordinates of the points and surface normals, or other geometric features. The focus of

these methods is to solve the nonlinear least-square problem in an accurate and efficient way. Two main methods are ICP-based and globally optimal methods.

The point registration problem can be solved with geometric-based approach with the Iterative Closest Point (ICP) algorithm as the most popular technique [11]. ICP solves the problem in two steps iteratively, 1) finding the correspondence points between frames (or alignment phase), and 2) minimizing the sum of squared residuals to find the rigid transformation (or registration phase). Several variants of ICP method improving the original method has been introduced such as ICPp2pl, ICPpl2pl, GICP [70, 115, 113]. But, one major limitation of ICP-based methods is that it can only converge to a local optimum near the initialization, and its convergence region is fairly small, especially when there are noise or outliers in the point cloud data.

Other geometric-based methods are globally optimal methods that try to find the global solution, and ultimately avoid the sensitivity to initialization [136, 137]. The globally optimal methods are mostly using branch-and-bound optimization techniques [18, 82, 94, 89] or other global optimization approaches (e.g. truncated L_2-norm, second order cone programming (SOCP), consensus set maximization [3, 60, 68]) to find the global solution. But unfortunately these type of methods are very slow, make a lot of unrealistic assumptions, and they are only practical in some limited scenarios, and clearly they are not suitable for real-time applications. In addition to ICP-based method, some other schemes use the concept of point set correlation (i.e. soft assignment) to iteratively harden the point-to-point correspondence. The logic behind these methods follows the fact that the convergence area of ICP can be widened by soft assignment techniques [40, 126, 26], so that the algorithm can converge to a better solution.

Finally, there are also some probabilistic methods that can be grouped as a subcategory of geometric-based methods [104, 72, 86]. These methods are distinct in their probabilistic interpretation of alignment phase between two point cloud frames, so with that the least-square problem in their geometric-based counterparts are modeled as a maximum likelihood problem. The probabilistic approach is based on soft assignment–the process of assigning multiple points to a point across frames with a probability–instead of hard-assignment or one-to-one association of points in the frames.

Deep Learning Point Registration

Very recently, the learning-based approaches for point registration has been introduced, which is gaining popularity in the community due to its benefits over geometric approaches. The methods are mostly focused on the correspondence phase or alignments (authors put forward various combinations of back bone feature extractors [102] for 3D point clouds, transformer networks [127] for feature representation, and outlier filtering techniques like RANSAC [35] for refining the final results [25]), and for the registration phase (or pose optimization) they still use the singular value decomposition to solve the problem similar to classical counterparts. In this sense, they are hybrid method, using both geometric- and learning-based modules to solve the problem.

Authors in [139] have incorporated both point correspondence and registration phases in an end-to-end trainable network with a match matrix at the end which resembles the kernel correlation in classical methods, so it needs to be deployed iteratively in order to converge to the final alignment. Similarly, [71] uses an end-to-end formalism for registration, utilizing 3D CNN as the correspondence generating layer to final alignment in one iteration.

Using only fully connected layers (i.e. MLPs) in the deep neural network architecture will result poor discriminative ability, which leads to a large proportion of incorrect point correspondences, and consequently results an inaccurate point registration due to the wrong alignment of frames [138, 111, 2].

Inspired by natural language processing, authors in [133] use a combination of point encoding network, attention-based module and Pointer Networks [128] to solve the registration problem. The results show that the method should be applied iteratively (or recursively) in order to refine the alignment and ultimately improve the registration quality, which may takes more time required for a on-line inference algorithm.

In this section, we describe the point registration problem formulation, later we introduce the graph network representation of point cloud frames. The input and output of each block of graph network should be a graph with the same structure but with different attributes. We explain the details of this specific framework and the graph representation of the point clouds in the following sections.

5.3 Problem Formulation

Preliminaries

Point registration problem Throughout the paper we use the following notations. M, N are the number of data points in the *target* and *source* point clouds (subscribed with $\mathcal{T}$ and $\mathcal{S}$), respectively. $\mathbf{P}_{\mathcal{T}} = \{\boldsymbol{y}_i\}_{m=1,\dots,M},\ \boldsymbol{y}_i \in \mathbb{R}^{3+d}$ and $\mathbf{P}_{\mathcal{S}} = \{\boldsymbol{x}_i\}_{n=1,\dots,N},\ \boldsymbol{x}_i \in \mathbb{R}^{3+d}$ denote the target and source point cloud Cartesian position plus any additional $d-$dimensional feature space which can be extracted by any feature off-the-shelf descriptor and used in our pipeline. Given two point cloud frames, we seek to find the rigid transformation $\mathbf{T}\,(\boldsymbol{R},\ \boldsymbol{t}) \in SE(3)$ between the target and source frames, which can be represented generally as an operator $\mathbf{P}_{\mathcal{T}} = \mathbf{T}_{\boldsymbol{R},\boldsymbol{t}}\,(\mathbf{P}_{\mathcal{S}})$. Since in general we do not assume the one-to-one feature correspondence between frames, therefore the point clouds can have different number of points, i.e. $M \neq N$.

Graph We define graph as a 3-tuple $G = (\boldsymbol{u}, V, E)$, which is directed with attributes on nodes $V = \{\boldsymbol{v_i}\}_{i=1:N^v}$, vertices $E = \{\boldsymbol{e}_k,\ r_k,\ s_k\}_{k=1:N^e}$, and a global or graph-level attribute $\boldsymbol{u}$. Nodes can be a receiver or a sender, where r_k and s_k are the index of receiver and sender nodes, respectively.

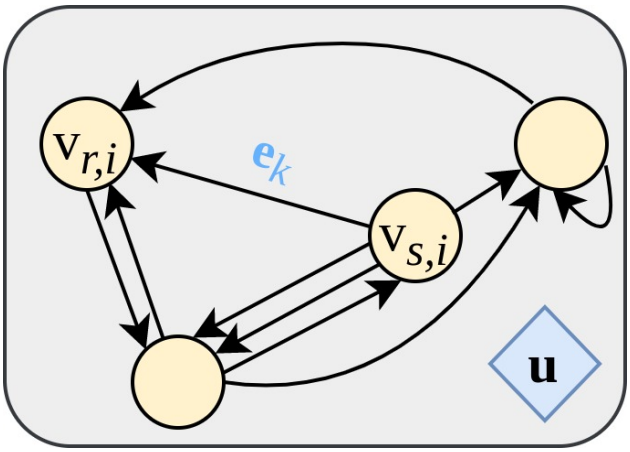

Figure 5.1: *Graph Network* (GN) Block. The building block of our proposed method is based on the GN block. It consist of nodes, edges, and graph–level attribute $\boldsymbol{u}$. Input/output of the block should also be in standard graph form explained in this work. An input graph attributes pass through a set of computational steps based on update/aggregation functions to result the output graph.

Graph Network (GN) Terminology

Inspired by the new graph network (GN) formalism in [7], we briefly explain the main block of GN and its internal structure. This frameworks defines a class of functions for relational reasoning over graph-structured representation of point clouds. The GNs make the building blocks of our proposed graph-based framework for 3D point registration. The GN block consists of three main entities: 1) nodes, 2) edges, and 3) graph-level or global attribute, as seen in Fig. 5.1. Nodes and edges carry out attributes that can be in any structural form like vector, matrix, or tensor. Note that the input and output of GN blocks should be in graph form as well.

In its complete form, within each block there exists three *update functions* $\phi(\cdot)$ and three *aggregation functions* $\rho(\cdot)$, responsible for per-edge, per-node, and per-graph updates and for aggregating information that comes from nodes and edges, respectively. Per-edge and per-node functions are reused across all edges and nodes. This means that a single graph network can be used on any graph representations with different size and shape.

$$
\begin{aligned}
\boldsymbol{e'}_k &= \phi^e\left(\boldsymbol{e}_k,\ \boldsymbol{v}_{r_k},\ \boldsymbol{v}_{s_k},\ \boldsymbol{u}\right) & \bar{\boldsymbol{e}}'_i &= \rho^{e\to v}\left(E'_i\right) \\
\boldsymbol{v'}_i &= \phi^v\left(\bar{\boldsymbol{e}}'_i,\ \boldsymbol{v}_i,\ \boldsymbol{u}\right) & \bar{\boldsymbol{e}}' &= \rho^{e\to u}\left(E'\right) \\
\boldsymbol{u'} &= \phi^u\left(\bar{\boldsymbol{e}}',\ \bar{\boldsymbol{v}},\ \boldsymbol{u}\right) & \bar{\boldsymbol{v}}' &= \rho^{v\to u}\left(V'\right)
\end{aligned}
\tag{5.1}
$$

where $E'_i = \{\boldsymbol{e'}_k, r_k, s_k\}_{r_k=i,\ k=1:N^e}$, $V' = \{\boldsymbol{v'}_i\}_{i=1:N^v}$, and $E' = \bigcup_i E'_i = \{\boldsymbol{e'}_k, r_k, s_k\}_{k=1:N^e}$. Obviously, the aggregation must be a symmetric function with respect to the input ar-

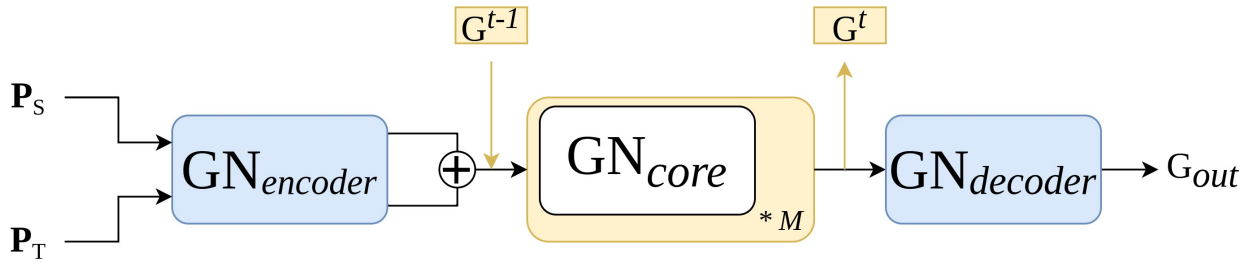

Figure 5.2: Graph Network Blocks. The proposed architecture for the task of point registration consists of three main graph network blocks: Encoder, Core, and Decoder. The Encoder embeds the first latent graph from graph representation of point clouds $\boldsymbol{P}_{\mathcal{S}}$ and $\boldsymbol{P}_{\mathcal{T}}$. The core block has been repeated M times to represent the message–passing processes in graph theory. M steps are constructed recurrently with shared parameters to also capture the temporal information which exists during iterative point cloud alignment. The Decoder extracts rigid transformation information from the last latent graphs. For more detail see Fig. 5.3

guments (i.e. permutation invariant, since in general the order of nodes/edges is ambiguous). A GN block can have any arbitrary combination of update and aggregation functions depending on its functionality and design purpose. Finally, in a specific order update/aggregation functions should be executed on any input graph (i.e the computation proceeds from edges, to nodes, and to the global level). For more detail see [7].

Graph Representation

The initial source and target graphs are constructed in such a way that connects neighbors of each point in Cartesian coordinate $\boldsymbol{p} = (x, y, z)$ within a certain radius. A standard $kd-$tree algorithm is utilized for this task. Given the point clouds $\boldsymbol{P}_{\mathcal{S}}, \boldsymbol{P}_{\mathcal{T}}$, we create a graph representation with data points as nodes $\boldsymbol{p}_i$, and directed edges $(d_{ij}, s_k = i, r_k = j)$ (both sides as multi-graph) as the distance connection between points in each point cloud frame. This initial graph later be feed into the encoder block for node and edge embedding.

We choose the radius to rather be large enough in order to capture long-distance dependencies in the structure of 3D point clouds (at the extreme end, this can be a fully connected multi-graph representation of a frame with relatively moderate number of points). Although this initial graph topology can be seen simple and inadequate, but our aim is to embed the latent node/edge features automatically by the use of proposed Encoder-Core-Decoder architecture which we explain next.

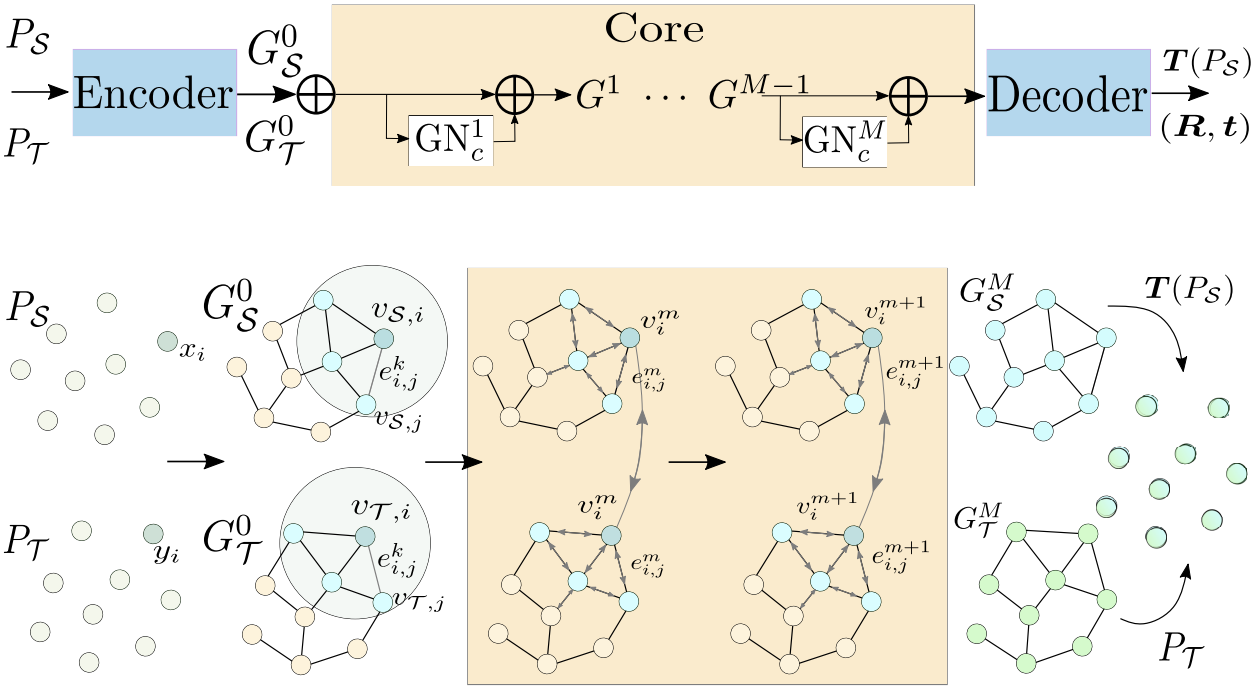

Figure 5.3: Diagram of the proposed deep graph network operating on a pair of point clouds. **Top Diagram** shows three main blocks of the model. The Core blocks does M –step message–passing with shared parameter in each $\mathbf{GN}^m_c$. **Bottom Diagram** shows the details of the model and the input/output in each part of the model. The first latent graphs $\boldsymbol{G}^0_{\mathcal{S}}$ $\boldsymbol{G}^0_{\mathcal{T}}$ carry the embeddings based on non–local mechanism introduced in Encoder section 5.4. The concatenation of both graphs enters the Core block when message–passing process aggregates information across both graphs, and also captures temporal relationships due to the use of shared functions and parameters in the structure of Core block. The Decoder simply extract rigid transformation which was encoded as the global entity in the graphs. At the end, the transformation performed on the source frame estimates the target frame (This has been used during the training phase for computing the loss function comparing it with the true transformation between frames).

