國立臺灣大學電機資訊學院電機工程學系

碩士論文

Department of Electrical Engineering
College of Electrical Engineering and Computer Science
National Taiwan University

Master's Thesis

基於網格平面的線特徵與矩形幾何約束之單眼視覺里程計

Monocular Visual Odometry Based on Line Features on Grid Mesh Plane and Rectangular Geometric Constraint

李昀誠

Yun-Xian Li

指導教授：連豊力 博士

Advisor: Feng-Li Lian, Ph.D.

中華民國 113 年 7 月

July 2024

doi:10.6342/NTU202402036

doi:10.6342/NTU202402036

國立臺灣大學碩士學位論文
口試委員會審定書
MASTER'S THESIS ACCEPTANCE CERTIFICATE
NATIONAL TAIWAN UNIVERSITY

基於網格平面的線特徵與矩形幾何約束之單眼視覺里程計

Monocular Visual Odometry Based on Line Features on Grid Mesh Plane and Rectangular Geometric Constraint

本論文係 李昀誠 (學號 R10921130) 在國立臺灣大學電機工程學系完成之碩士學位論文，於民國一百一十三年七月二十五日承下列考試委員審查通過及口試及格，特此證明。

The undersigned, appointed by the Department of Electrical Engineering on 25 July 2024 have examined a Master's thesis entitled above presented by Yun-Xian Li (ID R10921130) candidate and hereby certify that it is worthy of acceptance.

口試委員 Oral examination committee:

連豐力 （指導教授 Advisor）		連豐力
王富正		王富正
江明理		江明理
系主任 Director	李建模	李建模

誌謝

當初因為不喜歡讀書而選擇讀高職的我，一定想不到現在會在臺大完成我的碩士論文。誠摯地感謝一路上許多貴人的幫助與鼓勵。

感謝指導教授**連豊力 博士**，讓我從研究過程中學到控制的精神在於回授，若沒有如此頻繁地與老師討論研究的進度，以我的個性也許至今仍無法完成這篇論文。感謝教授鼓勵我們參與學術活動，人生第一次出國便是到日本橫濱參加 ICRA 2024，即使我沒有投稿，也能與實驗室的各位一同參與頂尖的學術會議，絕對是我在碩士期間最難忘的經驗。感謝口試委員**王富正 教授**與**江明理 教授**，在遇到颱風假還是如期口試，給我許多建議，使我的碩士論文更加完整。

感謝**優式機器人**的各位，在研究期間給我的建議與支援，讓我瞭解到從研究要轉變為產品所需考量的各種因素。感謝 NCSLab 的各位: 謝謝**有田**除了幫助我們處理實驗室的各種事務，也會針對研究給予建議；謝謝**昱翔**不管是在我的研究，亦或是和公司討論時都給我非常多的建議與幫助；謝謝**晁維**和**昱圣**，在機器人學一起完成有趣的期末專題；謝謝**雁丞**，你強大的行動力總是激勵著實驗室的大家；謝謝**高逵**與**凱正**，在研究的空檔跟我一同挑戰環騎台北；謝謝**暄坐**與**之蕙**，讓我搶先獲得老師對於準備口試簡報的建議；謝謝**旅青**，不論是研究上的問題還是動漫話題都和你聊得非常開心，我很敬佩你對研究的熱忱，祝你的博士之路一切順利；謝謝**芳緯**，在日本的行程都是靠你的推薦，讓我第一次出國的體驗非常棒，也祝你在工研院的研究順利；謝謝**芳源**，在研究之餘還能一起談論各種時事話題，排解研究遇到瓶頸時的壓力；謝謝**瑋恩**、**宜儒**、**偉銘**、**敬祥**、**祐誠**、**勝凱**和**林靖**，與實驗室的大家在進度報告與期刊報告時總是非常的熱鬧、有趣。另外也非常感謝從大安高工到臺科大一起度過七年，以及一起到臺大與我共度九年的同學們，每次臨時起意約吃飯或是看電影都可以隨時出現，和大家一起的時光非常開心。

最後感謝我的家人們，讓我在學業以外的一切生活瑣事都不必擔心，一路上也都完全支持我所做的決定，使我的求學生涯如此精彩。

李昀誠 謹誌

中華民國一百一十三年七月三十日

doi:10.6342/NTU202402036

基於網格平面的線特徵與矩形幾何約束之單眼視覺里程計

研究生：李昀誠　　指導教授：連豊力 博士

國立臺灣大學　電機工程學系

摘要

科技發展迅速，機器人已經可以獨自執行各種複雜的任務，不需要額外的人為參與，從而提升我們日常生活的便利性。為了讓機器人能夠自主執行任務，機器人必須能夠感知周遭的環境，以便在場景中移動與避開障礙物。

本論文提出一個單眼視覺里程計，用於機器人在都市環境的定位。透過線段偵測、合併、分群與排除異常值，偵測出畫面中建築物外牆上的水平與垂直結構線，並透過矩形幾何約束，從偵測到的線條在三維空間中建立網格平面，經由追蹤建立的網格平面來估測機器人的位置。為了讓本論文所提出的演算法能在即時條件下執行，加入了可靠性測試，透過重投影特徵將估測資料做降取樣處理，進而提升運算的效率。

本論文透過模擬、室內與室外的實驗驗證了演算法的效能，展示在實際的都市環境中定位的能力。

關鍵字:

視覺里程計、線特徵、幾何約束、單眼視覺、影像處理

doi:10.6342/NTU202402036

Monocular Visual Odometry Based on Line Features on Grid Mesh Plane and Rectangular Geometric Constraint

Student: Yun-Xian Li Advisor: Feng-Li Lian, Ph.D.

Department of Electrical Engineering

National Taiwan University

ABSTRACT

Nowadays, mobile robots can perform complex tasks independently, reducing the requirement for human involvement and thus significantly improving the convenience of our everyday lives. To perform tasks independently, mobile robots need to determine their location, sense its surroundings for navigation and avoid obstacles.

This thesis proposes a monocular visual odometry for a mobile robot localization in urban environments. The proposed algorithm detects the horizontal and vertical structural lines on a building facade using image processing techniques including line segment detection, segments merging, clustering and outlier removal. Then, the rectangular geometric constraint is applied to form the grid mesh plane in 3-D space using the detected line features, and the mobile robot position is estimated by tracking the plane in camera images. For the proposed algorithm to operate in the real-time, a reliability testing is involved to enhance the calculation efficiency by downsampling data via feature reprojection.

doi:10.6342/NTU202402036

The proposed algorithm in this thesis has validated the performance through simulations and both indoor and outdoor experiments, demonstrating its capability for localization in noisy urban environments.

Keywords:

Visual odometry, line features, geometric constraint, monocular vision, image processing

doi:10.6342/NTU202402036

CONTENTS

doi:10.6342/NTU202402036

doi:10.6342/NTU202402036

LIST OF FIGURES

doi:10.6342/NTU202402036

doi:10.6342/NTU202402036

doi:10.6342/NTU202402036

doi:10.6342/NTU202402036

LIST OF TABLES

doi:10.6342/NTU202402036

Chapter 1
Introduction

In this chapter, the motivation of this thesis is first presented in Section 1.1. The problem formulation is provided in Section 1.2. The contribution of this thesis is summarized in Section 1.3. Finally, the organization of this thesis is provided in Section 1.4.

1.1 Motivation

Nowadays, mobile robots are able to perform complex tasks without human intervention and have been widely used in various fields such as military, agriculture and domestic appliances fields, making our lives more convenient.

For a mobile robot to perform tasks autonomously, it needs to be able to determine its location, sense its surroundings for navigation and avoid obstacles. The traditional method is Global Positioning System (GPS) [48: Limitd 2007] with a prior map. A GPS device must receive signals from at least four satellites to get reliable position estimation [49: Penn State], as illustrated in Figure 1.1, the GPS cannot be used indoors or in sheltered environments since the radio signals from the satellites are affected by walls and other obstacles. But according [1: Vatansever & Butun 2017], even in outdoor environments, there are many factors that can affect GPS accuracy, such as satellite and receiver clock error, electronic noise, uncertainty in the position of the satellites and multipath error (Figure 1.2).

In complex environments like forests and urban canyons, there are various sensors can improve positioning accuracy. For example, in [4: 廖育萱 & 彭彥嘉 2023], the authors improve the GPS accuracy with real time kinematic(RTK) technique, and utilize LiDAR for creating point cloud data of the surroundings for localization in orchards.

doi:10.6342/NTU202402036

Other examples in urban environment, Liao et al. [5: Liao *et al.* 2023] provide a novel dataset and benchmarks which includes color image from fisheye cameras, stereo image from perspective cameras, point cloud data from LiDAR and pose data of the car from GPS/IMU. Yin et al. [6: Yin et al. 2022] provide both indoor and outdoor dataset using a ground robot with multiple sensors, includes color camera, infrared camera, event camera, stereo camera, LiDAR, IMU and GNSS receiver. These datasets allow researchers to study various positioning methods with multiple sensors.

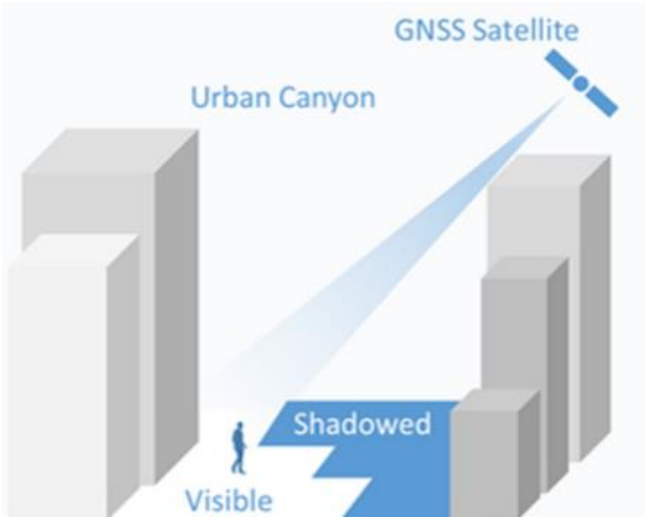


Figure 1.1: The radio single from satellite is blocked by obstacles.[3: Strandjord et al. 2020]

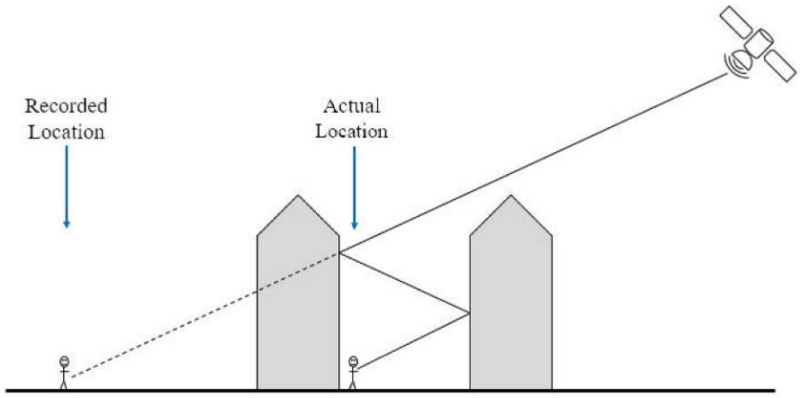


Figure 1.2: The multipath error due to GPS signals reflecting off of tall buildings.[2: Hutt et al. 2021]

doi:10.6342/NTU202402036

However, with improving computing performance, increasing sensor and maintenance costs, commercial applications prefer visual methods over other ranging sensing methods for localization. In [50: Tesla], Tesla uses vision module to replace ultrasonic sensors for close-range sensing, claiming that the vision module can achieve the same function with ultrasonic sensors through software update. By the test in [51: It's Only Electric], the vision module can achieve the same function as ultrasonic sensors, but its performance is poorer. Such as the minimum safety distance and larger blind zone as shown in Figure 1.3.

(a) Safety distance test

(b) Obstacle test

(c) Tesla Vision dead angle

Figure 1.3: The test result from [51: It's Only Electric]

Considering the trend of research in visual localization and the applications in the urban environments, when the mobile robot is near a building, the building will block the GPS signals. This thesis will concentrate on the visual localization method near the

doi:10.6342/NTU202402036

buildings with specific features, such as the building facade, doors and windows. The goal is to achieve visual localization in these challenging environments.

1.2 Problem Formulation

This thesis is aiming to achieve monocular visual localization near the buildings under following constraints: planar, repetitive and regular grid-like features. Some example scenarios are shown in Figure 1.4. These features are not suitable for standard visual localization methods due to their lack of uniqueness and varied depth information. As illustrated in Figure 1.5, to achieve localization in such environments, there are two main tasks need to be addressed.

The first task is to find the target plane, the camera captures the target plane and surrounding scenes, so it is necessary to find the boundaries of the plane, then extract valid features from the plane for use in subsequent localization processes, such as the structure lines and the corner points on the plane.

After identifying the target plane, the second task is to establish the camera's position relative to the target plane. Determine the relative angle between the plane and the camera using the identified features, then track the target plane in camera’s view and estimate the 6 degrees-of-freedom poses of the camera.

1.3 Contribution

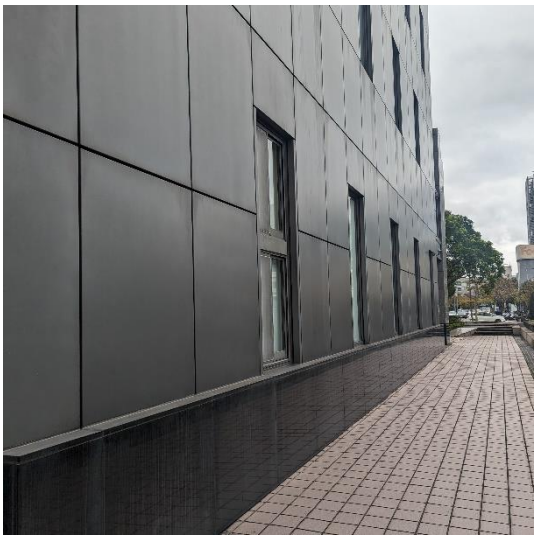
The contributions of this thesis can be divided into following two parts.

First is the plane tracking method with monocular camera, this thesis uses structure lines as features similar to [7: Liu et al. 2022] and [8: Zhou et al. 2015], However, unlike matching features through pixel patches on the lines, this thesis matches features based on their position and geometric relation, so the matching process is not affected by changes in time, light and shadow.

doi:10.6342/NTU202402036

Second, the camera pose can be estimated in single camera frame, the common estimation method is to solve five-point relative pose problem between two views [9: Lu et al. 2000] [10: Nister 2004]. Since the scenario in this thesis is aiming the grid-like features on the plane, the depth estimation of features can be simplified as linear equations without iteration.

(a) Glass curtain wall



(b) Building facades

(c) Windows in hallway

(d) Doors in hallway

Figure 1.4: The scenarios of grid-like planar features

doi:10.6342/NTU202402036

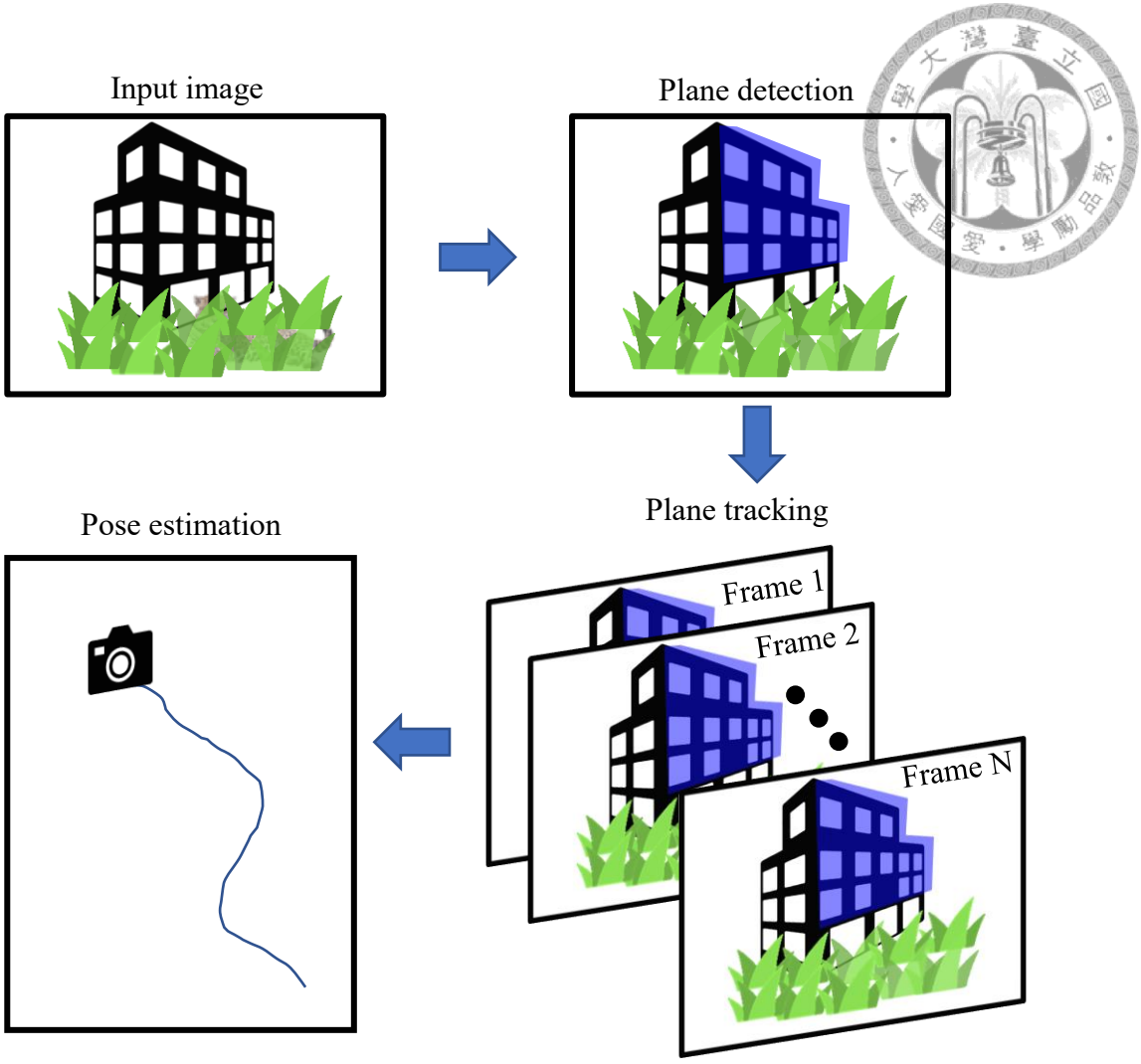


Figure 1.5: Schematic diagram of localization process

1.4 Organization of the Thesis

The rest of this thesis is organized as follows. The background and relative literature are discussed in Chapter 2. And some related algorithm implemented in the proposed algorithm are introduced in Chapter 3. In Chapter 4, the details of the proposed algorithm are introduced, including an overview of entire algorithm, feature extraction and motion estimation. The experiment result and analysis are presented in Chapter 5, Finally, the conclusions and future works of this thesis is discussed in Chapter 6.

doi:10.6342/NTU202402036

Chapter 2
Background and Literature Survey

In this chapter, the background and literature survey of the visual localization methods are discussed. Section 2.1 introduce the diverse visual sensors employed in visual odometry. The comparison of different data processing methods and estimation methods are provided in Section 2.2 and Section 2.3. The overview classification is summarized in Figure 2.1.

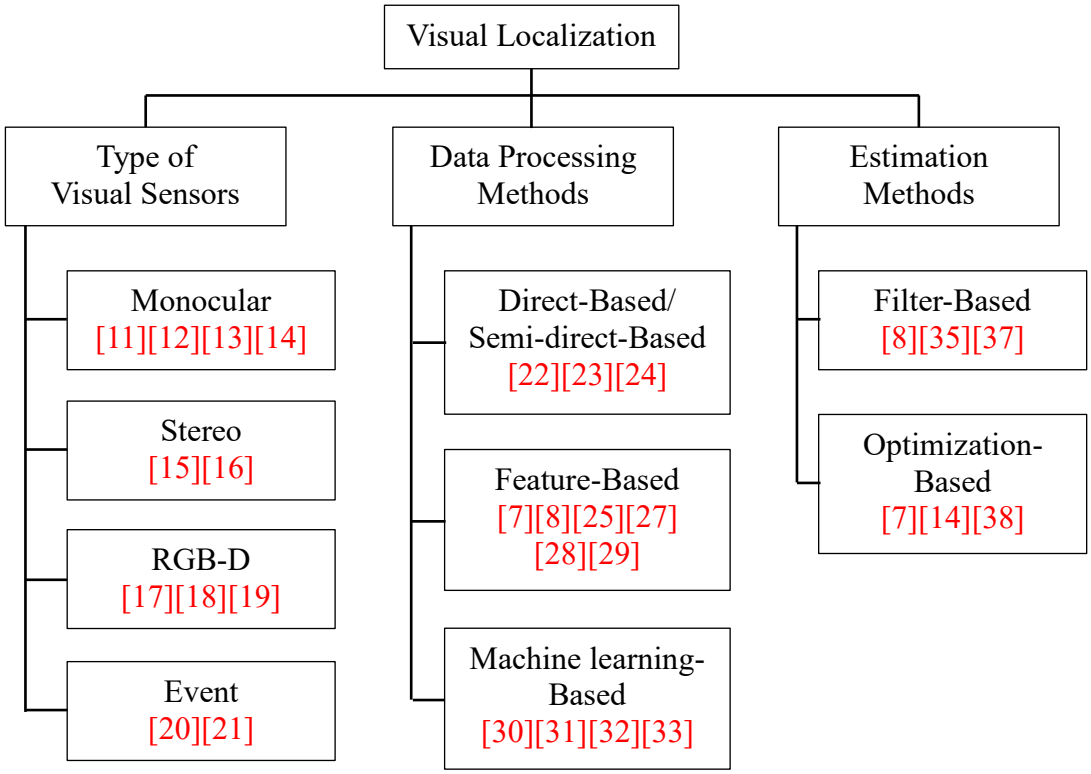


Figure 2.1: Different methods for visual localization

2.1 Type of Visual Sensors

The visual odometry relies on visual sensors to offer the information about the environment. The typical sensor types include monocular, stereo, RGB-D and event

doi:10.6342/NTU202402036

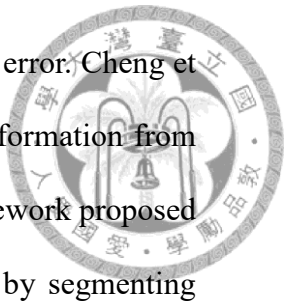
camera, each type has its own advantages and disadvantages, and the data processing methods and suitable scenarios are also different.

Monocular cameras are cost-effective, common and easy-to-use sensors. However, they suffer from scale ambiguity problem, making it challenging to accurately estimate the length of translational movement from features [11: Choi et al. 2013]. ORB-SLAM [12: Mur-Artal et al. 2015] uses ORB features to compute the relative pose between two frames in parallel two geometrical models, try to recover the unique solution of relative pose. Structure-SLAM [13: Li et al. 2020] uses convolutional neural network and Manhattan World assumption to estimate pose from point, line and planar features. Qin et al. [14: Qin et al. 2018] present a method that tightly couples pre-integrated IMU measurements and feature observations to achieve highly accurate and robust estimation.

Stereo cameras use a pair of cameras, the left camera is generally taken as the pose of the stereo camera, and the pose of right camera is fixed from left camera, so the depth of the pixels in the image can be estimated. But it requires higher computational cost to calculate depth from two images, and the camera needs to be well calibrated to get accurate results. Lin et al. [15: Lin et al. 2022] use stereo camera to obtain point cloud map, and detect the point cloud changes to estimate trajectory. Zhang et al. [16: Zhang et al. 2015] present a graph-based SLAM using 3-D straight lines as features, and uses two different representations to parameterize 3-D lines for initialization and optimization, performs better result in line-rich environment.

RGB-D cameras combine an RGB camera with a depth sensor, like an infrared light camera and projector, utilizing structured-light or time-of-flight (ToF) methods to measure the pixel depth directly, simplifying the depth estimation process. However, the depth sensors are interference by observed material and sunlight, and the measurement range is restricted. Lin et al. [17: Lin et al. 2023] propose a key-frame based method with

doi:10.6342/NTU202402036



intrinsic keyframe selection mechanism, effectively reduces the tracking error. Cheng et al. [18: Cheng et al. 2023] present a system that fuses 2-D semantic information from RGB image and 3-D geometric information from depth sensor. The framework proposed in [19: Zhao et al. 2019] is able to operate in dynamic environments by segmenting objects and categorizing them as static or dynamic objects.

Unlike other conventional visual sensors that provide the entire images at a fixed frequency, event cameras have independent pixels that transmit data solely when brightness changes in the scene at the time they occur. Thus, the sensor output is asynchronous and has high temporal resolution, resulting in low power consumption and suitable for tracking fast motion and high-speed dynamics without suffering from motion blur. The system proposed in [20: Weikersdorfer et al. 2013] tracks the events at edges from the scenes and achieves real-time performance on standard computing hardware. Rebecq et al. [21: Rebecq et al. 2017] present an event-based visual odometry that can track fast camera motions, unaffected by motion blur, and operates very well in high dynamic environments.

According to the problem to be solved in this thesis, the featureless planar scenarios are not suitable for stereo camera and event camera, and considering the hardware cost and computation cost, a monocular camera is selected for this thesis. The depth estimation for 3-D reconstruction can be addressed using the data processing techniques outlined in Section 2.2.

2.2 Data Processing Methods

The data processing methods can be categorized into three types: direct based, feature-based and machine learning-based. These approaches have their trade-offs and considerations, such as system complexity, computation cost, and application scenarios.

doi:10.6342/NTU202402036

Direct-based methods directly operate on the pixel intensity or color values from sensor raw data. Semi-direct methods are similar to direct-based method but concentrate on the pixels with high intensity gradients. LSD-SLAM [22: Engel et al. 2014] allows to estimate in large-scale environments with pose estimation based on direct image alignment and 3-D reconstruction with semi-direct depth maps. The visual SLAM system in [23: Silveira et al. 2008] aligns the reference image with successive frames directly and selects image regions to estimate motion by plane-based epipolar geometric method. SVO [24: Forster et al. 2017] uses direct-based methods to track and triangulate pixels, and uses feature-based methods for optimization of structure and motion.

Feature-based methods focus on specific areas within an image that contain unique information known as features. These features can take on various forms, including points, lines, planes, markers, or specific objects. ORB-SLAM3 [25: Campos et al. 2021] is one of the fundamental feature-based SLAM that extracts the features using ORB descriptor [26: Rublee et al. 2011]. In order to extract more useful information from images, in [7: Liu et al. 2022], [8: Zhou et al. 2015], [27: Gomez-Ojeda et al. 2019], and [28: Guan et al. 2023], both point features and line features are extracted to obtain more robustness and accurately estimation. Sun et al. [29: Sun et al. 2018] propose a statistical information grid-based plane extraction algorithm for tracking planes in indoor environments, achieving high accuracy and robustness in both on-board and hand-held applications.

Machine learning-based methods integrate valuable additional information from the environment with artificial neural networks to train models that can replace conventional processes. The framework proposed in [30: Gelen & Atasoy 2023] uses three different models to achieve pose estimation with an event camera. The SLAM system proposed in [31: Yang & Scherer 2019] improves the accuracy of monocular SLAM by integrating semantic scene understanding with traditional methods during feature extraction. The

doi:10.6342/NTU202402036

SLAM systems proposed in [32: Liu & Miura 2021] and [33: Ran et al. 2021] use semantic segmentation to recognize dynamic objects, and achieve precise estimation in dynamic environments.

Since corner points and structure lines are valuable information in scenarios with grid-like planar features, feature-based methods are selected in this thesis.

2.3 Estimation Methods

The estimation methods can be separated into two main categories: filter-based methods and optimization-based methods.

Filter-based methods consist of two main parts: a prediction step and an update step, they compare predicted state with the measurement from sensor data to rectify the current state, reducing estimation error. Such as extended Kalman filter (EKF) [34: Kalman 1960] used in [8: Zhou et al. 2015] and [35: Bloesch et al. 2015]. And error-state EKF [36: Solà 2015] used in [37: Chamorro et al. 2022].

Optimization-based methods achieve the optimal estimation by minimizing the designed cost function. Consequently, they outperform filter-based methods in terms of accuracy but requires more computational resources. In [14: Qin et al. 2018], the system optimizes camera pose with pose graph optimization. Another commonly used method is bundle adjustment, which is used in [7: Liu et al. 2022]. The visual-inertial odometry proposed in [38: Mueggler et al. 2018] expresses the optimization problem as a nonlinear least square problem and apply it with standard numerical solvers.

Since the monocular camera is the sole sensor utilized in this thesis, the proposed system is relatively simple compared to existing methods. As a result, optimization-based methods can achieve accurate estimation without requiring extensive computational resources.

doi:10.6342/NTU202402036

Chapter 3
Related Algorithms

In this chapter, the existing algorithms and techniques related to the propose algorithm are presented. This includes the pinhole camera model in Section 3.1, representation method of line features in Section 3.2, the clustering algorithm used in the proposed algorithm in Section 3.3, and the optimization algorithm in Section 3.4.

3.1 Pinhole Camera Model

According to [52: Savarese & Bohg 2023] and [53: Collins 2007], the pinhole camera model is a mathematical representation used to establish a direct correspondence between 3-D points in the real world and their projection onto a 2-D image in pixels.

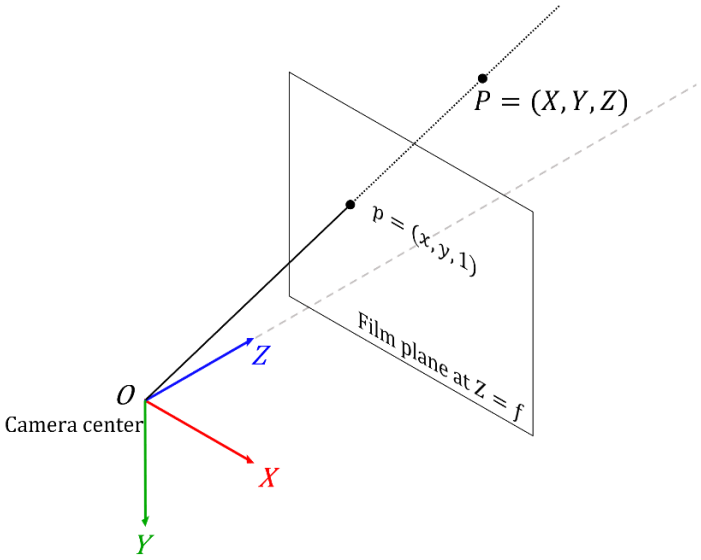


Figure 3.1: Illustration for pinhole camera model

The pinhole camera model projects a 3-D scene point onto a 2-D image point through perspective projection. The camera's intrinsic parameters, such as the image center (c_x, c_y) and focal length f, are well-defined. As illustrated in Figure 3.1, consider a 3-D point P at position (X, Y, Z) relative to the camera center O, where the Z axis is the

doi:10.6342/NTU202402036

optic axis of the camera. First, project point P onto the film plane (X-Y plane at Z=f) using perspective projection as Equation (3.1), and obtain the projected point p at position (x, y, f).

$$x = f\frac{X}{Z}$$

$$y = f\frac{Y}{Z} \tag{3.1}$$

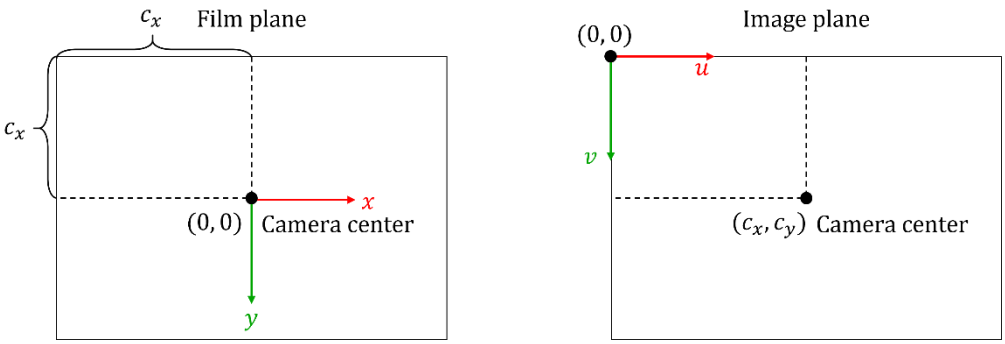


Figure 3.2: Coordinates of film plane and image plane

After determining the position of point p, the next step is mapping the point p to the 2-D image coordinate in pixels. As illustrated in Figure 3.2, the position (x, y, f) of the point p on film plane in real world scale can be mapped to the position (u, v) on the image plane in pixels as shown in Equation (3.2).

$$u = c_x + x$$

$$v = c_y + y \tag{3.2}$$

On the contrary, Equation (3.1) and Equation (3.2) can also map the point from image pixels to film plane in real world coordinates. However, the 2-D image lacks the depth information, resulting in the transformation outputting homogeneous coordinates [54: Wikipedia] without providing the actual depth from camera images.

doi:10.6342/NTU202402036

3.2 Line Representation

The common way to represent a line segment is by using endpoints, like LSD [39: Grompone von Gioi et al. 2010], EDLines [40: Akinlar & Topal 2011] and FLD used in [41: Lee et al. 2014], they all output the endpoints of the detected line segments. But in this thesis, the geometry relation of line features as the input for localization, it is more effective to compare the distance and angle between line segments using line equations rather than endpoints in computation.

The straight line can be represented in general form as Equation (3.3), and can be parameterized by a point (b, m). However, this parameterization method has a problem when it encounters a vertical line, resulting in unbounded values for the slope parameter m.