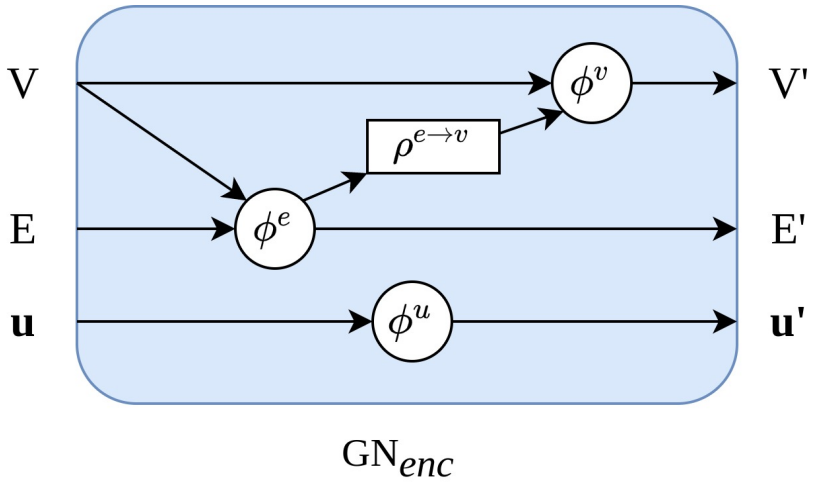

Figure 5.4: Encoder GN Block. For implementing the non–local mean, the edge update function $\phi^e(\cdot)$ is a MLP with arguments as the dot product of positional data points (a scalar pairwise–interaction function) and a vector–valued non–pairwise function (also represented as a MLP). The edge–to–node aggregation function $\rho^{e\to v}(\cdot)$ is only summation (or mean in case of normalizing the sum). The rest are also MLPs which will be described in Experiential Evaluation section 5.5.

5.4 Graph-based Point Registration

Now that all the necessary elements are defined, the details of the proposed graph-based model will be presented in this section. The overall goal of the proposed deep learning architecture is to find the discrepancy in pose structure–which is indicative of the rigid transformation–between the two graph representation of point clouds. But since the point correspondence is not one-on-one between frames, the model should learn other local/global structural characteristics in frames in order to be able to ultimately find an accurate alignment in the presence of outliers, noise, occlusion, and non-overlap effects. In the following the mechanisms used in the proposed model will be explained that address these essential requirements such that results a robust model for point registration task.

Encoder graph net

The encoder block, Enc : $\mathcal{P} \to \mathcal{G}$, embeds the point cloud representation $\boldsymbol{P}_{\mathcal{S}}$ (or $\boldsymbol{P}_{\mathcal{T}}$) as a latent graph $\boldsymbol{G}^0_{\mathcal{S}}$ (or $\boldsymbol{G}^0_{\mathcal{T}}$), where $G = (\boldsymbol{u}, V, E)$ is the graph described in section 5.3. The node embedding functions $\boldsymbol{v'}_i = \phi^v(\boldsymbol{p}_i)$ are learned MLPs on data points position. The edge functions $\boldsymbol{e'}_{ij} = \phi^e(d_{ij})$ are also MLPs on pairwise relative distance, which embed the topological relation between pairs, such as local geometry, long range relation, or directional dependencies. $\boldsymbol{e'}_{ij}$ are directed edges from sender i to receiver j. The

graph-level attribute $\boldsymbol{u}$ representing global properties is the point cloud pose which can be embedded separately but also it is possible to be as part of nodes' features (as a relative pose between correspondence pairs). The encoder block defined here is shared (i.e. shared MLPs) between the two initial graph representations. The Non-local embedding mechanism as an important part in the encoder block will be explained below.

Non-local Mechanism One of the main functionality of the encoder is the non-local mechanism [131]. *Non-local mean* can be formulated as the normalized summation of a pairwise and unitary functions on neighboring nodes in the graph as

$$y_i \propto \sum_{\forall j} f(v_i, v_j)\boldsymbol{g}(v_j),$$

where v_j's are neighboring nodes of v_i. The non-local mechanism captures the long-range dependencies between entities of its input signals. In our proposed framework, it expresses the global structural properties of the point clouds as a latent feature space. In graph matching, both local geometry of each point as well as the global structure and topology are considered in establishing correspondences, so that more correct correspondences are found. So it is beneficial to exploit the exact concept in point clouds for establishing more accurate correspondences between two frames by use of non–local mechanism. We consider dot product as the pairwise-interaction function, which is a proxy for the similarity between any location in the input signal. In another words, the global features are the representation of pairwise relationship between points with their neighbors. Non-local mechanism is robust to outliers, and it is insensitive to noise which both are rather a local phenomenon than global.

For implementation, as you see in Fig. 5.4, it is simply possible to incorporate the equations within the encoder block. The updates are the pairwise and unitary MLPs, and the aggregation function is the summation (a permutation–invariant function) as it is the case for the non-local mean.

Core graph net

The core block, Core : $\mathcal{G} \rightarrow \mathcal{G}$, is a full GN with all the update and aggregation functions. The M steps applied to the input graphs $\boldsymbol{G}_{\mathcal{S}}^0$, $\boldsymbol{G}_{\mathcal{T}}^0$ with shared weights as a sequence. The target frame can be interpreted as the transformed version of the source frame, in another words $\boldsymbol{G}_{\mathcal{T}}^M = \boldsymbol{T}_{\boldsymbol{\theta}}\left(\boldsymbol{G}_{\mathcal{S}}^0\right)$, where $\boldsymbol{\theta} = (\boldsymbol{R}, \boldsymbol{t})$ is the learned weights of the model. So in this case, the core block is a learned function that transforms the latent source graph to the latent target graph. In the case of point registration, both the message-passing as well as recurrent setting are natural to the structure of the problem.

As the two graph representation of point clouds are fed into the model, they are merged (concatenated) along the entire model passing through several modules, so by keep tracking their indices, they will be reconstructed at the end of the model after the decoder block.

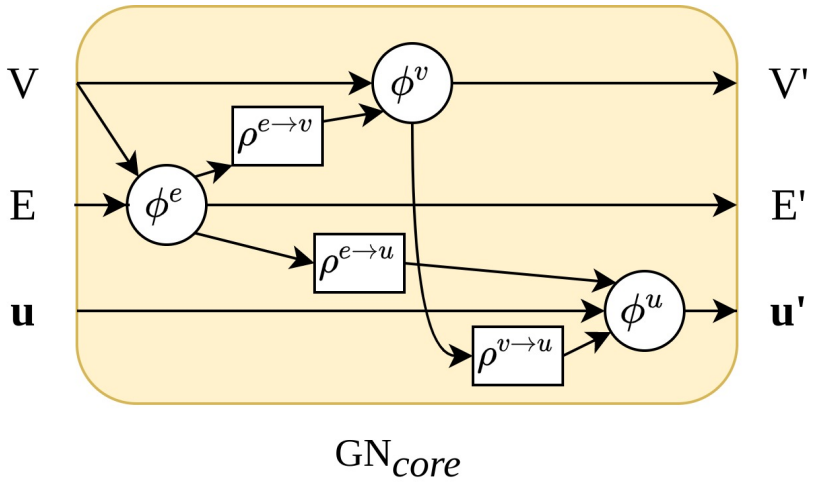

Figure 5.5: Core GN Block. All the update functions $\phi^e(\cdot),\ \phi^v(\cdot), \phi^u(\cdot)$ in this block are MLPs with shared parameters across pair of point clouds and all nodes and edges. All the aggregation functions $\rho^{e\to v}(\cdot),\ \rho^{e\to u}(\cdot),\ \rho^{v\to u}(\cdot)$ are just normalized sum of the functions' input. The message-passing and recurrent property also is implemented in the structure of the Core block.

Message-passing The M–step pass resembles the message-passing operation on graphs, the process of propagating information across the graph. The message-passing process is similar to the breaking down the process of registration of one frame to another into smaller infinitesimal steps.
Recurrent behavior The recurrent architecture also has been built when the core is applied in a sequential setting. The recurrent formalism guarantees the sequential nature of rigid transformation preventing any discontinuity or jump in the process of learning point registration (improves stability of convergence).

Decoder graph net

The decoder block Dec : $\mathcal{G} \to \mathcal{P}$ extracts information from the last latent graph $\boldsymbol{G}^M$. No aggregation function is used in this block. The nodes, edges, and global attributes of the decoded output graph can be used for any task-specific purposes. in addition to extracting the rigid transformation, the decoder merges the two latent graph $\boldsymbol{G}_{\mathcal{S}}^M$ and $\boldsymbol{G}_{\mathcal{T}}^M$ by applying the learned transformation on the source point cloud.

Loss function

In geometric-based methods, the registration can be done by iteratively minimizing the residuals in the least-square sense; or by maximum likelihood problem similar to

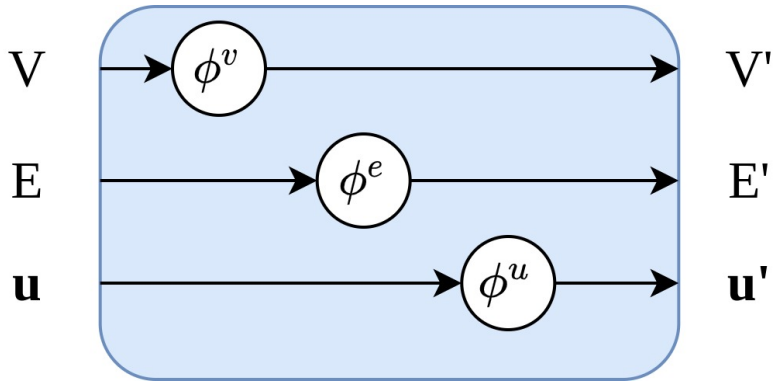

Figure 5.6: Decoder GN Block. This block only updates $\phi^e(\cdot),\ \phi^v(\cdot), \phi^u(\cdot)$ nodes, edges, and graph–level entities with no aggregation as the final step in the model. The decoder independently decodes the edge, node, and global attributes (does not compute relations, etc.), on each message-passing step.

expectation-maximization approach. For the training of deep learning models a similar formulation can be used for the loss function. Specifically, the minimization applies to the residual of estimated rigid transformation matrix with the true transformation. For the loss function the Frobenius norm has been chosen. The norm measures the difference between ground truth transformation with the estimate from the proposed model. Another term also added to incorporate a soft constraint that should be enforced on any rotation estimation to be an orthogonal matrix.

$$\text{Loss} = \|\boldsymbol{G}_{est}^{-1}\boldsymbol{G}_{gt} - \mathbb{I}\|_F + \alpha\|\boldsymbol{R}_{est}^{-1}\boldsymbol{R}_{est} - \mathbb{I}\|_F \tag{5.2}$$

where $\boldsymbol{G}_{est},\ \boldsymbol{G}_{gt} \in SE(3)$ are estimated and ground truth rigid transformations, and $\boldsymbol{R}_{est} \in SO(3)$ is the ground truth rotation matrix. The training loss is computed for each processing step of the core block. The reason is to encourage the model to solve the problem in as few steps as possible, and force the model to learn the infinitesimal changes in transformation matrix. This is possible due to the fact that we synthesize the data points needed for the training, and breaking down the registration task into smaller transformations, which allows computing the loss of the intermediate steps along the way.

5.5 Experimental validation

The process of training and testing of the proposed graph neural network constructed based on combination of train/test point clouds, and evaluation and comparison with other geometric and deep state–of–the–art methods in terms of estimation error, training time and inference time.

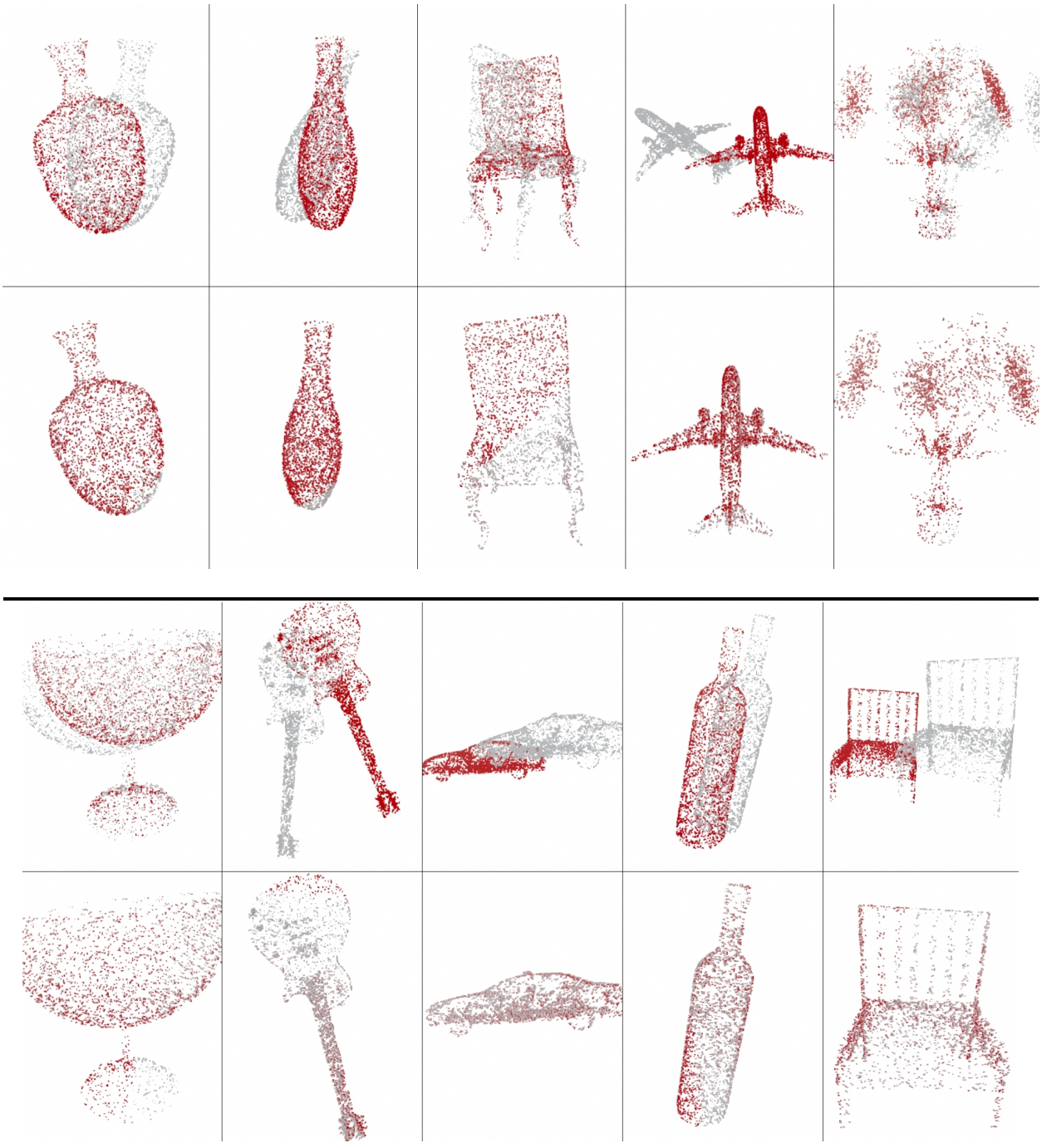

Figure 5.7: Examples of input to the trained model and the output registered results. **Row 1 & 3:** Two point clouds (source and target), each containing 2048 points, are set as the input to the model. **Row 2 & 4:** The output point clouds have been aligned and translated, showing that the trained model correctly finds the rigid transformation between frames.

Table 5.1: Comparison of Deep point registration Methods, in [deg.] & [m]

error metric	ICP	FGR	PointNetLK	RPM-Net	DCP-v2	Ours
Avg. $e_{\boldsymbol{R}_{est}}$	2.99	0.936	1.58	3.35	1.14	3.11
Avg. $e_{\boldsymbol{t}_{est}}$	0.29	0.014	0.022	0.086	0.002	0.051

Dataset

Modelnet40 has been used as the main dataset for training of the proposed model [135]. They contains of CAD model (mesh) of 40 object categories that are divided into training and testing portions. All the dataset are randomly sampled from surface and volume of meshes into point clouds with 4096 positional points. All the points have been normalized into a unit box centered at origin. Only a subset of data points are used for training and testing, since 4096 is excessive particularly when the number of neighboring is large to cover the global topology of the objects, bu comparison has been made for the different inference time when using different number of points for estimation.

Training

In terms of the training of deep point registrations, depending on the model architecture, and in particular the graph-based models, it is easily possible to train the model with synthesized data that are noise and outlier free, dense, and uniform across the topology of objects.[1] Also synthetic rigid transformation can be used, so the labels for a supervised learning training process are ready to use. Also, comparing with geometric methods that need to be run on CPU, the proposed method can be run directly on GPU as part of a neural network pipeline. And this allows to run some part of the model in parallel, and ultimately have faster solution time for point registration.

All the source point clouds have been rotated and translated randomly in the range of $[-30, +30]^{\circ}$ and $[-0.5,\ 0.5]\ m$, respectively. The combined values represent a rigid transformation matrix that works as the label for the computation of loss function at the end of each forward pass. The random rotation for each input is divided into infinitesimal rotation as a batch, which can be fed into the model and the model can learn infinitesimal rotations at the core block 5.4, when $M = 10$ steps of message–passing takes place. Small

[1] Although in practice deep neural networks tend to better generalize with adding noise into the input data, but one can argue this is not the case for point cloud registration. The noise adding has been seen as a regularization [13] in loss optimization problem or data augmentation [41]. But generalization was the main goal by indicating the use of graph representation of point clouds as the way of introducing Relational Inductive Biases, so that no such noise adding be needed. In addition, point clouds are inherently unordered and unstructured, so adding noise does not help in that front. The results in this chapter shows the training improvement. Although noise are not added for training but they were for the testing to measure the robustness of the proposed model.

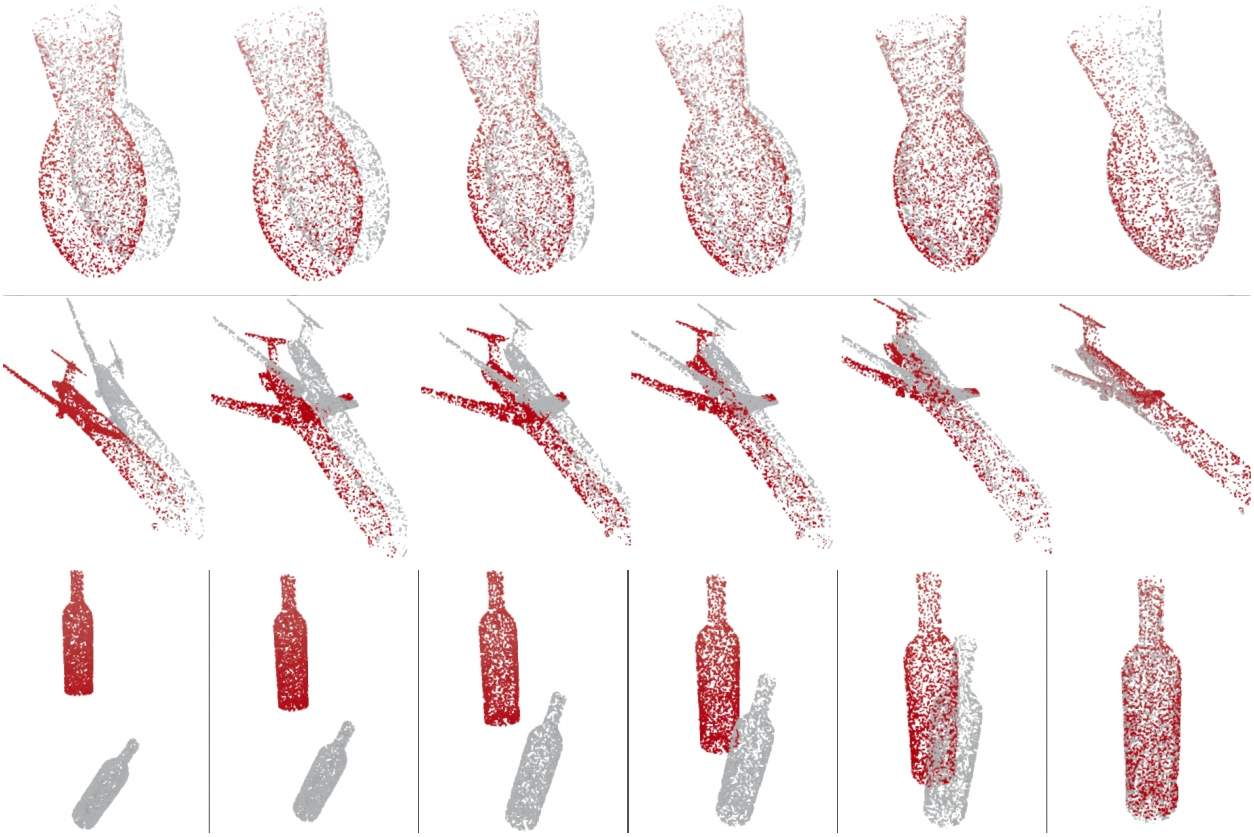

Figure 5.8: **Left to Right:** The progress in objects alignment (three objects: vase, airplane, and bottle) during M–step message passing process inside the core block 5.4. The training loss is computed for each processing step of the core block. The reason is to encourage the model to solve the problem in as few steps as possible, and force the model to learn the infinitesimal changes in transformation matrix.

step rotations are more numerically stable since they have simple representation and the composition of infinitesimal rotations is commutative.

All update functions in the model including those in the encoder, core, and decoder blocks are MLPs (fully connected layers). All MLPs have two hidden layers with ReLU activation followed by an output layer. Each layer has size 128, and all layers are followed by normalization layer [4], which is found to be improving the training stability.

Table 5.2: Accuracy results in the presence of noise & outliers

% outlier/noise	FGR	PointNetLK	RPM-Net	DCP-v2	Ours
5	1.05	1.89	3.37	1.11	2.41
10	1.06	1.89	3.42	1.13	2.42
15	2.11	2.24	3.76	1.18	2.59
20	2.67	3.08	3.88	1.34	2.71

Results

We adopt the error metric for comparison of estimated and true rotation matrices from [129] as,

$$e_{\boldsymbol{R}_{est}} = \cos^{-1}\left(\frac{\mathrm{tr}(\boldsymbol{R}_{gt}^{\top}\boldsymbol{R}_{est}) - 1}{2}\right)$$

where $\boldsymbol{R}_{est}$ and $\boldsymbol{R}_{gt}$ are estimated and true rotation matrix, respectively. The metric $e_{\boldsymbol{R}_{est}}$ returns the angle of rotation matrix $\boldsymbol{R}_{gt}^{\top}\boldsymbol{R}_{est}$ in degrees. For a perfect estimation this should be zero since the composition of a rotation matrix with its transpose is the identity matrix. And for the translation error we have $l2$-norm of the difference of estimated and true translation as

$$e_{\boldsymbol{t}_{est}} = ||\boldsymbol{t}_{est} - \boldsymbol{t}_{gt}||_2$$

Seen in Table 5.1 are the average rotation and translation error over a random set of point clouds drawn randomly from the the test portion of dataset, and also randomly chosen from different 40 categories. As seen, though different with other methods in structure and use of features, the proposed model still produces competitive results with some of the methods listed in the table.

Seen in Fig. 5.9 are examples of point registration for different object with Gaussian noise (with zero mean and unit variance) synthetically injected into the dataset. Objects are aligned relatively correct even with noise, showing that the proposed method is robust enough to demonstrate good results with corrupted data (noise or outliers in point cloud). Table 5.2 shows the accuracy of rigid transformation estimates when different levels of noise has been injected into dataset. The comparison in the table shows the results are competitive with other deep learning methods. One important property in the result points to the fact that by increasing the amount noise in the dataset the proposed method performs relatively stable with small error, confirming the long–range features as the source of alignment rather than local features that can be corrupted by noise or existence of outliers in point clouds.

In Table 5.3 the inference time has be compared with other methods with different number of points in point clouds. Subsampling from objects in *ModelNet40* dataset it is

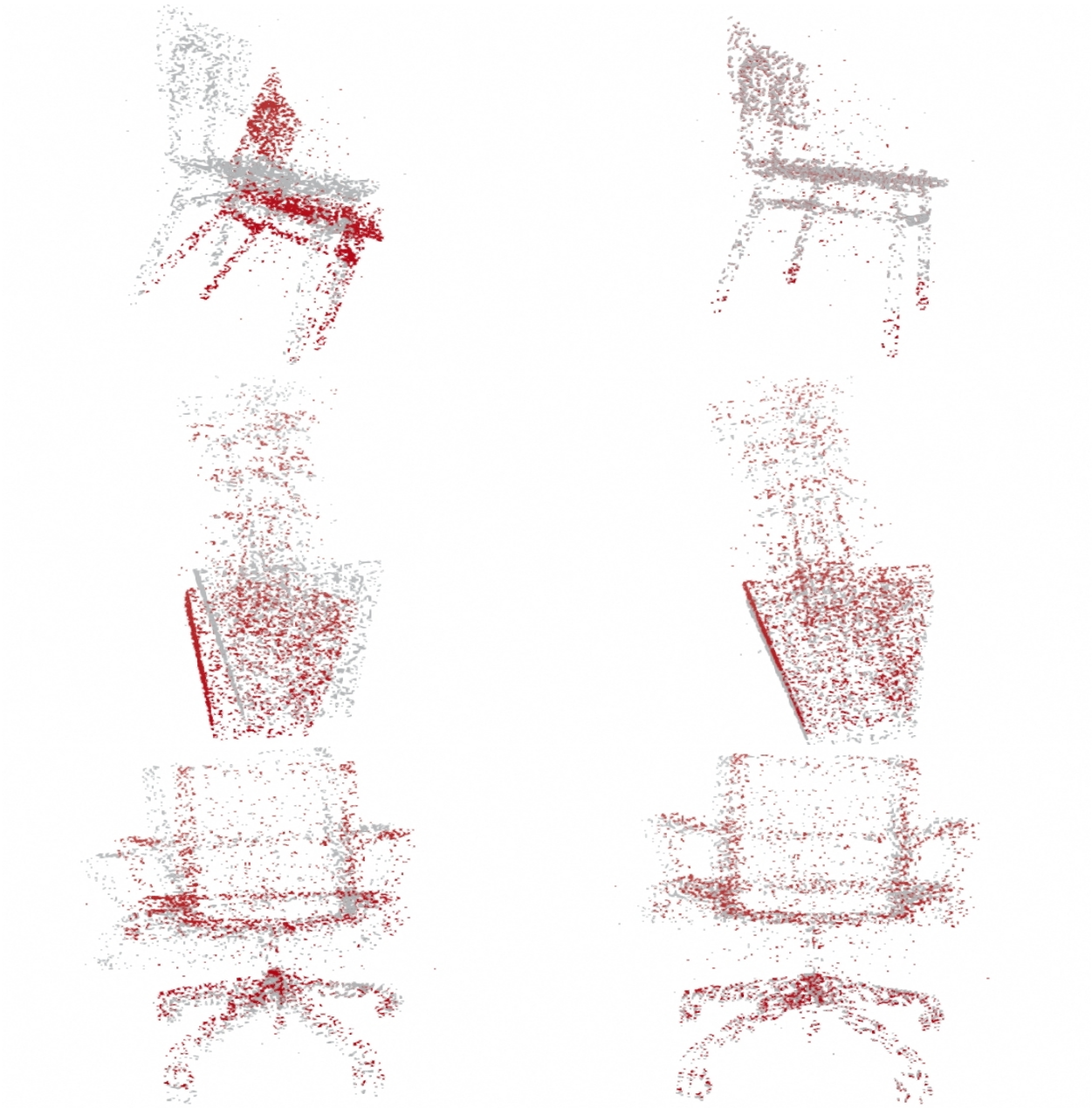

Figure 5.9: Three examples of point matching in the presence of noise and outliers.