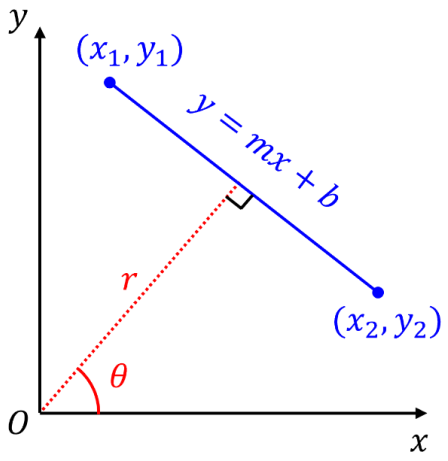


Figure 3.3: The parameters for a line

$$y = mx + b \tag{3.3}$$

$$r = x\cos\theta + y\sin\theta \tag{3.4}$$

Thus, for computational reasons, [42: Duda & Hart 1972] proposed the use of the Hesse normal form to express the equation of a line as Equation (3.4) and illustrated in Figure 3.3, and can be parameterized by a point (r, θ).

doi:10.6342/NTU202402036

According to Equation (3.3) and Equation (3.4), a line with two known points $(x_1, y_1), (x_2, y_2)$ can be represented as a point (r, θ) using Equation (3.5).

$$\theta = \tan^{-1}\left(-\frac{x_2 - x_1}{y_2 - y_1}\right)$$

$$r = x_1 cos\theta + y_1 sin\theta \tag{3.5}$$

3.3 Density-Based Spatial Clustering of Applications with Noise

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a clustering algorithm proposed in [43: Ester et al. 1996]. Unlike other clustering algorithms, it groups together points with many nearby neighbors and treats points with distant neighbors as outliers, so it is not necessary to know the number of groups before clustering.

DBSCAN requires two parameters: the radius ε to define the neighborhood with respect to other points, and the minimum cluster size $MinPts$. The points in a dataset D follow these definitions:

Definition 1: The ε-neighborhood of a point p is defined by Equation (3.6).

$$N_\varepsilon(p) = \{q \in D | dist(p, q) \leq \varepsilon\} \tag{3.6}$$

Definition 2: As shown in Equation (3.7), a point p is core point if its neighbors with in distance ε are larger than $MinPts$.

$$|N_\varepsilon(p)| \geq MinPts \tag{3.7}$$

Definition 3: A point p is directly reachable from a point q with respect to $\varepsilon, MinPts$ if it follows Equation (3.8).

$$\begin{cases} p \in N_\varepsilon(q) \\ |N_\varepsilon(p)| \geq MinPts \end{cases} \tag{3.8}$$

doi:10.6342/NTU202402036

Definition 4: A point p is reachable from a point q if there is a path $p_1, \ldots, p_n, p_1 = q, p_n = p$ such that p_{i+1} is directly reachable from p_i.

Definition 5: All points not reachable from any other point are outliers.

After classifying all the points in D, if p is a core point, then it forms a cluster together with all points that are reachable from it. An example is shown in Figure 3.4.

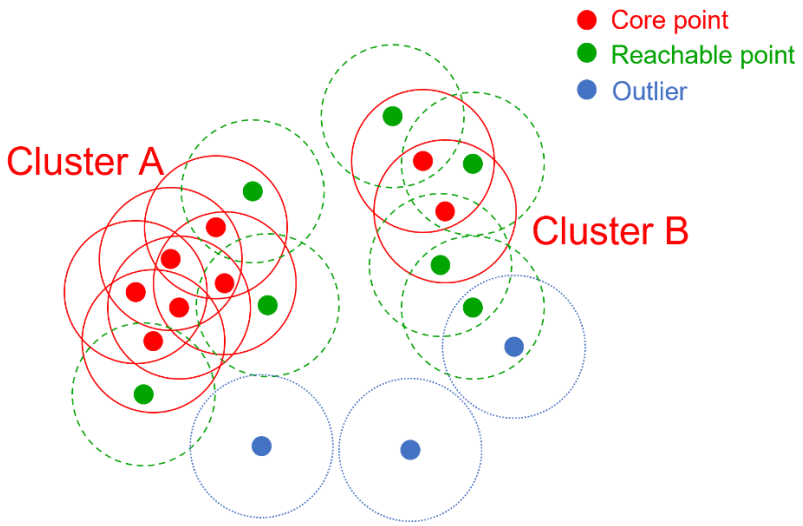


Figure 3.4: An example of clustering result: The circle radius represents ε, MinPts=4. Red and green points indicate cluster members, while blue points signify outliers.

3.4 Random Sample Consensus

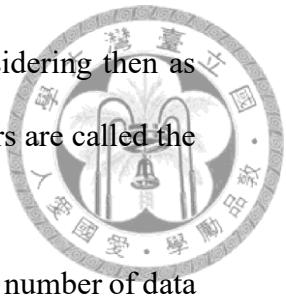
Random Sample Consensus (RANSAC) is an iterative method proposed in [44: Fischler & Bolles 1981], it estimates parameters of a mathematical model from a set of observed data without being influenced by outliers. Therefore, it also can be interpreted as an outlier detection method.

The RANSAC algorithm consists of following steps that are iteratively repeated:

Step 1: A sample subset containing minimal data items is randomly selected as hypothetical inliers.

Step 2: A fitting model is calculated using only the elements of the hypothetical inliers.

doi:10.6342/NTU202402036



Step 3: All data are then tested against the fitted model, considering then as inliers and outliers based on a defined error threshold, the inliers are called the consensus set.

Step 4: The estimated model is quite reliable when a significant number of data points have been classified as part of the consensus set.

This process is iterated a set number of times, refining model with a consensus set size larger than the previous one.

The number of iterations k can be roughly determined based on the desired probability of success p and the size of hypothetical inliers n. Assuming w represent the probability of selecting an inlier from the entire data.

$$w = \frac{\text{number of inliers in data}}{\text{number of points in data}} \tag{3.9}$$

A common case is that w is not well known because the number of inliers in data is unknown before running the RANSAC algorithm, but a rough value can be given. Given a rough value of w, select the size of hypothetical inliers n, let p be the probability of at least one successful model estimation occurring, the probability that the algorithm failing to produce a successful model estimation can be expressed as Equation (3.10), then k can be determined by taking the logarithm of both sides. Figure 3.5 illustrates an example of line fitting for a dataset containing outliers.

$$1 - p = (1 - w^n)^k$$

$$k = \frac{\log(1-p)}{\log(1-w^n)} \tag{3.10}$$

doi:10.6342/NTU202402036

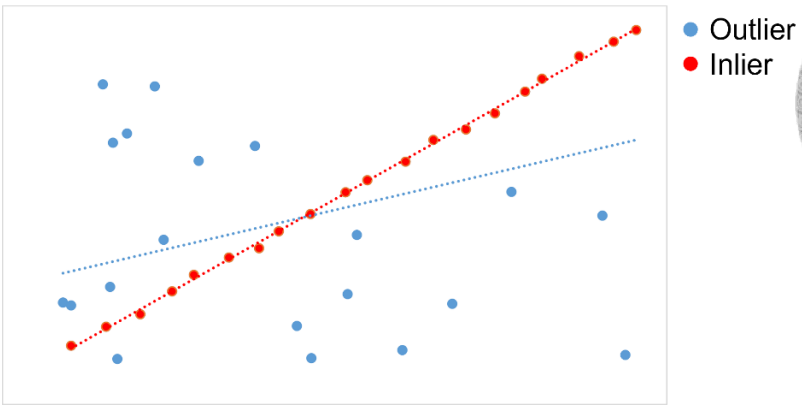


Figure 3.5: An example of line fitting with RANSAC, red dot line is the fitted line with inlier, blue dot line is the fitted line with entire dataset.

doi:10.6342/NTU202402036

Chapter 4
Proposed Algorithm

In this chapter, the details of the proposed algorithm are discussed. the overview of the algorithm is presented in Section 4.1, line segment detection and merging methods are discussed in Section 4.2, Section 4.3 introduces the feature extraction process, and motion estimation procedures are explained in Section 4.4. Lastly, the state updating rules are introduced in Section 4.5.

4.1 Overview

The goal of the proposed algorithm is to achieve the visual localization using a monocular camera in environments with grid-like planar features. The algorithm takes the camera stream as input and outputs the camera motion between camera images as shown in Figure 4.1.

An overview of the algorithm is depicted in Figure 4.2, the algorithm firstly finds the target plane by detect the lines, and extract features from the target plane. Estimates motion with RANSAC after matching the features with previous result, and updates the features before start next estimation.

There are four coordinate systems used in the algorithm: image frame $\{I\}$, world frame $\{W\}$, camera frame $\{C\}$, and plane frame $\{P\}$. A schematic diagram of the coordinate systems is shown in Figure 4.3.

doi:10.6342/NTU202402036

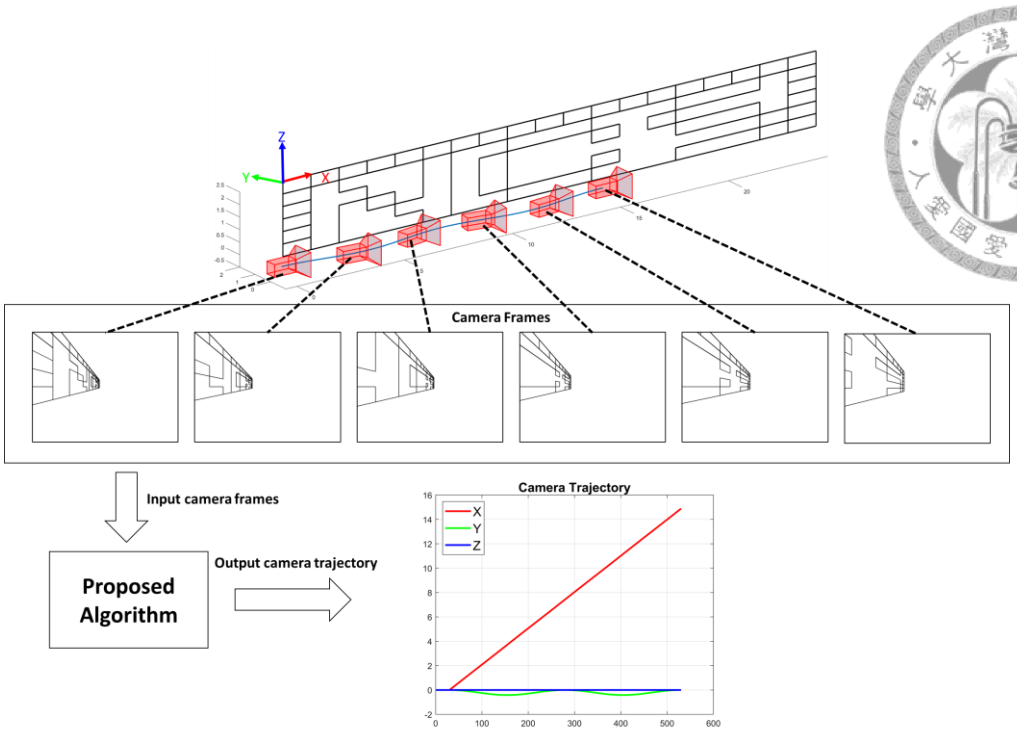


Figure 4.1: The process of solving visual localization problem

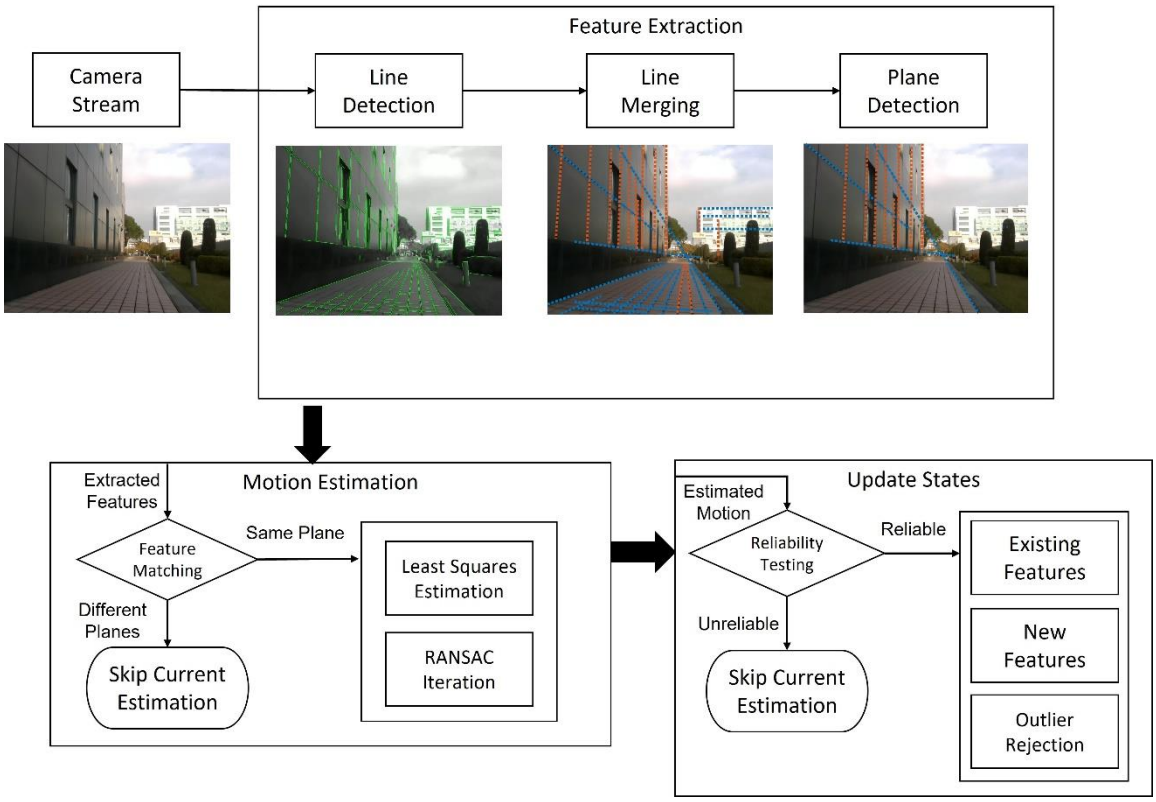


Figure 4.2: Overview of the algorithm process

doi:10.6342/NTU202402036

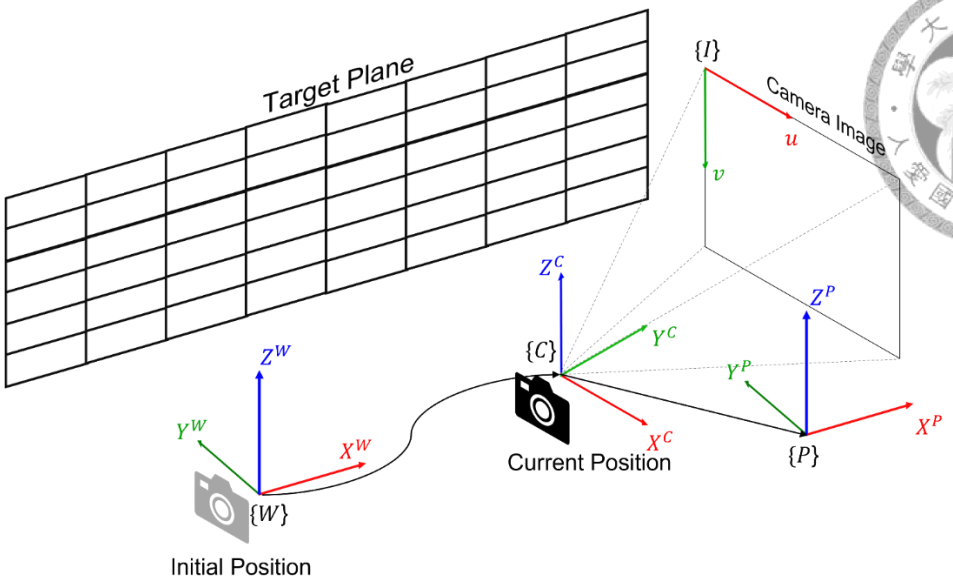


Figure 4.3: The coordinate systems used in the proposed algorithm

The image frame is a 2-D camera image with origin on the top left of image. The world frame is the coordinate system defined by the target plane, the X and Z axes are land on the plane with horizontal and vertical direction, and the Y axis is parallel to the normal vector of the target plane, the origin is fixed as the initial position of the camera. The camera frame is the coordinate system with the camera position as origin, X axis is parallel to the u axis of the image frame, Y axis is the optical axis of the camera. The plane frame is the same coordinate system as the world frame, but the origin moves with the camera center.

The notations used in the proposed algorithm are defined in Table 4.1.

doi:10.6342/NTU202402036

Table 4.1: The notations used in the proposed algorithm

Notation	Description
L	Set of detected line segments
L_w	Set of weighted line segments
L_h	Set of merged horizontal line segments
L_v	Set of merged vertical line segments
R_q	Rotation matrix of a quadrilateral
R_t	Rotation matrix of the target plane
F_h	Set of horizontal line features
F_v	Set of vertical line features
d_t	distance between target plane and camera
T_l	Threshold for minimum segment length
T_w	Threshold for add weights
T_θ	Threshold for identify horizontal and vertical line segments
T_t	Threshold for the tolerance distance of quadrilateral corner
T_e	Threshold for minimum edge length of quadrilateral
T_p	Threshold for minimum angle difference of the target plane
T_n	Threshold for minimum distance of add new features
T_r	Threshold for success rate of invalid features

4.2 Line Segments Detection and Merging

To detect the target plane in the camera scene, the algorithm must identify structure lines to create planes. The line extraction method is adopted from [41: Lee et al. 2014], it will generate the endpoints of detected line segments. However, the detected line segments may be affected by the light or camera resolution, causing one structure line to be split into multiple line segments. Therefore, the line segments belonging to the same structure line will be merged in this process. An example of the process is shown in Figure 4.4.

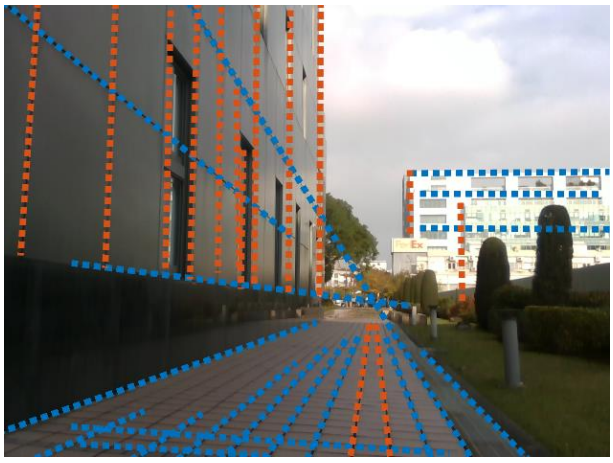
Let L be the set of all detected line segments, defined as Equation (4.1), where line segment l_n contains the endpoints position p_n^1, p_n^2 in image frame, parameters of the line equation r_n, θ_n introduced in Equation (3.5).

$$L = \{l_1, l_2, \dots, l_n\}$$
$$l_n = (p_n^1, p_n^2, r_n, \theta_n)$$
$$p_n^i = (u_n^i, v_n^i) \quad (4.1)$$

doi:10.6342/NTU202402036

(a) The imput image

(b) The output of line segments detector

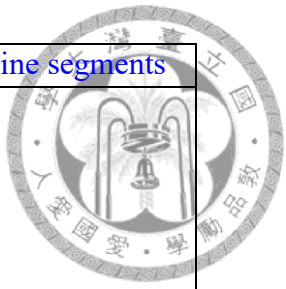


(c) Line merging result

Figure 4.4: The process of detection and merging line segments

First, as shown in Algorithm 4.1, remove line segments with a length less than a threshold T_l, and allocate weights by replicating elements according to a threshold T_w to generate L_w.

doi:10.6342/NTU202402036



Algorithm 4.1: Remove invalid line segments and add weights to valid line segments

Input: L, T_l, T_w

Output: L_w

$L_r = \{l_i \in L \mid \text{dist}(p_i^1, p_i^2) > T_l\}$

$L_d \leftarrow \emptyset$

for all $l_i \in L_r$:

  $w = \left\lfloor \frac{\text{dist}(p_i^1, p_i^2)}{T_w} \right\rfloor$

  while $w > 0$:

    $w = w - 1$

    $L_d \leftarrow add\_element(l_i)$

  endwhile

  $L_w \leftarrow merge\_set(L_r, L_d)$

endfor

Then, as shown in Algorithm 4.2, apply the DBSCAN algorithm using the 2-D position (r, θ) of the line segments in L_w, For each cluster S_j, merge the line segments into one structural line segment, the parameters r, θ of the merged line segment are the average value of elements in S_j. Project all the endpoints of the line segments onto the merged line, then the outer boundary of the projected endpoints will be the endpoints of the merged line segment, as shown in Figure 4.5. Lastly, the merged line segments are divided into horizontal lines L_h and vertical lines L_v by a predefined threshold T_θ.

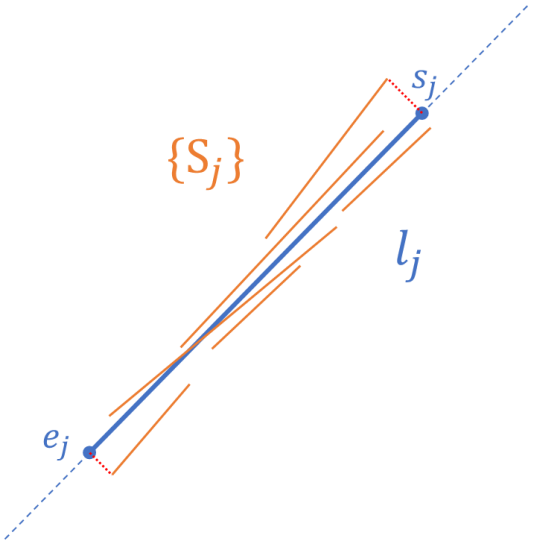
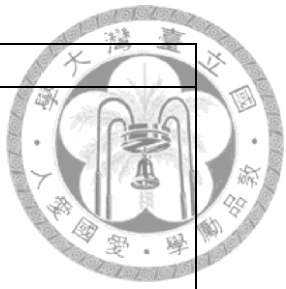


Figure 4.5: Schematic of merging line segments within a cluster.

doi:10.6342/NTU202402036



Algorithm 4.2: merge line segments of each cluster

Input: L_w, T_θ

Output: L_h, L_v

$S = \{S_1, S_2, \dots, S_j\} \leftarrow DBSCAN(L_w)$

$L_h \leftarrow \emptyset, L_v \leftarrow \emptyset$

for each $S_j \in S$:

  $r_j = mean(\forall r \in l \subseteq S_j)$

  $\theta_j = mean(\forall \theta \in l \subseteq S_j)$

  $P \leftarrow \emptyset$

  for all endpoints $p_i(u_i, v_i)$ in $l \subseteq S_j$:

    $r' = u_i sin\theta_j - v_i cos\theta_j$

    $A = \begin{bmatrix} \cos\theta_j & sin\theta_j \\ \sin\theta_j & cos\theta_j \end{bmatrix}$

    $b = \begin{bmatrix} r_j \\ r' \end{bmatrix}$

    Let $p^T = A^{-1}b$

    $P \leftarrow add\_element(p)$

  endfor

  $(p_j^1, p_j^2) = (p_1, p_2) \in P | \text{dist}(p_1, p_2)$ is maximum

  $l_j = (p_j^1, p_j^2, r_j, \theta_j)$

  if $\theta_j > T_\theta$:

    $L_h \leftarrow add\_element(l_j)$

  else:

    $L_v \leftarrow add\_element(l_j)$

  endif

endfor

The reason for duplicating the line segments as weights is that DBSCAN clusters data based on their density, if a structure line is perfectly detected, DBSCAN will treat it as an outlier because the density is not high enough. Therefore, duplicating the line segments based on their length will increase the density and allow the data to be processed in a standard DBSCAN without modifying the algorithm.

doi:10.6342/NTU202402036

4.3 Extract Line Features from Grid Mesh Plane

After generate the horizontal lines L_h and vertical lines L_v in Section 4.2, the next step is to identify which lines belong to the target plane and estimate the rotation between the target plane and the camera frame.

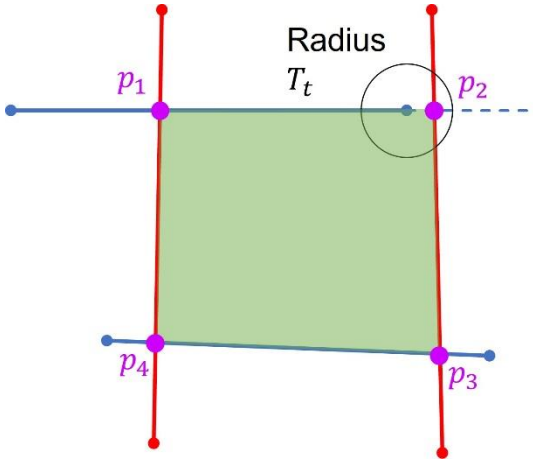
The extraction process is split into two parts, estimating the relative depth ratio between lines (Section 4.3.1) and identifying the lines on the same plane to create the target plane (Section 4.3.2).

4.3.1 Depth Estimation with Rectangular Constraint

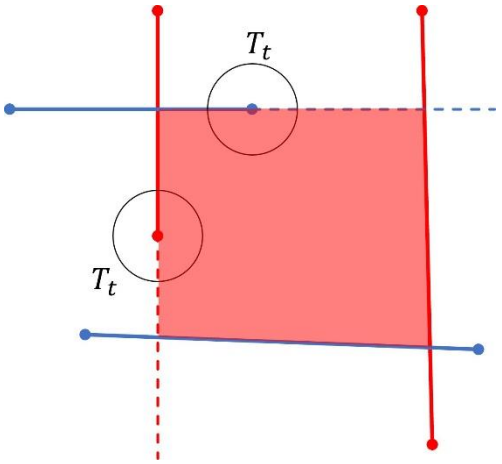
Since the scenario of this thesis is grid-like planes, the property of rectangular in 3-D space can be used to calculate the relative depth of corner points composed of vertical and horizontal line segments.

First, choose two vertical lines and two horizontal lines from L_h and L_v, if they can create a convex quadrilateral within a tolerance threshold T_t (Figure 4.6) and the shortest edge length is larger than a threshold T_e to ensure the noise cause by pixel error is small, the relative depth ratios of corner points can be determined by assuming the quadrilateral is the perspective projection of a rectangle in 3-D space as shown in Figure 4.7. Using the property that the opposite sides are parallel and have same length, and using the vector of adjacent sides along with their cross vector as the rotation matrix of the rectangle with respect to the camera frame, as shown in Algorithm 4.3.

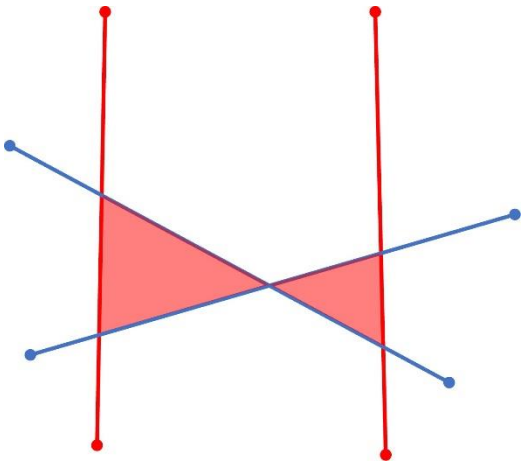
doi:10.6342/NTU202402036



(a) A valid quadrilateral with actual cross points or distance from intersection to endpoint within T_t.



(b) An invalid quadrilateral which distance from intersection to endpoint greater than T_t.



(c) An invalid quadrilateral that isn't convex quadrilateral.

Figure 4.6: Examples of composing a quadrilateral from line segments.

doi:10.6342/NTU202402036

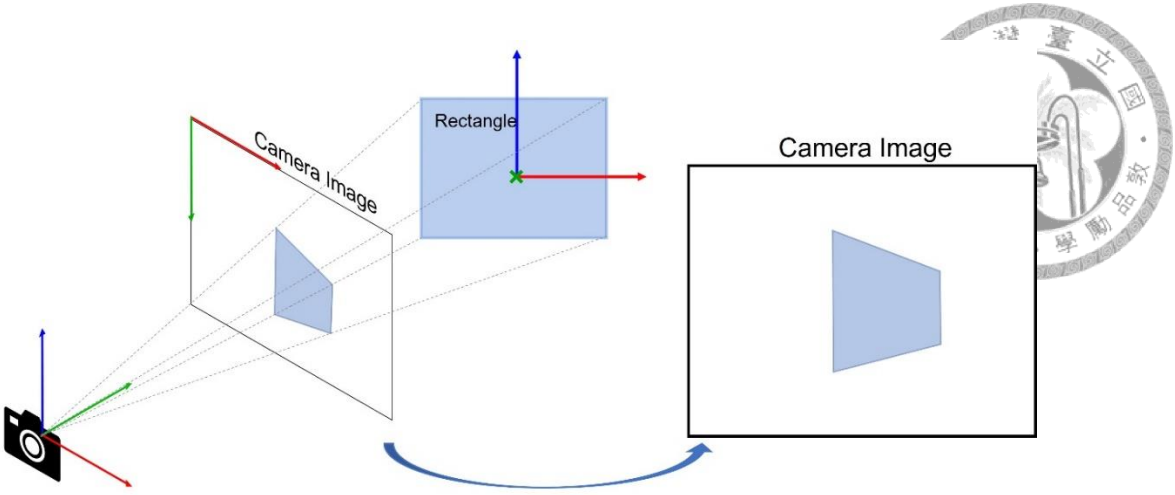


Figure 4.7: The rectangle in the camera image as seen through perspective projection

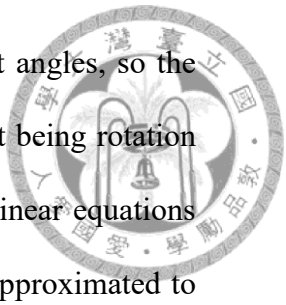
Algorithm 4.3: Estimating rotation matrix of a quadrilateral.

Input: $p_1, p_2, p_3, p_4, f, c_x, c_y$

Output: R'_q

1. $p_1(u_1, v_1), p_2(u_2, v_2), p_3(u_3, v_3), p_4(u_4, v_4) \leftarrow$ the corner points of a convex quadrilateral in image frame, $p_i \in \mathbb{R}^{2\times1}$
2. Let $P_1(X_1, Y_1, Z_1), P_2(X_2, Y_2, Z_2), P_3(X_3, Y_3, Z_3), P_4(X_4, Y_4, Z_4)$ be the 3-D positions of p_1, p_2, p_3, p_4 in camera frame, $P_i \in \mathbb{R}^{3\times1}$
3. Assume $\overrightarrow{P_1P_4}$ and $\overrightarrow{P_2P_3}$ are same vectors:
   $$\begin{bmatrix} X_4 - X_1 \\ Y_4 - Y_1 \\ Z_4 - Z_1 \end{bmatrix} = \begin{bmatrix} X_3 - X_2 \\ Y_3 - Y_2 \\ Z_3 - Z_2 \end{bmatrix}$$
4. multiply by focal length f:
   $$\begin{bmatrix} fX_4 - fX_1 \\ Y_4 - Y_1 \\ fZ_4 - fZ_1 \end{bmatrix} = \begin{bmatrix} fX_3 - fX_2 \\ Y_3 - Y_2 \\ fZ_3 - fZ_2 \end{bmatrix}$$
5. with Equation (3.1) and Equation (3.2) of camera model, obtain:
   $$\begin{bmatrix} u_4Y_4 - u_1Y_1 \\ Y_4 - Y_1 \\ v_4Y_4 - v_1Y_1 \end{bmatrix} = \begin{bmatrix} u_3Y_3 - u_2Y_2 \\ Y_3 - Y_2 \\ v_3Y_3 - v_2Y_2 \end{bmatrix}$$
6. Let $Y_1 = 1$, solve the relative depth Y_2, Y_3, Y_4 respect to Y_1 as a linear equation below:
   $$\begin{bmatrix} u_2 & -u_3 & u_4 \\ v_2 & -v_3 & v_4 \\ 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} Y_2 \\ Y_3 \\ Y_4 \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}$$
7. The 3-D points $P_i = (X_i, Y_i, Z_i)^T = \left(\frac{(u_i - c_x)Y_i}{f}, Y_i, \frac{(c_y - v_i)Y_i}{f}\right)^T$
8. $V_X = \frac{P_3 - P_4}{|P_3 - P_4|}$
9. $V_Z = \frac{P_1 - P_4}{|P_1 - P_4|}$
10. $V_Y = V_Z \times V_X$
11. $R'_q = [V_X \quad V_Y \quad V_Z]$

doi:10.6342/NTU202402036

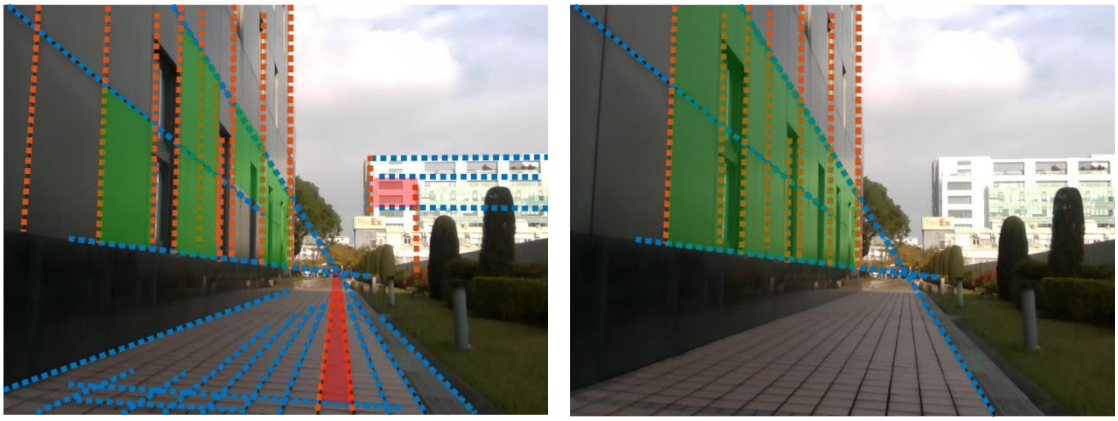


However, Algorithm 4.3 does not check if all four angles are right angles, so the constraint is suitable for a parallelogram, resulting the output matrix not being rotation matrix. If the right angles constraint is added, it cannot be solvable as linear equations and increase the computational complexity. In this thesis, the result is approximated to rotation matrix R_q using singular value decomposition, as shown in Equation (4.2), but a drawback of the approximation is that the algorithm cannot determine the camera scene does not adhere the grid-like assumption.

$$U\Sigma V^T = SVD(R_q{}')$$
$$R_q = UV^T \tag{4.2}$$

4.3.2 Target Plane Detection and Line Features Extraction

After calculating the rotation matrices of all the possible quadrilaterals, the next step is to identify which quadrilaterals are the rectangles lying on the same plane in 3-D space and extract the position of the lines on the plane. an example is shown in Figure 4.8, note that the quadrilaterals in the figure are just for indication, not complete results.