Table 5.3: Comparison of Inference time in millisecond

no. points	ICP	FGR	PointNetLK	RPM-Net	DCP-v2	Ours
512	7.97	22.01	161.2	65.85	7.9	210.21
1024	16.68	84.30	176.34	143.4	8.3	276.30
2048	28.634	147.65	209.07	445.67	73.7	634.21

possible to test the trained model with different number of points. As expected, as the number of points increases so the inference time for estimation. Since the graph network tries to find global and long-range dependencies in the point clouds the process of inference takes relatively more time comparing with other methods, so this can be further improved in future works by incorporating other attention and relational mechanism that can capture global dependencies faster.

5.6 Conclusion

A deep graph-based framework has been proposed for the task of point registration between two point clouds. Since the model is based on correspondence between two frames in the latent space and not initialization of the frames, it makes the solution less sensitive and more robust with to initialization errors.

Using non–local mechanism in the model helps to encode the global features of point clouds that makes the correspondence less affected by noise. The structure of graph network in this work naturally supports the combinatorial generalization, since the computations are not strictly at the system level, but rather they apply shared weights and computations across the entities and relations as well (i.e. The training of the model is not driven purely by the input signals, but the relationship between different parts of the architecture plays an important role in learning the underlying task of registration in this example). This allows never–before–seen signals to be reasoned about using the graph network models.

By utilizing the proposed model, it is possible to combine and utilize other recently proposed mechanisms such as attention–based, interaction–based, and relational networks within this model. This helps to use the benefits of each of these modules under a unified setting of graph network which makes it easy to understand, accessible, and easy to implement by using only one neural network platform.

Part III

Control and Deep Learning

Chapter 6

In Proximity of ReLU Architecture and eMPC

6.1 Overview

In recent years, deep neural networks (DNN) has had tremendous success in computer vision, speech recognition, and other areas of machine learning [63, 42, 117, 95]. Despite all these unprecedented performances in learning tasks, a theoretical understanding of DNN's architecture, features, and properties is still unexplored. Also, all of these successes are related to the supervised learning and are concerned mostly with function fitting (e.g. classification, function approximation, and regression). In contrast, in reinforcement learning (RL), the concept of feedback makes it hard to study in theory since the statistical properties are dynamic/changing, and also they are hard to train in practice. Another shortcoming of DNN in RL is the absence of theoretical guarantees regarding stability, robustness, and convergence. All these issues need a great deal of consideration.

On the other hand, model predictive control is a powerful tool for control and decision making in robotics and other safety-concerned applications due to its adaptability, robustness, and stability-safety guarantees. In specific, *explicit* MPC allows us to pre-compute the optimal control policy $u_t^* = f(x(t))$ as a function of current state $x(t)$, and deploy it on-line in real-time. This prevents the issue of solving optimization problem in real-time on embedded systems which are typically limited with regard to memory capacity and computation power. But deployment of an explicit MPC suffers from increasing number of regions which grows exponentially (in the worst case) with the number of constraints [1, 9]. This demands significant amount of storage and computational complexity.

Several attempts have been made to address those shortcomings in *explicit* MPC [8, 38, 56, 55, 43, 119]. But in contrast, regarding deep reinforcement learning, all the attempts were mainly focused on empirical results, and analyzing its architecture only to appear in literature in very recent years. Here we focus more to mention some of these new findings regarding the DNN. Authors in [116] investigate the complexity of DNN by studying

the number of polytopic regions that they can attain. The paper also provides a tighter upper-bound (compared to previous bounds [80]) on the maximal number of regions that can be partitioned by a ReLU DNN. The paper [34] discusses the geometric properties of DNN for classification and how to improve the robustness of such DNN to perturbation by analyzing those properties. In [10] authors present a method that adds stability guarantee to the deep gradient descent algorithm.

Although these two specific areas of research (deep RL and MPC) have strong connection in (adaptive) optimal control theory [124], but from mathematical point of view there is another link between these two: Both ReLU DNN and solution to the mp-LP or mp-QP in *explicit* MPC represent a PWA function on polyhedra. This gives a great amount of motivation to investigate the possibility of reconstructing one from the other in order to benefit from advantages in both approaches.

Since the presumption concerning DNN that they have tens of thousands of parameters (weights and biases) seems reasonable for vision or language applications, but in fact a DNN can represent a very complex function with much less number of parameters. This is a compelling property when we are dealing with representing a control policy as a DNN. As an example, Fig. 6.1 shows a 2–layer ReLU network with just 84 parameters chosen randomly. The plot shows how a very small size network can subdivide the input-space to many polytopes and different affine policy pieces over each region.

In the following, we first provide mathematical definition of ReLU DNN and its structural properties in Section 6.2. Then in Section 6.3, we present a brief overview of existing theorems that represent connection between ReLU nets and PWA functions, and discuss challenges which prevent us to have an explicit association. Also we present a sample-based method in order to identify the underlying PWA function that a ReLU DNN can represent. Finally in Section 6.4, we provide a numerical example that examine a simple network and its equivalent PWA function.

6.2 Preliminaries and Problem Formulation

In this section we define feedforward ReLU DNN and discuss some properties of these models and their ability to map input-space to the complex family of PWA functions.

Notation and Definitions

Definition 1 *A rectifier (ReLU) feedforward network is a layered neural network with $L \in \mathbb{N}$ hidden layers (*depth *of the net) with input and output dimensions $n_0, n_{L+1} \in \mathbb{N}$ respectively. Each hidden layer l is composed of an affine transformation $f_l : \mathbb{R}^{n_l} \to \mathbb{R}^{n_{l+1}}$ followed by a rectifier activation function* $\mathsf{rect}(x) : x \mapsto \max(x, 0)$

$$
\begin{aligned}
f_l &= W_l h_{l-1} + b_l \\
h_l &= \mathsf{rect}(f_l) = \max\{f_l,\ 0\},
\end{aligned}
$$

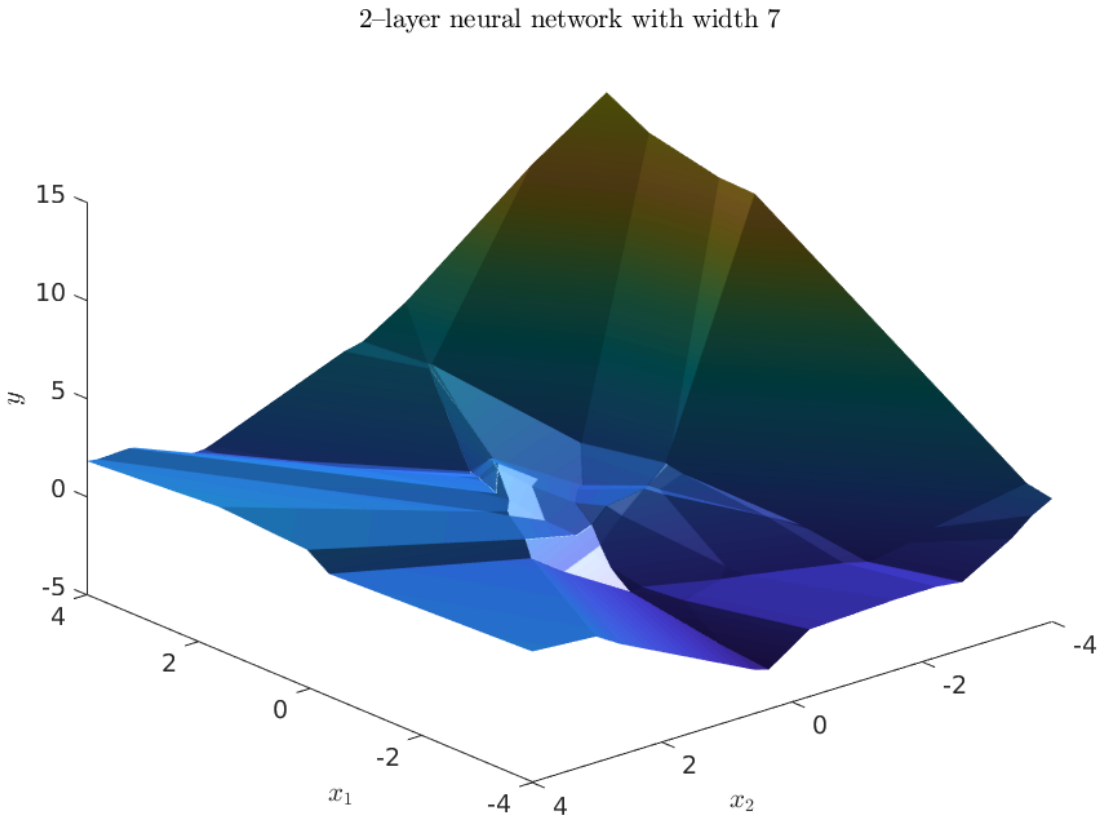

Figure 6.1: This example shows the level of complexity that a feedforward neural network (NN) can represent. The plot shows how just a 2-layer NN maps input-space $\boldsymbol{x} \in \mathbb{R}^2, (x_1,\ x_2)$ to the output-space $y \in \mathbb{R}$ with 7 ReLU activation units in each layer (total of 84 parameters). The network creates a complex continuous PWA function which can be a close approximation of a highly nonlinear function.

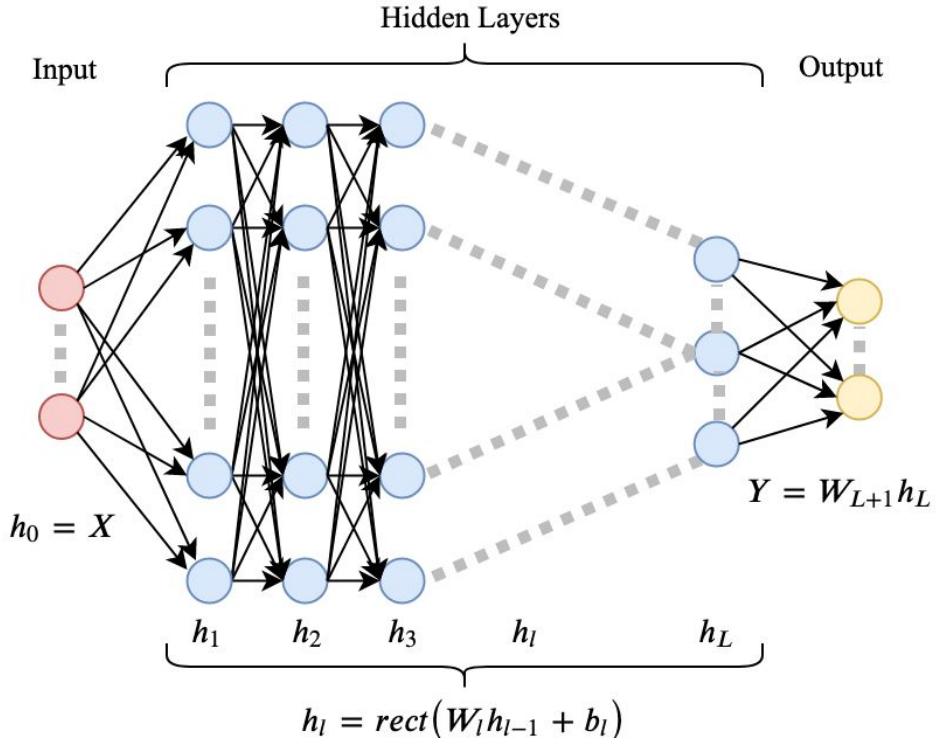

Figure 6.2: Illustration of L–layer ReLU DNN $f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_{L+1}}$. Depending on application and complexity of function to be approximated, DNN can have an arbitrary number of layers (i.e. depth) and activation units in each layer (i.e width). Note that DNNs are recognized just by the number of *hidden layers*. Also as seen, the output layer is just a linear transformation of last hidden layer without activation mapping.

where the max is an element-wise function, $W_l \in \mathbb{R}^{n_l \times n_{l-1}}$, $b_l \in \mathbb{R}^{n_l}$, $f_l \in \mathbb{R}^{n_l}$, $h_l \in \mathbb{R}^{n_l}$, *and* $h_0 \in \mathbb{R}^{n_0}$ *is defined as the input to the network. We call* f_l pre-activation *and* h_l post-activation *functions at hidden layer* l*. The output layer is just a linear transformation* $W^{(L+1)}$ *and does not count as part of the hidden layers. Finally, any ReLU net with* $L > 1$ *layers is called* L–layer DNN *and can be represented as a function* $f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_{L+1}}$

$$f = W_{L+1} \circ h_L \circ f_L \circ \ldots \circ h_2 \circ f_2 \circ h_1 \circ f_1$$

where $\circ$ *denotes function decomposition.*

Definition 2 *Every layer* $l \in \{1, 2, \ldots, L\}$ *of a ReLU DNN has* n_l activation units *which is called* width *of the layer. Each activation unit receives the rectified weighted sum of the previous*

post-activation values h_{l-1} plus a bias. The jth activation unit in layer l is denoted by $h_{l,j} \in \mathbb{R}$

$$f_{l,j} = W_{l,j}^{\top} h_{l-1} + b_{l,j}$$
$$h_{l,j} = \max\{f_{l,j}, 0\},$$

where $W_{l,j}^{\top}$ and $b_{l,j}$ are the jth row and the jth element of matrix W_l and vector b_l respectively.

An illustration of a L–layer ReLU DNN is shown in Fig. 6.2. Each blue circle in the figure represent an activation unit. Depending on the structure of DNN it can have any width size for each hidden layer. The total number of parameters for each DNN $\theta = \{W_{1:L+1}, b_{1:L}\}$ can be a basis to compare different architectures by varying depths and widths.

ReLU DNN Expressiveness

Despite the DNN's empirical successes, some fundamental questions about how and why these results are achieved is absent in literature. *Neural net expressivity* is a subject that tries to answer some of these questions such as how the depth, width, and the type of layers impact the function that the network represents, and also how these properties affect its performance. Here we try to provide some of these findings. First we present a set of theorems that deal with these types of questions.

First, since the post-activation $h(s) := \max\{s, 0\}$ is itself a PWA function and also the structure of ReLU networks is a series of composition of affine and post-activation functions, therefore the result is a PWA function that is defined over the regions of the input-space. This has been stated in the following theorem.

Theorem 1 *Given a neural network with ReLU activation, the input-space is partitioned into convex polytopes.*

Proof 2 *The complete proof can be found in [103]. But as sketch of proof, consider the first layer $l = 1$; each pre–activation function establishes a hyperplane on the input-space since $f_{1,j} = W_{1,j}x + b_{1,j} = 0$. All such hyperplanes associated to each unit provide a* hyperplane arrangement *which partitions the input-space into polytopes. By induction, it can be shown that this is true for all other layers in DNN. Fig. 6.3 illustrates the theorem for a 2-layer DNN.*

Another important property of ReLU networks is the number of polytopic regions that they can realize on their input-spaces. This helps on two fronts: 1) To understand the complexity of a specific architecture based on the lower- and upper-bound of the number of regions and 2) To design an architecture based on the number of regions that is necessary for an specific application. A *lower–bound* on the number of regions is described in the following theorem

Theorem 2 *The maximal number of regions computed by a ReLU neural network, with n_0 inputs, L hidden layers, and widths $n_l \geq n_0 \; \forall l \in \{1, 2, \ldots, L\}$, is lower-bounded by*

$$\left(\prod_{l=1}^{L-1} \lfloor \frac{n_l}{n_0} \rfloor^{n_0} \right) \sum_{j=0}^{n_0} \binom{n_L}{j}. \tag{6.1}$$

where $\lfloor \cdot \rfloor$ is the floor function on fractions.

Proof 3 *Proof can be found in [81] or [91].*

From the hyperplane arrangement it can be shown that the maximal number of regions for any ReLU networks with a total of N activation units is bounded from above by 2^N [81]. This bound is very loose, and not very useful. But there is also a tighter *upper-bound* on the number of regions,

Theorem 3 *The maximal number of regions of a ReLU neural network, with n_0 inputs, L hidden layers, and widths $n_l \geq n_0 \; \forall l \in \{1, 2, \ldots, L\}$, is upper-bounded by*

$$\sum_{(j_1, \ldots, j_L) \in J} \prod_{l=1}^{L} \binom{n_l}{j_l} \tag{6.2}$$

where $J = \{(j_1, \ldots, j_L) \in \mathbb{Z}^L : 0 \leq j_l \leq \min\{n_0, n_1 - j_1, \ldots, n_{l-1} - j_{l-1}, n_l\}, \; \forall l \in [L]\}$

Proof 4 *See Theorem 1. in [116].*

These theoretical backgrounds give us better understanding of how a structure of neural network impacts its performance and also helps us to use some of these properties in order to construct the link with explicit MPC in the following sections.

explicit MPC and PWA functions

Given a dynamical system, the purpose of a constrained optimal control is to solve an optimization problem with a set of constraints on states x_t and actions u_t in order to find a sequence of actions $u^*_{0:\infty}$ that controls the system to a desired/reference state. We can formulate such problem as an infinite-horizon optimization problem

$$\begin{aligned} J^*_\infty(x(0)) = \min_{u_0, u_1, \ldots} & \sum_{t=0}^{\infty} q(x_t, u_t) \\ \text{s.t.} \quad & x_{t+1} = Ax_t + Bu_t, \\ & x_t \in \mathcal{X}, \; u_t \in \mathcal{U}, \\ & x_0 = x(0), \\ & \forall t = 0, 1, \ldots \; . \end{aligned} \tag{6.3}$$

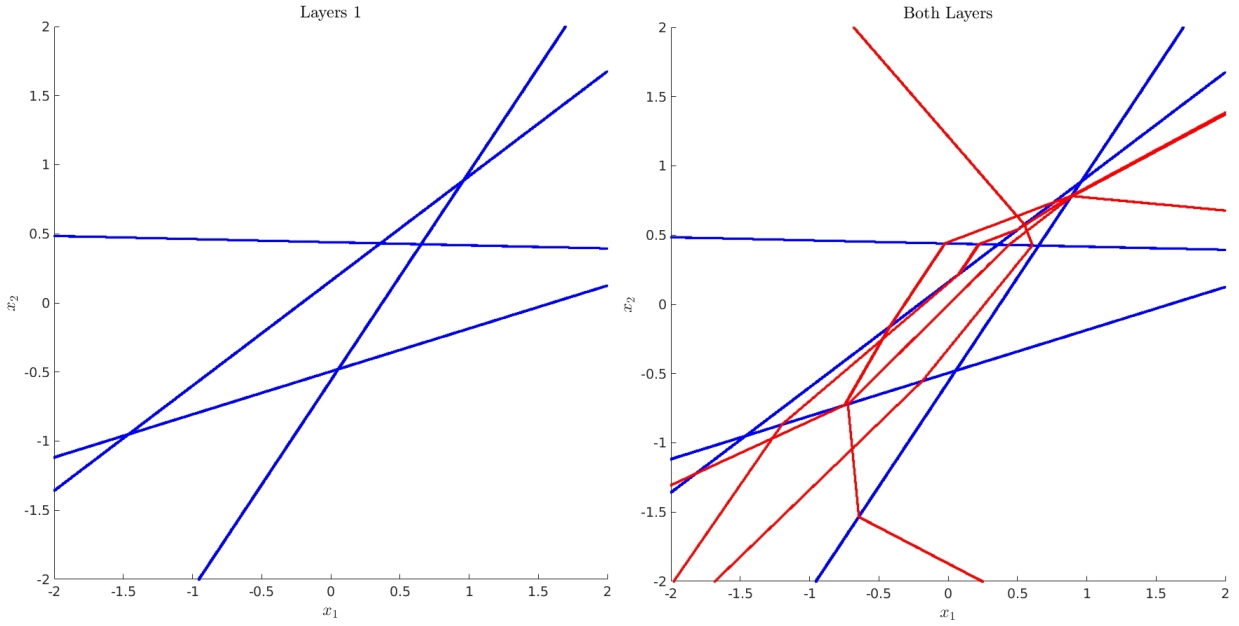

Figure 6.3: A ReLU DNN subdivides the input-space into polytopes. In fact, each hidden layer divides the input-space from the previous layer h_{l-1}, and this recursively subdivides the input–space of the whole network for the deeper layers forward. Here we have a 2–layer ReLU net with input $x \in \mathbb{R}^2$ and four activation units in each layer. The left plot shows the pre-activation functions $f_1 = W_1 x + b_1$ that is equivalent to four hyperplanes in $\mathbb{R}^2$. Hidden units are activated in one side of their corresponding hyperplanes. The right plot shows both hyperplanes from the first and second layers in blue and red, respectively. The hyperplanes in the second layer, as seen in the plot, are not straight lines, but rather they are bent at the first layer boundaries (blue lines). When those hyperplanes pass through different regions partitioned by the first layer, they will be bent. Therefore we still have four activation boundaries for four units in layer 2, but they are not straight lines. In the right plot we can see all the regions that the network can partition on the input-space. Also it represents different affine functions over each polytope.

This problem (6.3) cannot be solved easily due to its infinite horizon nature with constraints on states and actions [15]; instead model predictive control (i.e. *receding horizon control*) is a suitable approach to follow, which mimics (6.3) by appropriate choice of $p(x_N)$, $q(x_k, u_k)$, and $\mathcal{X}_f$ as the following,

$$\begin{aligned} J_0^*(x(t)) = \min_{u_{0:(N-1)}} \; & p(x_N) + \sum_{k=0}^{N-1} q(x_k, u_k) \\ \text{s.t} \quad & x_{k+1} = Ax_k + Bu_k, \\ & x_k \in \mathcal{X}, \; u_k \in \mathcal{U}, \\ & x_N \in \mathcal{X}_f, \\ & x_0 = x(t), \\ & \forall k = 0, \dots, N-1. \end{aligned} \tag{6.4}$$

Equation (6.4) can be seen as a multiparametric program (mp) in which $x(t)$ is the vector of parameters. In particular for the case of linear and quadratic cost functions with polyhedral constraints, it transpires that the solution to problem (6.4) is in fact a PWA function of the parameters $u^*(t) = f(x(t))$, an *explicit* solution to the MPC controller.

In a number of instances we may be interested in the constructing the PWA function corresponding to a ReLU net which is also the solution of a mp–LP/mp–QP problem. This may arise when, for example, we want to measure the suboptimality of a trained network with the solution of an explicit MPC. *Inverse mp–LP/QP* studies this idea, constructing such optimization problems from PWA functions. The following theorem expresses this in detail,

Theorem 4 *Every continuous piecewise affine function $f : \mathbb{R}^m \to \mathbb{R}^n$ can be obtained as a linear map of the unique explicit solution $\hat{f}(x)$ of multi-parametric linear program in the form of*

$$\begin{aligned} \hat{f}(x) \; \in & \; arg\, min_z \;\; J(z, x) \\ \text{s.t.} \quad & (z, x) \in \Omega, \end{aligned} \tag{6.5}$$

with dimension $\hat{n}$, when $\hat{n} \leq 2n$.

The proof presented in [45] is constructive, that means the proof establishes a procedure that results to the formulation of a mp-LP from a PWA function. The proof follows from the fact that every PWA function can be decomposed to two convex function and from there it is straightforward to construct a mp-LP for a convex PWA function. Note that, although the proof is constructive, it is still very hard (or even impossible) to implement it as an algorithm.

Now, referring to Fig. 6.4 we can have a better understanding of the whole picture in relating the ReLU feedforward to the PWA functions, mp-LP/mp-QP, and ultimately explicit MPC. Although it is possible to use learning to find an approximation of an explicit MPC policy, yet constructing a deep network from a PWA function needs to be studied further in the future works.

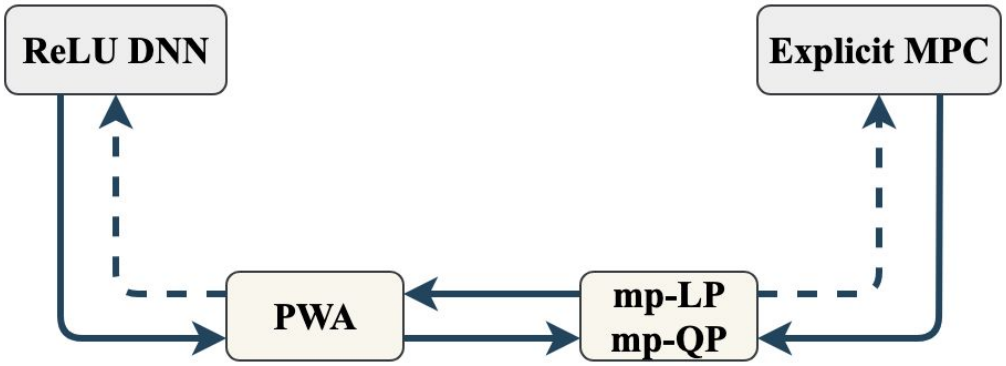

Figure 6.4: This illustration gives an entire perspective that this chapter tries to depict. ReLU DNNs represent PWA function on polyhedra which subdivide the input-space. Assuming that the input to the neural network is the parameter $x(t)$, the network can exactly act as an explicit state feedback policy. The dashed arrows indicate the needs for further study of methods -analytical or approximate- which can reconstruct the mathematical structures of each block from the other in a constructive manner.

6.3 Explicit MPC and ReLU DNN

In this section we connect the ReLU DNN and explicit MPC through their underlying connection which is PWA functions. As mentioned in previous sections we know that every ReLU DNN has a continuous PWA function representation on the input-space, and vice versa (but not necessary in an explicit closed form, since constructing such a connection is not easy in general).

Identification of Input-Space in ReLU DNN

In order to identify the different regions partitioned by ReLU NN on the input-space, we present an approximate method here that is an extension of the method introduced in [81]. We will show that it is possible to construct each pieces of a PWA function by extending the PWA representation of a shallow network (i.e. $L = 1$).

Since every dimension of the output-space can be treated independently, here we assume the construction of a scalar-valued function $f : \Omega \subset \mathbb{R}^{n_0} \to \mathbb{R}$ from a DNN model (i.e the output-space is scalar $n_{L+1} = 1$), but as mentioned, the proposed method can be applied separately for each dimension in the case of vector-valued DNN models. Any scalar-valued affine function which is defined over its convex region Ω_i can be written as

$$f_i(x) = u^\top x + c, \quad x \in \Omega_i, \tag{6.6}$$

where $u^\top \in \mathbb{R}^{n_0}$ and $c \in \mathbb{R}$. In order to construct $u^\top$ and c in (6.6), we first consider a NN with one layer and then extend it to the deep nets.

Shallow Network

Note that we can reformulate a scalar rectifier function as

$$\text{rect}(h) = \mathbf{I}(h) \cdot h, \tag{6.7}$$

where $\mathbf{I}(h)$ is an indicator function defined as follows

$$\mathbf{I}(h) = \begin{cases} 1 & h > 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.8}$$

Now considering a single layer NN $f : \mathbb{R}^{n_0} \to \mathbb{R}$, we rewrite it with the help of indicator function as

$$f(x) = W_2 \text{diag}\left(\begin{bmatrix} \mathbf{I}(W_{1,1}x + b_{1,1}) \\ \mathbf{I}(W_{1,2}x + b_{1,2}) \\ \vdots \\ \mathbf{I}(W_{1,n_1}x + b_{1,n_1}) \end{bmatrix} \right) (W_1 x + b_1). \tag{6.9}$$

Simplifying (6.9), $f(x)$ can be written more compactly as

$$f(x) = W_2 \text{diag}(\mathbf{I}_{f_1}(x)) W_1 x + W_2 \text{diag}(\mathbf{I}_{f_1}(x)) b_1, \tag{6.10}$$

where $\text{diag}(\mathbf{I}_{f_l}(x))$ is the compact form of indicator function for pre-activation f_l in layer l. From (6.10) we can see that given input x weight $u^\top$ and bias c can be computed.