(a) Clustering quadrilaterals (b) Identified plane and valid features

Figure 4.8: An example of plane detection and features extraction

Let $R = \{R_1, R_2, \dots, R_n\}$ be the set contains all possible rotation matrices of quadrilaterals obtained from Section 4.3.1, clustering them using DBSCAN, and the

doi:10.6342/NTU202402036

distance function of calculate the angle θ_{ij} between rotation matrices R_i, R_j is shown in Equation (4.3).

$$|\theta_{ij}| = \cos^{-1}\frac{tr(R_i Rj^T) - 1}{2} \tag{4.3}$$

After finishing clustering, select the largest cluster as the candidate target plane, calculate the rotation matrix R_t of the target plane respect to the camera frame.

Let $C = \{R_1, R_2, \dots, R_k\}$ be the set of the largest cluster

$$USV^T = SVD(\Sigma R_k)$$

$$R_t = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = UV^T \tag{4.4}$$

Let a point $P_b(X_b, Y_b, Z_b)$ be the reference point on the target plane in 3-D space in camera frame, obtain the plane equation with P_b and R_t (Equation (4.5)). This thesis selects one of the endpoints belongs to a line segment in the cluster C as reference point.

$$Ax + By + Cz + D = 0$$

Where:

$$\begin{aligned} A &= r_{21}r_{33} - r_{31}r_{23} \\ B &= r_{13}r_{31} - r_{11}r_{33} \\ C &= r_{11}r_{23} - r_{13}r_{21} \\ D &= -(AX_b + BY_b + CZ_b) \end{aligned} \tag{4.5}$$

Determine the 3-D positions ${}^cP({}^cX, {}^cY, {}^cZ)$ of each line's endpoints (u, v) in camera frame by solving Equation (4.6), transform the endpoints from camera frame to plane frame using Equation (4.7) with R_t from Equation (4.4), the target plane is the X-Z plane in world frame and the Y positions of all endpoints are equal the distance from the target plane to the camera, this distance lacks the scale factor to transform the position to real-world scale during initialization. In experiments, the scale factor is given by user through measurement.

doi:10.6342/NTU202402036

$$\begin{cases} A^{C}X + B^{C}Y + C^{C}Z + D = 0 \text{ (plane equation)} \\ u - c_x = f\dfrac{^{C}X}{^{C}Y} \\ c_y - v = f\dfrac{^{C}Z}{^{C}Y} \text{ (pinhole camera model)} \end{cases} \tag{4.6}$$

$$^{P}P = R_t^T\, ^{C}P \tag{4.7}$$

Calculate line equation parameters r, θ with the endpoints P^P on X-Z plane in plane frame using Equation (3.5). Then approximate the parameters to horizontal (X-axis) and vertical (Z-axis) lines by ignore parameters θ of the lines, the parameters r will be the z and x positions of horizontal and vertical lines to obtain features F_h, F_v. And d_t is the distance between target plane and camera on Y axis in plane frame ,which can be obtained by the y position of endpoints P^P in plane frame. A schematic diagram is shown in Figure 4.9.

$$F_h = \{z_1, z_2, \dots, z_i\}$$

$$F_v = \{x_1, x_2, \dots, x_j\}$$

d_t is the minimum distance between target plane and camera center. (4.8)

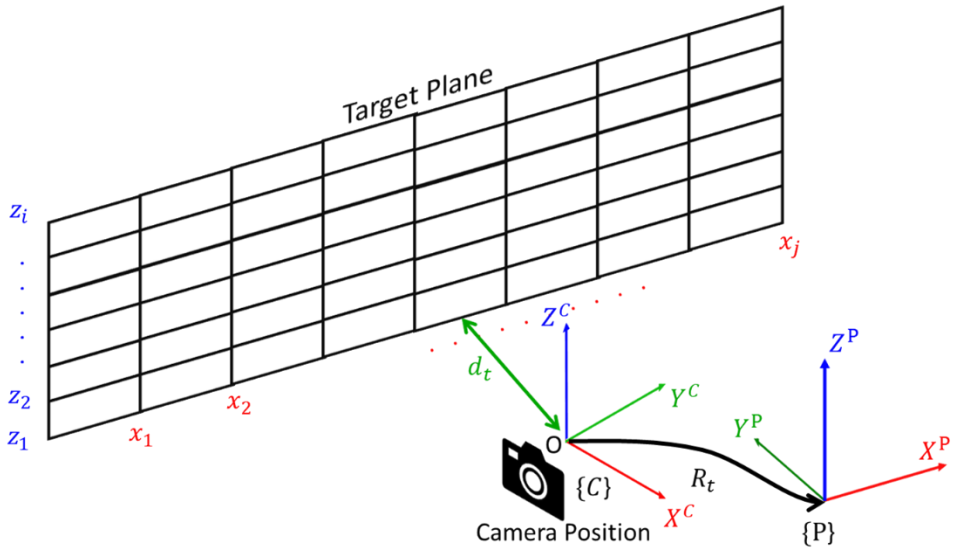


Figure 4.9: A schematic diagram about parameterize target plane

doi:10.6342/NTU202402036

4.4 Plane Tracking with RANSAC

In this section, the motion estimation is divided to two steps: feature matching rules (Section 4.4.1) and motion estimation with RANSAC (Section 4.4.2).

4.4.1 Feature Matching

After finishing the feature extraction in Section 4.3, the first step is to verify whether the detected plane and the stored target plane are the same plane. This can be done by comparing the angle between rotation matrices with Equation (4.3). If the angle is less than a threshold T_p, treat them as same plane then matching the features with nearest distance; if the angle is greater than T_p, identified them as different planes and skip current estimation and wait for next estimation.

F_h, F_v are the features of detected plane, let F_{h-1}, F_{v-1} are the features of stored target plane from previous estimation. The matching pairs M_h, M_v can be obtained by Equation (4.9).

$$\begin{aligned} M_h &= \{(z_1, z_{1-1}), (z_2, z_{2-1}), \dots, (z_i, z_{i-1})\} \\ s.t.\ \ & z_i \in F_h, z_{i-1} \in F_{h-1}, \text{dist}(z_i, z_{i-1}) \text{ is minimun} \end{aligned}$$

$$\begin{aligned} M_v &= \{(x_1, x_{1-1}), (x_2, x_{2-1}), \dots, (x_j, x_{j-1})\} \\ s.t.\ \ & x_j \in F_v, x_{j-1} \in F_{v-1}, \text{dist}(x_j, x_{j-1}) \text{ is minimun} \end{aligned} \tag{4.9}$$

4.4.2 Motion Estimation with RANSAC

The motion estimation is optimized using least squares method. It estimates the motion between frames by minimizing the square of the distance of matching pairs. These matching pairs are sampled and estimated iteratively with RANSAC to eliminate false matches.

The cost function can be written as Equation (4.10), s is the scale factor; v, h are the shift of features in X, Z axes in plane frame.

doi:10.6342/NTU202402036

$$\min_{s,h,v} e = \sum_i \sum_j (sz_i - v - z_{i-1})^2 + \left(sx_j - h - x_{j-1}\right)^2$$

$s.t.$ $z_i, z_{i-1} \in M'_h; x_j, x_{j-1} \in M'_v$; M'_h, M'_v are the sampled set in RANSAC process with select i, j elements from M_h, M_v (4.10)

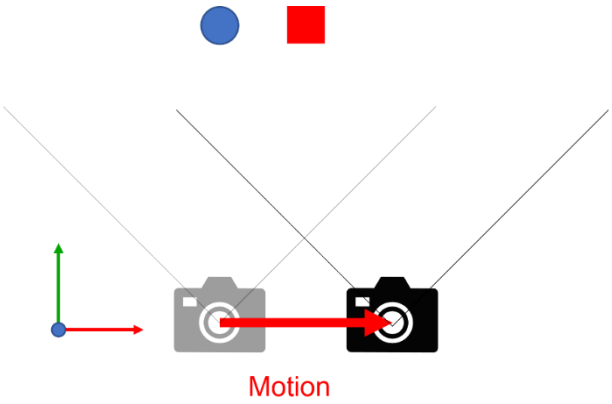
The motion in plane frame can be determined by Equation (4.11), d_{t-1} is the distance to the target plane from previous estimation. The current camera orientation R_c and position p_c in the world frame can be determined by R_t in Equation (4.4) accumulating motion in Equation (4.12).

$$t_c = -(v, sd_t - d_{t-1}, h) \tag{4.11}$$

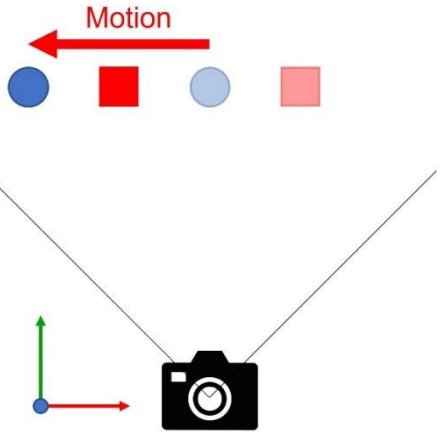
$$R_c = R_t^T$$
$$p_c = p_{c-1} + t_c$$
Where p_{c-1} is the previous position (4.12)

The reason the camera motion is opposite to the estimated is because the cost function is designed to estimate feature motion with a static camera, and the camera image is as same as a moving camera and static features, but in the opposite direction of estimation. A schematic is shown in Figure 4.10.

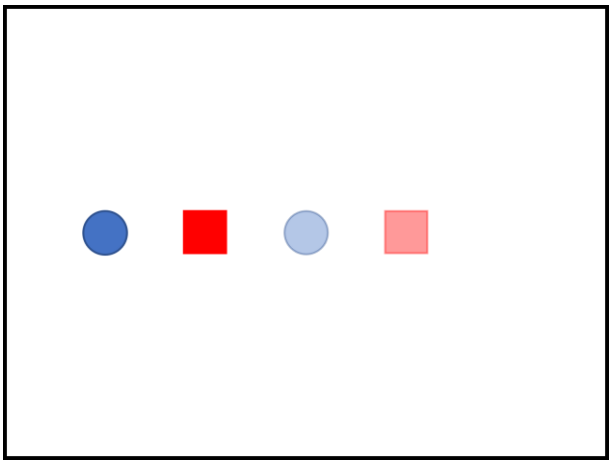
doi:10.6342/NTU202402036



(a) Camera moving right



(b) Features moving left



(c) Camera image

Figure 4.10: The scenarios (a) and (b) are having same camera image (c), and the camera and features motion are in opposite directions.

doi:10.6342/NTU202402036

4.5 Update States

In this section, the estimated result will run a reliability testing(Section 4.5.1), then update the stored features if the estimation is reliable, includes update the existing features (Section 4.5.2), add new features (Section 4.5.3) and eliminate invalid features (Section 4.5.4).

4.5.1 Reliability Testing

After calculating the orientation and position of the camera with respect to the world frame, it is necessary to verify the reliability of the estimation results as it does not account for noise from the camera, such as distortion from camera resolution or motion blur.

In the reliability testing, the stored features at previous position will be reprojected onto the image frame at the current estimated position. A schematic diagram is shown in Figure 4.11, the reprojected line l_c' does not overlap with observation l_c because of estimation error. To determine the distance between feature lines, first, represent the lines in Hough space (r, θ) which is also a 2-D polar coordinate system as shown in Figure 4.12. Define the distance between lines as Equation (4.13), if the average distance between reprojected lines l_c' and current observations l_c is greater than the average distance between previous observations l_{c-1} and current observations l_c, treat this estimate as an unreliable one, skip current estimated result and wait for next camera image; on the country, if the average distance between reprojections l_c' and observations l_c is less than average distance across observations l_{c-1} and l_c.

After reliability testing, unreliable estimates will be discarded, and the process can be seen as downsampling data to improve the algorithm speed. Only features with reliable estimates will be updated follow Section 4.5.2, Section 4.5.3 and Section 4.5.4. This process discards unreliable estimates without optimization, it has a drawback that when

doi:10.6342/NTU202402036

the camera frames lost the target plane, the algorithm will pause the estimation until the camera frame find back the target plane.

$$D = \sqrt{r_1^2 + r_2^2 - 2r_1 r_2 \cos(\theta_1 - \theta_2)} \tag{4.13}$$

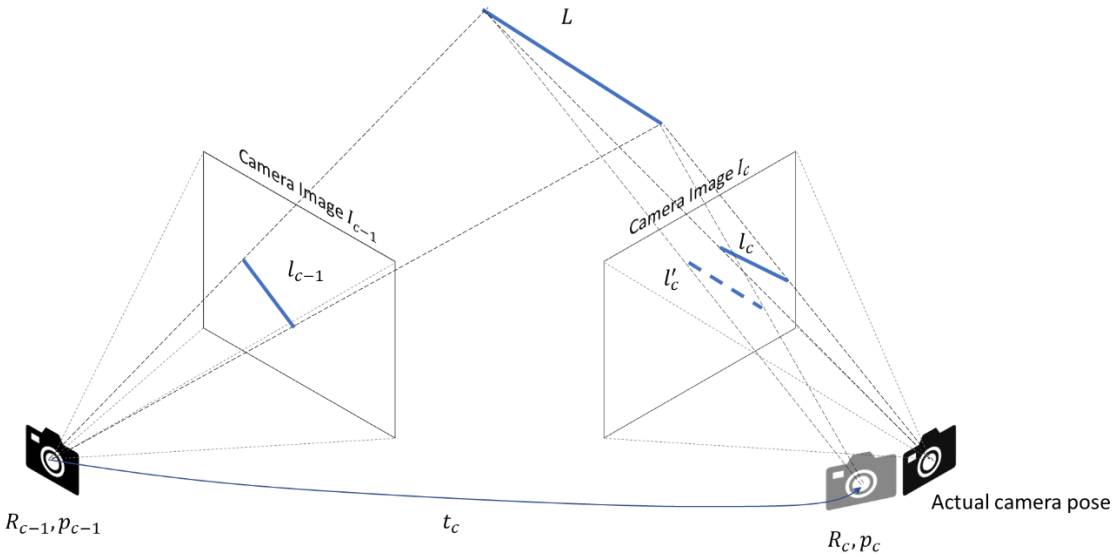


Figure 4.11: A schematic diagram of line reprojection

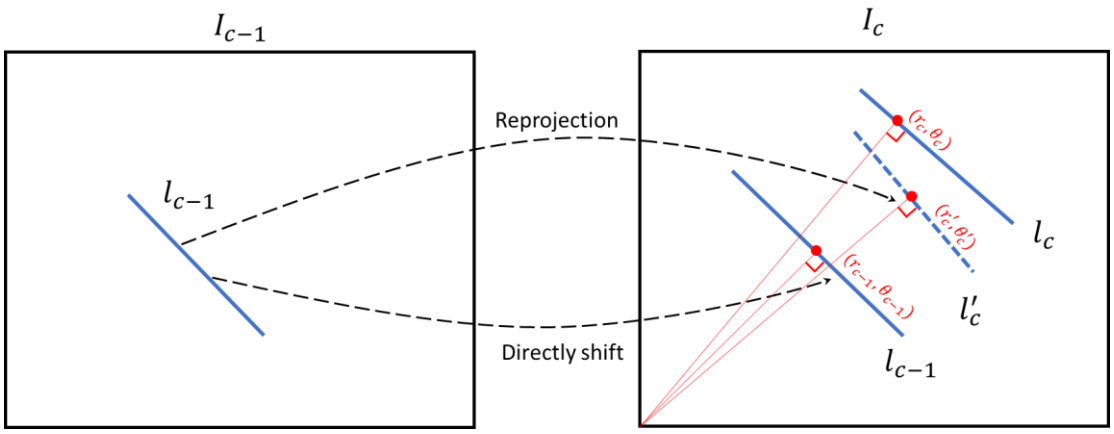


Figure 4.12: Represent lines with Hough space (r, θ)

4.5.2 Update Existing Features

The matched features after estimation with RANSAC will update their positions. For each matched pair, the feature is updated using exponential moving average, as shown in Equation (4.14), where z_t, x_t and z_{t-1}, x_{t-1} are the current and previous feature

doi:10.6342/NTU202402036

position in a matched pair, they will align with the current position before averaging. This moving average helps in noise reduction and provides smoother data.

$$\begin{cases} z_t \leftarrow 0.5sz_t + 0.5(z_{t-1} - v) \\ x_t \leftarrow 0.5sx_t + 0.5(x_{t-1} - h) \end{cases} \tag{4.14}$$

And the distance between camera and the target plane is updated by the scale factor s from estimation to align the actual scale to plane frame.

$$d_t \leftarrow sd_t \tag{4.15}$$

4.5.3 Add New Features

For the new features that belong to the target plane and do not match to the existing features will be added to them. To prevent false features from being considered inliers in RANSAC, the new features must have a minimum distance to the existing features greater than a threshold T_n. This strategy helps keep the stored features stay sparse and maintain calculation speed.

4.5.4 Remove Invalid Features

For each stored feature, the algorithm has stored the matching success rate W, which is the ratio about number of successful matching and features shown in camera FOV in the past n frames as shown in Equation (4.16).

$$W = \frac{\text{number of successful matching in past } n \text{ frames}}{\text{number of shown in FOV in past } n \text{ frames}} \tag{4.16}$$

If the W of a feature is less than a threshold T_r, the feature will be identified as invalid and unable to provide reliable information, and will be removed from stored features.

doi:10.6342/NTU202402036

Chapter 5
Synthetic and Real-World Experiments

In this chapter, the experiments based on the propose algorithm are discussed. the experiments in simulation and real-world are evaluated. In Section 5.1, the experiments in simulation are verified for the accuracy in noisy environments and the performance with adding reliability testing. The real-world experiments performed with real-time computation are detailed in Section 5.2.

5.1 Evaluate Performance in Simulation

In this section, the construction of synthetic scenes and designed trajectories are introduced in Section 5.1.1. Method to measure performance of the propose algorithm is discussed in Section 5.1.2. The estimate results of simulations are present in Section 5.1.3. A concise summary of the simulations is presented in Section 5.1.4.

5.1.1 Synthetic Scenes

To evaluate the performance of the propose algorithm, a simple synthetic scene involves a 29.9×3 m grid mesh plane with 24 vertical lines and 7 horizontal lines, parallel to the X-Z plane of the world frame, as shown in Figure 5.1. In noisy scenes, some noise line segments are randomly positioned with varying lengths, creating different levels of noise based on the number and length of line segments. All the noisy synthetic scenes are shown in Table 5.1, the columns of the table represent different length ranges of noise line segments, and the rows of the table represent varying numbers of noise line segments.

A virtual perspective camera moves in the synthetic scenes and generates the sequential images for the algorithm input, the virtual camera has 90×74° FOV(H×V),

doi:10.6342/NTU202402036

640×480 pixels resolution without any distortion and motion blur. An example of camera imaging in the scene is shown in Figure 5.2.

For the length of noise line segments, in cases with the same number of noise line segments, the length of line segments can be categorized into three types. The centers of the line segments are consistent, and the length of the segments is double that of the previous type. The camera imaging of noise line segments with the same number but different lengths is shown in Figure 5.3.

For the number of line segments, there is a noise set containing 50 noise line segments, in the cases from no noise to 50 noise line segments, it retains the previously selected noise line segments and adds 10 more noise line segments selected from the noise set. The camera imaging of noise line segments of the same length but with different numbers is shown in Figure 5.4.

For each synthetic scene in the simulations, the virtual camera moves along the grid mesh plane following various trajectories in X-Y plane, and the grid mesh plane is captured in all the camera frames of the sequential images. There are four trajectories for the simulations: Straight, Straight and U-turn, Wave, Wave and U-turn.

Straight trajectory

In the Straight trajectory, the virtual camera first stops at origin and facing the X-axis direction for 30 frames and then moves forward 15 meters along X-axis for 500 frames, resulting in a total of 530 sequential images.

Straight and U-turn trajectory

In the Straight and U-turn trajectory, the virtual camera first stops at origin and facing the X-axis direction for 30 frames and then moves forward 15 meters along X-axis for 500 frames, then turn around along a counterclockwise circular path with

doi:10.6342/NTU202402036

a radius of 0.125 meters for 200 frames, and continues forward for 15 meters in the negative X-direction for 500 frames, resulting in a total of 1230 sequential images.

Wave trajectory

In the Wave trajectory, the virtual camera first stops at origin and facing the X-axis direction for 30 frames and then the yaw angle oscillates between plus and minus 10 degrees as it moves forward, with a total distance of 15 meters for 500 frames, resulting in a total of 530 sequential images.

Wave and U-turn trajectory

In the Wave and U-turn trajectory, the virtual camera first stops at origin and facing the X-axis direction for 30 frames and then the yaw angle oscillates between plus and minus 10 degrees as it moves forward, with a total distance of 15 meters for 500 frames, then turn around along a counterclockwise circular path with a radius of 0.125 meters for 200 frames, and continues forward with the yaw angle oscillates between plus and minus 10 degrees for a total distance of 15 meters in 500 frames, resulting in a total of 530 sequential images.

The camera pose for each trajectory is detailed in Table 5.2 and Figure 5.5.

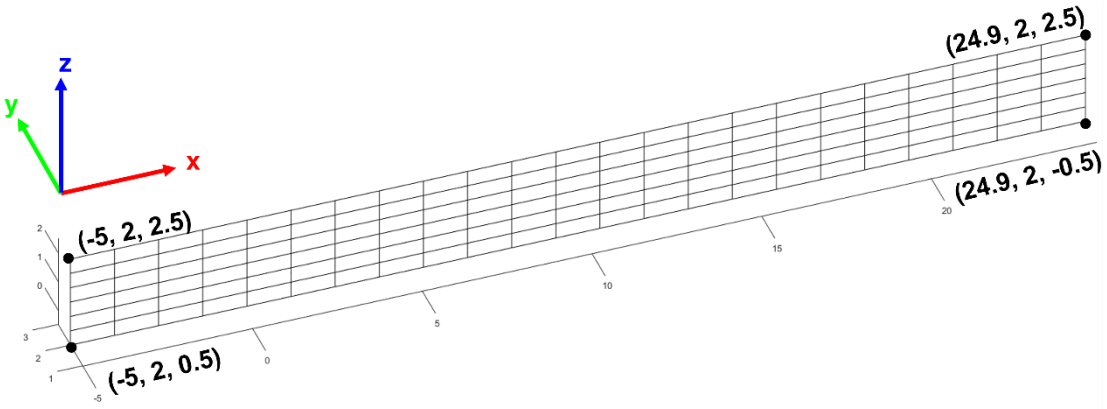
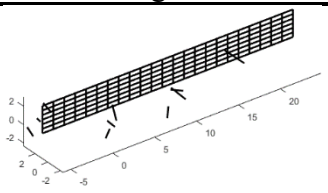
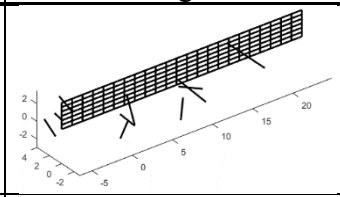
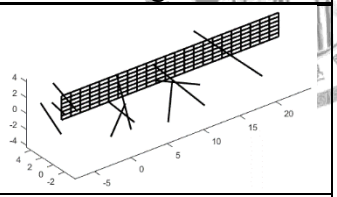
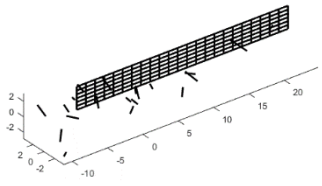
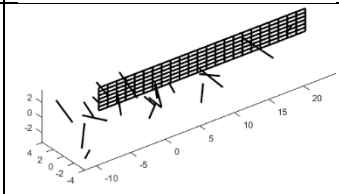
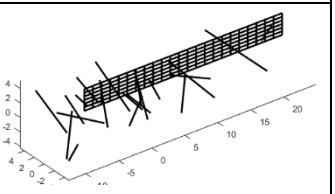
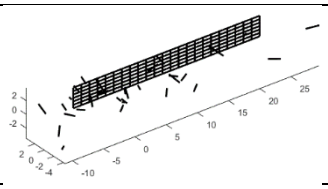
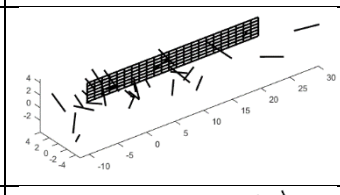
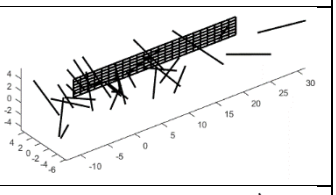
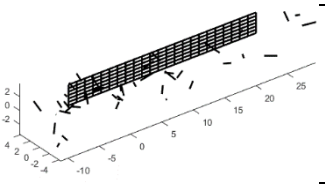
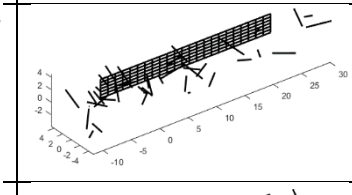
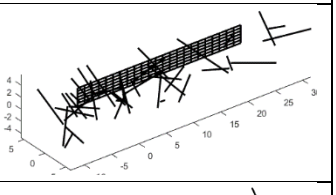
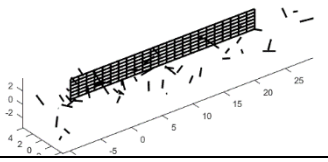
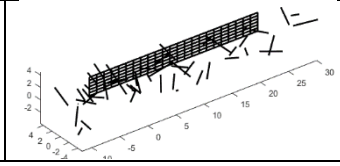
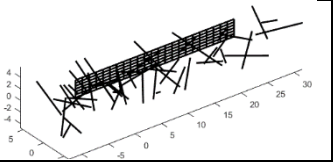
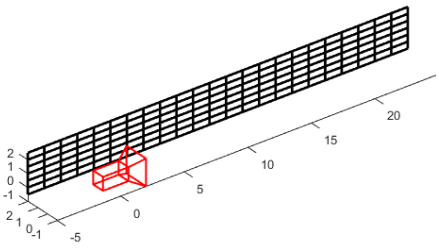


Figure 5.1: The synthetic scene with a grid mesh plane

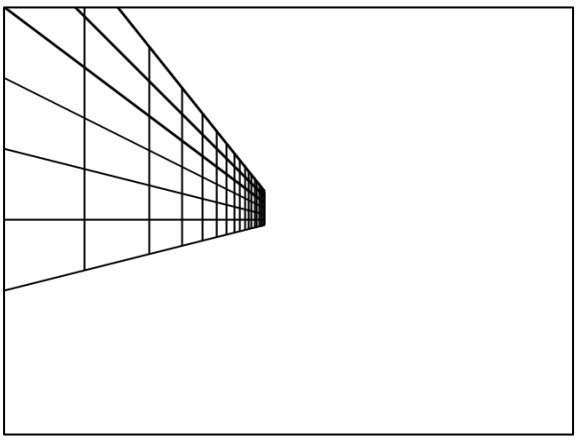
doi:10.6342/NTU202402036

Table 5.1: Overview of the noisy scenes

	A 0m < Length < 2.5 m	B 0m < Length < 5 m	C 0m < Length < 10 m
10			
20			
30			
40			
50			



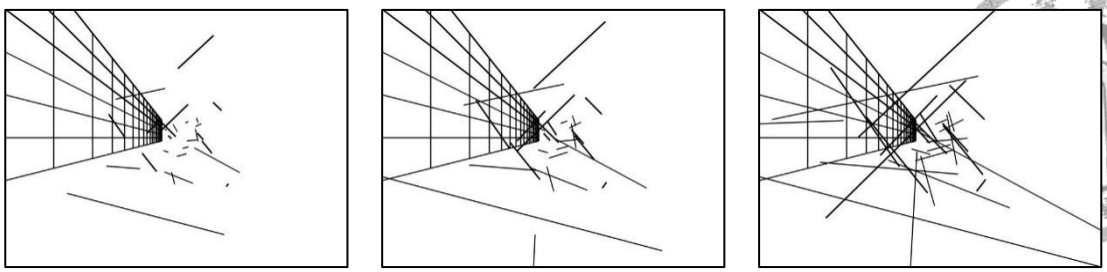
(a) Camera at origin and the optical axis overlaps with the x-axis.



(b) Camera imaging at origin

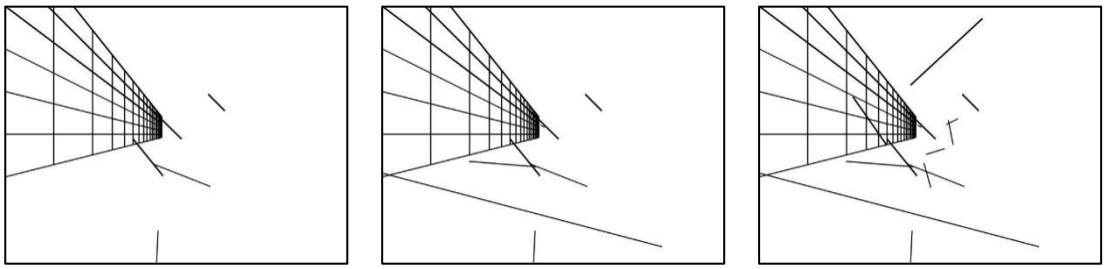
Figure 5.2: The virtual camera imaging in the synthetic scene

doi:10.6342/NTU202402036

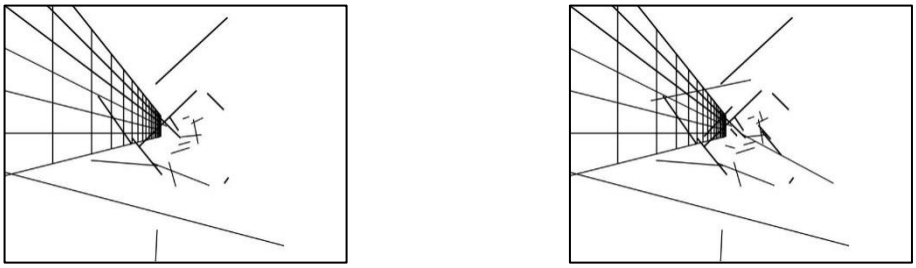


(a) Length A (b) Length B (c) Length C

Figure 5.3: Different lengths of noise line segments in 50 noise line scenes



(a) 10 line segments (b) 20 line segments (c) 30 line segments



(d) 40 line segments (e) 50 line segments

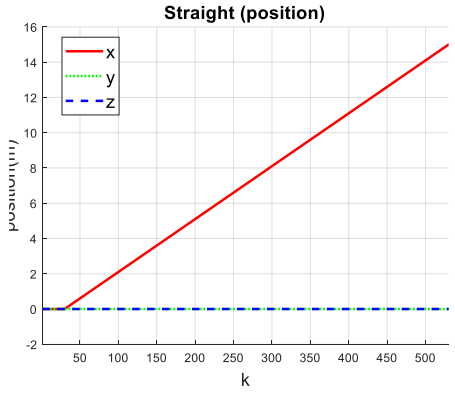
Figure 5.4: Different number of noise line segments in Length B scenes

doi:10.6342/NTU202402036

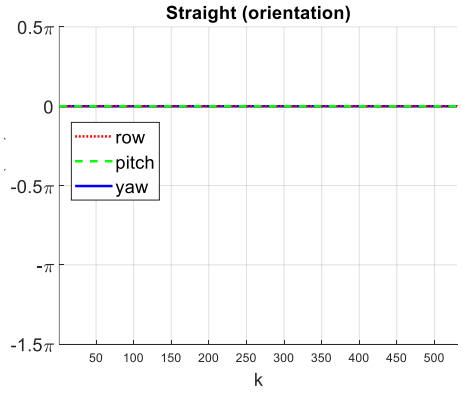
Table 5.2: 6-DOF camera poses in designed trajectories

Frame count k	Straight	Straight and U-turn	Wave and U-turn	Wave
[1,30] Stop	$x[k] = 0$ $y[k] = 0$ $z[k] = 0$ $u[k] = 0$ $v[k] = 0$ $w[k] = 0$			
[31,530] Forward	$x[k] = x[k-1] + 0.03$ $y[k] = y[k-1]$ $z[k] = z[k-1]$ $u[k] = u[k-1]$ $v[k] = v[k-1]$ $w[k] = w[k-1]$		$x[k] = x[k-1] + 0.03\cos(w[k])$ $y[k] = y[k-1] - 0.03sin(w[k])$ $z[k] = z[k-1]$ $u[k] = u[k-1]$ $v[k] = v[k-1]$ $w[k] = \frac{\pi}{18} sin\left(\frac{2\pi(k-30)}{250}\right)$	
[531,730] U-turn		$x[k] = x[k-1] + \frac{\pi}{8 \times 200} cos(w[k])$ $y[T] = y[k-1] - \frac{\pi}{8 \times 200} sin(w[k])$ $z[k] = z[k-1]$ $u[k] = u[k-1]$ $v[k] = v[k-1]$ $w[k] = w[k-1] - \frac{\pi}{200}$		
[731,1230] Return		$x[k] =$ $x[k-1] - 0.03$ $y[k] = y[k-1]$ $z[k] = z[k-1]$ $u[k] = u[k-1]$ $v[k] = v[k-1]$ $w[k] =$ $w[k-1]$	$x[k] =$ $x[k-1] + 0.03cos(w[k])$ $y[k] =$ $y[k-1] - 0.03sin(w[k])$ $z[k] = z[k-1]$ $u[k] = u[k-1]$ $v[k] = v[k-1]$ $w[k] =$ $-\pi - \frac{\pi}{18} sin\left(\frac{2\pi(k-730)}{250}\right)$	

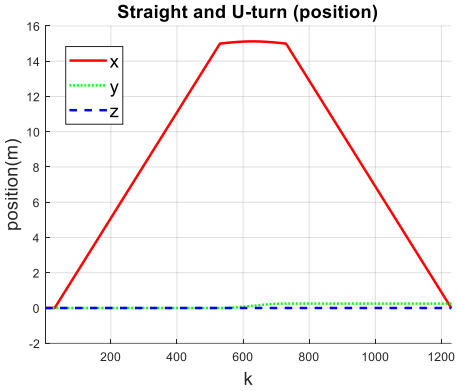
doi:10.6342/NTU202402036



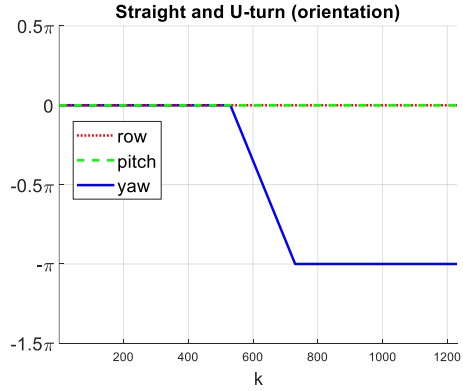
(a) Position of Straight trajectory



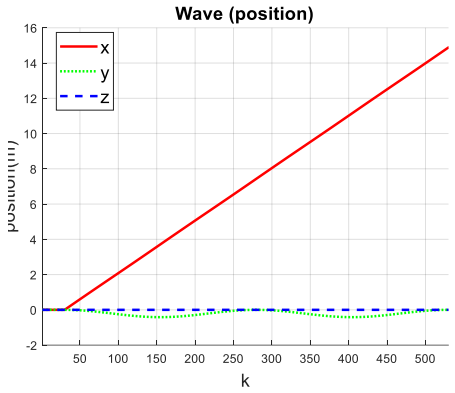
(b) Orientation of Straight trajectory



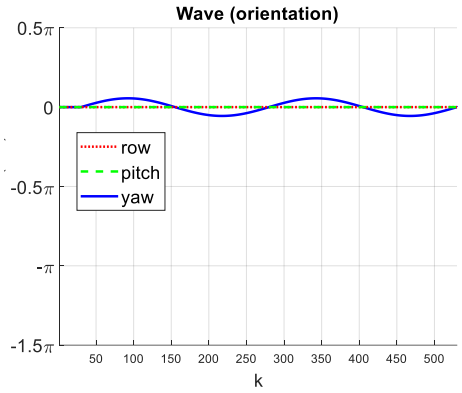
(c) Position of Straight and U-turn trajectory



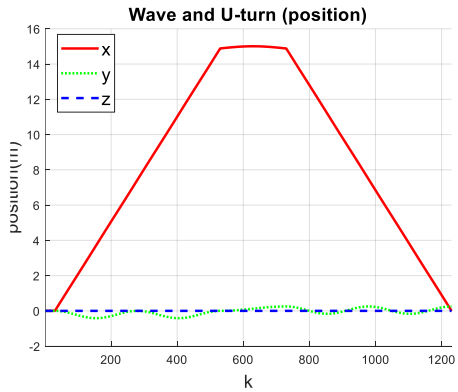
(d) Orientation of Straight and U-turn trajectory



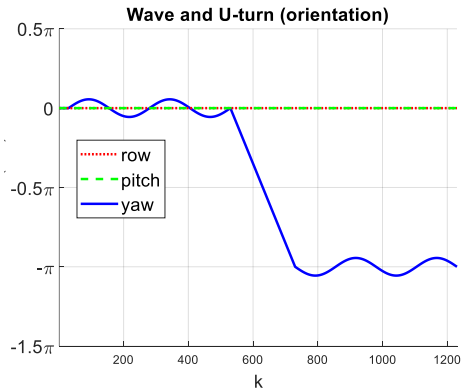
(e) Position of Wave trajectory



(f) Orientation of Wave trajectory



(e) Position of Wave and U-turn trajectory



(f) Orientation of Wave and U-turn trajectory

Figure 5.5: Position and orientation of all trajectories

doi:10.6342/NTU202402036

5.1.2 Evaluate Performance Using Dynamic Time Warping

To evaluate the similarity between estimation and ground truth, dynamic time warping (DTW) is utilized to align and quantify the performance. DTW is a distance measure that compares two time series after optimally aligning them [46: Mueen & Keogh 2016], it can calculate the similarity between two sequences, even if they differ in speed or length. Compare to calculate the Euclidean distance based on time sequence, DTW can provide reasonable performance when the trajectory drifts.