Deep Network

Now we can extend the derivation in (6.10) for deep network. Given an input x from a region Ω_i we can construct the corresponding weight $u^\top$ and bias c for each affine map f_i. The weight is computed by

$$\begin{aligned} u^\top = W_{L+1} \text{diag}(\mathbf{I}_{f_L}(x)) W_L \ \ldots \\ \text{diag}(\mathbf{I}_{f_2}(x)) W_2 \ \text{diag}(\mathbf{I}_{f_1}(x)) W_1, \end{aligned} \tag{6.11}$$

A bias of the affine map c also can be computed similarly

$$\begin{aligned} c = & W_{L+1} \text{diag}(\mathbf{I}_{f_L}) W_L \ \ldots \ \text{diag}(\mathbf{I}_{f_2}) W_2 \ \text{diag}(\mathbf{I}_{f_1}) b_1 \\ & + W_{L+1} \text{diag}(\mathbf{I}_{f_L}) W_L \ \ldots \text{diag}(\mathbf{I}_{f_2}) b_2 \\ & + \vdots \\ & + W_{L+1} \text{diag}(\mathbf{I}_{f_L}) b_L \end{aligned} \tag{6.12}$$

Both equations (6.11) and (6.12) depend on input x, so we need to use a (large enough) set of samples from the input-space to be able to identify different affine responses of the

output. It is worth mentioning that from (6.11) and (6.12) it is also possible to derive the corresponding affine function for each activation unit up to a specific hidden layer instead of the whole network. This means that any activation unit in any stage of a deep neural network can be written as a PWA function over the input-space of the network. This needs further study, but as a preliminary, we can ask "is there any connection between layers of a neural network and for example the horizon in model predictive control?"

Learning DNN with Exact Architecture

Several literature study the concept of learning approximate MPC through supervised or reinforcement learning process [22, 47]. But we can utilize the structure of PWA control policy to further improve the process of learning [58]. The following theorem is the key concept in the process.

Theorem 5 *Any convex PWA function $f : \mathbb{R}^{n_0} \to \mathbb{R}$, which also can be formulated as pointwise maximum of N affine functions $f(x) := \max_{i=1,\dots,N} f_i(x)$, can be exactly presented by a ReLU DNN with width $n_l = n_0 + 1,\ \forall l \in [L]$ and depth N.*

Proof 5 *See Theorem 2 in [44].*

Depending on the dimension of control input $u \in \mathbb{R}^m$, it may be needed to train up to $2m$ networks. In fact, every element in the control input vector can be treated separately. Each element $u^*(x(t)) : \mathbb{R}^{n_0} \to \mathbb{R}$ is a PWA function which needs to be decomposed into the difference of two convex PWA functions. Finally, theorem 5 gives an exact design structure for each network. And presumably, this should result a better learning (smaller loss value, faster convergence, better accuracy), which ultimately impacts the performance of the controller that the trained network substitutes.

6.4 Experimental validation

In this section we present a simple example (as a proof of concept) to illustrate the way to construct a mp-LP from a ReLU DNN. We examine a 2-layer feedforward net with $n_1 = n_2 = 2$ (i.e. total of four activation units) and one dimensional input/output $x,\ y \in \mathbb{R}$. The two hidden layers are constructed as follows

$$\begin{aligned} h_1 = \begin{bmatrix} h_{1,1} \\ h_{1,2} \end{bmatrix} &= \max\{0, W_1 x + b_1\} \\ &= \max\left\{0, \begin{bmatrix} -3/2 \\ 2 \end{bmatrix} x + \begin{bmatrix} 3 \\ -2 \end{bmatrix}\right\} \end{aligned}$$

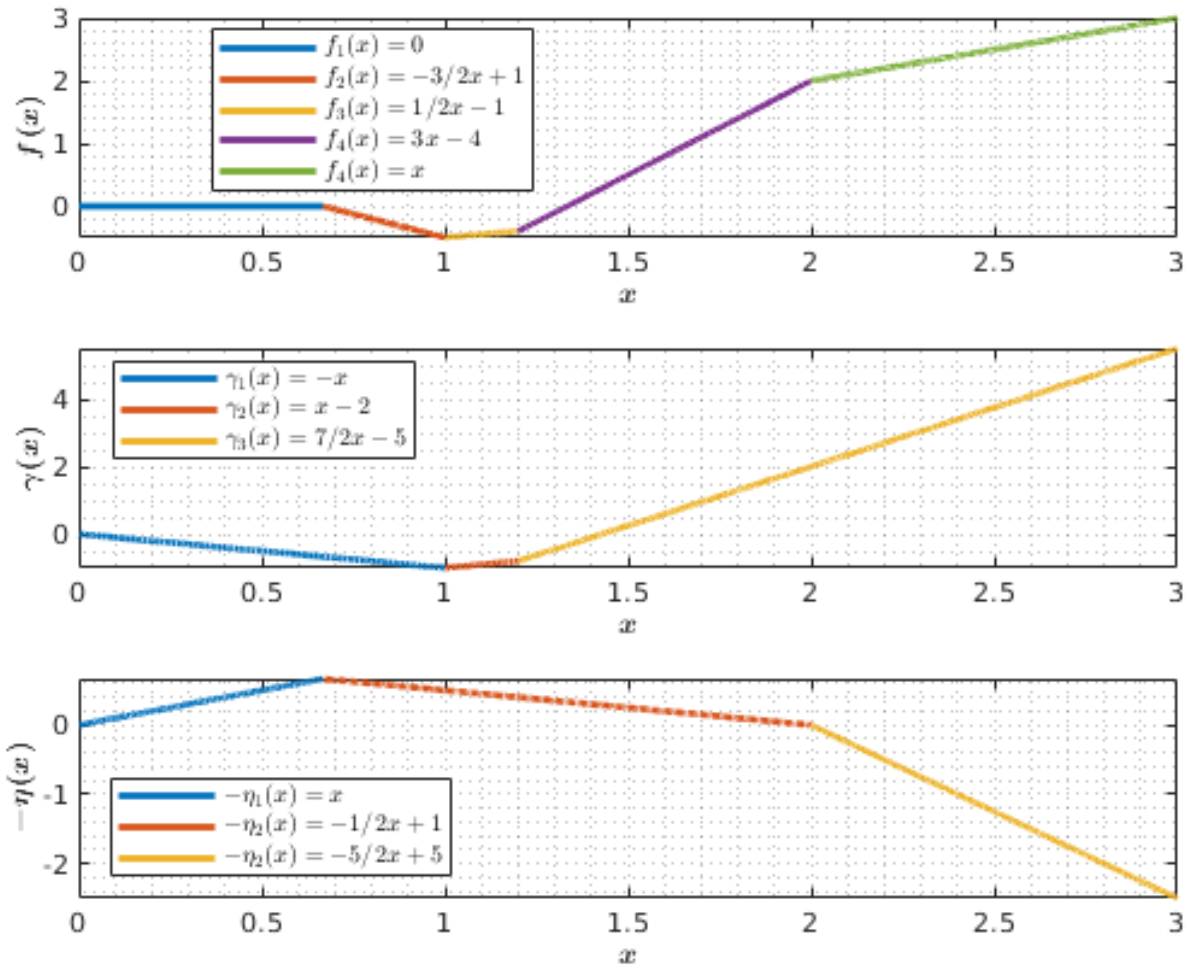

Figure 6.5: The plots show a scalar PWA function and its decomposition to two convex functions, $f(x) = \gamma(x) - \eta(x)$.

$$h_2 = \begin{bmatrix} h_{2,1} \\ h_{2,2} \end{bmatrix} = \max\{0, W_2 h_1 + b_2\}$$
$$= \max\left\{0, \begin{bmatrix} -1 & -1 \\ 1/2 & -1 \end{bmatrix} h_1 + \begin{bmatrix} 1 \\ 2 \end{bmatrix}\right\},$$

and the linear map for the output layer is

$$y = W_3 h_2 = \begin{bmatrix} 1 & -1 \end{bmatrix} h_2.$$

The PWA function equivalent to the above feedforward network is

$$y = f(x) = \begin{cases} 0 & x \leq \frac{2}{3} \\ -\frac{3}{2}x + 1 & \frac{2}{3} \leq x \leq 1 \\ \frac{1}{2}x - 1 & 1 \leq x \leq \frac{6}{5} \\ -3x - 4 & \frac{6}{5} \leq x \leq 2 \\ x & 2 \leq x \end{cases} \tag{6.13}$$

since the PWA function (6.13) is not convex nor concave, we can decompose it into the difference of two convex functions $\gamma(x)$ and $\eta(x)$ as follows

$$\gamma(x) = \begin{cases} -x & x \leq 1 \\ x - 2 & 1 \leq x \leq \frac{6}{5} \\ \frac{7}{2}x - 5 & \frac{6}{5} \leq x \end{cases}$$
$$\eta(x) = \begin{cases} -x & x \leq \frac{2}{3} \\ \frac{1}{2}x - 1 & \frac{2}{3} \leq x \leq 2 \\ \frac{5}{2}x - 5 & 2 \leq x \end{cases}$$

Then we can construct the mp-LP counterpart that its solution is the same as PWA function (6.13). Introducing decision variable $z \in \mathbb{R}^2$, we can write

$$\begin{aligned} J^*(x) = \min_{z \in \mathbb{R}^2} \quad & \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \\ \text{s.t.} \quad & -x \leq z_1 \quad x \geq z_2 \\ & x - 2 \leq z_1 \quad -\frac{1}{2}x + 1 \geq z_2 \\ & \frac{7}{2}x - 5 \leq z_1 \quad -\frac{5}{2}x + 5 \geq z_2, \\ & x \in [0, 3] \end{aligned} \tag{6.14}$$

and then we can construct $f(x)$ with linear map T as

$$f(x) = T\hat{f}(x) = \begin{bmatrix} 1 & 1 \end{bmatrix} \hat{f}(x)$$

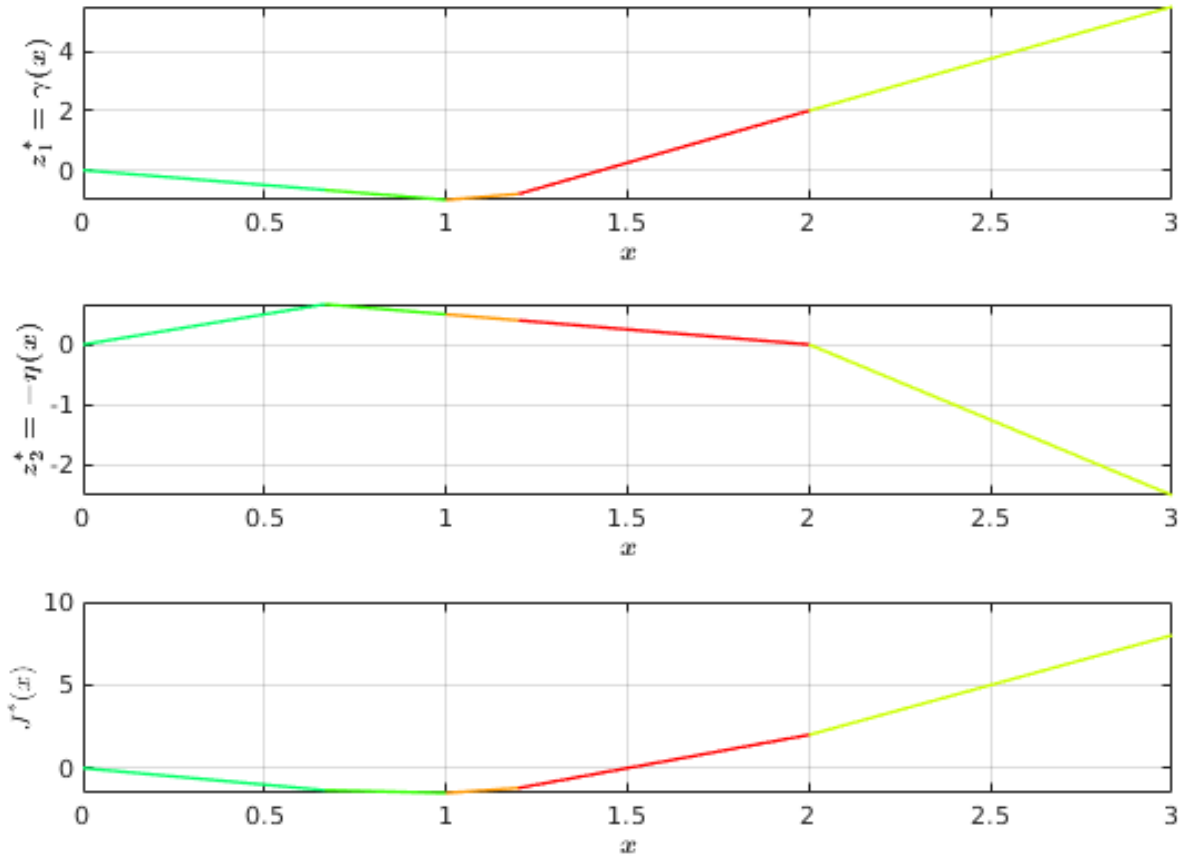

Figure 6.6: The plots show the explicit solution to the mp-LP (6.14) using MPT3 toolbox [46]. **Top & center:** show plots of the optimizers $z_1^* = \gamma(x)$ and $z_2^* = -\eta(x)$ which both are functions of the parameter x. the solution exactly results the PWA function (6.13). **bottom:** The plot depicts the optimal value $J^*(x)$ which is a function of parameter x and also is convex and PWA as we know from the theory of mp-LP (corollary 11.5 in [15]).

In fact the explicit solutions to the mp-LP are

$$\begin{aligned} z_1^* &= \gamma(x), \\ z_2^* &= -\eta(x), \\ \hat{f}(x) &= \begin{bmatrix} \gamma(x) & -\eta(x) \end{bmatrix}^T, \end{aligned}$$

which exactly follows the constructive proof in [45].

6.5 Conclusion

We presented an overview of ReLU deep neural networks; and discussed several structural properties of such models which are key concepts of using a ReLU network as an explicit state feedback policy for a model predictive controller. Specifically, we argued since any ReLU network models a PWA function on polyhedra, it would be a perfect choice to use a ReLU network instead of state feedback policy computed by an explicit MPC procedure considering storage and execution complexity of such controllers in real-time. We also presented a sample–based method that identifies different affine pieces of a ReLU networks. For future work, alongside further development of some initial findings in this dissertation, other very recently new ideas such as representing ReLU DNN as a mixed-integer linear problem [116] can be the subject of further investigation.

Bibliography

[1] Alessandro Alessio and Alberto Bemporad. "A Survey on Explicit Model Predictive Control". In: *Nonlinear Model Predictive Control: Towards New Challenging Applications*. Ed. by Lalo Magni, Davide Martino Raimondo, and Frank Allgöwer. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 345–369. ISBN: 978-3-642-01094-1.

[2] Yasuhiro Aoki et al. "PointNetLK: Robust & Efficient Point Cloud Registration Using PointNet". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[3] Erik Ask, Olof Enqvist, and Fredrik Kahl. "Optimal Geometric Fitting under the Truncated L2-Norm". In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 1722–1729. DOI: 10.1109/CVPR.2013.225.

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. 2016. arXiv: 1607.06450 [stat.ML].

[5] Arindam Banerjee et al. "Clustering on the Unit Hypersphere Using Von Mises-Fisher Distributions". In: *J. Mach. Learn. Res.* 6 (Dec. 2005), pp. 1345–1382. ISSN: 1532-4435.

[6] Martin Barczyk and Alan F Lynch. "Integration of a triaxial magnetometer into a helicopter UAV GPS-aided INS". In: *IEEE transactions on Aerospace and Electronic Systems* 48.4 (2012), pp. 2947–2960.

[7] Peter Battaglia et al. "Relational inductive biases, deep learning, and graph networks". In: *arXiv* (2018). URL: https://arxiv.org/pdf/1806.01261.pdf.

[8] A. Bemporad and C. Filippi. "Suboptimal Explicit Receding Horizon Control via Approximate Multiparametric Quadratic Programming". In: *Journal of Optimization Theory and Applications* 117.1 (Apr. 2003), pp. 9–38. ISSN: 1573-2878.

[9] Alberto Bemporad. "Model Predictive Control Design: New Trends and Tools". In: Jan. 2007, pp. 6678–6683.

[10] Felix Berkenkamp et al. "Safe Model-based Reinforcement Learning with Stability Guarantees". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 908–918.

[11] Paul J. Besl and Neil D. McKay. "A Method for Registration of 3-D Shapes". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 14.2 (Feb. 1992), pp. 239–256. ISSN: 0162-8828.

[12] Seth Billings and Russell Taylor. "Generalized iterative most likely oriented-point (G-IMLOP) registration". In: *International Journal of Computer Assisted Radiology and Surgery* 10.8 (Aug. 2015), pp. 1213–1226. ISSN: 1861-6429.

[13] Chris M. Bishop. "Training with Noise is Equivalent to Tikhonov Regularization". In: *Neural Computation* 7.1 (Jan. 1995), pp. 108–116. ISSN: 0899-7667. DOI: 10.1162/neco.1995.7.1.108. eprint: https://direct.mit.edu/neco/article-pdf/7/1/108/812990/neco.1995.7.1.108.pdf. URL: https://doi.org/10.1162/neco.1995.7.1.108.

[14] Elizabeth R. Boroson and Nora Ayanian. "3D Keypoint Repeatability for Heterogeneous Multi-Robot SLAM". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 6337–6343. DOI: 10.1109/ICRA.2019.8793609.

[15] F. Borrelli, A. Bemporad, and M. Morari. *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017. ISBN: 9781107652873.

[16] N. Boumal et al. "Manopt, a Matlab Toolbox for Optimization on Manifolds". In: *Journal of Machine Learning Research* 15 (2014), pp. 1455–1459.

[17] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004, p. 387. ISBN: 0521833787.

[18] Thomas M. Breuel. "Implementation techniques for geometric branch-and-bound matching methods". In: *Comput. Vis. Image Underst.* 90 (2003), pp. 258–294.

[19] D. Briese, H. Kunze, and G. Rose. "UWB localization using adaptive covariance Kalman Filter based on sensor fusion". In: *2017 IEEE 17th International Conference on Ubiquitous Wireless Broadband (ICUWB)*. Sept. 2017, pp. 1–7. DOI: 10.1109/ICUWB.2017.8250968.

[20] C. Cadena et al. "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age". In: *IEEE Transactions on Robotics* 32.6 (Dec. 2016), pp. 1309–1332. ISSN: 1552-3098. DOI: 10.1109/TRO.2016.2624754.

[21] Michael Chang et al. "Automatically Composing Representation Transformations as a Means for Generalization". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=B1ffQnRcKX.

[22] Steven Chen et al. "Approximating Explicit Model Predictive Control using Constrained Neural Networks". In: *American Control Conference (ACC)*. 2018.

[23] Am Cho et al. "Wind estimation and airspeed calibration using a UAV with a single-antenna GPS receiver and pitot tube". In: *IEEE transactions on aerospace and electronic systems* 47.1 (2011), pp. 109–117.

[24] Siddharth Choudhary et al. “Information-based reduced landmark SLAM”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 4620–4627. DOI: 10.1109/ICRA.2015.7139839.

[25] Christopher Choy, Wei Dong, and Vladlen Koltun. “Deep Global Registration”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 2511–2520. DOI: 10.1109/CVPR42600.2020.00259.

[26] H. Chui and A. Rangarajan. “A feature registration framework using mixture models”. In: *Proceedings IEEE Workshop on Mathematical Methods in Biomedical Image Analysis. MMBIA-2000 (Cat. No.PR00737)*. 2000, pp. 190–197. DOI: 10.1109/MMBIA.2000.852377.

[27] Jose A. Costa, Neal Patwari, and Alfred O. Hero III. “Distributed Weighted multi-dimensional Scaling for Node Localization in Sensor Networks”. In: *ACM Trans. Sen. Netw.* 2.1 (Feb. 2006), pp. 39–64. ISSN: 1550-4859. DOI: 10.1145/1138127.1138129.

[28] Matthew N. Dailey and Manukid Parnichkun. “Landmark-based Simultaneous Localization and Mapping with Stereo Vision”. In: *In Proc. Asian Conference on Industrial Automation and Robotics*. 2005, pp. 108–113.

[29] T. Deissler and J. Thielecke. “Feature based indoor mapping using a bat-type UWB radar”. In: *2009 IEEE International Conference on Ultra-Wideband*. Sept. 2009, pp. 475–479. DOI: 10.1109/ICUWB.2009.5288802.

[30] T. Deißler and J. Thielecke. “UWB SLAM with Rao-Blackwellized Monte Carlo data association”. In: *2010 International Conference on Indoor Positioning and Indoor Navigation*. Sept. 2010, pp. 1–5. DOI: 10.1109/IPIN.2010.5647596.

[31] J. Djugash et al. “Range-only SLAM for robots operating cooperatively with sensor networks”. In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* 2006, pp. 2078–2084. DOI: 10.1109/ROBOT.2006.1642011.

[32] Shaoyi Du et al. “Robust rigid registration algorithm based on pointwise correspondence and correntropy”. In: *Pattern Recognition Letters* (2018). ISSN: 0167-8655.

[33] Georgios D. Evangelidis and Radu Horaud. “Joint Alignment of Multiple Point Sets with Batch and Incremental Expectation-Maximization”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 40.6 (2018), pp. 1397–1410.

[34] Alhussein Fawzi et al. “Classification regions of deep neural networks”. In: *CoRR* abs/1705.09552 (2017). arXiv: 1705.09552.

[35] Martin A. Fischler and Robert C. Bolles. “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”. In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: https://doi.org/10.1145/358669.358692.

[36] Christian Forster et al. “Collaborative monocular slam with multiple micro aerial vehicles”. In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE. 2013, pp. 3962–3970.

[37] Andreas Geiger et al. “Vision meets Robotics: The KITTI Dataset”. In: *International Journal of Robotics Research (IJRR)* (2013).

[38] Tobias Geyer, Fabio D. Torrisi, and Manfred Morari. “Optimal Complexity Reduction of Polyhedral Piecewise Affine Systems”. In: *Automatica* 44.7 (July 2008), pp. 1728–1740. ISSN: 0005-1098.

[39] S. Gezici et al. “Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks”. In: *IEEE Signal Processing Magazine* 22.4 (July 2005), pp. 70–84. ISSN: 1053-5888. DOI: 10.1109/MSP.2005.1458289.

[40] Steven Gold et al. “New Algorithms for 2D and 3D Point Matching: Pose Estimation and Correspondence”. In: *Advances in Neural Information Processing Systems 7, [NIPS Conference, Denver, Colorado, USA, 1994]*. Ed. by Gerald Tesauro, David S. Touretzky, and Todd K. Leen. MIT Press, 1994, pp. 957–964. URL: http://papers.nips.cc/paper/977-new-algorithms-for-2d-and-3d-point-matching-pose-estimation-and-correspondence.

[41] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[42] Ian Goodfellow et al. “Maxout Networks”. In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, June 2013, pp. 1319–1327.

[43] P. Grieder and M. Morari. “Complexity reduction of receding horizon control”. In: *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*. Vol. 3. Dec. 2003, 3179–3190 Vol.3.

[44] B. Hanin. “Universal Function Approximation by Deep Neural Nets with Bounded Width and ReLU Activations”. In: *ArXiv e-prints* (Aug. 2017). arXiv: 1708.02691 [stat.ML].

[45] Andreas Hempel, Paul J. Goulart, and John Lygeros. “Every Continuous Piecewise Affine Function Can Be Obtained by Solving a Parametric Linear Program”. en. In: *2013 European Control Conference (ECC)*. European Control Conference (ECC 2013); Conference Location: Zürich, Switzerland; Conference Date: July 17-19, 2013; . IEEE, 2013, pp. 2657–2662. ISBN: 978-3-033-03962-9.

[46] M. Herceg et al. “Multi-Parametric Toolbox 3.0”. In: *Proc. of the European Control Conference*. http://control.ee.ethz.ch/~mpt. Zürich, Switzerland, July 2013, pp. 502–510.

[47] Michael Hertneck et al. “Learning an Approximate Model Predictive Controller with Guarantees”. In: *CoRR* abs/1806.04167 (2018). arXiv: 1806.04167.

[48] R. Horaud et al. "Rigid and Articulated Point Registration with Expectation Conditional Maximization". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.3 (Mar. 2011), pp. 587–602.

[49] Kurt Hornik et al. "Spherical k-Means Clustering". In: *Journal of Statistical Software* 50 (Sept. 2012), pp. 1–22.

[50] J.P. How et al. "Real-time indoor autonomous vehicle test environment". In: *IEEE Control Systems Magazine* 28.2 (Apr. 2008), pp. 51–64. ISSN: 0272-1708.

[51] Yang Hui et al. "An unmanned air vehicle (UAV) GPS location and navigation system". In: *Microwave and Millimeter Wave Technology Proceedings, 1998. ICMMT'98. 1998 International Conference on*. IEEE. 1998, pp. 472–475.

[52] J Jang and C Tomlin. "Longitudinal stability augmentation system design for the DragonFly UAV using a single GPS receiver". In: *AIAA Guidance, Navigation, and Control Conference and Exhibit*. 2003, p. 5592.

[53] Mohammed Javed, Md Meraz, and Pavan Chakraborty. "A Quick Review on Recent Trends in 3D Point Cloud Data Compression Techniques and the Challenges of Direct Processing in 3D Compressed Domain". In: *CoRR* abs/2007.05038 (2020). arXiv: 2007.05038. URL: https://arxiv.org/abs/2007.05038.

[54] Bing Jian and Baba Vemuri. "Robust Point Set Registration Using Gaussian Mixture Models". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 33 (Sept. 2011), pp. 1633–1645.

[55] T. A. Johansen and A. Grancharova. "Approximate explicit constrained linear model predictive control via orthogonal search tree". In: *IEEE Transactions on Automatic Control* 48.5 (May 2003), pp. 810–815. ISSN: 0018-9286.

[56] Tor Arne Johansen. "Approximate explicit receding horizon control of constrained nonlinear systems". In: *Automatica* 40 (2004), pp. 293–300.

[57] G. Kantor and S. Singh. "Preliminary results in range-only localization and mapping". In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*. Vol. 2. 2002, 1818–1823 vol.2. DOI: 10.1109/ROBOT.2002.1014805.

[58] Benjamin Karg and Sergio Lucia. "Efficient representation and approximation of model predictive control laws via deep learning". In: *arXiv preprint arXiv:1806.10644* (2018).

[59] John T. Kent. "The Fisher-Bingham Distribution on the Sphere". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 44.1 (1982), pp. 71–80. ISSN: 00359246.

[60] Jae-Hak Kim, Hongdong Li, and Richard Hartley. "Motion estimation for multi-camera systems using global optimization". In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. 2008, pp. 1–8. DOI: 10.1109/CVPR.2008.4587680.

[61] Klaas Klasing et al. "Comparison of surface normal estimation methods for range sensing applications". In: *Proceedings - IEEE International Conference on Robotics and Automation* (May 2009), pp. 3206–3211.

[62] Sivanand Krishnan et al. "A UWB based localization system for indoor robot navigation". In: *Ultra-Wideband, 2007. ICUWB 2007. IEEE International Conference on*. IEEE. 2007, pp. 77–82.

[63] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105.

[64] M. J. Kuhn et al. "Adaptive leading-edge detection in UWB indoor localization". In: *2010 IEEE Radio and Wireless Symposium (RWS)*. Jan. 2010, pp. 268–271. DOI: 10.1109/RWS.2010.5434259.

[65] A. Ledergerber and R. D'Andrea. "Ultra-Wideband Range Measurement Model with Gaussian Processes". In: *Proceedings of the IEEE Conference on Control Technology and Applications*. IEEE. 2017.

[66] R. Leishman et al. "Quadrotors and Accelerometers: State Estimation with an Improved Dynamic Model". In: *IEEE Control Systems Magazine* 34.1 (2014), pp. 28–41. ISSN: 1066-033X. DOI: 10.1109/MCS.2013.2287362.

[67] Christophe Ley and Christophe Ley. *Applied Directional Statistics: Modern Methods and Case Studies*. CRC Press, 2018.

[68] Hongdong Li. "Consensus set maximization with guaranteed global optimality for robust geometry estimation". In: *2009 IEEE 12th International Conference on Computer Vision*. 2009, pp. 1074–1080. DOI: 10.1109/ICCV.2009.5459398.

[69] Weiping Liu et al. "Deep Learning on Point Clouds and Its Application: A Survey". In: *Sensors* 19.19 (2019). ISSN: 1424-8220. DOI: 10.3390/s19194188. URL: https://www.mdpi.com/1424-8220/19/19/4188.

[70] Kok-Lim Low. *Linear Least-Squares Optimization for Point-to-Plane ICP Surface Registration*. Jan. 2004.