For example, there are three 1-D trajectories shown in Figure 5.6 with following definitions:

$$\begin{aligned} C_1 &= \sin(2\pi k) \\ C_2 &= \cos(2\pi k) \\ C_3 &= 0.5\sin(\pi k) \\ k &= [0, 0.01, \ldots, 0.99, 1] \end{aligned} \tag{5.1}$$

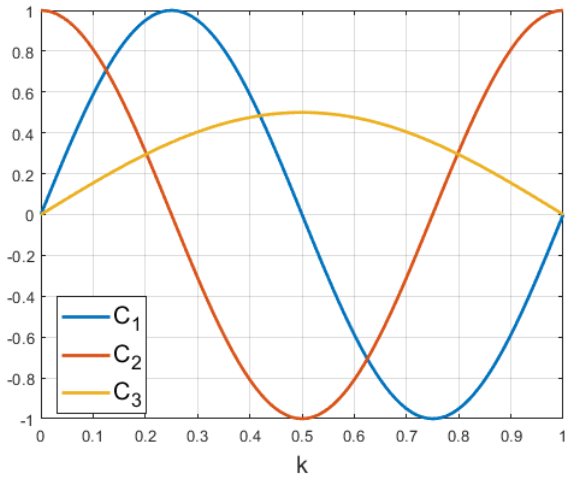


Figure 5.6: 1-D trajectories C_1, C_2, and C_3

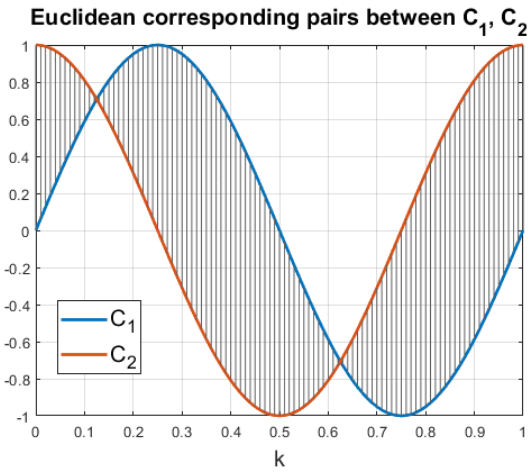
Using Euclidean distance based on time sequence, Figure 5.7 shows the corresponding pairs with gray lines and the squared distance for each corresponding pair between C_1, C_2 and C_1, C_3. The cumulative squared distance indicates that C_3 bears greater resemblance to C_1 than C_2, yielding an erroneous outcome given their trajectory

doi:10.6342/NTU202402036

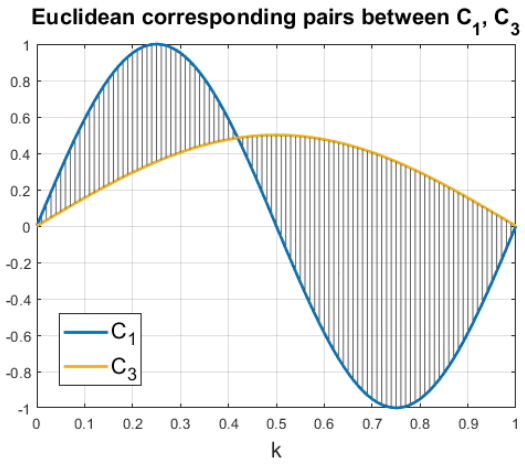
patterns. On the other hand, as in Figure 5.8, the cumulative squared distance based on the corresponding pairs from DTW indicated that C_2 is more similar to C_1 than C_3.

Note that in Figure 5.7 and Figure 5.8, the gray lines represent matched pairs, their length does not reflect the actual distance as the x-axis denotes time.

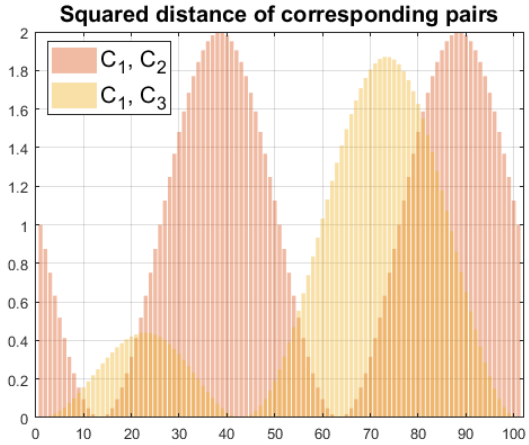
Due to the advantages of the DTW method, which can measure similarities with data of different lengths, it is the best method for evaluating the performance of the proposed algorithm and the effect of adding reliability testing.



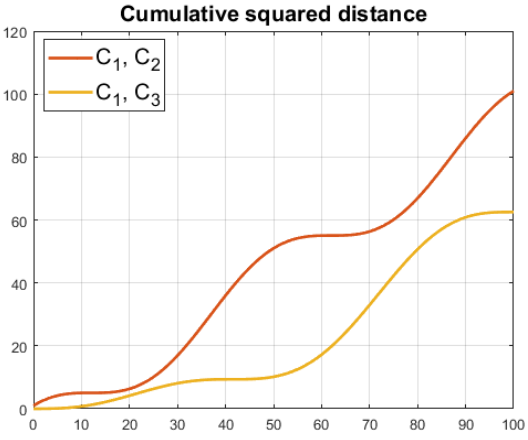
(a) Euclidean corresponding pairs between C_1, C_2



(b) Euclidean corresponding pairs between C_1, C_3



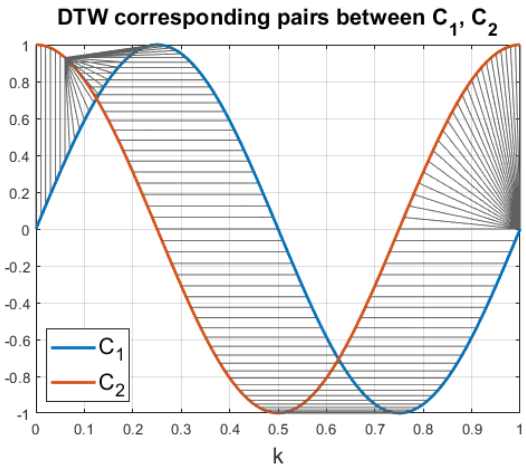
(c) Squared distance of corresponding pairs in C_1, C_2 and C_1, C_3



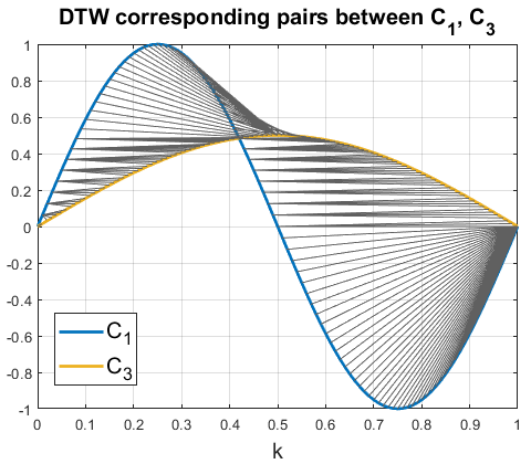
(d) Cumulative squared distance of C_1, C_2 and C_1, C_3

Figure 5.7: The matching result using Euclidean distance based on time sequence

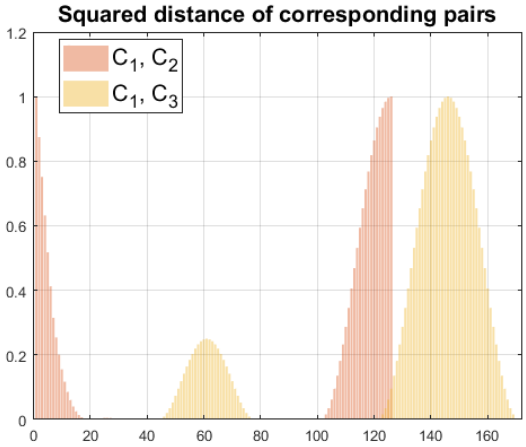
doi:10.6342/NTU202402036



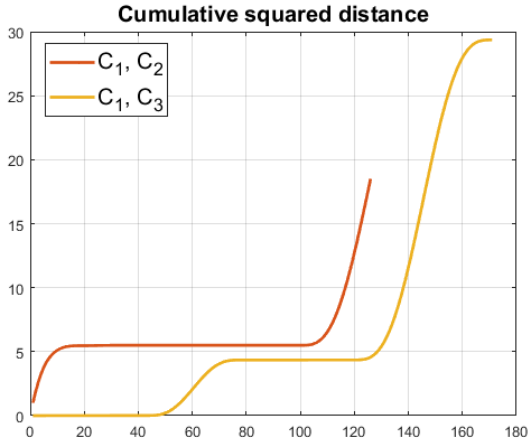
(a) DTW corresponding pairs between C_1, C_2



(b) DTW corresponding pairs between C_1, C_3



(c) Squared distance of corresponding pairs in C_1, C_2 and C_1, C_3



(d) Cumulative squared distance of C_1, C_2 and C_1, C_3

Figure 5.8: The matching result using dynamic time warping method

5.1.3 Estimate Without and With Reliability Testing

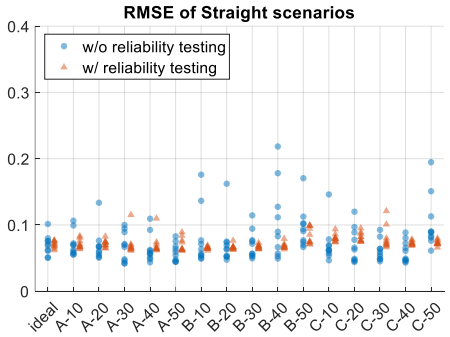
In this section, the synthetic scenarios are executed without and within reliability testing. In the simulations without reliability testing, all input frames were estimated and recorded as trajectory waypoints; on the contrary, in the simulations within reliability testing, the unreliable estimations won't update the stored information as mention in Section 4.5.1.

The following parts of this section compare performance under different levels of noise and the difference with or without reliability testing based on camera trajectories.

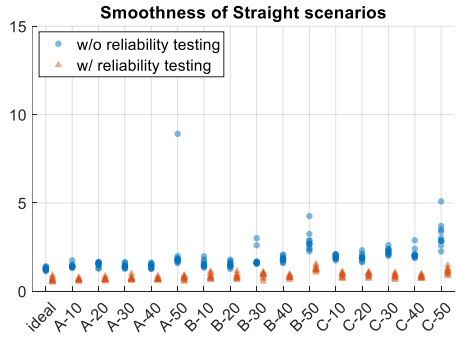
doi:10.6342/NTU202402036

In each scenario, the proposed algorithm is executed 10 times each with and without reliability testing to ensure the consistency of the estimation results. And some figures illustrate the root-mean-square error (RMSE) and the smoothness determined by the integral of the squared second derivative of each estimation, with some instances depicting the contrast between with and without reliability testing.

Figure 5.9 to Figure 5.12 are the RMSE and smoothness of 10 estimations of each trajectory performed in all synthetic scenes in both with and without reliability testing, and the average value of 10 estimations are shown in Table 5.3 to Table 5.5.

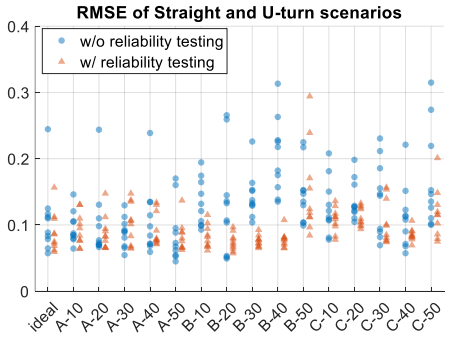


(a) RMSE of estimated results

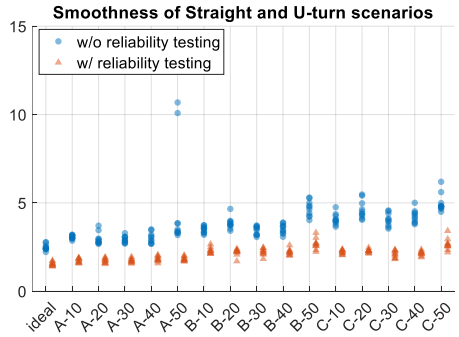


(b) Smoothness of estimated results

Figure 5.9: Estimated results of Straight trajectory in various synthetic scenes



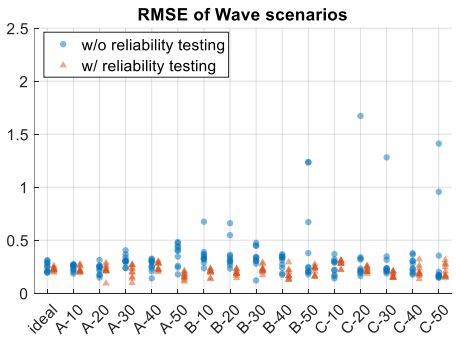
(a) RMSE of estimated results



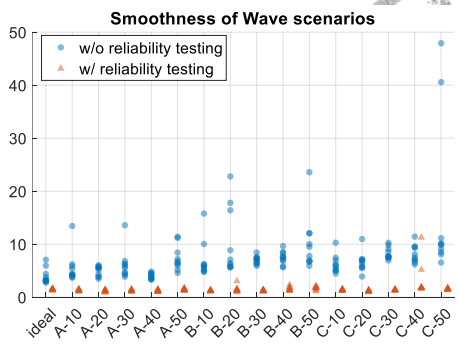
(b) Smoothness of estimated results

Figure 5.10: Estimated results of Straight and U-turn trajectory in various synthetic scenes

doi:10.6342/NTU202402036

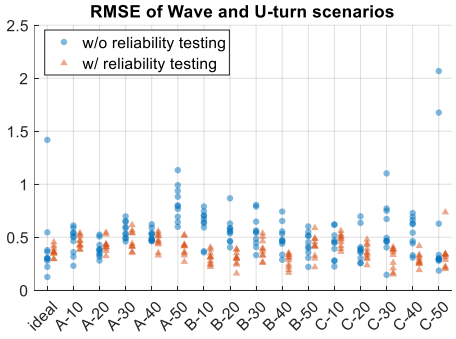


(a) RMSE of estimated results

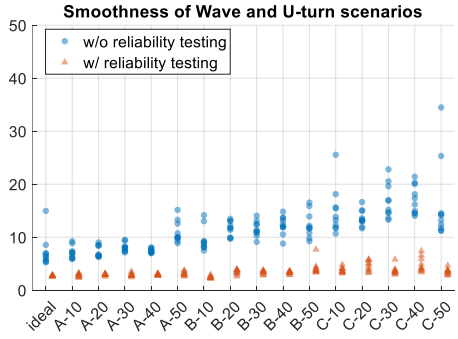


(b) Smoothness of estimated results

Figure 5.11: Estimated results of Wave trajectory in various synthetic scenes



(a) RMSE of estimated results



(b) Smoothness of estimated results

Figure 5.12: Estimated results of Wave and U-turn trajectory in various synthetic scenes

Table 5.3: Average RMSE of 10 estimations across all scenarios (RT: reliability testing)

Unit: m	Straight		Straight and U-turn		Wave		Wave and U-turn	
	w/o RT	w/ RT	w/o RT	w/ RT	w/o RT	w/ RT	w/o RT	w/ RT
Ideal	0.0697	0.0705	0.1081	0.0888	0.2492	0.2264	0.4232	0.3638
A-10	0.0709	0.071	0.0956	0.0933	0.2404	0.2242	0.4668	0.4548
A-20	0.0689	0.0716	0.1009	0.0883	0.2265	0.2187	0.3888	0.4277
A-30	0.0670	0.0706	0.0884	0.1017	0.3096	0.2052	0.5685	0.4658
A-40	0.0637	0.0703	0.1075	0.0929	0.2630	0.2537	0.5179	0.4612
A-50	0.0585	0.0700	0.0868	0.0826	0.3667	0.1638	0.8237	0.4032
B-10	0.0788	0.0652	0.1331	0.0821	0.3627	0.1965	0.6209	0.3063
B-20	0.0713	0.0662	0.1303	0.0741	0.3648	0.1923	0.5639	0.299
B-30	0.0702	0.0668	0.1440	0.0755	0.3388	0.2258	0.5570	0.3835
B-40	0.1034	0.0683	0.2036	0.078	0.2918	0.1815	0.4944	0.2748
B-50	0.0959	0.0843	0.1465	0.1506	0.4793	0.2166	0.4322	0.4104
C-10	0.0705	0.0801	0.1277	0.1076	0.2483	0.2788	0.4298	0.4699
C-20	0.0698	0.0814	0.1371	0.1121	0.3866	0.2217	0.4051	0.3605
C-30	0.0613	0.0800	0.1409	0.1042	0.3459	0.1736	0.5725	0.3036
C-40	0.0599	0.0724	0.1117	0.0851	0.2658	0.2056	0.5423	0.2877
C-50	0.1028	0.0729	0.1681	0.1173	0.3886	0.2116	0.6346	0.3320

doi:10.6342/NTU202402036

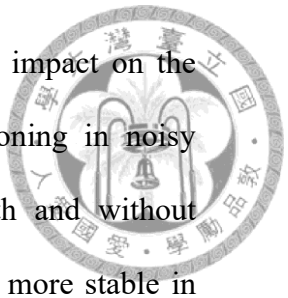
Table 5.4: Ratio between RMSE and trajectory length ($\frac{RMSE}{trajectory\ length} \times 100\%$) from 10 estimations in all scenarios (RT: reliability testing)

Unit: %	Straight		Straight and U-turn		Wave		Wave and U-turn	
	w/o RT	w/ RT	w/o RT	w/ RT	w/o RT	w/ RT	w/o RT	w/ RT
Ideal	0.4647	0.4701	0.3555	0.292	1.6615	1.5095	1.3924	1.1968
A-10	0.4723	0.4735	0.3147	0.3069	1.6029	1.4949	1.536	1.4963
A-20	0.4592	0.4775	0.3320	0.2907	1.5103	1.4583	1.2794	1.4071
A-30	0.4466	0.4703	0.2909	0.3345	2.064	1.3679	1.8706	1.5325
A-40	0.425	0.4689	0.3538	0.3057	1.7531	1.6916	1.7039	1.5174
A-50	0.3901	0.4664	0.2857	0.2718	2.4446	1.0919	2.7103	1.3268
B-10	0.5251	0.4347	0.4379	0.2701	2.4179	1.3102	2.0428	1.0078
B-20	0.4752	0.4415	0.4287	0.2438	2.4321	1.2823	1.8555	0.9839
B-30	0.4677	0.4455	0.4737	0.2486	2.2584	1.5052	1.8327	1.2617
B-40	0.6892	0.4552	0.6698	0.2568	1.9452	1.2103	1.6267	0.9043
B-50	0.6396	0.5619	0.4821	0.4955	3.1955	1.4439	1.4220	1.3503
C-10	0.4702	0.5343	0.4202	0.3539	1.6551	1.8585	1.4140	1.5462
C-20	0.4652	0.5423	0.4510	0.3689	2.5775	1.4778	1.3328	1.1862
C-30	0.4085	0.5332	0.4636	0.343	2.3060	1.157	1.8837	0.9988
C-40	0.3994	0.4827	0.3674	0.2801	1.7718	1.3705	1.7843	0.9466
C-50	0.6851	0.4862	0.5530	0.3859	2.5905	1.4108	2.0882	1.0924

Table 5.5: Average smoothness value of 10 estimations across all scenarios (RT: reliability testing)

	Straight		Straight and U-turn		Wave		Wave and U-turn	
	w/o RT	w/ RT	w/o RT	w/ RT	w/o RT	w/ RT	w/o RT	w/ RT
Ideal	1.2731	0.6565	2.4887	1.5420	3.9713	1.4852	7.2420	2.7249
A-10	1.4358	0.6482	3.0617	1.7343	5.5358	1.3230	7.0931	2.8179
A-20	1.5163	0.6794	2.9700	1.7282	4.9453	1.1943	7.3412	2.9457
A-30	1.4315	0.7220	2.9294	1.7326	6.1652	1.2629	8.0859	2.8995
A-40	1.4307	0.6930	3.0176	1.8253	3.9906	1.2276	7.4620	2.9285
A-50	2.4862	0.7338	4.8483	1.8253	7.3627	1.4506	10.9646	3.1431
B-10	1.5637	0.8355	3.4929	2.2949	7.0011	1.2297	9.5392	2.4587
B-20	1.5147	0.8241	3.8474	2.2046	10.2598	1.4835	11.6095	3.4666
B-30	1.8527	0.8865	3.4349	2.2269	7.0053	1.2485	11.5816	3.3768
B-40	1.8234	0.8068	3.5262	2.2152	7.4558	1.5757	12.2732	3.3349
B-50	2.8299	1.2681	4.6767	2.6771	10.1421	1.7392	12.3536	4.2038
C-10	1.9463	0.9082	4.0749	2.2189	6.2542	1.3682	14.8427	3.7885
C-20	1.9197	0.9250	4.6156	2.2734	6.6151	1.1976	13.645	4.6000
C-30	2.2507	0.8487	4.0506	2.0964	8.2717	1.3752	16.8236	3.7061
C-40	2.1256	0.8235	4.2438	2.1476	8.2345	3.0639	17.1378	4.8072
C-50	3.2074	1.1281	4.9876	2.6579	16.1809	1.5609	16.2715	3.4944

doi:10.6342/NTU202402036



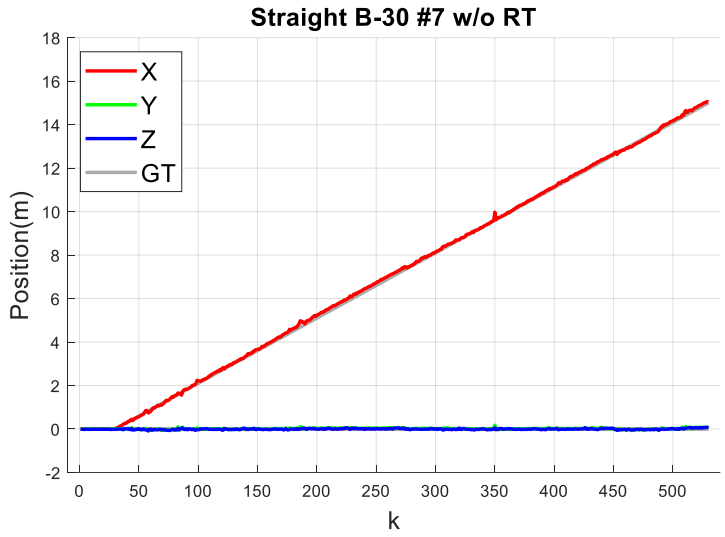
The estimated results indicate that the level of noise has a slight impact on the estimated accuracy, showing that the algorithm is capable of functioning in noisy environments. And comparing the accuracy of estimated results with and without reliability testing, adding reliability testing makes the estimated results more stable in repetitive tests, but the improvement in accuracy is marginal, because the reliability testing only discards unreliable estimations without optimizing the remaining estimations to guarantee the proposed algorithm's real-time capability.

For the trajectory smoothness, the smoothness of the estimated results is determined by the integral of the squared second derivative of estimated trajectories, the lower results indicate smoother trajectories. As the figures and Table 5.5 show, adding reliability testing makes the estimated trajectories smoother than without it. The reliability testing discards unreliable estimations which often cause trajectory drift, improving trajectory smoothness by reducing drift and high frequency errors.

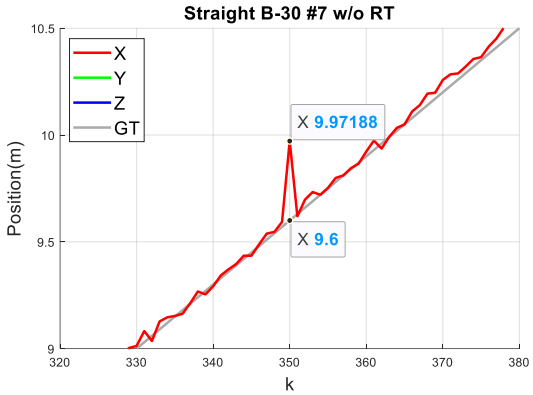
Following are some estimated results to show the performance and the difference of adding reliability testing.

Figure 5.13 shows an estimated result of Straight trajectory in the synthetic scene B-30 without reliability testing, with an RMSE of 0.0495 m (0.33 %) and a smoothness of 2.6011. Overall, the performance is well in going straight without any camera rotation, and hardly affect by noise, only occurs a drift at k=350 (Figure 5.13 (b)), the estimated trajectory has slight error due to limitation of camera resolution (Figure 5.13 (c)).

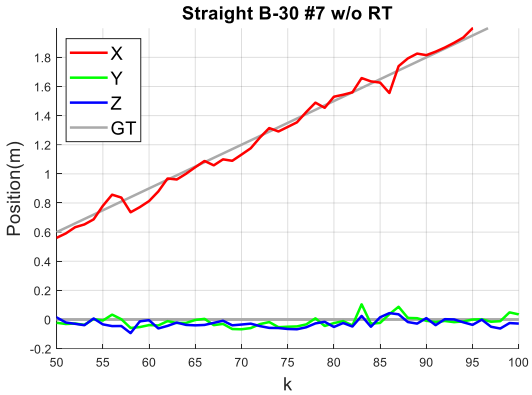
doi:10.6342/NTU202402036



(a) Estimated trajectory



(b) Drift at k=350



(c) Slight error cause by camera resolution

Figure 5.13: Estimated results of Straight trajectory in scene B-30 without reliability testing

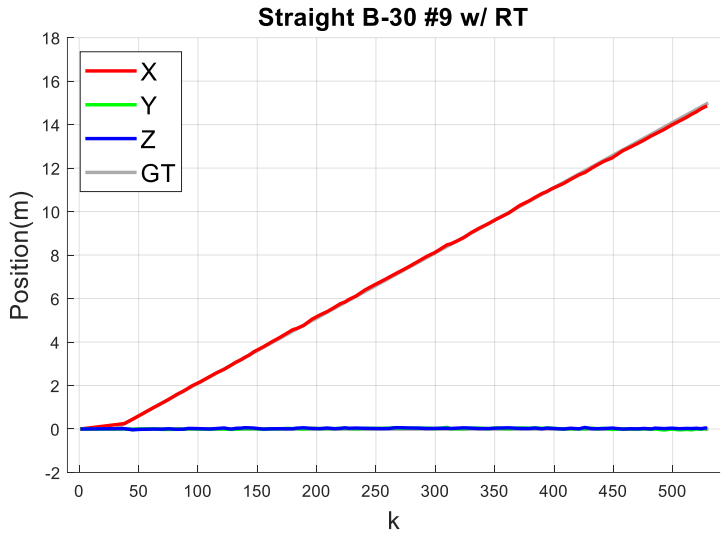
Figure 5.14 shows an estimated result of Straight trajectory in the synthetic scene B-30 with reliability testing, with an RMSE of 0.0693 m (0.462 %) and a smoothness of 0.5496. Compare to estimated results without reliability testing like Figure 5.13, because the original estimation is good enough and almost no drifts occur, the RMSE is slightly increase due to data downsampling during reliability testing, the number of waypoints between estimation and ground truth are different, the ground truth that cannot align by time will optimally match the estimated position at another time (Figure 5.14 (b)) through

doi:10.6342/NTU202402036

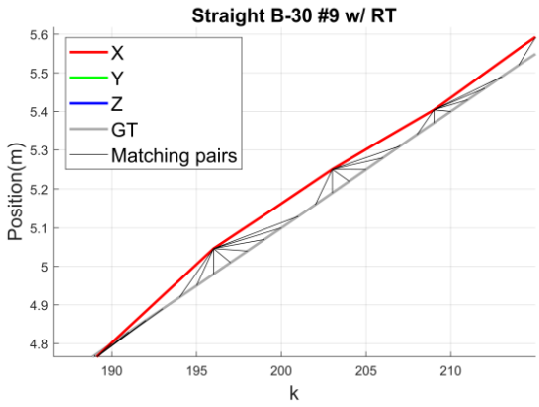
DTW method as discussed in Section 5.1.2. The trajectory smoothness improves by downsampling data during reliability testing, reducing high-frequency errors of the estimated trajectory and resulting a smoother estimation as shown in Figure 5.14 (c).

Figure 5.15 shows an estimated result of Wave trajectory in the synthetic scene B-30 without reliability testing, with an RMSE of 0.2791 m (1.8607 %) and a smoothness of 6.4503. Compare to the Straight trajectory, the RMSE is increased and less smooth due to changes in camera's yaw angle during the estimation, the estimated result shows the proposed algorithm has good performance when moving in X-Y plane. And Figure 5.16 shows an estimated result of Wave trajectory in the synthetic scene B-30 with reliability testing, with an RMSE of 0.2229 m (1.486 %) and a smoothness of 1.0481. In the complex trajectory, the reliability testing can improve performance in both RMSE and smoothness.