[71] Weixin Lu et al. "DeepVCP: An End-to-End Deep Neural Network for Point Cloud Registration". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 12–21. DOI: 10.1109/ICCV.2019.00010.

[72] Bin Luo and Edwin R. Hancock. "Structural Graph Matching Using the EM Algorithm and Singular Value Decomposition". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 23.10 (Oct. 2001), pp. 1120–1136. ISSN: 0162-8828.

[73] Sergei Lupashin et al. "A platform for aerial robotics research and demonstration: The Flying Machine Arena". In: *Mechatronics* 24.1 (2014), pp. 41–54. ISSN: 0957-4158.

[74] M.R. Mahfouz et al. "Investigation of High-Accuracy Indoor 3-D Positioning Using UWB Technology". In: *IEEE Transactions on Microwave Theory and Techniques* 56.6 (2008), pp. 1316–1330. ISSN: 0018-9480.

[75] K. V. Mardia. "Statistics of Directional Data". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 37.3 (1975), pp. 349–393. ISSN: 00359246. URL: http://www.jstor.org/stable/2984782.

[76] F Landis Markley. "Attitude error representations for Kalman filtering". In: *Journal of guidance, control, and dynamics* 26.2 (2003), pp. 311–317.

[77] Nathan Michael et al. "The grasp multiple micro-UAV testbed". In: *IEEE Robotics & Automation Magazine* 17.3 (2010), pp. 56–65.

[78] Z. Min, J. Wang, and M. Q. Meng. "Robust Generalized Point Cloud Registration With Orientational Data Based on Expectation Maximization". In: *IEEE Transactions on Automation Science and Engineering* (2019), pp. 1–15. ISSN: 1545-5955.

[79] Maher Moakher. "Means and Averaging in the Group of Rotations". In: *SIAM J. Matrix Anal. Appl.* 24.1 (Jan. 2002), pp. 1–16. ISSN: 0895-4798.

[80] Guido Montufar. "Notes on the number of linear regions of deep neural networks". In: Mar. 2017.

[81] Guido F Montufar et al. "On the Number of Linear Regions of Deep Neural Networks". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 2924–2932.

[82] David M Mount, Nathan S Netanyahu, and Jacqueline Le Moigne. "Efficient algorithms for robust feature matching". In: *Pattern Recognition* 32.1 (1999), pp. 17–38. ISSN: 0031-3203. DOI: https://doi.org/10.1016/S0031-3203(98)00086-7. URL: https://www.sciencedirect.com/science/article/pii/S0031320398000867.

[83] M.W. Mueller, M. Hehn, and Raffaello D'Andrea. "Covariance correction step for Kalman filtering with an attitude". In: *Journal of Guidance, Control, and Dynamics* (2016), pp. 1–7.

[84] Mark W Mueller, Michael Hamer, and Raffaello D'Andrea. "Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadrocopter state estimation". In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 1730–1736.

[85] Arthur G. O. Mutambara. *Decentralized Estimation and Control for Multisensor Systems*. 1st. Boca Raton, FL, USA: CRC Press, Inc., 1998. ISBN: 0849318653.

[86] Andriy Myronenko and Xubo Song. "Point Set Registration: Coherent Point Drift". In: *IEEE transactions on pattern analysis and machine intelligence* 32 (Dec. 2010), pp. 2262–75.

[87] Hien Nguyen. *A Novel Algorithm for Clustering of Data on the Unit Sphere via Mixture Models*. Sept. 2017.

[88] Gabriel Nützi et al. "Fusion of IMU and vision for absolute scale estimation in monocular SLAM". In: *Journal of intelligent & robotic systems* 61.1 (2011), pp. 287–299.

[89] Carl Olsson, Fredrik Kahl, and Magnus Oskarsson. "Branch-and-Bound Methods for Euclidean Registration Problems". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.5 (2009), pp. 783–794. DOI: 10.1109/TPAMI.2008.131.

[90] F. Olsson, J. Rantakokko, and J. Nygårds. "Cooperative localization using a foot-mounted inertial navigation system and ultrawideband ranging". In: *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. Oct. 2014, pp. 122–131. DOI: 10.1109/IPIN.2014.7275476.

[91] Razvan Pascanu, Guido Montúfar, and Yoshua Bengio. "On the number of inference regions of deep feed forward networks with piece-wise linear activations". In: *CoRR* abs/1312.6098 (2013). arXiv: 1312.6098.

[92] David Peel, William J Whiten, and Geoffrey J McLachlan. "Fitting Mixtures of Kent Distributions to Aid in Joint Set Identification". In: *Journal of the American Statistical Association* 96.453 (2001), pp. 56–63.

[93] Arthur Pewsey and Eduardo Garc'ia-Portugu'es. "Recent advances in directional statistics". In: *arXiv: Methodology* (2020).

[94] Frank Pfeuffer, Michael Stiglmayr, and Kathrin Klamroth. "Discrete and geometric Branch and Bound algorithms for medical image registration". In: *Annals of Operations Research* 196 (2012), pp. 737–765.

[95] Chris Piech et al. "Deep Knowledge Tracing". In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes et al. Curran Associates, Inc., 2015, pp. 505–513.

[96] François Pomerleau et al. "Challenging data sets for point cloud registration algorithms". In: *The International Journal of Robotics Research* 31.14 (Dec. 2012), pp. 1705–1711.

[97] Florent Poux et al. "SMART POINT CLOUD: DEFINITION AND REMAINING CHALLENGES". In: vol. IV-2/W1. Oct. 2016. DOI: 10.5194/isprs-annals-IV-2-W1-119-2016.

[98] Marcelo O. R. Prates. "Learning to solve NP-complete problems". In: 2019.

[99] Nissanka B Priyantha et al. "The cricket compass for context-aware mobile applications". In: *Proceedings of the 7th annual international conference on Mobile computing and networking*. ACM. 2001, pp. 1–14.

[100] Amanda Prorok et al. “Indoor navigation research with the Khepera III mobile robot: An experimental baseline with a case-study on ultra-wideband positioning”. In: *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE. 2010, pp. 1–9.

[101] C. Qi et al. “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space”. In: *NIPS*. 2017.

[102] Charles R Qi et al. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *arXiv preprint arXiv:1612.00593* (2016).

[103] Maithra Raghu et al. “On the Expressive Power of Deep Neural Networks”. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, Aug. 2017, pp. 2847–2854.

[104] Anand Rangarajan et al. “A robust point matching algorithm for autoradiograph alignment”. In: *Visualization in Biomedical Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 277–286. ISBN: 978-3-540-70739-4.

[105] Ioannis M. Rekleitis, Gregory Dudek, and Evangelos E. Milios. “Multi-robot Exploration of an Unknown Environment, Efficiently Reducing the Odometry Error”. In: *Proceedings of the Fifteenth International Joint Conference on Artifical Intelligence - Volume 2*. IJCAI’97. Nagoya, Japan: Morgan Kaufmann Publishers Inc., 1997, pp. 1340–1345.

[106] S. I. Roumeliotis and G. A. Bekey. “Distributed multirobot localization”. In: *IEEE Transactions on Robotics and Automation* 18.5 (Oct. 2002), pp. 781–795. ISSN: 1042-296X. DOI: 10.1109/TRA.2002.803461.

[107] Radu Bogdan Rusu. *Semantic 3D Object Maps for Everyday Robot Manipulation*. Springer Publishing Company, Incorporated, 2013. ISBN: 9783642354786.

[108] Radu Bogdan Rusu et al. “Towards 3D Point Cloud Based Object Maps for Household Environments”. In: *Robot. Auton. Syst.* 56.11 (Nov. 2008), pp. 927–941. ISSN: 0921-8890.

[109] J Sachs. “Handbook of Ultra-Wideband Short-Range Sensing: Theory, Sensors, Applications”. In: (Dec. 2012), p. 25.

[110] Zafer Sahinoglu, Sinan Gezici, and Ismail Gvenc. *Ultra-wideband Positioning Systems: Theoretical Limits, Ranging Algorithms, and Protocols*. New York, NY, USA: Cambridge University Press, 2011.

[111] Vinit Sarode et al. “PCRNet: Point Cloud Registration Network using PointNet Encoding”. In: *ArXiv* abs/1908.07906 (2019).

[112] Davide Scaramuzza et al. “Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in GPS-denied environments”. In: *IEEE Robotics & Automation Magazine* 21.3 (2014), pp. 26–40.

[113] Aleksandr Segal, Dirk Hähnel, and Sebastian Thrun. “Generalized-ICP”. In: *Proc. of Robotics: Science and Systems* (June 2009).

[114] Marcelo J Segura, Vicente A Mut, and Hector D Patiño. “Mobile robot self localization system using IR-UWB sensor in indoor environments”. In: *Robotic and Sensors Environments, 2009. ROSE 2009. IEEE International Workshop on*. IEEE. 2009, pp. 29–34.

[115] Jacopo Serafin and Giorgio Grisetti. “NICP: Dense Normal Based Point Cloud Registration”. In: *Proceedings of the ... IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE/RSJ International Conference on Intelligent Robots and Systems* (Sept. 2015), pp. 742–749.

[116] Thiago Serra, Christian Tjandraatmadja, and Srikumar Ramalingam. “Bounding and Counting Linear Regions of Deep Neural Networks”. In: *CoRR* abs/1711.02114 (2017). arXiv: 1711.02114.

[117] David Silver et al. “Mastering the game of Go with deep neural networks and tree search”. In: *Nature* 529 (2016), pp. 484–503.

[118] Dan Simon. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[119] J. Spjøtvold, P. Tøndel, and T. A. Johansen. “Continuous Selection and Unique Polyhedral Representation of Solutions to Convex Parametric Quadratic Programs”. In: *Journal of Optimization Theory and Applications* 134.2 (Aug. 2007), pp. 177–189. ISSN: 1573-2878.

[120] John R. Spletzer. *A New Approach to Range-only SLAM for Wireless Sensor Networks*.

[121] Przemysław Spurek et al. “HyperFlow: Representing 3D Objects as Surfaces”. In: *ArXiv* abs/2006.08710 (2020).

[122] Suvrit Sra. “Directional Statistics in Machine Learning: A Brief Review Suvrit Sra”. In: *Applied Directional Statistics* (2018).

[123] T. Stoyanov, M. Magnusson, and A. J. Lilienthal. “Point set registration through minimization of the L2 distance between 3D-NDT models”. In: *2012 IEEE International Conference on Robotics and Automation*. Apr. 2012, pp. 5196–5201.

[124] R. S. Sutton, A. G. Barto, and R. J. Williams. “Reinforcement learning is direct adaptive optimal control”. In: *IEEE Control Systems Magazine* 12.2 (Apr. 1992), pp. 19–22. ISSN: 1066-033X.

[125] Queens Maria Thomas, Oliver Wasenmüller, and Didier Stricker. “DeLiO: Decoupled LiDAR Odometry”. In: *CoRR* abs/1904.12667 (2019). arXiv: 1904.12667.

[126] Yanghai Tsin and Takeo Kanade. "A Correlation-Based Approach to Robust Point Set Registration". In: *Computer Vision - ECCV 2004*. Ed. by Tomáš Pajdla and Jiří Matas. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 558–569. ISBN: 978-3-540-24672-5.

[127] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[128] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. "Pointer Networks". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'15. Montreal, Canada: MIT Press, 2015, pp. 2692–2700.

[129] Di Wang et al. "Accurate mix-norm-based scan matching". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 1665–1671.

[130] X. Wang et al. "Sensor selection based on the fisher information of the Kalman filter for target tracking in WSNs". In: *Proceedings of the 33rd Chinese Control Conference*. July 2014, pp. 383–388. DOI: 10.1109/ChiCC.2014.6896653.

[131] Xiaolong Wang et al. "Non-local Neural Networks". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7794–7803. DOI: 10.1109/CVPR.2018.00813.

[132] Yue Wang. "Linear least squares localization in sensor networks". In: *EURASIP Journal on Wireless Communications and Networking* 2015.1 (Mar. 2015), p. 51. ISSN: 1687-1499. DOI: 10.1186/s13638-015-0298-1.

[133] Yue Wang and Justin M. Solomon. "Deep Closest Point: Learning Representations for Point Cloud Registration". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), pp. 3522–3531.

[134] Jan Wendel et al. "An integrated GPS/MEMS-IMU navigation system for an autonomous helicopter". In: *Aerospace Science and Technology* 10.6 (2006), pp. 527–533.

[135] Zhirong Wu et al. "3D ShapeNets: A deep representation for volumetric shapes". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1912–1920. DOI: 10.1109/CVPR.2015.7298801.

[136] Jiaolong Yang, Hongdong Li, and Yunde Jia. "Go-ICP: Solving 3D Registration Efficiently and Globally Optimally". In: *2013 IEEE International Conference on Computer Vision*. 2013, pp. 1457–1464. DOI: 10.1109/ICCV.2013.184.

[137] Jiaolong Yang et al. "Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.11 (2016), pp. 2241–2254. DOI: 10.1109/TPAMI.2015.2513405.

[138] Zi Jian Yew and Gim Hee Lee. "3DFeat-Net: Weakly Supervised Local 3D Features for Point Cloud Registration". In: *ECCV*. 2018.

[139] Zi Jian Yew and Gim Hee Lee. "RPM-Net: Robust Point Matching using Learned Features". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[140] Maciej Zamorski et al. "Adversarial autoencoders for compact representations of 3D point clouds". In: *Computer Vision and Image Understanding* 193 (Feb. 2020), p. 102921. DOI: 10.1016/j.cviu.2020.102921.

[141] Kuangen Zhang et al. "Linked dynamic graph cnn: learning on point cloud via linking hierarchical features". In: *arXiv:1904.10014 [cs]* (Apr. 2019).

[142] Zhengyou Zhang. "Iterative point matching for registration of free-form curves and surfaces". In: *International journal of computer vision* 13 (Oct. 1994), pp. 119–152.

[143] Zhiyuan Zhang, Yuchao Dai, and Jiadai Sun. "Deep learning based point cloud registration: an overview". In: *Virtual Reality & Intelligent Hardware* 2.3 (2020). 3D Visual Processing and Reconstruction Special Issue, pp. 222–246. ISSN: 2096-5796.

[144] Jie Zhou et al. "Graph neural networks: A review of methods and applications". In: *AI Open* 1 (2020), pp. 57–81. ISSN: 2666-6510. DOI: https://doi.org/10.1016/j.aiopen.2021.01.001. URL: https://www.sciencedirect.com/science/article/pii/S2666651021000012.