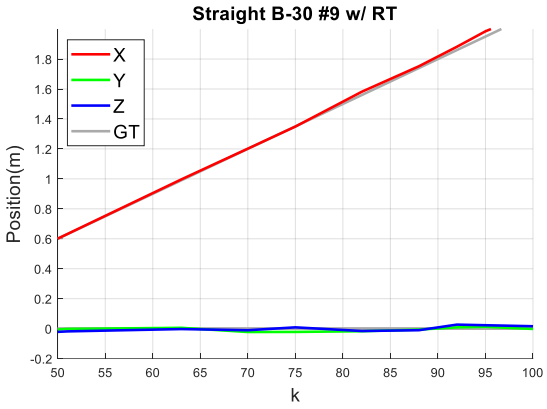
doi:10.6342/NTU202402036



(a) Estimated trajectory



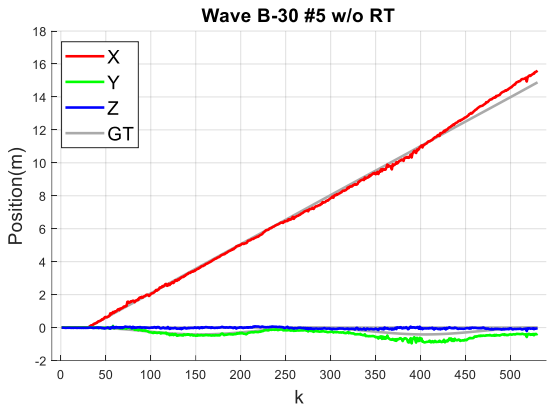
(b) DTW matching pairs

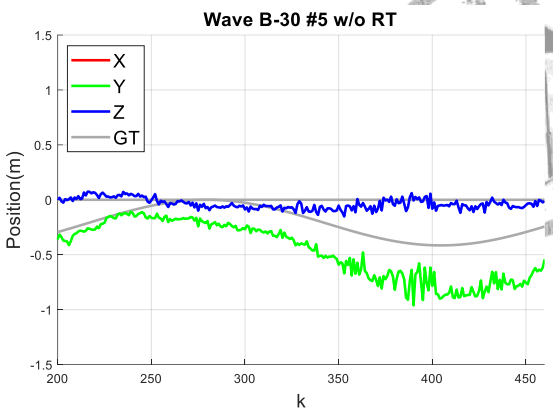


(c) Smoother trajectory after adding reliability testing

Figure 5.14: Estimated results of Straight trajectory in scene B-30 with reliability testing

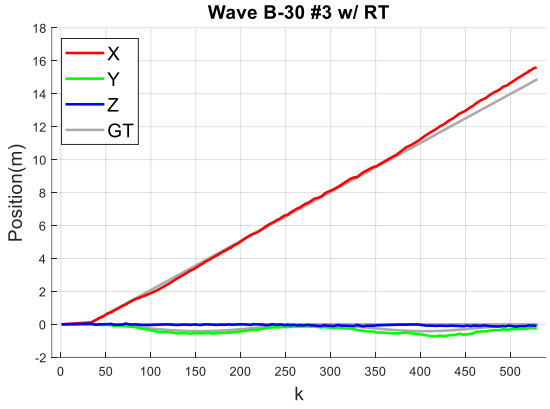
doi:10.6342/NTU202402036

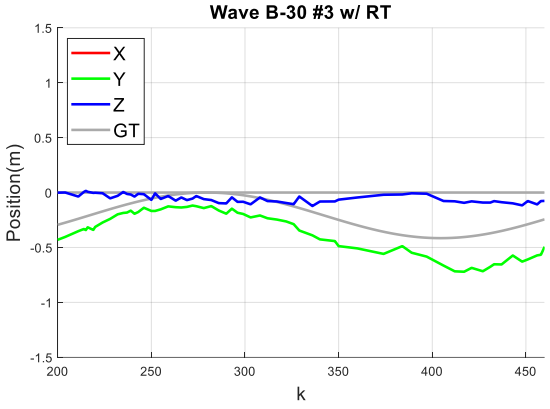




(a) Estimated trajectory (b) Details of the trajectory

Figure 5.15: Estimated results of Wave trajectory in scene B-30 without reliability testing





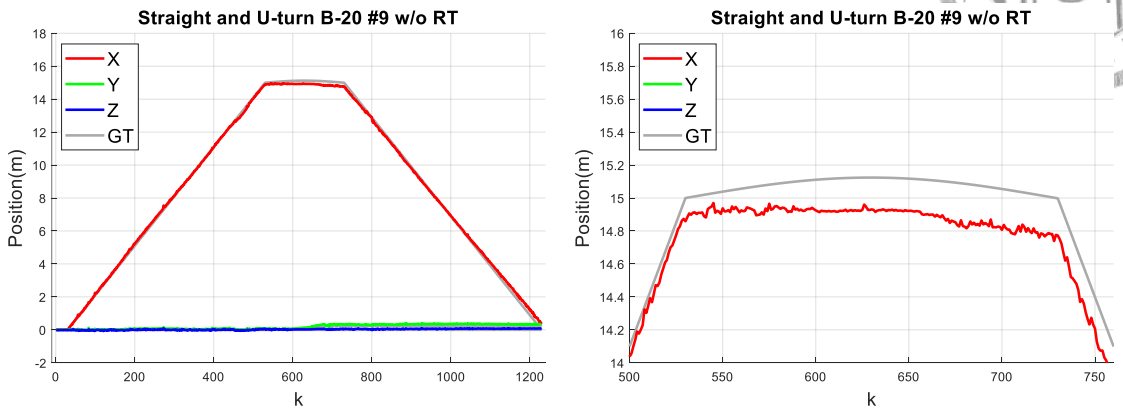
(a) Estimated trajectory (b) Details of the trajectory

Figure 5.16: Estimated results of Wave trajectory in scene B-30 with reliability testing

For the Straight and U-turn trajectory, and Wave and U-turn trajectory in the synthetic scenes, the estimated results are shown in Figure 5.17 to Figure 5.20, as discussed in Straight and Wave trajectories, the reliability testing can yield improved performance in both RMSE and smoothness, However, when examining the performance of each segment in the trajectory, it is observed that the U-turn segment at k=531 to 730 has a higher error. this phenomenon happened because the target plane changes greatly in camera imaging as shown in Figure 5.21, the accumulated error has little effect on the accuracy in the Return segment, the estimated trajectory pattern still reflects the ground truth pattern. This conclusion can be obtained from Table 5.6 to Table 5.9 that calculate

doi:10.6342/NTU202402036

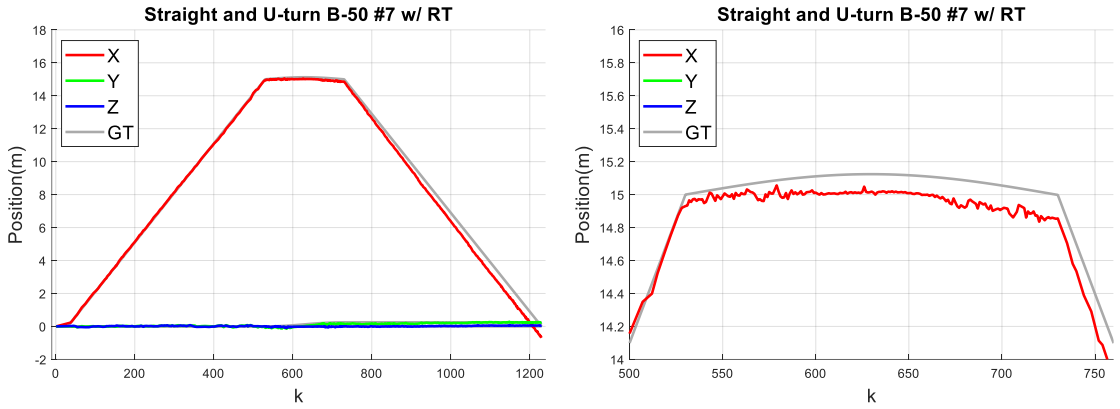
the RMSE from segments separately with remove accumulated error from previous segment.



(a) Estimated result with RMSE = 0.1068m(0.35%), smoothness=3.4201

(b) Trajectory details of (a) at U-turn segment

Figure 5.17: Estimated results of Straight and U-turn trajectory in scene B-20 without reliability testing

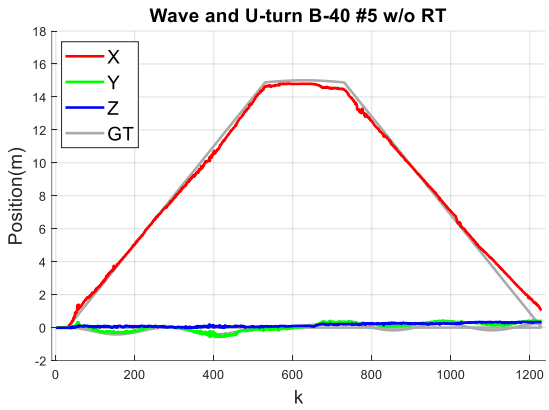


(a) Estimated result with RMSE = 0.0965m(0.32%), smoothness=2.5527

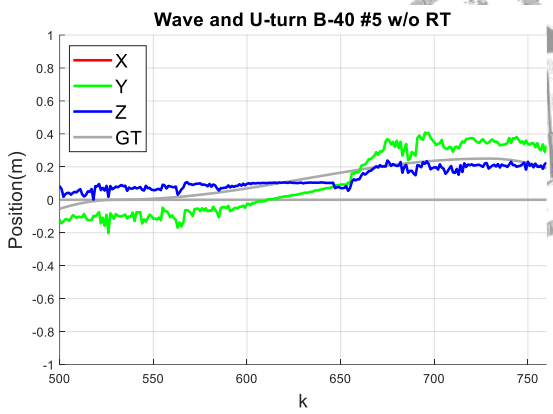
(b) Trajectory details of (a) at U-turn segment

Figure 5.18: Estimated results of Straight and U-turn trajectory in scene B-50 with reliability testing

doi:10.6342/NTU202402036

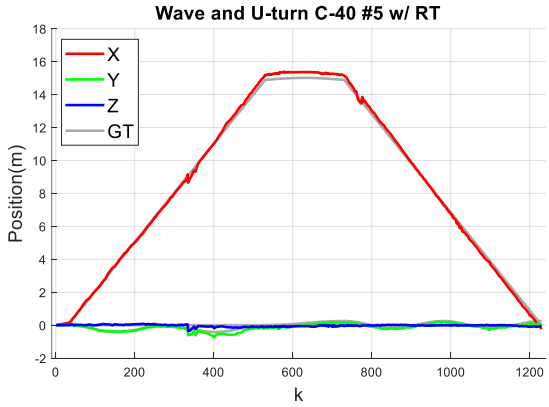


(a) Estimated result with RMSE = 0.2874m(0.95%), smoothness=8.7969

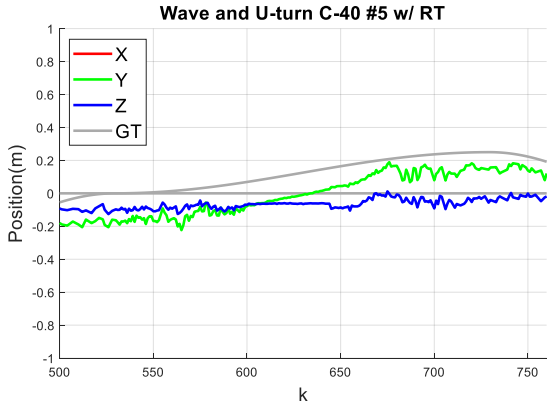


(b) Trajectory details of (a) at U-turn segment

Figure 5.19: Estimated results of Wave and U-turn trajectory in scene B-40 without reliability testing



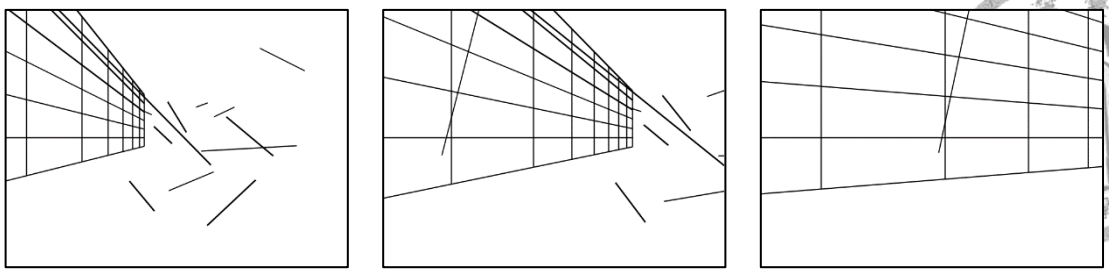
(a) Estimated result with RMSE = 0.1911m(0.63%), smoothness=6.5840



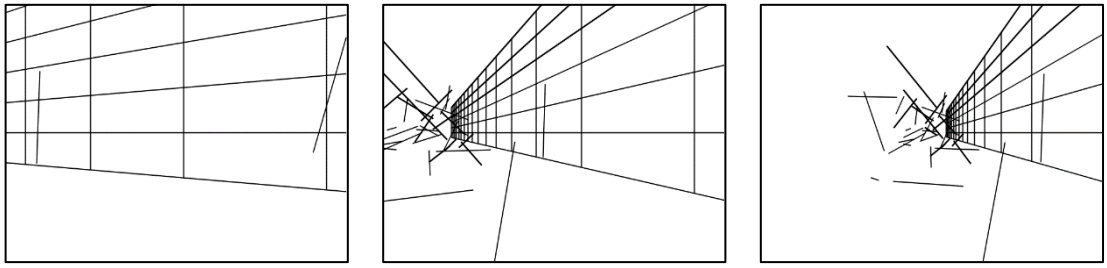
(b) Trajectory details of (a) at U-turn segment

Figure 5.20: Estimated results of Wave and U-turn trajectory in scene C-40 with reliability testing

doi:10.6342/NTU202402036



(a) k=531 (b) k=570 (c) k=610



(d) k=650 (e) k=690 (f) k=730

Figure 5.21: Camera images at U-turn segment of Wave and U-turn trajectory in scene B-30.

Table 5.6: Average RMSE of the segments of Straight and U-turn trajectory in synthetic scenes, estimated without reliability testing.

	Forward	U-turn	Return
Ideal	0.0719m (0.48%)	0.0474m (12.08%)	0.0773m (0.52%)
A-10	0.0585m (0.39%)	0.0424m (10.79%)	0.0648m (0.43%)
A-20	0.0700m (0.47%)	0.0451m (11.49%)	0.0932m (0.62%)
A-30	0.0632m (0.42%)	0.0411m (10.47%)	0.0880m (0.59%)
A-40	0.0581m (0.39%)	0.0488m (12.42%)	0.1380m (0.92%)
A-50	0.0609m (0.41%)	0.0563m (14.34%)	0.0853m (0.57%)
B-10	0.0738m (0.49%)	0.0507m (12.91%)	0.0976m (0.65%)
B-20	0.0806m (0.54%)	0.0447m (11.39%)	0.0719m (0.48%)
B-30	0.0769m (0.51%)	0.0580m (14.76%)	0.0803m (0.54%)
B-40	0.1067m (0.71%)	0.0529m (13.46%)	0.1045m (0.70%)
B-50	0.0825m (0.55%)	0.0520m (13.25%)	0.1356m (0.90%)
C-10	0.0664m (0.44%)	0.0543m (13.84%)	0.0559m (0.37%)
C-20	0.0520m (0.35%)	0.0471m (12.00%)	0.1256m (0.84%)
C-30	0.0765m (0.51%)	0.0468m (11.93%)	0.1198m (0.80%)
C-40	0.0646m (0.43%)	0.0508m (12.94%)	0.0887m (0.59%)
C-50	0.0929m (0.62%)	0.0664m (16.91%)	0.2402m (1.60%)

doi:10.6342/NTU202402036

Table 5.7: Average RMSE of the segments of Straight and U-turn trajectory in synthetic scenes, estimated with reliability testing.

	Forward	U-turn	Return
Ideal	0.0697m (0.46%)	0.0379m (9.64%)	0.0677m (0.45%)
A-10	0.0724m (0.48%)	0.0486m (12.39%)	0.0787m (0.52%)
A-20	0.0700m (0.47%)	0.0460m (11.71%)	0.0757m (0.50%)
A-30	0.0696m (0.46%)	0.0544m (13.85%)	0.1123m (0.75%)
A-40	0.0667m (0.44%)	0.0473m (12.05%)	0.1048m (0.70%)
A-50	0.0685m (0.46%)	0.0493m (12.56%)	0.0669m (0.45%)
B-10	0.0675m (0.45%)	0.0622m (15.85%)	0.0795m (0.53%)
B-20	0.0640m (0.43%)	0.0544m (13.86%)	0.0664m (0.44%)
B-30	0.0659m (0.44%)	0.0648m (16.51%)	0.0553m (0.37%)
B-40	0.0694m (0.46%)	0.0626m (15.93%)	0.0591m (0.39%)
B-50	0.0775m (0.52%)	0.0774m (19.71%)	0.1448m (0.97%)
C-10	0.0806m (0.54%)	0.0704m (17.94%)	0.0637m (0.42%)
C-20	0.0857m (0.57%)	0.0847m (21.56%)	0.0574m (0.38%)
C-30	0.0799m (0.53%)	0.0825m (21.00%)	0.0578m (0.39%)
C-40	0.0754m (0.50%)	0.0914m (23.27%)	0.0584m (0.39%)
C-50	0.0727m (0.48%)	0.0516m (13.15%)	0.0900m (0.60%)

Table 5.8: Average RMSE of the segments of Wave and U-turn trajectory in synthetic scenes, estimated without reliability testing.

	Forward	U-turn	Return
Ideal	0.2980m (1.99%)	0.1007m (25.65%)	0.3543m (2.36%)
A-10	0.2193m (1.46%)	0.0648m (16.49%)	0.5484m (3.66%)
A-20	0.2130m (1.42%)	0.0785m (20.00%)	0.2836m (1.89%)
A-30	0.3043m (2.03%)	0.0809m (20.59%)	0.5356m (3.57%)
A-40	0.2821m (1.88%)	0.0785m (20.00%)	0.4250m (2.83%)
A-50	0.3739m (2.49%)	0.0605m (15.40%)	1.0688m (7.13%)
B-10	0.3266m (2.18%)	0.0752m (19.16%)	0.7821m (5.21%)
B-20	0.3271m (2.18%)	0.0778m (19.81%)	0.6117m (4.08%)
B-30	0.3463m (2.31%)	0.0799m (20.34%)	0.5433m (3.62%)
B-40	0.2981m (1.99%)	0.1085m (27.62%)	0.4585m (3.06%)
B-50	0.2731m (1.82%)	0.0817m (20.80%)	0.3465m (2.31%)
C-10	0.2556m (1.70%)	0.0871m (22.19%)	0.3805m (2.54%)
C-20	0.2050m (1.37%)	0.0770m (19.61%)	0.4164m (2.78%)
C-30	0.2596m (1.73%)	0.0638m (16.24%)	0.7469m (4.98%)
C-40	0.2716m (1.81%)	0.1003m (25.54%)	0.6743m (4.50%)
C-50	0.3364m (2.24%)	0.1188m (30.26%)	0.7277m (4.85%)

doi:10.6342/NTU202402036

Table 5.9: Average RMSE of the segments of Wave and U-turn trajectory in synthetic scenes, estimated with reliability testing.

	Forward	U-turn	Return
Ideal	0.2350m (1.57%)	0.0620m (15.79%)	0.2351m (1.57%)
A-10	0.2523m (1.68%)	0.0573m (14.58%)	0.4143m (2.76%)
A-20	0.2359m (1.57%)	0.0561m (14.28%)	0.3883m (2.59%)
A-30	0.2376m (1.58%)	0.0536m (13.64%)	0.4973m (3.32%)
A-40	0.2403m (1.60%)	0.0674m (17.17%)	0.4384m (2.92%)
A-50	0.1754m (1.17%)	0.0530m (13.50%)	0.4443m (2.96%)
B-10	0.1882m (1.25%)	0.0593m (15.09%)	0.1899m (1.27%)
B-20	0.1758m (1.17%)	0.0586m (14.91%)	0.2213m (1.48%)
B-30	0.2215m (1.48%)	0.0804m (20.47%)	0.2932m (1.95%)
B-40	0.1652m (1.10%)	0.0716m (18.24%)	0.1855m (1.24%)
B-50	0.2352m (1.57%)	0.0705m (17.96%)	0.3255m (2.17%)
C-10	0.2743m (1.83%)	0.0769m (19.59%)	0.4038m (2.69%)
C-20	0.2148m (1.43%)	0.0738m (18.79%)	0.3118m (2.08%)
C-30	0.1803m (1.20%)	0.0569m (14.48%)	0.2647m (1.76%)
C-40	0.1843m (1.23%)	0.0764m (19.45%)	0.1749m (1.17%)
C-50	0.2067m (1.38%)	0.0958m (24.40%)	0.2277m (1.52%)

5.1.4 Summary of Performance in Simulation

In this section, the performance of the proposed algorithm is verified through estimated in 4 different trajectory and 16 synthetic scenes with varying noise levels, resulting in a total of 64 different scenarios. The proposed algorithm is estimated in scenarios with and without reliability testing, and the estimation is repeated 10 times to validate the stability of the algorithm and ensure that no conclusions are drawn from outliers.

From the estimations in different noise levels, the results show that the propose algorithm can successfully detect and track the target plane in noisy environments, calculate camera trajectory from the sequential images. With adding reliability testing to the algorithm, estimation errors can be reduced by discarding unreliable estimates and obtaining a smoother trajectory through data downsampling. In the trajectories with U-turn segment, the estimated results show that the proposed algorithm can track the target plane when camera yaw angle changes greatly.

doi:10.6342/NTU202402036

The simulation results show that the proposed algorithm is capable of handling the estimation of a moving robot in the XY plane, and obtaining the camera's trajectory.

5.2 Real-World Experiments Executed in Real-Time

In this section, the experiments setup regarding the hardware used is described in Section 5.2.1, the indoor hallway experiments compared with the marker-based method are shown in Section 5.2.2, the outdoor experiments of the algorithm executed in the real-world noisy environments are presented in Section 5.2.2. A summary of experiments is shown in Section 5.2.3.

5.2.1 Experiments Setup

This thesis run the real-world experiments on a laptop equipped with AMD Ryzen 7 7840U CPU at 3.3 GHz, 16-GB RAM, and windows 11 operation system. The proposed algorithm is implemented in Python and provides real-time estimations during experiments.

An RGB camera of Intel RealSense D455 [55: Intel] is used for the real-world experiments, it has 90×65° FOV(H×V) with global shutter sensor, and the resolution is set at 640×480 with frame rate at 30 fps. As shown in Figure 5.22, the camera is taped on the laptop, and run the experiments by walking with the laptop in hand.

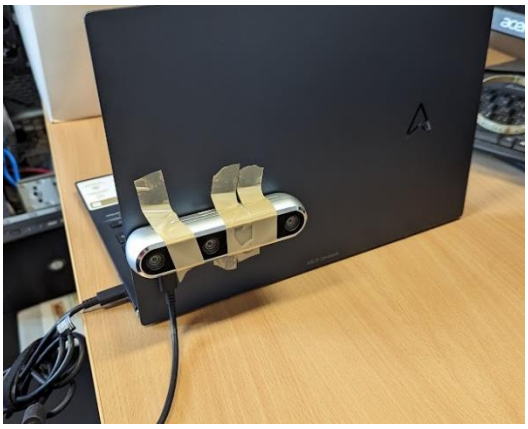


Figure 5.22 : The camera and the laptop

doi:10.6342/NTU202402036

As mention in Section 4.3.2, the initialization process requires the distance from camera to target plane to determine the scale factor from pixels to the real world, a laser range finder as shown in Figure 5.23 is used for measuring a rough distance to the target plane during initialization process.

Figure 5.23: The laser range finder used in experiments

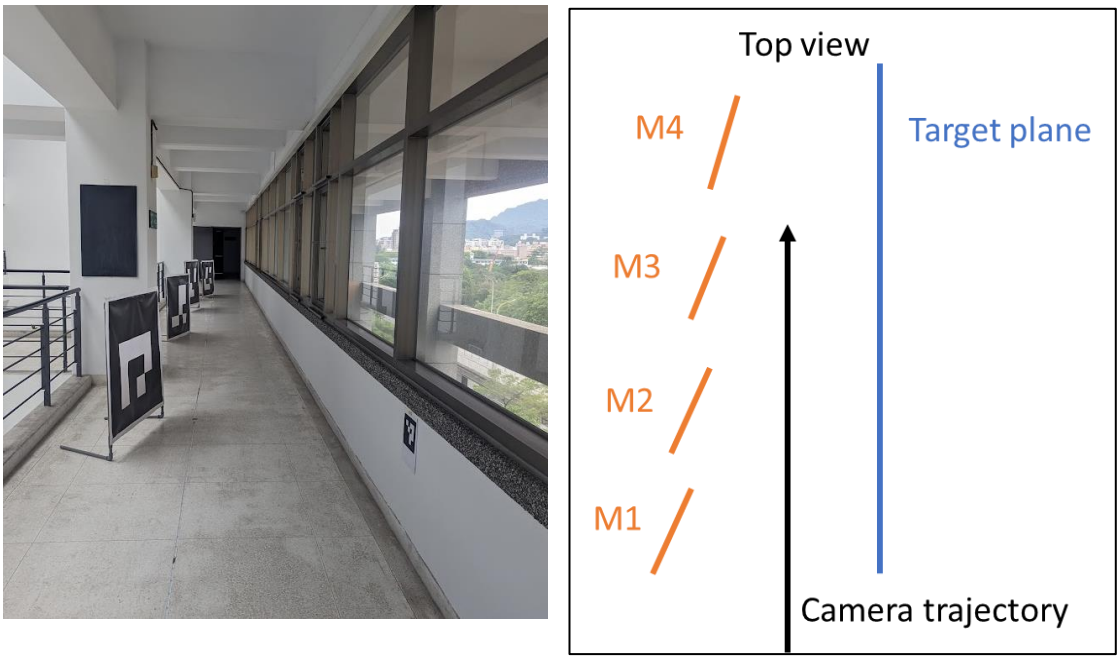
In order to verify the accuracy of the proposed algorithm, in the indoor hallway experiments, a marker-based pose estimation using ArUco markers [45: Garrido-Jurado et al. 2014] is employed as the ground truth in comparison to the algorithm's estimation. The ArUco markers are made by canvas with a marker size of 102×102cm as shown in Figure 5.24.

Figure 5.24: A 102×102cm ArUco marker made by canvas

doi:10.6342/NTU202402036

5.2.2 Indoor Hallway Experiments

In the indoor hallway experiments, the experiment scene is shown in Figure 5.25, the window frames on the right side of the camera image are the target plane of the experiment, and 4 ArUco markers $M1, M2, M3, M4$ arranged from near to far on the left side of the camera image are used to generate the ground truth of the camera trajectory. the camera trajectory is going straight in the experiment and repeat the experiment 10 times to confirm accuracy and consistency, the target plane and at least 2 of the ArUco markers are visible in the camera image during the estimation.



(a) Hallway scene (b) Schematic diagram of the scene

Figure 5.25: The hallway experiment scene

The ground truth trajectory is obtained by combining pose estimations from ArUco markers with OpenCV library [56: OpenCV] through the absolute trajectory error method from [57: Mur-Artal] according to [47: Horn 1987]. The trajectory estimated from each marker is based on its own coordinate system as shown in Figure 5.26. Align the trajectories by time and calculate the relative pose between markers by solving Equation (5.2), get $R_{12}, t_{12}, R_{23}, t_{23}, R_{34}, t_{34}$ from camera images, then transform all the

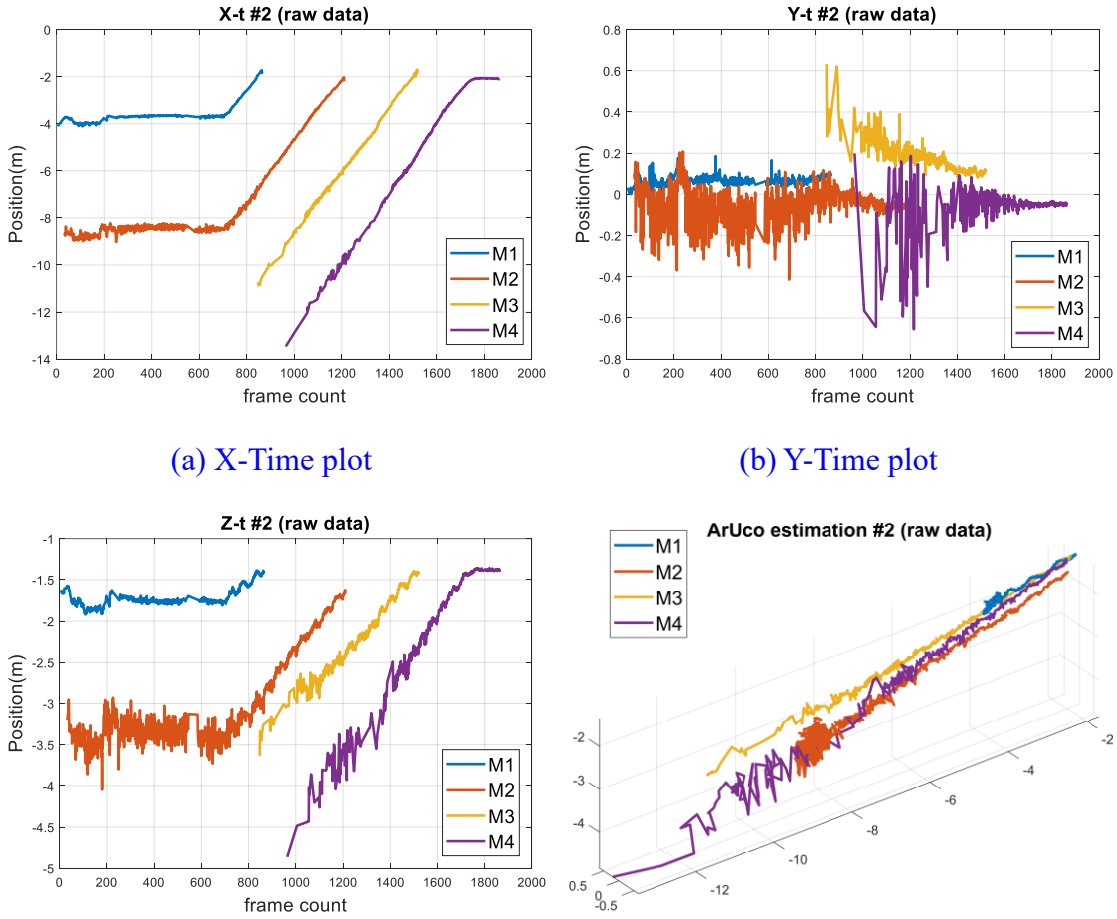
doi:10.6342/NTU202402036

coordinate systems to the coordinate system of M1 with Equation (5.3), as shown in Figure 5.27. For estimations from different markers at the same time, select the estimation from nearest marker as ground truth trajectory, the result is shown in Figure 5.28.

$$\min_{\boldsymbol{R}_{ij},\boldsymbol{t}_{ij}} \sum_{k} \left(\boldsymbol{R}_{ij}\boldsymbol{p}_{jk} + \boldsymbol{t}_{ij}\right) - \boldsymbol{p}_{ik}$$
$$\boldsymbol{R}_{ij} \in \mathrm{SO}(3), \boldsymbol{t} \in \mathbb{R}^{3\times 1}$$
$$\boldsymbol{p}_{jk}, \boldsymbol{p}_{ik} \in \mathbb{R}^{3\times 1}, \text{the } k\text{th matched pair estimated by } Mi \text{ and } Mj \tag{5.2}$$

$$^{M1}\boldsymbol{p}_i = \boldsymbol{R}_{1i}\boldsymbol{p}_i + \boldsymbol{t}_{1i}, \qquad i = 2,3,4$$
$$\boldsymbol{R}_{13} = \boldsymbol{R}_{12}\boldsymbol{R}_{23}$$
$$\boldsymbol{t}_{13} = \boldsymbol{t}_{12} + \boldsymbol{R}_{12}\boldsymbol{t}_{23}$$
$$\boldsymbol{R}_{14} = \boldsymbol{R}_{12}\boldsymbol{R}_{23}\boldsymbol{R}_{34}$$
$$\boldsymbol{t}_{14} = \boldsymbol{t}_{12} + \boldsymbol{R}_{12}\boldsymbol{t}_{23} + \boldsymbol{R}_{12}\boldsymbol{R}_{23}\boldsymbol{t}_{34} \tag{5.3}$$

Where $\boldsymbol{p}_i$ is the trajectory estimated from Mi



(a) X-Time plot

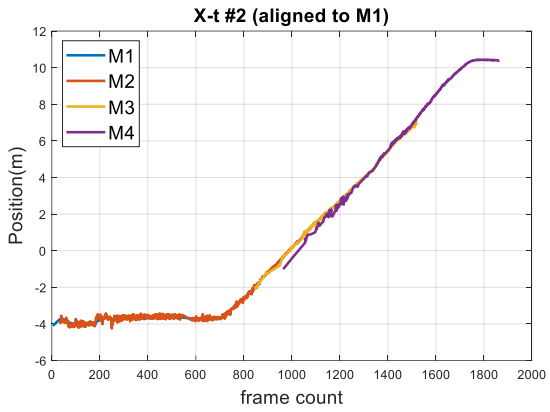
(b) Y-Time plot

(c) Z-Time plot

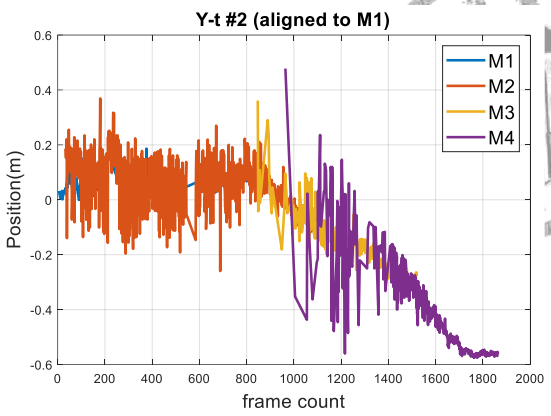
(d) 3-D plot

Figure 5.26: The initial trajectories estimated from ArUco markers

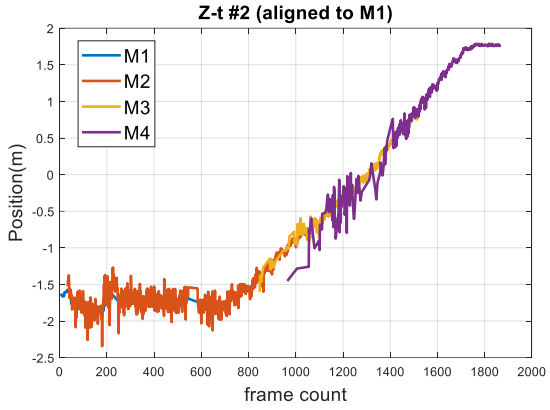
doi:10.6342/NTU202402036



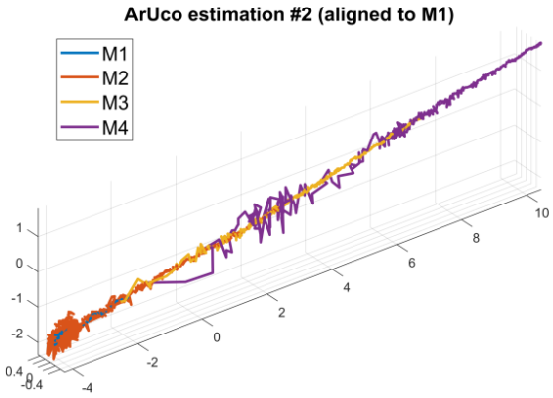
(a) X-Time plot



(b) Y-Time plot

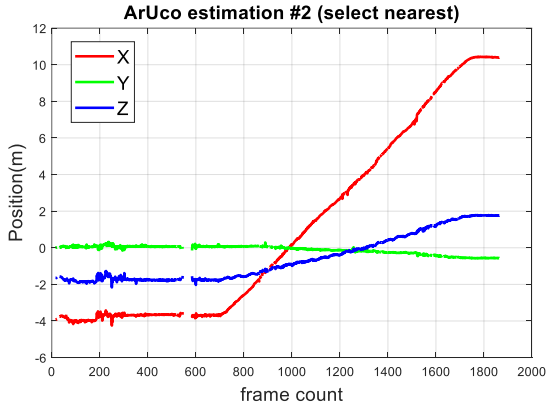


(c) Z-Time plot

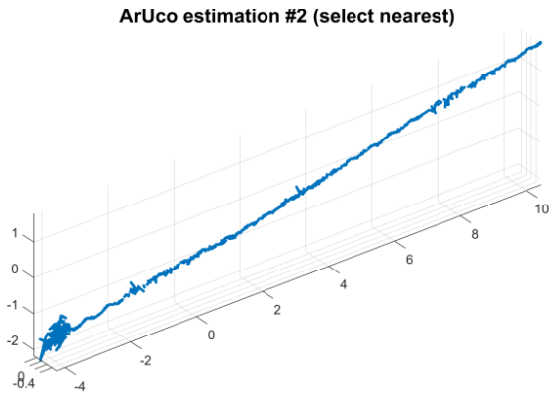


(d) 3-D plot

Figure 5.27: The aligned trajectories estimated from ArUco markers



(a) Aligned ground truth trajectory



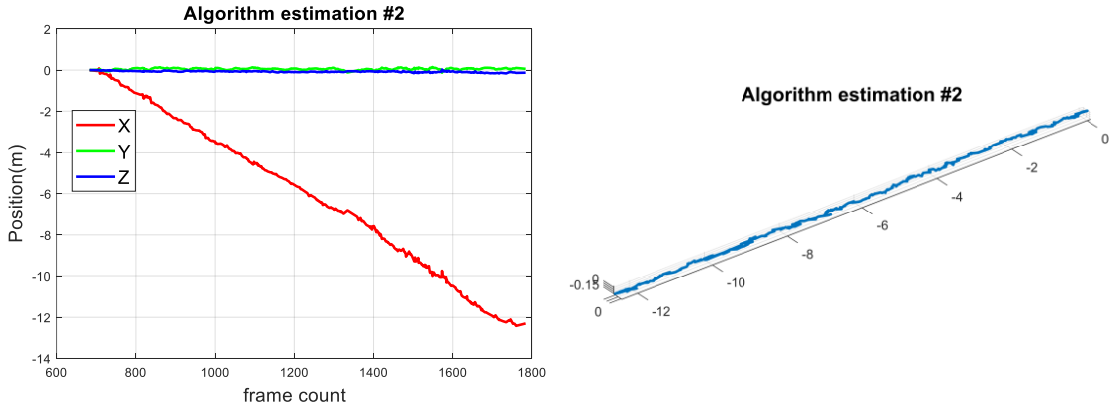
(b) 3-D plot of the trajectory

Figure 5.28: The ground truth trajectory estimated from ArUco Markers

The estimated result from the proposed algorithm is shown in Figure 5.29, and using world frame mentioned in Section 4.1 as the coordinate system of the estimated trajectory, and the scale factor to the real-world is not accurate as it is determined through manual

doi:10.6342/NTU202402036

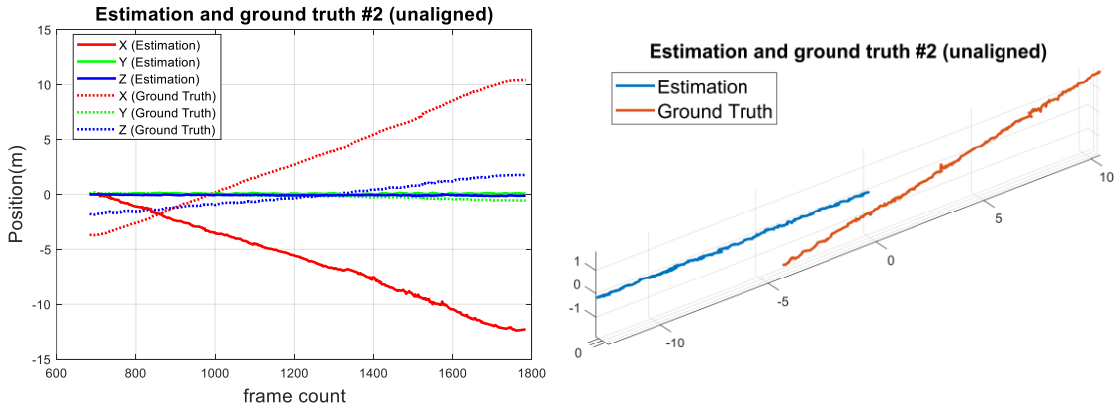
measurement. To determine the performance of the proposed algorithm in real-world scenarios, it is necessary to calculate the optimal scale factor and relative pose to align the estimation results with the ground truth.



(a) Estimated trajectory (b) 3-D plot of the trajectory

Figure 5.29: The estimated trajectory from proposed algorithm

In the indoor experiments, the performance is determined by the root-mean-square error (RMSE) over time indices with both ground truth and estimations from proposed algorithm, the matched trajectories are shown in Figure 5.30. Then calculate the scale factor and rotation between two trajectories by solving Equation (5.4), transform the coordinate system of ground truth to the world frame, and move the start position of two trajectories to the origin, the aligned result is shown in Figure 5.31.

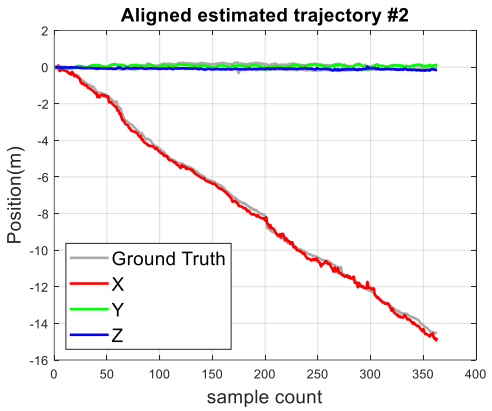


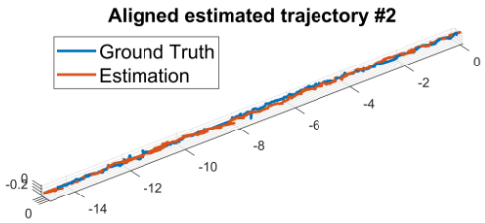
(a) Matched trajectories before alignment (b) 3-D plot of the trajectories

Figure 5.30: The trajectories matched by time indices before aligning the scale factor and coordinate system.

doi:10.6342/NTU202402036

$$\min_{\boldsymbol{R}_w, s} \sum_k \boldsymbol{R}_w \boldsymbol{p}_k - s\boldsymbol{q}_k$$
$$\boldsymbol{R}_{ij} \in \mathrm{SO}(3), s \in \mathbb{R}$$
$\boldsymbol{p}_k \in \mathbb{R}^{3\times 1}$,the position of kth matched pair from ground truth trajectory
$\boldsymbol{q}_k \in \mathbb{R}^{3\times 1}$,the position of kth matched pair from estimated trajectory (5.4)





(a) Estimated result after alignment (b) 3-D plot of the estimated result

Figure 5.31: the estimated trajectory after aligning the scale factor and coordinate system.

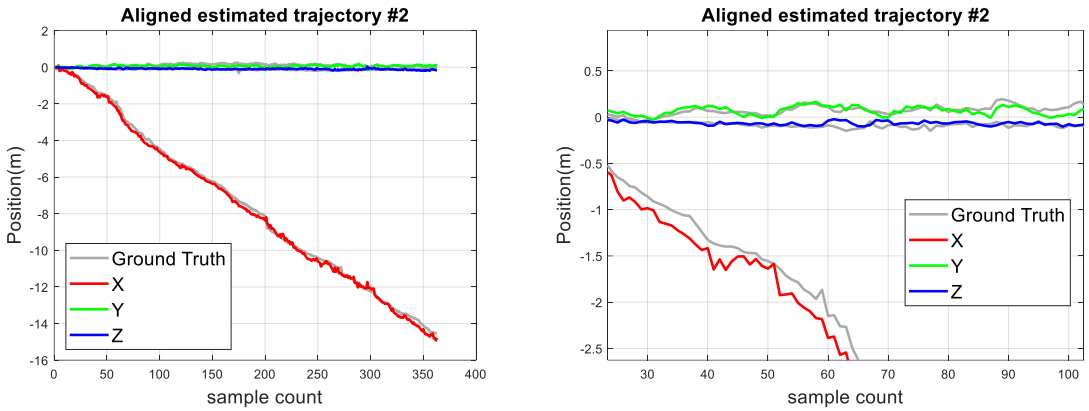
Table 5.10 shows the performance of all 10 experiments, including trajectory length, root-mean-square error (RMSE), maximum error (Max Error), the ratio between RMSE and the trajectory length, and the ratio between the number of output positions and the number of input camera images during estimation.

Table 5.10: The estimated results of indoor hallway experiments

Experiment Number	Trajectory Length	RMSE	Max Error	$\frac{RMSE}{\text{Trajectory Length}}$	Estimate Rate
#1	24.446 m	0.498 m	0.957 m	2.04 %	37.5 %
#2	21.422 m	0.212 m	0.493 m	0.99 %	34.1 %
#3	26.400 m	0.863 m	1.583 m	3.27 %	35.0 %
#4	29.185 m	0.684 m	1.282 m	2.34 %	38.8 %
#5	26.091 m	0.361 m	0.898 m	1.38 %	36.3 %
#6	24.654 m	0.526 m	1.111 m	2.13 %	38.1 %
#7	22.670 m	0.545 m	1.198 m	2.40 %	32.0 %
#8	23.773 m	0.490 m	1.189 m	2.06 %	34.4 %
#9	27.261 m	0.378 m	0.893 m	1.39 %	38.6 %
#10	28.267 m	0.271 m	0.730 m	0.96 %	40.9 %
Average	25.417 m	0.512 m		2.01 %	36.6 %

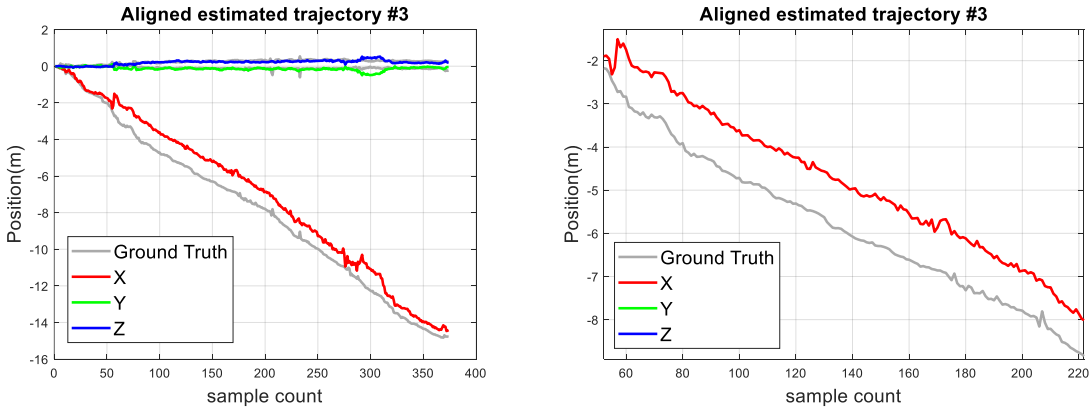
doi:10.6342/NTU202402036

The results show that the proposed algorithm can operate in real-world scenario, achieve real-time demand estimation by downsampling through reliability testing, in average, about 65% of input images are discarded and can still estimate the entire camera trajectory. While some experiments show higher errors due to drifts but all experiments are able to estimate the pattern throughout the trajectory. For example, Figure 5.32 shows the accurately estimated result, can even track the small movement in the Y-direction while walking. In Figure 5.33, the estimated trajectory has drifts, but the motion after drifts occur remains consistent with the ground truth trajectory. All estimated results are detailed in Appendix A.



(a) Estimated trajectory (b) Details of the trajectory

Figure 5.32: Estimated result of experiment #2



(a) Estimated trajectory (b) Details of the trajectory

Figure 5.33: Estimated result of experiment #3

doi:10.6342/NTU202402036

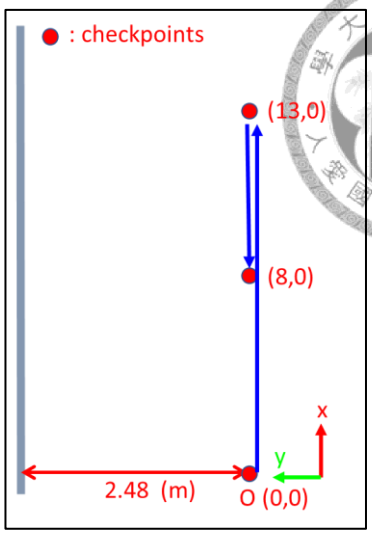
5.2.3 Outdoor Building Facade Experiments

In the outdoor building facade experiments, the main purpose of the experiments is to validate whether the propose algorithm can handle more complex environments. There are 4 cases in total, the first 2 cases are estimate along a building facade, with a few checkpoints measured roughly using a tape measure. The remaining 2 cases demonstrate the algorithm's capability to estimate across different scenes but have no comparable references.

Case 1: Moving straight and U-turn along building facade

In this case, as shown in Figure 5.34, the target plane is the building facade with windows in the left side of the camera frame, the camera moves forward for 13 meters, then turns around, continues forward for 5 meters and stop the estimation. Some camera frames during estimation are shown in Figure 5.35. The estimated trajectory is shown in Figure 5.36, the errors of the checkpoints are shown in Figure 5.37 and Figure 5.38. The estimated result shows that the pattern of trajectory and the designed trajectory are consistent, but the estimated trajectory has more error compared to indoor experiment. Possible reasons include the interference from floor tiles and inaccuracies in the scale factor to the world.

doi:10.6342/NTU202402036

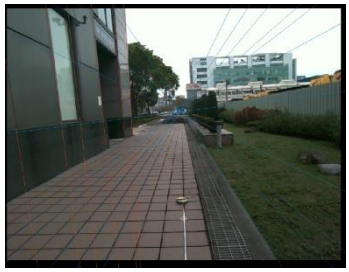


(a) Experiment scenes (b) Designed trajectory and checkpoints

Figure 5.34: The experiment scene of case 1

(a) Frame 1

(b) Frame 2



(c) Frame 3

(d) Frame 4

(e) Frame 5

(f) Frame 6

Figure 5.35: Camera frames from case 1 during estimation

doi:10.6342/NTU202402036

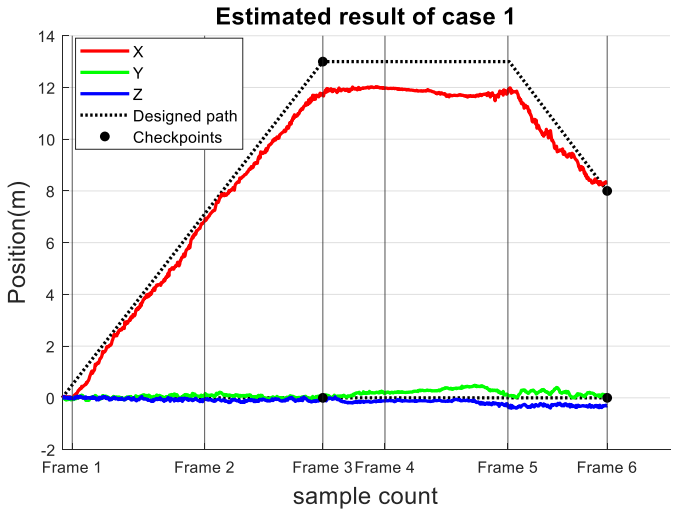
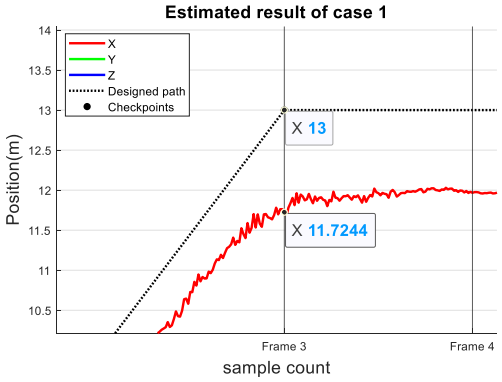
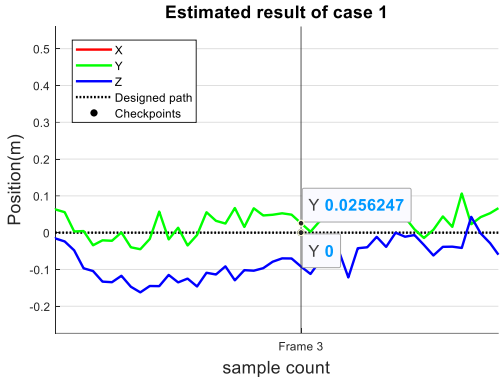


Figure 5.36: Estimated trajectory of case 1

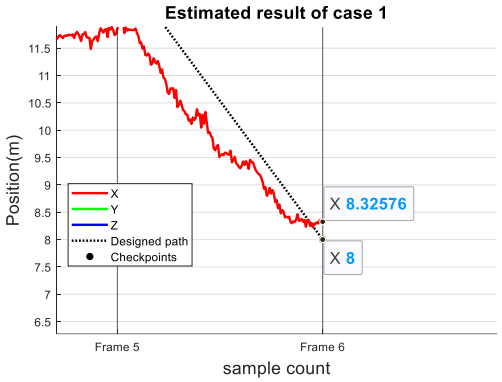


(a) 1.276 m error in X-direction at checkpoint (13,0)

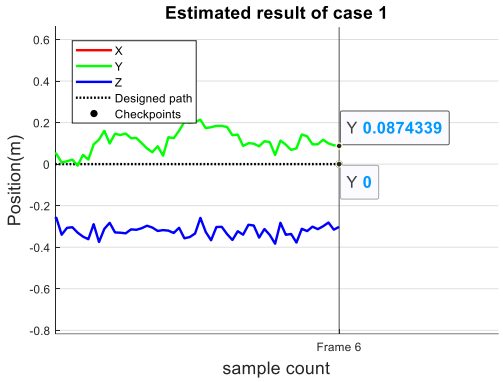


(b) 0.026 m error in Y-direction at checkpoint (13,0)

Figure 5.37: Estimated result at checkpoint (13,0)



(a) 0.326 m error in X-direction at checkpoint (8,0)



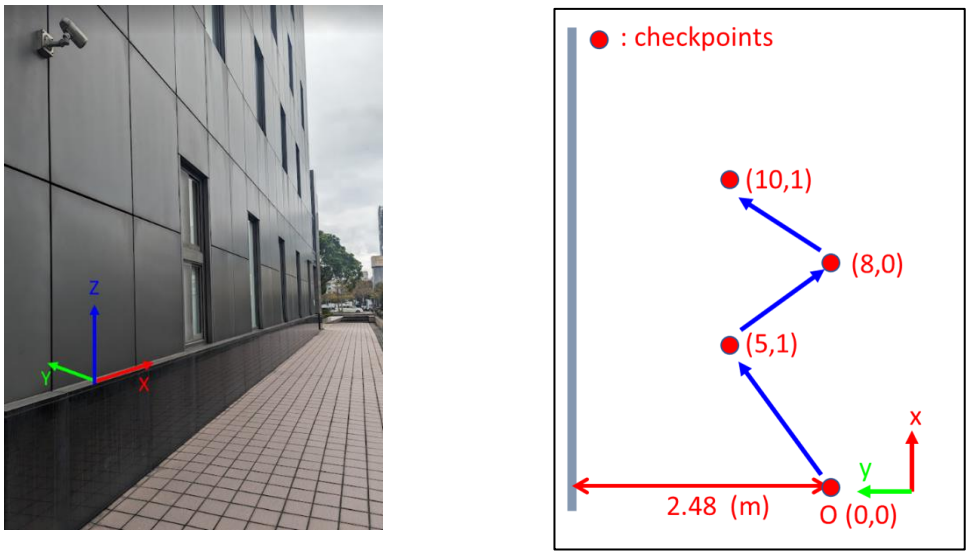
(b) 0.087 m error in Y-direction at checkpoint (8,0)

Figure 5.38: Estimated result at checkpoint (8,0)

doi:10.6342/NTU202402036

Case 2: Moving forward in XY plane along building facade

Case 2 and case 1 are executed in the same location, as shown in Figure 5.39, the designed trajectory moves forward in the X and Y directions, passing through the checkpoints located at (5,1), (8,0) and (10,1). Figure 5.40 shows the camera frames during the estimation. The estimated trajectory is shown in Figure 5.41, the errors of the checkpoints are shown in Figure 5.42, Figure 5.43 and Figure 5.44. The estimation result is similar to Case 1, the trajectory pattern is consistent with designed trajectory but has quite cumulative errors.



(a) Experiment scenes (b) Designed trajectory and checkpoints

Figure 5.39: The experiment scene of case 2

doi:10.6342/NTU202402036

(a) Frame 1

(b) Frame 2

(c) Frame 3

(d) Frame 4

(e) Frame 5

(f) Frame 6

Figure 5.40: Camera frames from case 2 during estimation

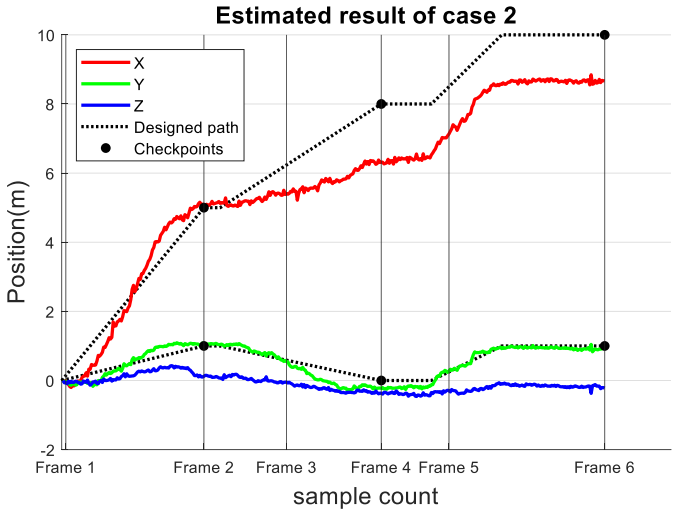
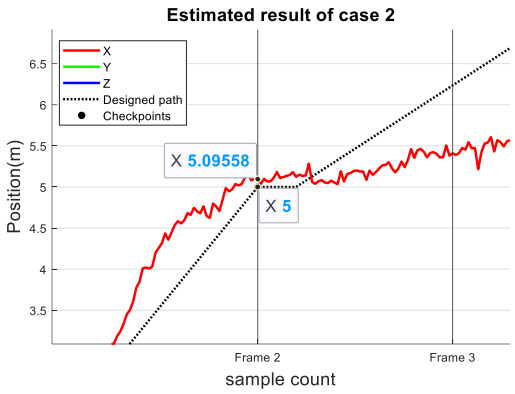
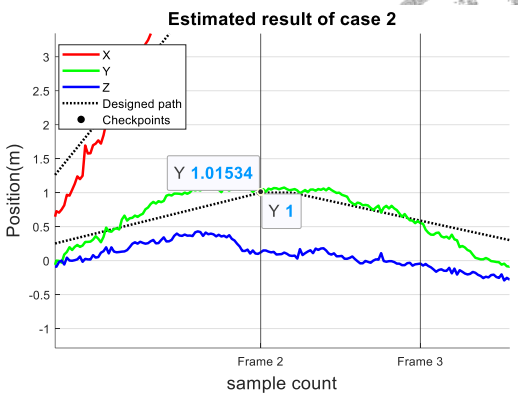


Figure 5.41: Estimated trajectory of case 2

doi:10.6342/NTU202402036

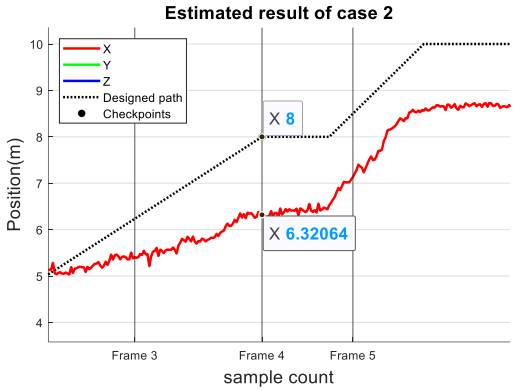


(a) 0.096 m error in X-direction at checkpoint (5,1)

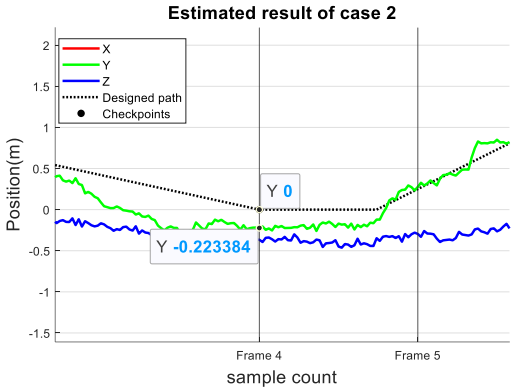


(b) 0.015 m error in Y-direction at checkpoint (5,1)

Figure 5.42: Estimated result at checkpoint (5,1)

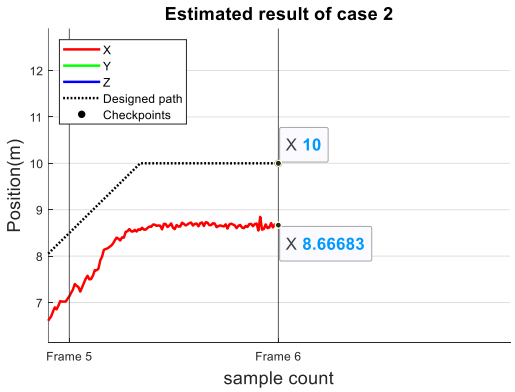


(a) 1.679 m error in X-direction at checkpoint (8,0)

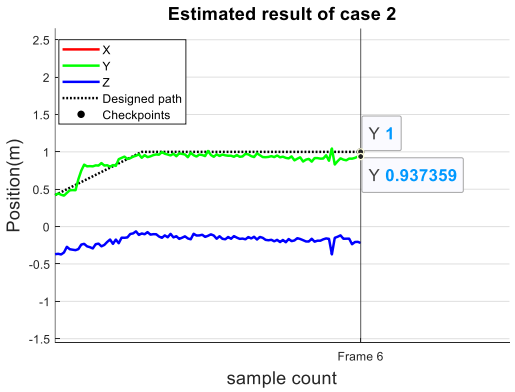


(b) 0.223 m error in Y-direction at checkpoint (8,0)

Figure 5.43: Estimated result at checkpoint (8,0)



(a) 1.333 m error in X-direction at checkpoint (10,1)



(b) 0.063 m error in Y-direction at checkpoint (10,1)

Figure 5.44: Estimated result at checkpoint (10,1)

doi:10.6342/NTU202402036

Case 3: Moving straight along building facade in a noisy environment

The case 3 tests the proposed algorithm with the camera moves in a straight line along a building, the distance from camera to the building is about 7 meters, and there are some bicycles between the camera and the building. The experiment scene is shown in Figure 5.45, the camera frames during estimation are shown in Figure 5.46, the estimated trajectory is shown in Figure 5.47. The estimated result shows that the propose algorithm can identify features in noisy environments and estimate the camera motion successfully.

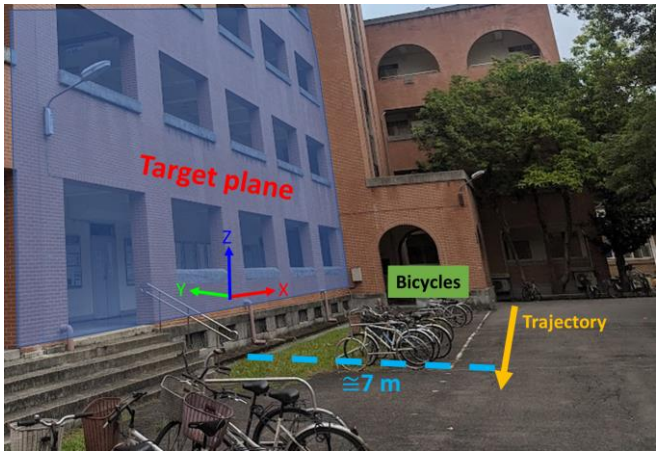


Figure 5.45: The experiment scene of case 3

(a) Frame 1

(b) Frame 2

(c) Frame 3

(d) Frame 4

(e) Frame 5

(f) Frame 6

Figure 5.46: Camera frames from case 3 during estimation

doi:10.6342/NTU202402036

Figure 5.47: Estimated trajectory of case 3

Case 4: Moving straight along building facade with window reflections

Case 4 tests the proposed algorithm along the building facade, the target plane is the windows with reflections on the building facade. As shown in Figure 5.48, the windows reflect the view with fences, trees and the road on the opposite side. the camera frames during estimation are shown in Figure 5.49, the estimated trajectory is shown in Figure 5.50. The proposed algorithm can identify the correct structural lines of the facade, without being disrupted by reflected scenes.

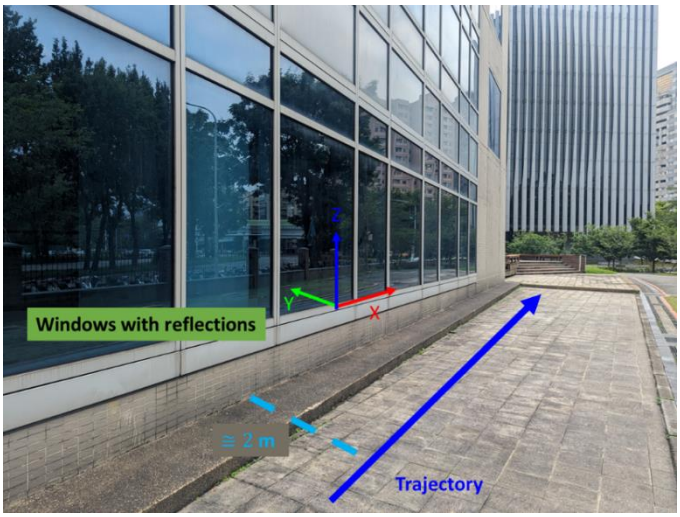


Figure 5.48: The experiment scene of case 4

doi:10.6342/NTU202402036

(a) Frame 1 (b) Frame 2 (c) Frame 3

(d) Frame 4 (e) Frame 5 (f) Frame 6

Figure 5.49: Camera frames from case 4 during estimation

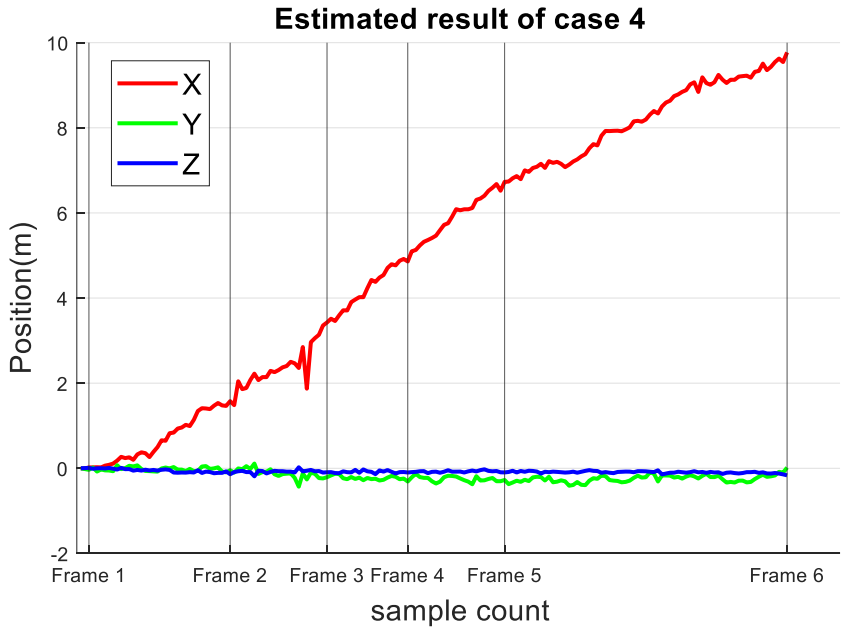


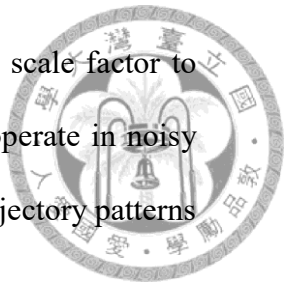
Figure 5.50: Estimated trajectory of case 4

5.2.4 Summary of Real-World Experiments

In this section, the proposed algorithm is verified in real-world environments. In Section 5.2.2, the indoor hallway experiments with marker-based ground truth demonstrate that the algorithm can operate in real-time demand and perform well in real-world scenarios with average RMSE of 0.512 m (2.01%). And the outdoor experiments

doi:10.6342/NTU202402036

are tested in Section 5.2.3, despite the lack of precise ground truth and scale factor to quantify performance, the results indicate the proposed algorithm can operate in noisy environments such like sunlight, obstacles and reflections, resulting in trajectory patterns that are consistent with camera motion.



doi:10.6342/NTU202402036

Chapter 6
Conclusions and Future Works

In this chapter, the conclusions of the thesis and potential improvements are discussed in Section 6.1 and Section 6.2.

6.1 Conclusions

In this thesis, a monocular visual odometry is utilized for localization in urban environments by tracking the grid mesh plane made up of horizontal and vertical structural lines. The proposed algorithm is validated through simulations, indoor and outdoor experiments, demonstrating its ability to estimate in real-time.

In the simulations, the simulation results show that the algorithm is able to estimate in noisy environments. And with adding reliability testing, it can make the algorithm estimate in real-time demand without sacrifice the estimated accuracy, even enhance the smoothness of the estimated trajectory.

In the real-world experiments, the estimated results from indoor experiments are compared with the ground truth generated by marker-based estimation, the comparison results show that the proposed algorithm performs well in real-world estimation. The outdoor experiments in 3 different scenarios show that the proposed algorithm is able to operate in complex environments with interferences like sunlight, obstacles and window reflections. Unfortunately, because of wind and light interference, the outdoor experiments do not have marker-based ground truth to quantify performance of the proposed algorithm in outdoor estimation, but the trajectory patterns are still consistent with the camera motion during estimation.

In generally, the estimated method of the proposed algorithm is similar to the marker-based method, so the performance of the proposed algorithm is compared to the marker-

doi:10.6342/NTU202402036

based estimation, shows that it can perform almost same performance without the need for additional markers in the scene. As for the disadvantages of the proposed algorithm, it requires structural lines that can form rectangles to estimate the camera pose, the application scenarios are very limited, some possible applications are patrol robot moving in warehouses or around buildings, and there is still some work to improve the algorithm so that it can be used in more scenarios, the potential improvements are discussed in Section 6.2.

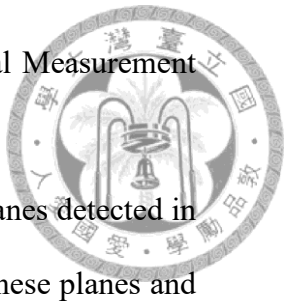
For the number of features needed and the execution time, the proposed algorithm can estimate the camera pose using only 20 to 30 lines and execute at 20Hz in Python environment without other optimization in computing speed. According the experiments in [12: Mur-Artal et al. 2015], ORB-SLAM is set to extract 1000 corner points in similar image resolution to ensure it can calculate the camera pose. In the system proposed in [8: Zhou et al. 2015], the feature points and feature lines are limited to 40 points and 24 lines, the authors implemented their system in MATLAB, which executes at about 2Hz, whereas the C++ implementation runs at an average of about 40Hz. Considering the number of extracted features, if the proposed algorithm can be implemented in more efficient programming languages such as C/C++, it will be more competitive in lower budget applications since fewer features are required.

6.2 Future Works

Although the proposed algorithm can track the grid mesh plane and localize the camera trajectory, there are some issues that need solving to enhance performance and apply it in a wider range of scenarios.

First issue is measuring the real-world scale factor. The current method is to manually measure the distance to the target plane and input the measurement into the algorithm before starting the estimation. The process can be completed automatically by

doi:10.6342/NTU202402036



adding additional sensors such as Time-of-Flight (ToF) sensor or Inertial Measurement Unit (IMU) to calculate the precise value.

Second issue is the number of target planes, in Section 4.4, if the planes detected in camera frame are not the stored target plane, the algorithm will discard these planes and only retains the information about the stored target plane. The process could be improved by storing the planes detected in camera frames, creating a map contains different planes and estimate camera pose with the planes individually, and then refine the results using filtering methods. It allows the algorithm to estimate in more scenarios without being constrained by moving along the same plane.

The third issue continues the second issue, if the algorithm could store complete information about the features, the loop closure process could be added to the algorithm to increase the estimate stability and long term accuracy.

doi:10.6342/NTU202402036

References

[1: Vatansever & Butun 2017]

Saffet Vatansever and Ismail Butun, "A broad overview of GPS fundamentals: Now and future," 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), pp. 1–6, Jan. 2017.

[2: Hutt et al. 2021]

Oliver Hutt, Kate Bowers, and Shane Johnson, "The effect of GPS refresh rate on measuring police patrol in micro-places," Crime Science, Vol. 10, Feb. 2021.

[3: Strandjord et al. 2020]

Kirsten L. Strandjord, Penina Axelrad, and Shan Mohiuddin, "Improved urban navigation with shadow matching and specular matching," NAVIGATION: Journal of the Institute of Navigation, Vol. 67, No. 3, Art. no. 3, Sep. 2020.

[4: 廖育萱 & 彭彥嘉 2023]

廖育萱 & 彭彥嘉, "農用戶外自主運行機器人定位技術介紹," 機械工業雜誌, No. 485, Art. no. 485, Aug. 2023.

[5: Liao *et al.* 2023]

Yiyi Liao, Jun Xie, and Andreas Geiger, "KITTI-360: A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 45, No. 3, pp. 3292–3310, Mar. 2023.

[6: Yin et al. 2022]

Jie Yin, Ang Li, Tao Li, Wenxian Yu, and Danping Zou, "M2DGR: A Multi-Sensor and Multi-Scenario SLAM Dataset for Ground Robots," IEEE Robotics and Automation Letters, Vol. 7, No. 2, pp. 2266–2273, Apr. 2022.

[7: Liu et al. 2022]

Zhe Liu, Dianxi Shi, Ruihao Li, Wei Qin, Yongjun Zhang, and Xiaoguang Ren, "PLC-VIO: Visual–Inertial Odometry Based on Point-Line Constraints," IEEE Transactions on Automation Science and Engineering, Vol. 19, No. 3, Art. no. 3, Jul. 2022.

[8: Zhou et al. 2015]

Huizhong Zhou, Danping Zou, Ling Pei, Rendong Ying, and Peilin Liu,

doi:10.6342/NTU202402036

"StructSLAM: Visual SLAM With Building Structure Lines," IEEE Transactions on Vehicular Technology, Vol. 64, No. 4, Art. no. 4, Apr. 2015.

[9: Lu et al. 2000]

C.-P. Lu, G.D. Hager, and E. Mjolsness, "Fast and globally convergent pose estimation from video images," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 6, Art. no. 6, Jun. 2000.

[10: Nister 2004]

D. Nister, "An efficient solution to the five-point relative pose problem," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 26, No. 6, pp. 756–770, Jun. 2004.

[11: Choi et al. 2013]

Sunglok Choi, Jaehyun Park, and Wonpil Yu, "Resolving scale ambiguity for monocular visual odometry," 2013 10th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), pp. 604–608, Oct. 2013.

[12: Mur-Artal et al. 2015]

Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," IEEE Transactions on Robotics, Vol. 31, No. 5, pp. 1147–1163, Oct. 2015.

[13: Li et al. 2020]

Yanyan Li, Nikolas Brasch, Yida Wang, Nassir Navab, and Federico Tombari, "Structure-SLAM: Low-Drift Monocular SLAM in Indoor Environments," IEEE Robotics and Automation Letters, Vol. 5, No. 4, pp. 6583–6590, Oct. 2020.

[14: Qin et al. 2018]

Tong Qin, Peiliang Li, and Shaojie Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," IEEE Transactions on Robotics, Vol. 34, No. 4, Art. no. 4, Aug. 2018.

[15: Lin et al. 2022]

Zihan Lin, Jincheng Yu, Lipu Zhou, Xudong Zhang, Jian Wang, and Yu Wang, "Point Cloud Change Detection With Stereo V-SLAM: Dataset, Metrics and Baseline," IEEE Robotics and Automation Letters, Vol. 7, No. 4, pp. 12443–12450, Oct. 2022.

[16: Zhang et al. 2015]

Guoxuan Zhang, Jin Han Lee, Jongwoo Lim, and Il Hong Suh, "Building a 3-D Line-Based Map Using Stereo SLAM," IEEE Transactions on Robotics, Vol. 31, No. 6, pp. 1364–1377, Feb. 2015.

doi:10.6342/NTU202402036

[17: Lin et al. 2023]

Xi Lin, Yewei Huang, Dingyi Sun, Tzu-Yuan Lin, Brendan Englot, Ryan M. Eustice, Maani Ghaffari, "A Robust Keyframe-Based Visual SLAM for RGB-D Cameras in Challenging Scenarios," IEEE Access, Vol. 11, pp. 97239–97249, 2023.

[18: Cheng et al. 2023]

Shuhong Cheng, Changhe Sun, Shijun Zhang, and Dianfan Zhang, "SG-SLAM: A Real-Time RGB-D Visual SLAM Toward Dynamic Scenes With Semantic and Geometric Information," IEEE Transactions on Instrumentation and Measurement, Vol. 72, pp. 1–12, 2023.

[19: Zhao et al. 2019]

Lili Zhao, Zhili Liu, Jianwen Chen, Weitong Cai, Wenyi Wang, and Liaoyuan Zeng, "A Compatible Framework for RGB-D SLAM in Dynamic Scenes," IEEE Access, Vol. 7, pp. 75604–75614, 2019.

[20: Weikersdorfer et al. 2013]

David Weikersdorfer, Raoul Hoffmann, and Jörg Conradt, "Simultaneous Localization and Mapping for Event-Based Vision Systems," in Proceedings of Computer Vision Systems, Berlin, Heidelberg, pp. 133–142, 2013.

[21: Rebecq et al. 2017]

Henri Rebecq, Timo Horstschaefer, Guillermo Gallego, and Davide Scaramuzza, "EVO: A Geometric Approach to Event-Based 6-DOF Parallel Tracking and Mapping in Real Time," IEEE Robotics and Automation Letters, Vol. 2, No. 2, pp. 593–600, Apr. 2017.

[22: Engel et al. 2014]

Jakob Engel, Thomas Schöps, and Daniel Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," Computer Vision – ECCV 2014, Cham, pp. 834–849, 2014.

[23: Silveira et al. 2008]

Geraldo Silveira, Ezio Malis, and Patrick Rives, "An Efficient Direct Approach to Visual SLAM," IEEE Transactions on Robotics, Vol. 24, No. 5, pp. 969–979, Oct. 2008.

[24: Forster et al. 2017]

Christian Forster, Zichao Zhang, Michael Gassner, Manuel Werlberger, and Davide Scaramuzza, "SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems," IEEE Transactions on Robotics, Vol. 33, No. 2, Art. no. 2, Apr. 2017.

doi:10.6342/NTU202402036

[25: Campos et al. 2021]

Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel, and Juan D. Tardós, "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM," IEEE Transactions on Robotics, Vol. 37, No. 6, pp. 1874–1890, Feb. 2021.

[26: Rublee et al. 2011]

Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski, "ORB: An efficient alternative to SIFT or SURF," 2011 International Conference on Computer Vision, pp. 2564–2571, Jan. 2011.

[27: Gomez-Ojeda et al. 2019]

Ruben Gomez-Ojeda, Francisco-Angel Moreno, David Zuñiga-Noël, Davide Scaramuzza, and Javier Gonzalez-Jimenez, "PL-SLAM: A Stereo SLAM System Through the Combination of Points and Line Segments," IEEE Transactions on Robotics, Vol. 35, No. 3, pp. 734–746, Jun. 2019.

[28: Guan et al. 2023]

Weipeng Guan, Peiyu Chen, Yuhan Xie, and Peng Lu, "PL-EVIO: Robust Monocular Event-Based Visual Inertial Odometry With Point and Line Features," IEEE Transactions on Automation Science and Engineering, pp. 1–17, 2023.

[29: Sun et al. 2018]

Qinxuan Sun, Jing Yuan, Xuebo Zhang, and Fengchi Sun, "RGB-D SLAM in Indoor Environments With STING-Based Plane Feature Extraction," IEEE/ASME Transactions on Mechatronics, Vol. 23, No. 3, pp. 1071–1082, Jun. 2018.

[30: Gelen & Atasoy 2023]

Aykut G. Gelen and Ayten Atasoy, "An Artificial Neural SLAM Framework for Event-Based Vision," IEEE Access, Vol. 11, pp. 58436–58450, 2023.

[31: Yang & Scherer 2019]

Shichao Yang and Sebastian Scherer, "Monocular Object and Plane SLAM in Structured Environments," IEEE Robotics and Automation Letters, Vol. 4, No. 4, pp. 3145–3152, Oct. 2019.

[32: Liu & Miura 2021]

Yubao Liu and Jun Miura, "RDMO-SLAM: Real-Time Visual SLAM for Dynamic Environments Using Semantic Label Prediction With Optical Flow," IEEE Access, Vol. 9, pp. 106981–106997, 2021.

doi:10.6342/NTU202402036

[33: Ran et al. 2021]

Teng Ran, Liang Yuan, Jianbo Zhang, Dingxin Tang, and Li He, "RS-SLAM: A Robust Semantic SLAM in Dynamic Environments Based on RGB-D Sensor," IEEE Sensors Journal, Vol. 21, No. 18, pp. 20657–20664, Sep. 2021.

[34: Kalman 1960]

R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," Journal of Basic Engineering, Vol. 82, No. 1, pp. 35–45, Mar. 1960.

[35: Bloesch et al. 2015]

Michael Bloesch, Sammy Omari, Marco Hutter, and Roland Siegwart, "Robust visual inertial odometry using a direct EKF-based approach," 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 298–304, Sep. 2015.

[36: Solà 2015]

Joan Solà, "Quaternion kinematics for the error-state KF," Mar. 2015.

[37: Chamorro et al. 2022]

William Chamorro, Joan Solà, and Juan Andrade-Cetto, "Event-Based Line SLAM in Real-Time," IEEE Robotics and Automation Letters, Vol. 7, No. 3, pp. 8146–8153, Jul. 2022.

[38: Mueggler et al. 2018]

Elias Mueggler, Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza, "Continuous-Time Visual-Inertial Odometry for Event Cameras," IEEE Transactions on Robotics, Vol. 34, No. 6, pp. 1425–1440, Feb. 2018.

[39: Grompone von Gioi et al. 2010]

Rafael Grompone von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall, "LSD: A Fast Line Segment Detector with a False Detection Control," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 32, No. 4, pp. 722–732, Apr. 2010.

[40: Akinlar & Topal 2011]

Cuneyt Akinlar and Cihan Topal, "EDLines: A real-time line segment detector with a false detection control," Pattern Recognition Letters, Vol. 32, No. 13, pp. 1633–1642, Oct. 2011.

[41: Lee et al. 2014]

Jin Han Lee, Sehyung Lee, Guoxuan Zhang, Jongwoo Lim, Wan Kyun Chung, and Il Hong Suh, "Outdoor place recognition in urban environments using straight lines,"

doi:10.6342/NTU202402036

2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 5550–5557, May 2014.

[42: Duda & Hart 1972]

Richard O. Duda and Peter E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," Communications of the ACM, Vol. 15, No. 1, pp. 11–15, Jan. 1972.

[43: Ester et al. 1996]

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," Second International Conference on Knowledge Discovery and Data Mining, Portland, Oregon, pp. 226–231, Aug. 2, 1996.

[44: Fischler & Bolles 1981]

Martin A. Fischler and Robert C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," Communications of the ACM, Vol. 24, No. 6, pp. 381–395, Jun. 1981.

[45: Garrido-Jurado et al. 2014]

S. Garrido-Jurado, R. Muñoz-Salinas, F.J. Madrid-Cuevas, and M.J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," Pattern Recognition, Vol. 47, No. 6, pp. 2280–2292, Jun. 2014.

[46: Mueen & Keogh 2016]

Abdullah Mueen and Eamonn Keogh, "Extracting Optimal Performance from Dynamic Time Warping," 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco California USA, pp. 2129–2130, Aug. 13, 2016.

[47: Horn 1987]

Berthold K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," Journal of the Optical Society of America A, Vol. 4, No. 4, p. 629, Apr. 1987.

Books

[48: Limitd 2007]

Trimble Navigation Limitd, "GPS: The First Global Navigation Satellite System," Trimble, 2007

doi:10.6342/NTU202402036

Websites

[49: Penn State]

Penn State, "17. Satellite Ranging," Penn State's Online Geospatial Education. [Online]. Available: https://www.e-education.psu.edu/natureofgeoinfo/book/export/html/1796.

[50: Tesla]

Tesla, "Tesla Vision Update: Replacing Ultrasonic Sensors with Tesla Vision," [Online]. Available: https://www.tesla.com/support/transitioning-tesla-vision

[51: It's Only Electric]

It's Only Electric, "Tesla Vision vs Parking sensors - Was it worth the wait?" YouTube. [Online]. Available: https://www.youtube.com/watch?v=-w1NS_eDz5k

[52: Savarese & Bohg 2023]

Silvio Savarese, Jeannette Bohg, "CS231A: Computer Vision, From 3D Reconstruction to Recognition." Stanford University, 2023. [Online]. Available: https://web.stanford.edu/class/cs231a/

[53: Collins 2007]

Robert Collins, "CSE486 Computer Vision I." Penn State University, 2007. [Online]. Available: https://www.cse.psu.edu/~rtc12/CSE486/

[54: Wikipedia]

"Homogeneous coordinates," Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Homogeneous_coordinates

[55: Intel]

"Depth Camera D455," Intel. [Online]. Available: https://www.intelrealsense.com/depth-camera-d455/

[56: OpenCV]

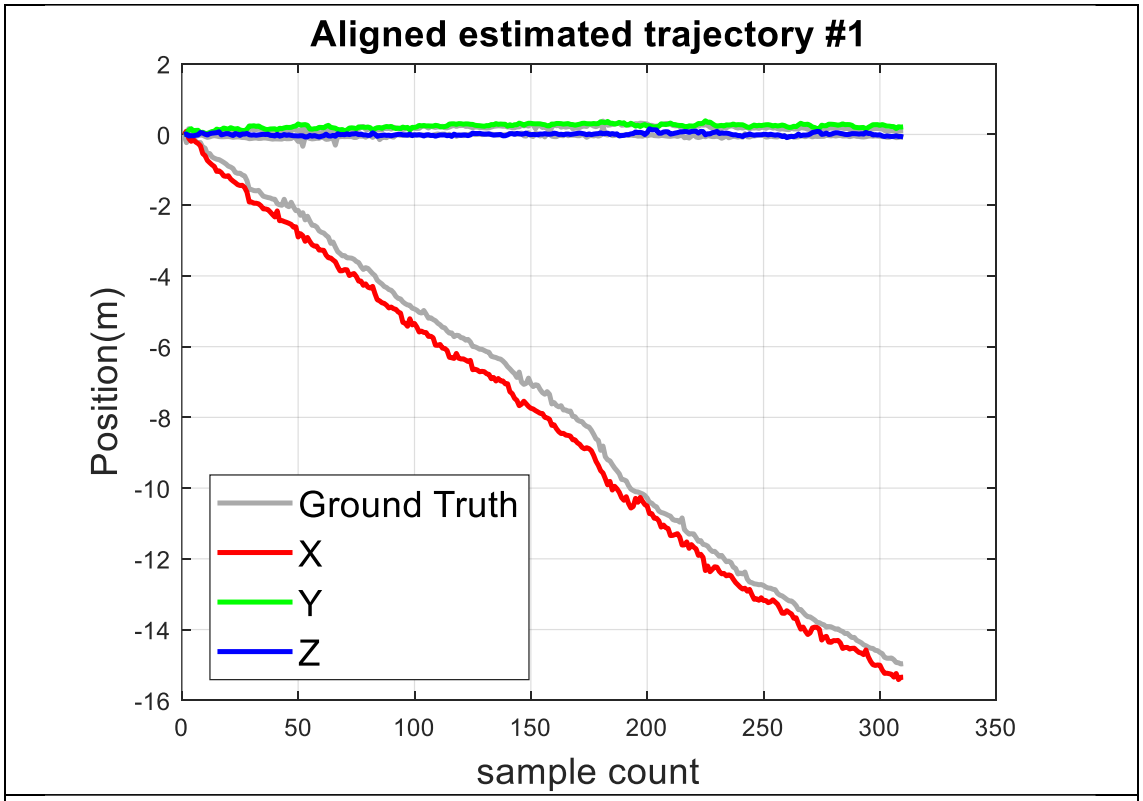
"OpenCV: Detection of ArUco Markers." OpenCV. [Online]. Available: https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html

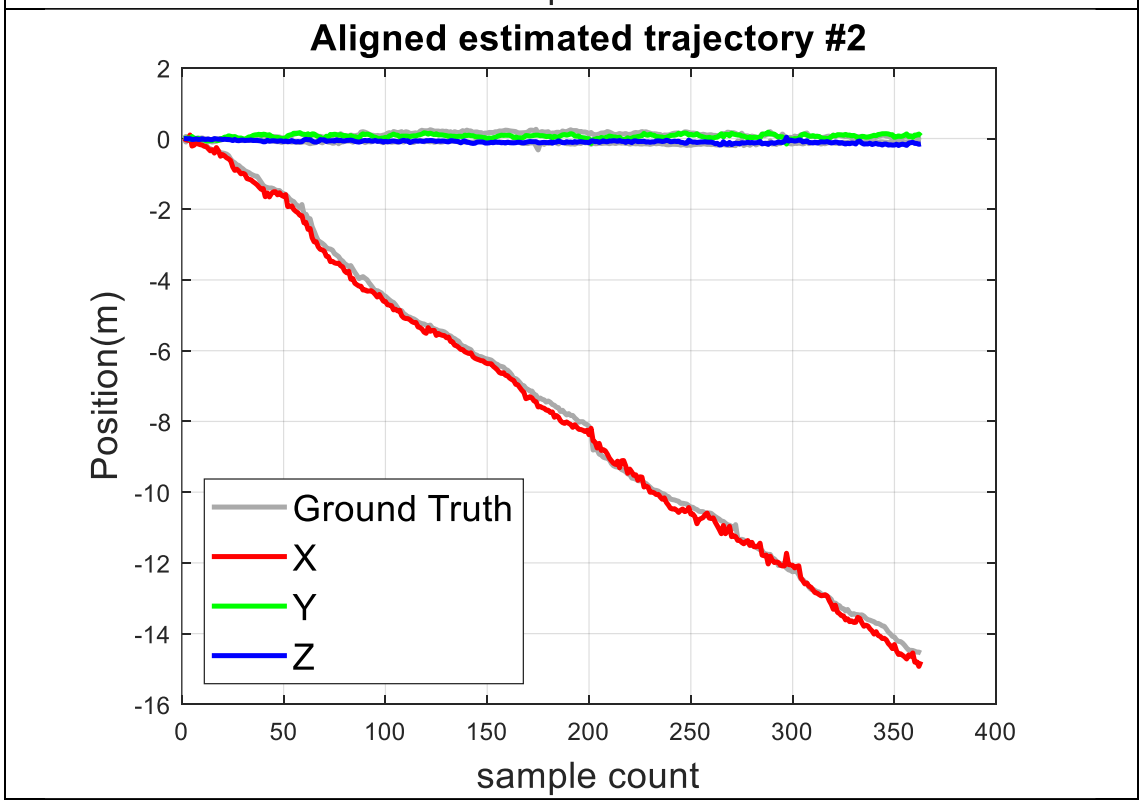
[57: Mur-Artal]

Mur-Artal, "evaluate_ate_scale." Github. [Online]. Available: https://github.com/raulmur/evaluate_ate_scale

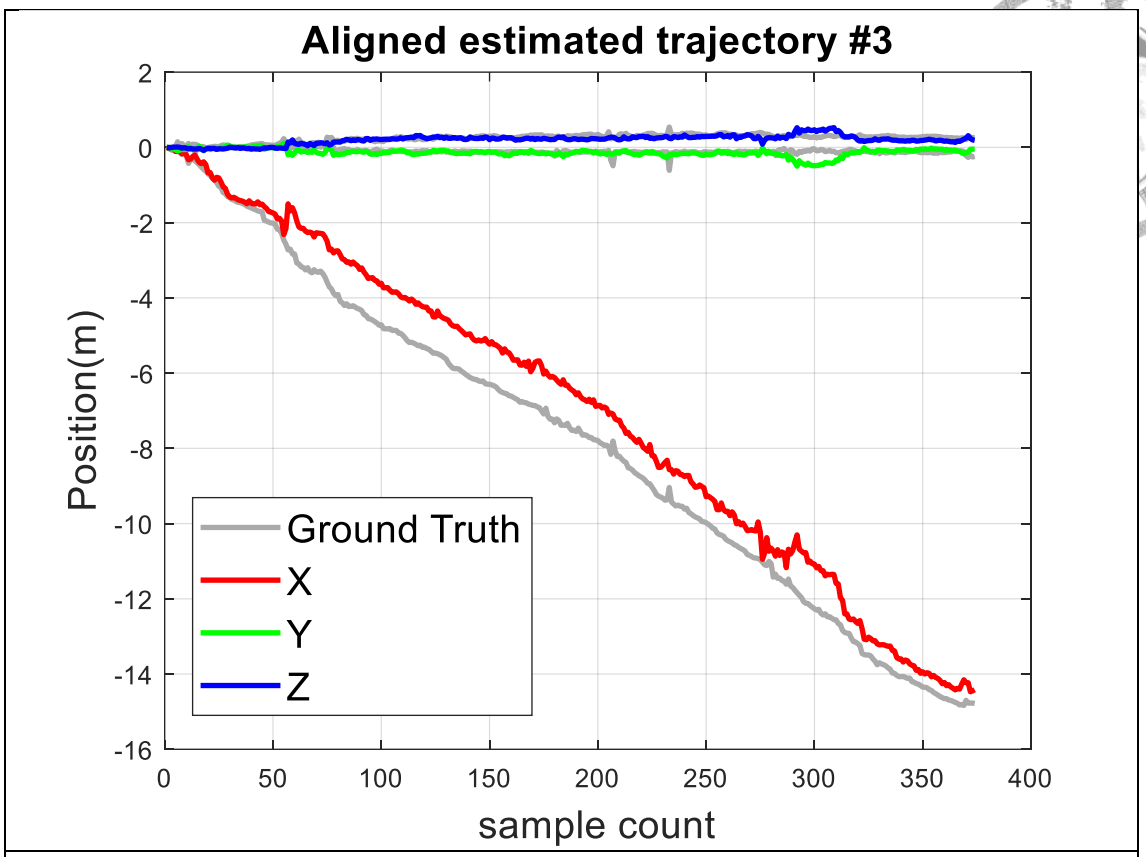
doi:10.6342/NTU202402036

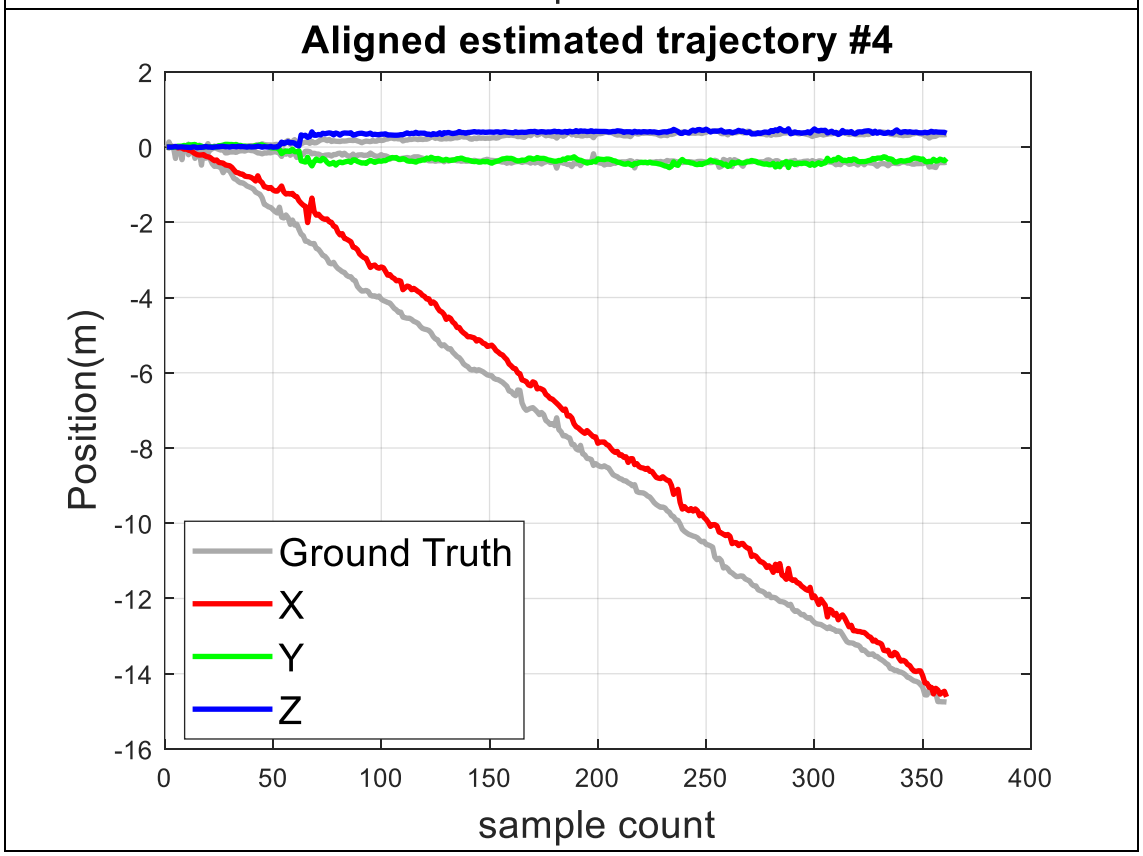
Appendix A
Estimated Results of Indoor Hallway Experiments



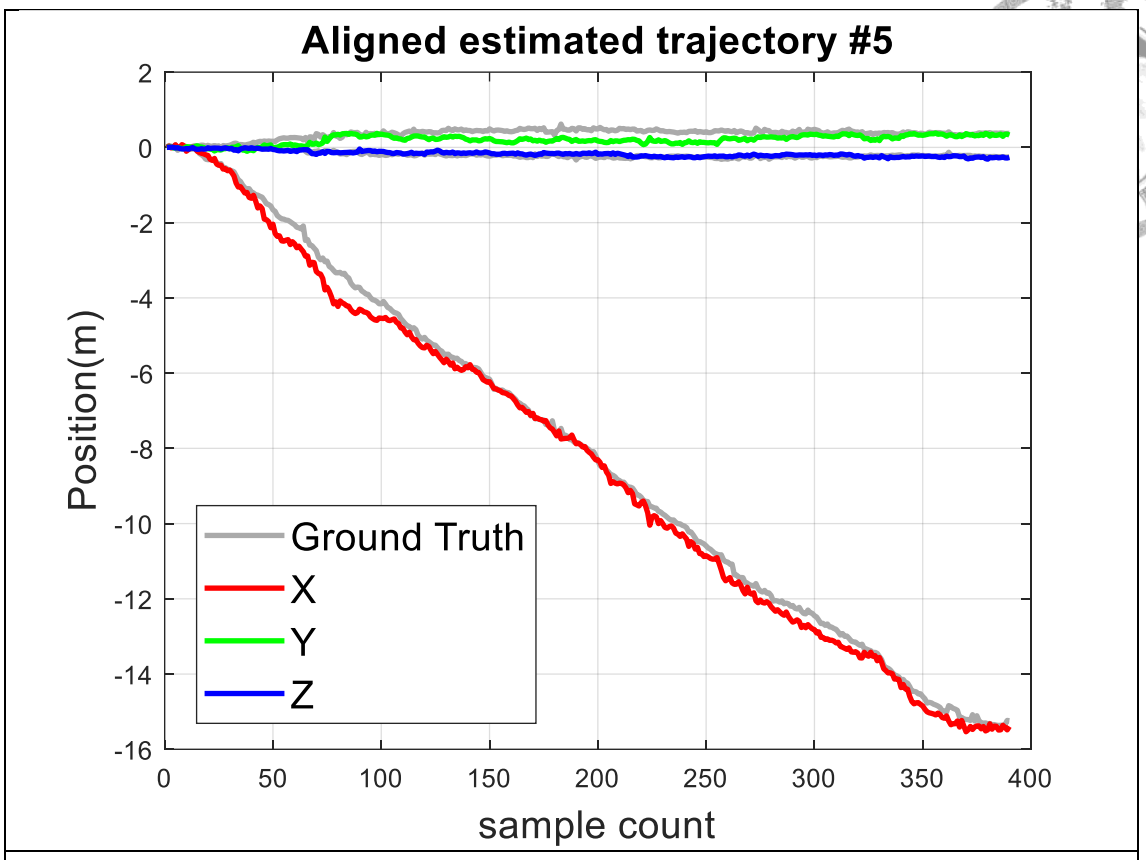


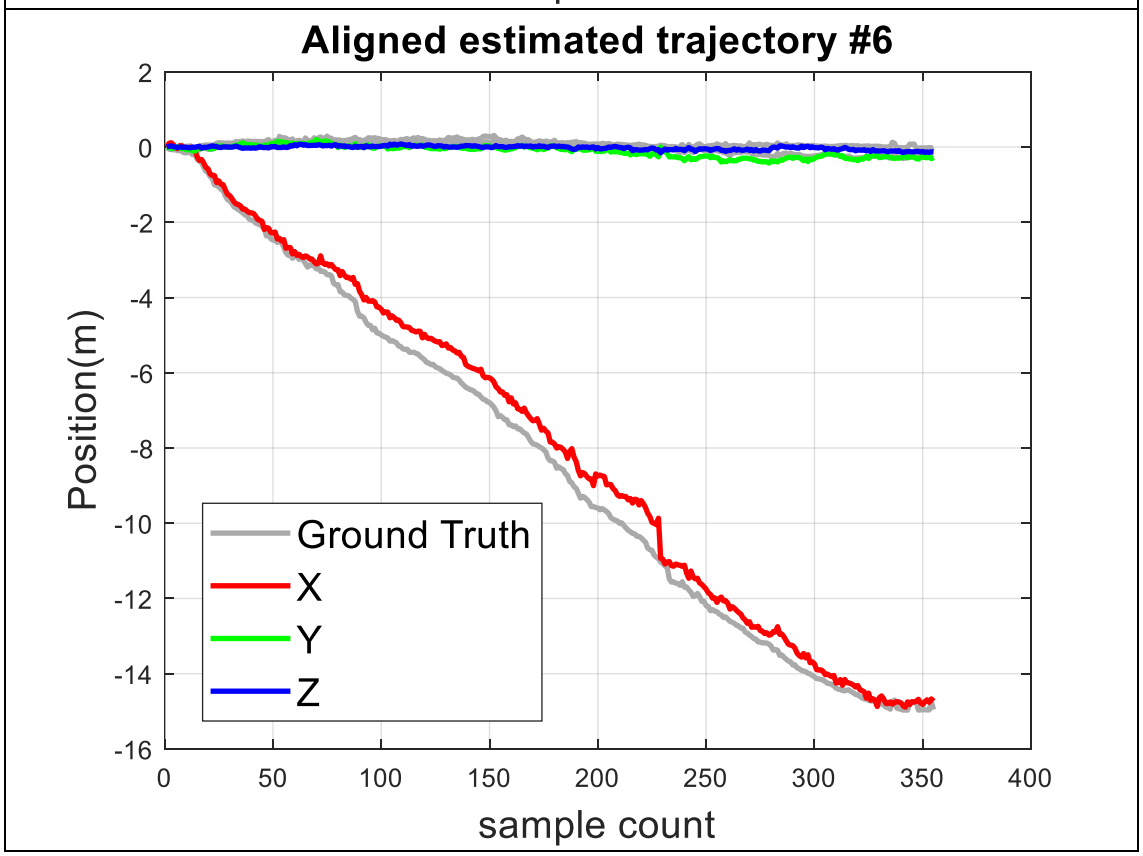
doi:10.6342/NTU202402036



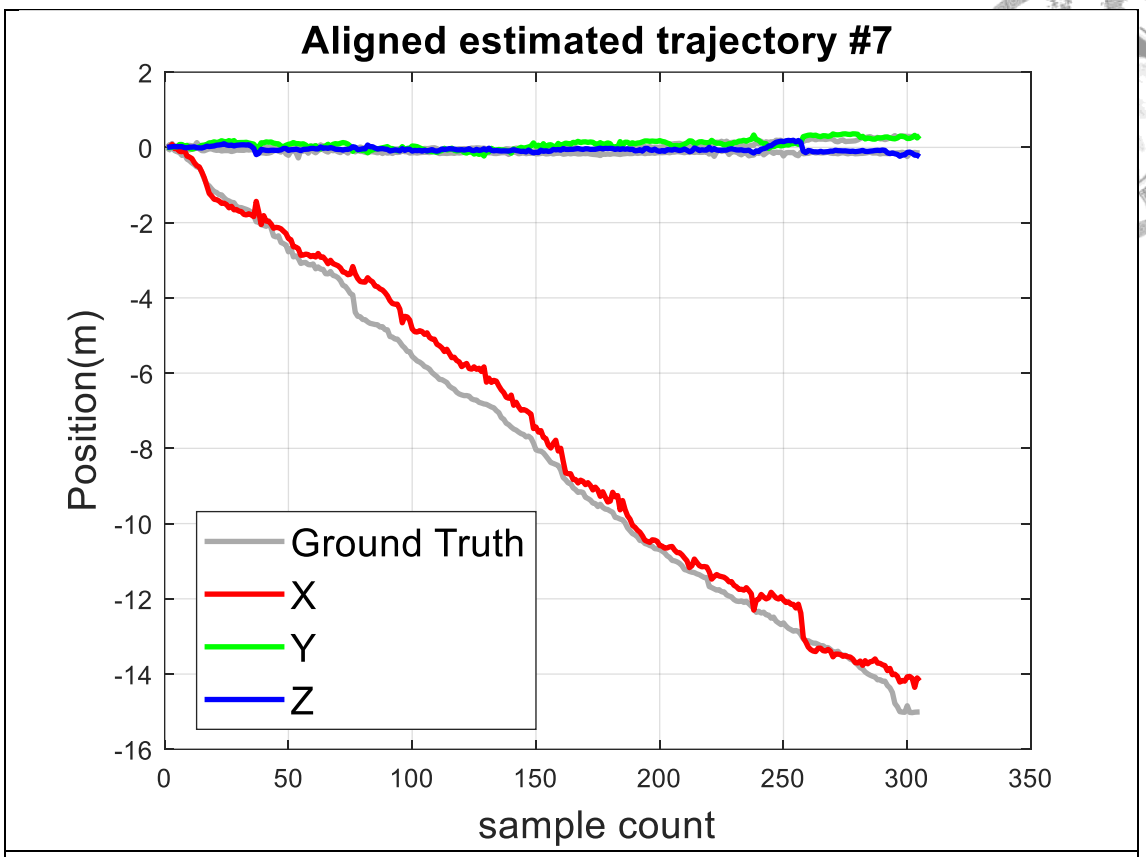


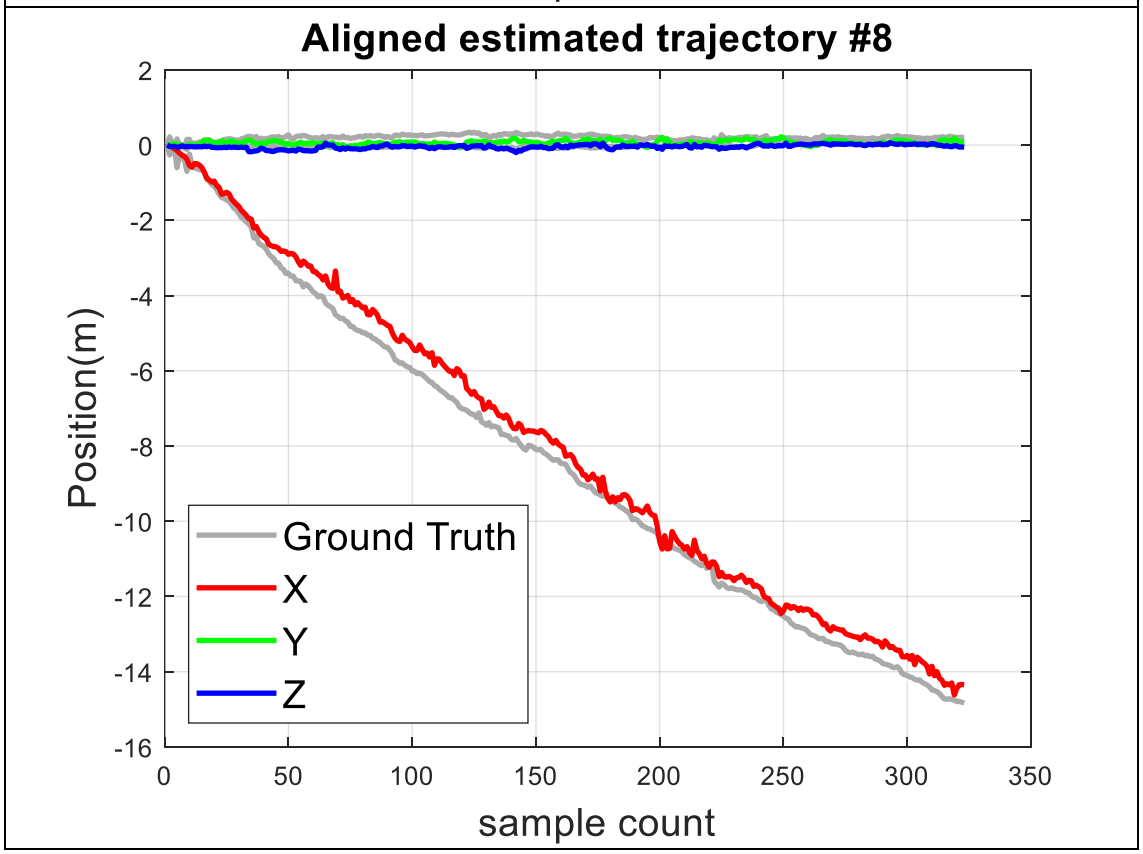
doi:10.6342/NTU202402036





doi:10.6342/NTU202402036





doi:10.6342/NTU202402036

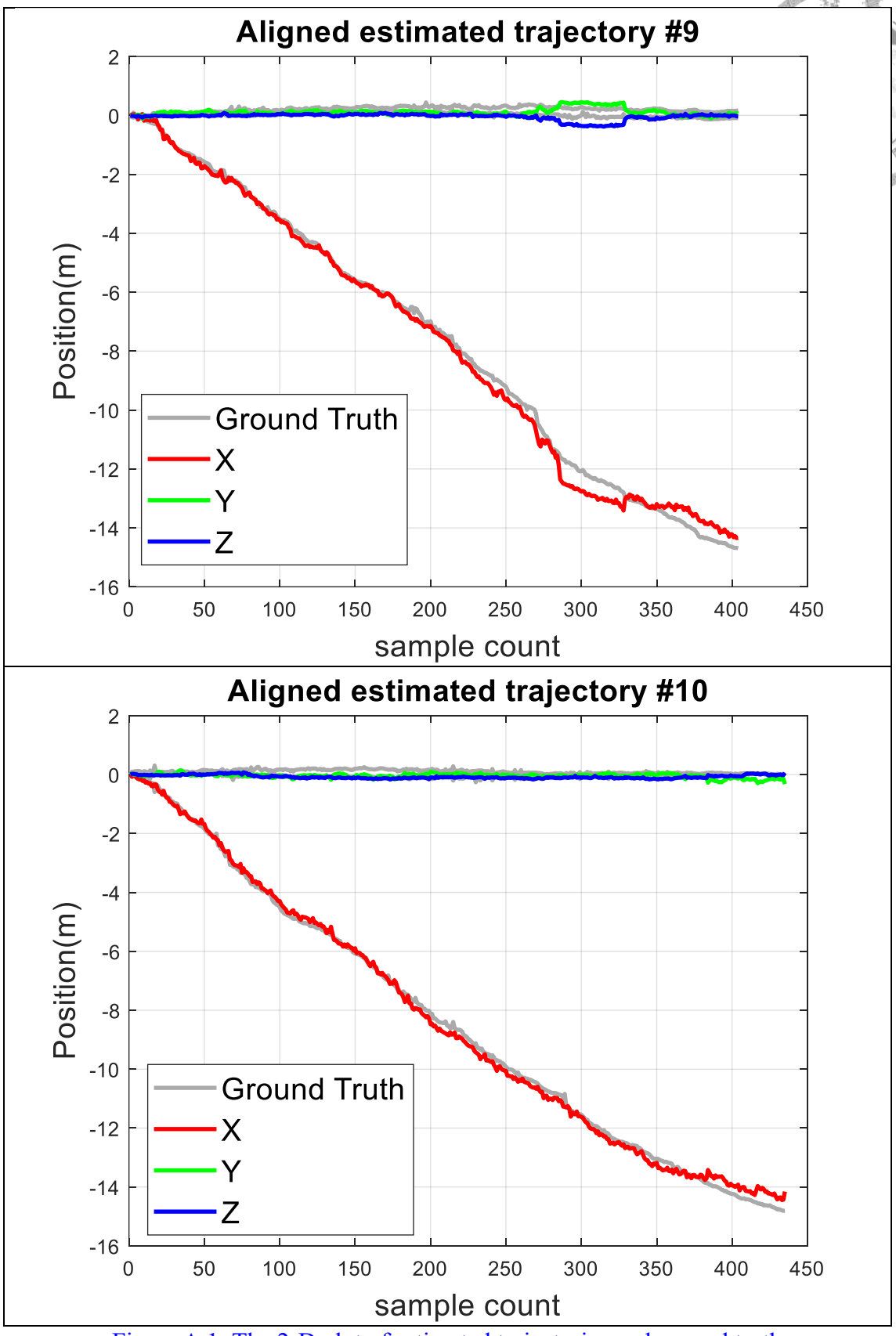
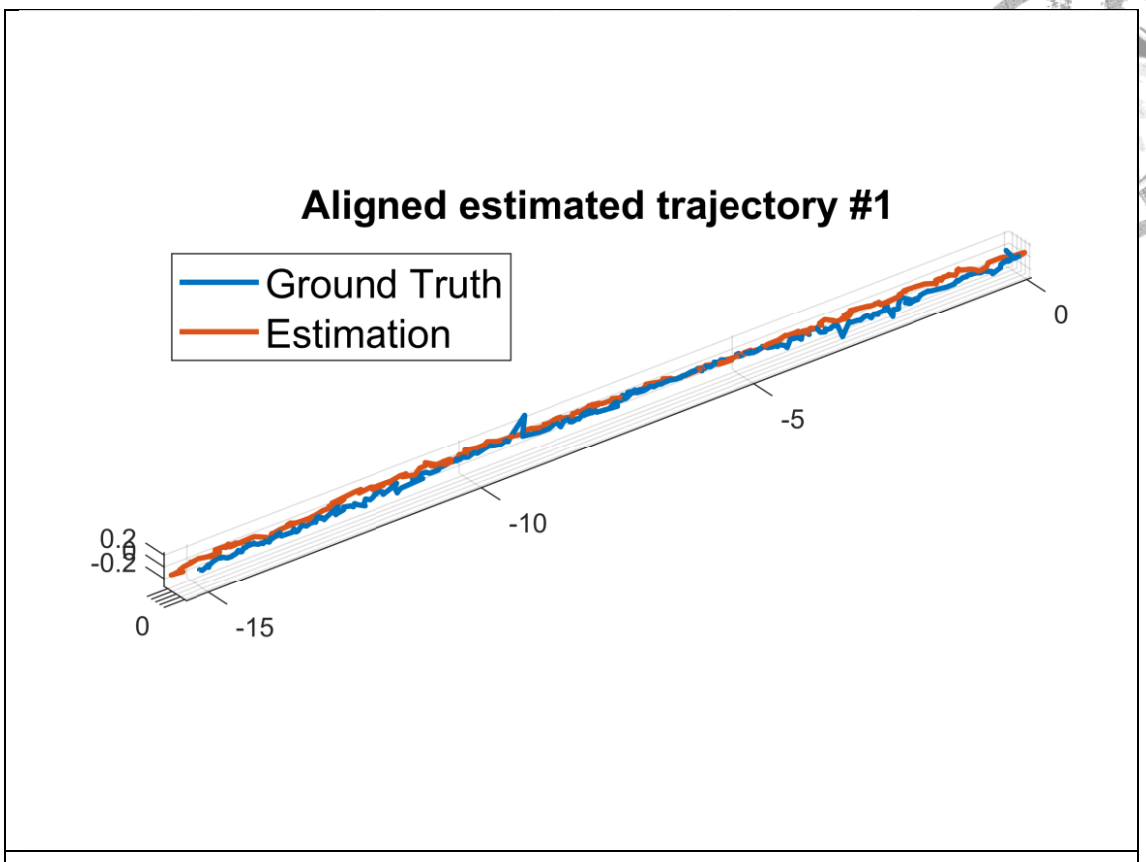
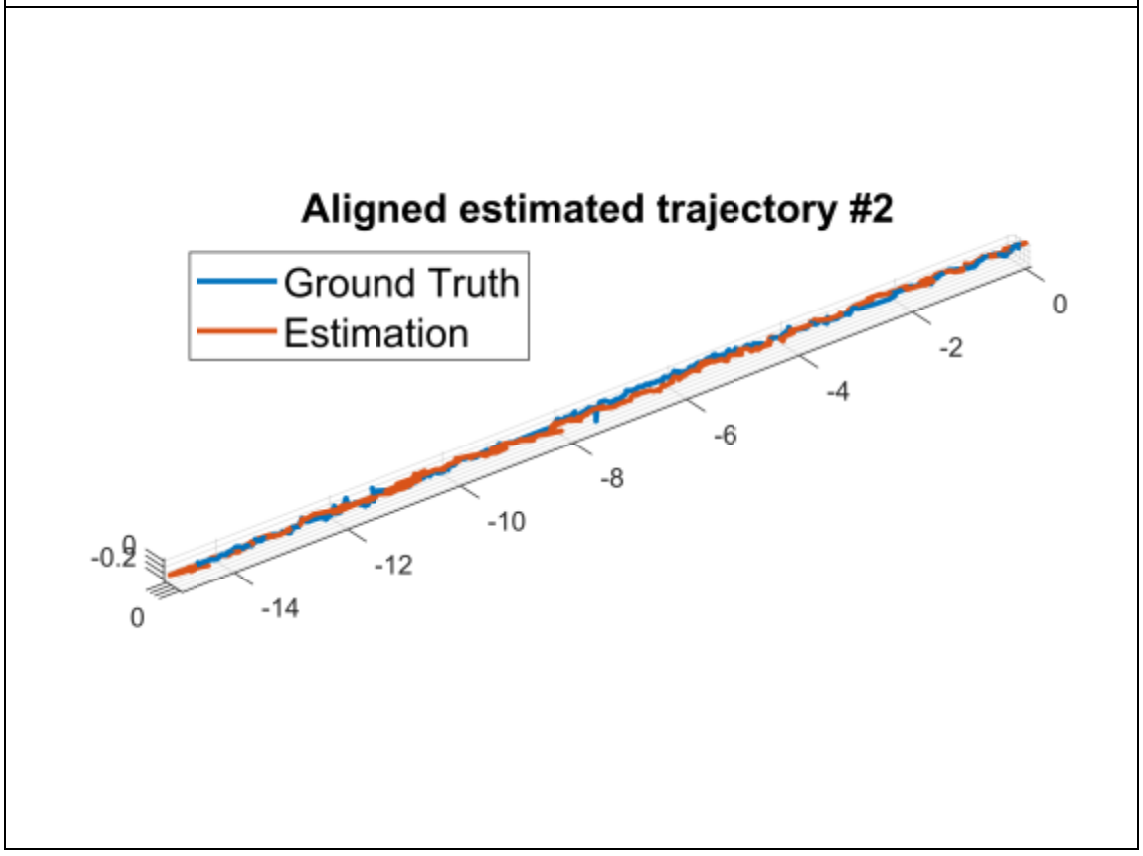


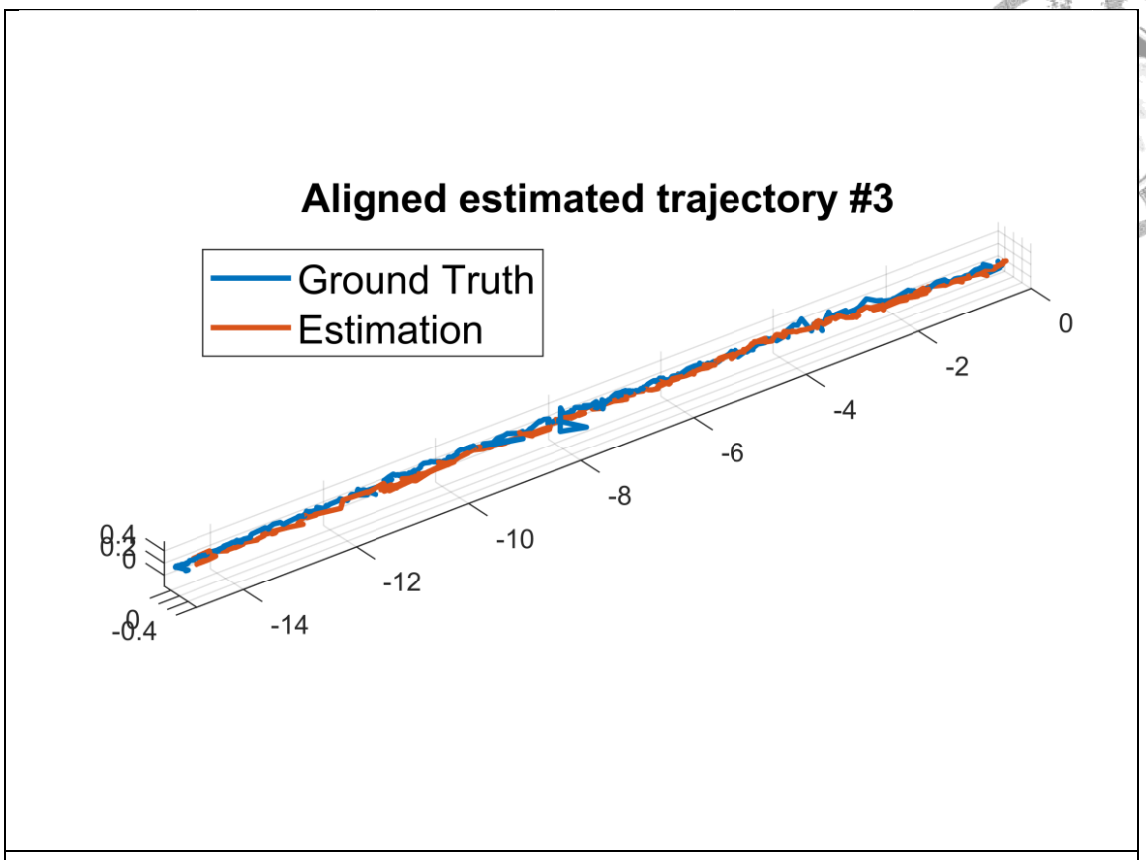
Figure A.1: The 2-D plot of estimated trajectories and ground truth

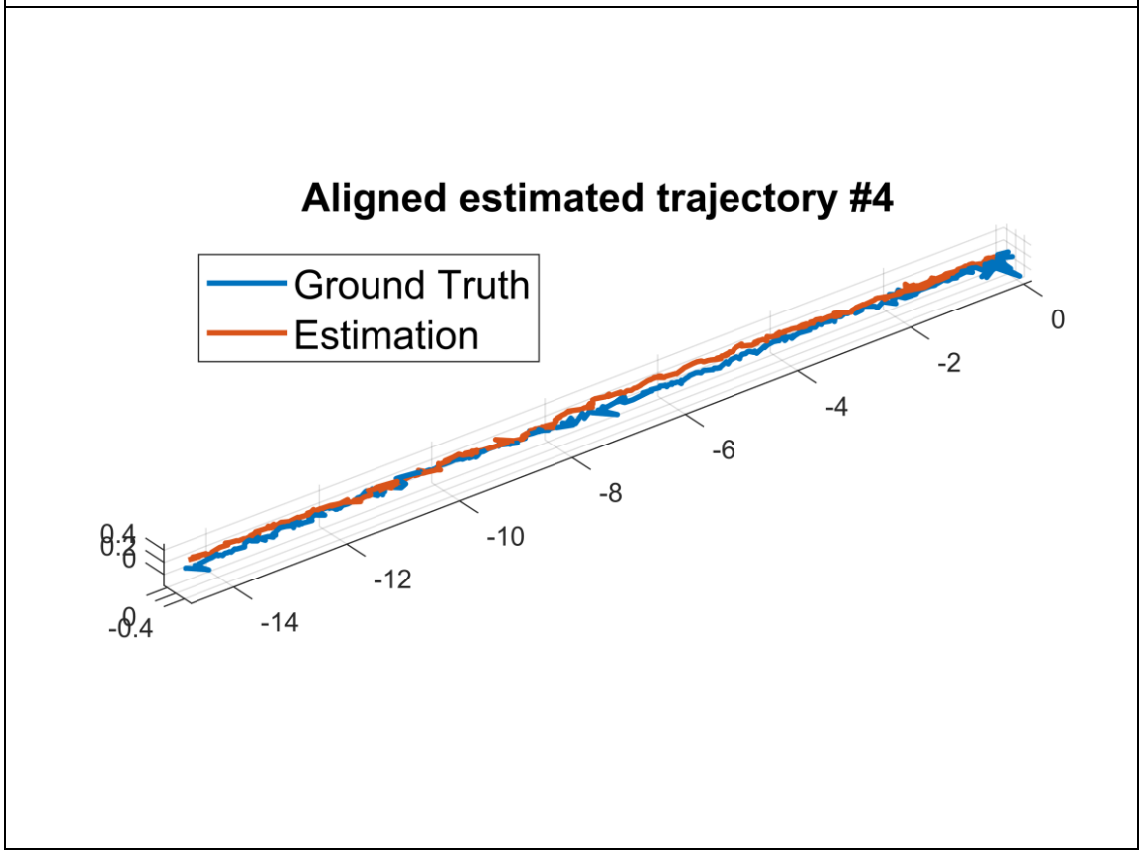
doi:10.6342/NTU202402036



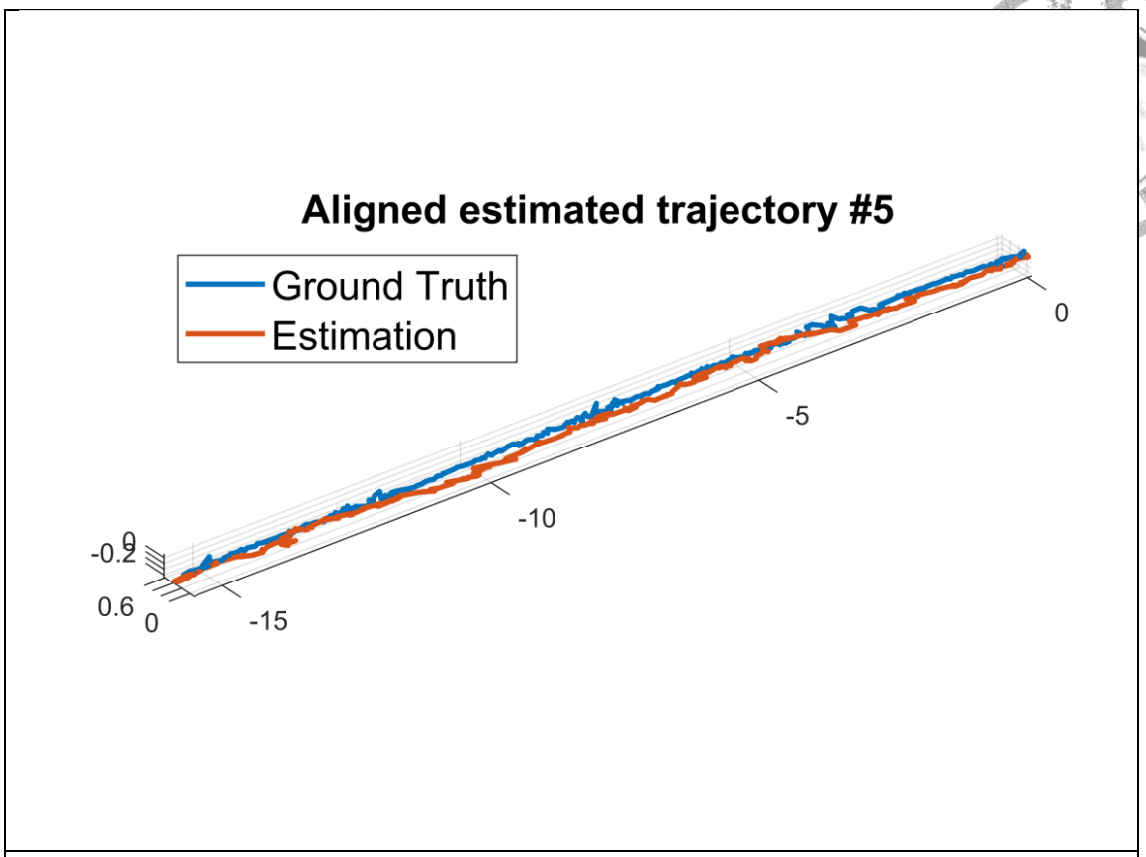


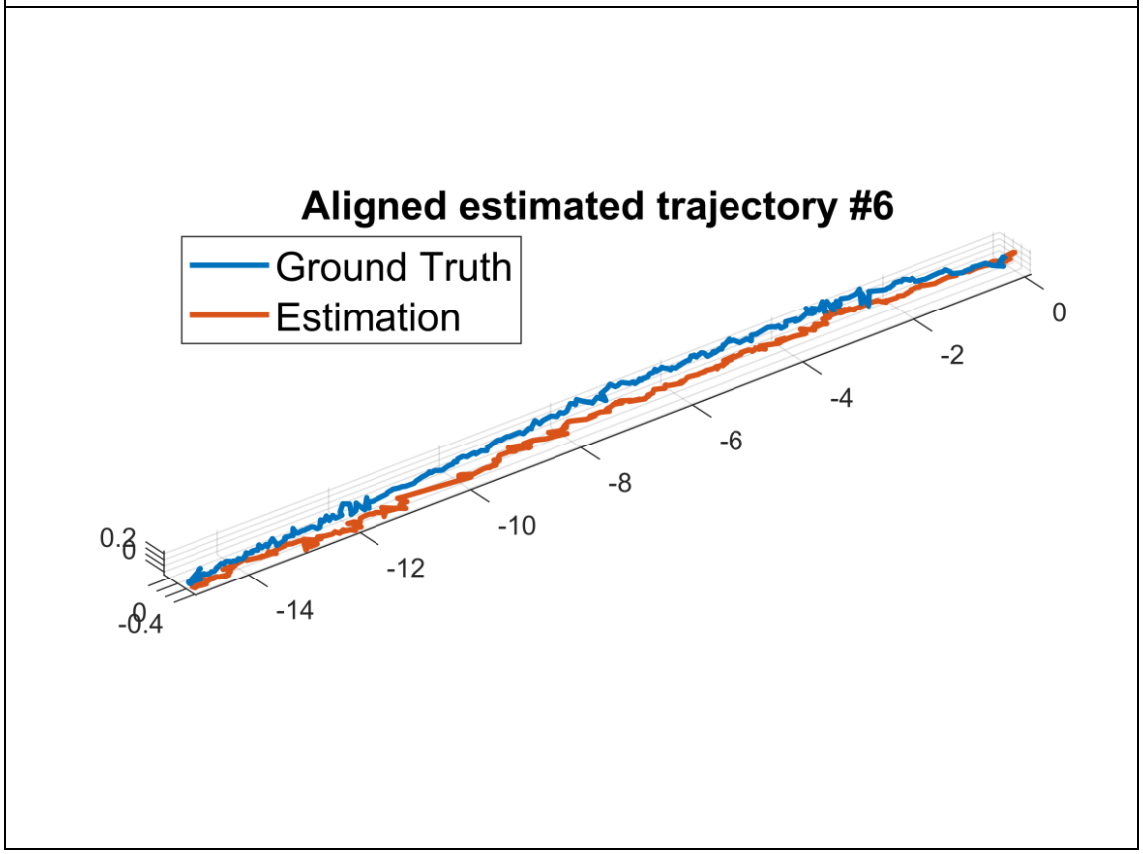
doi:10.6342/NTU202402036



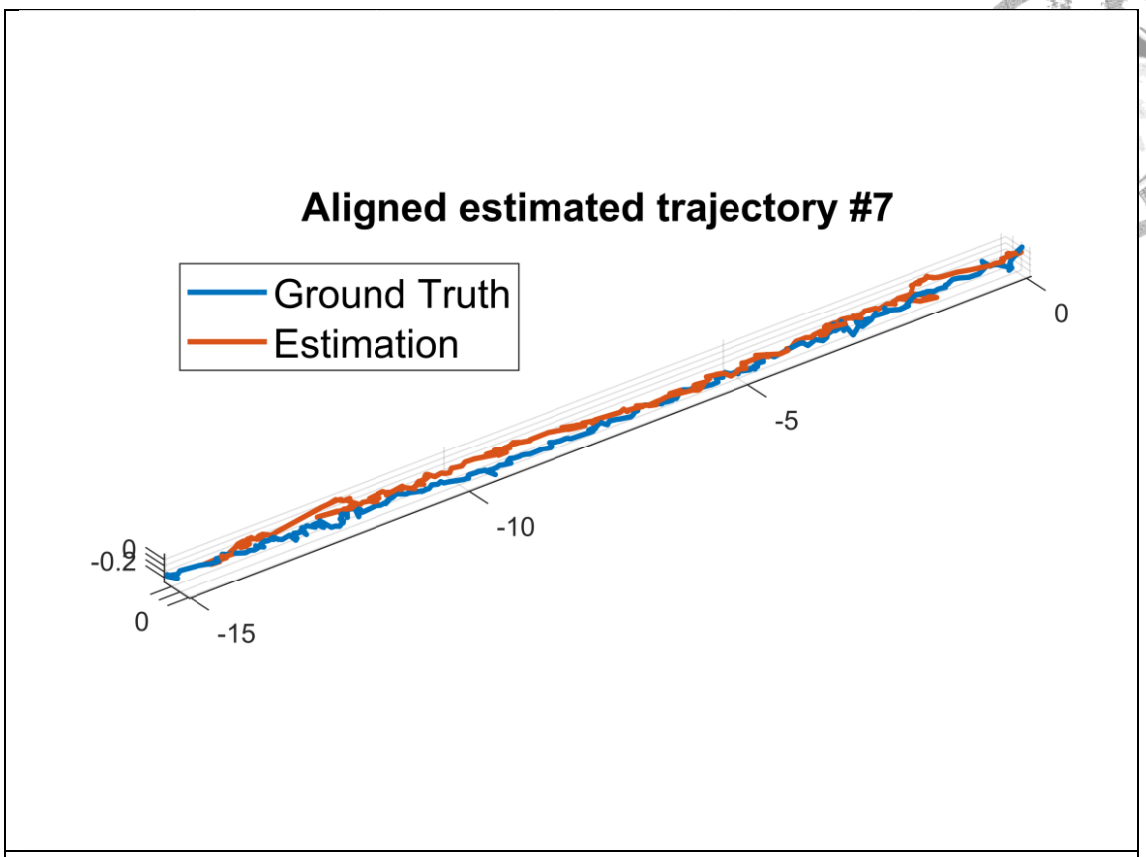


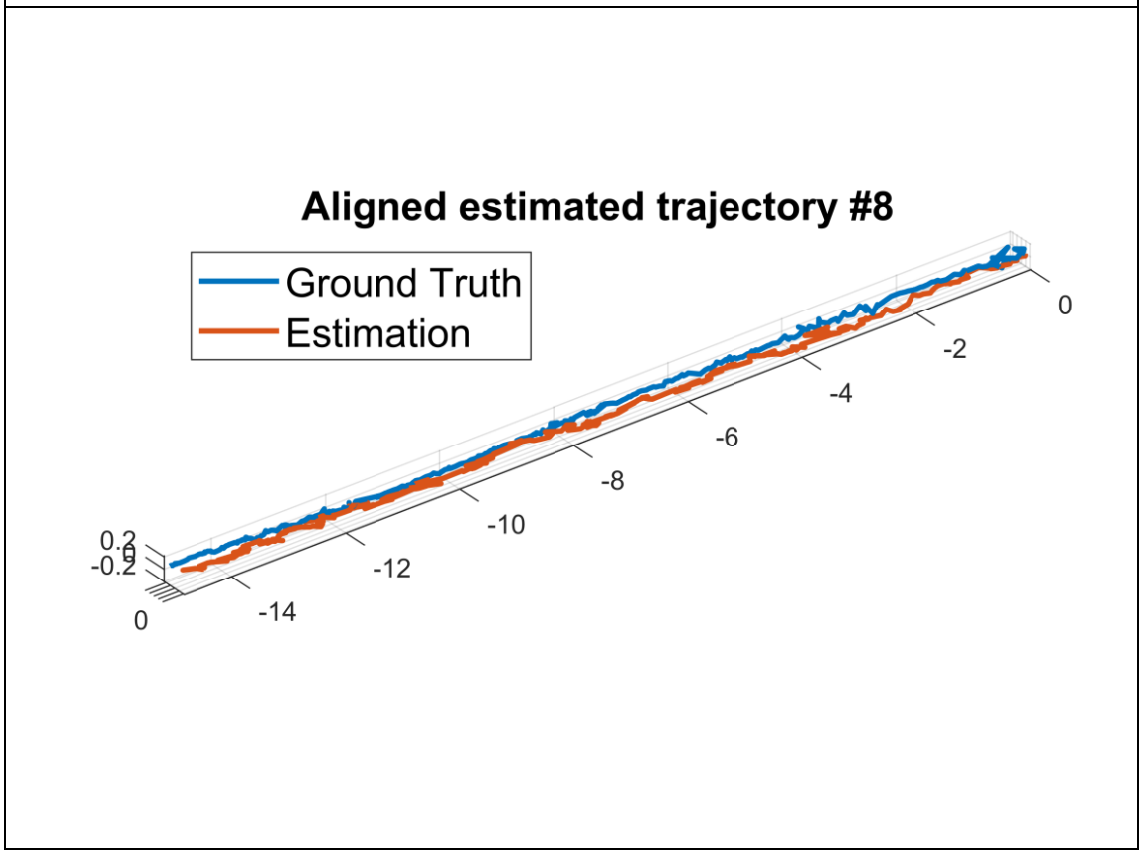
doi:10.6342/NTU202402036





doi:10.6342/NTU202402036





doi:10.6342/NTU202402036

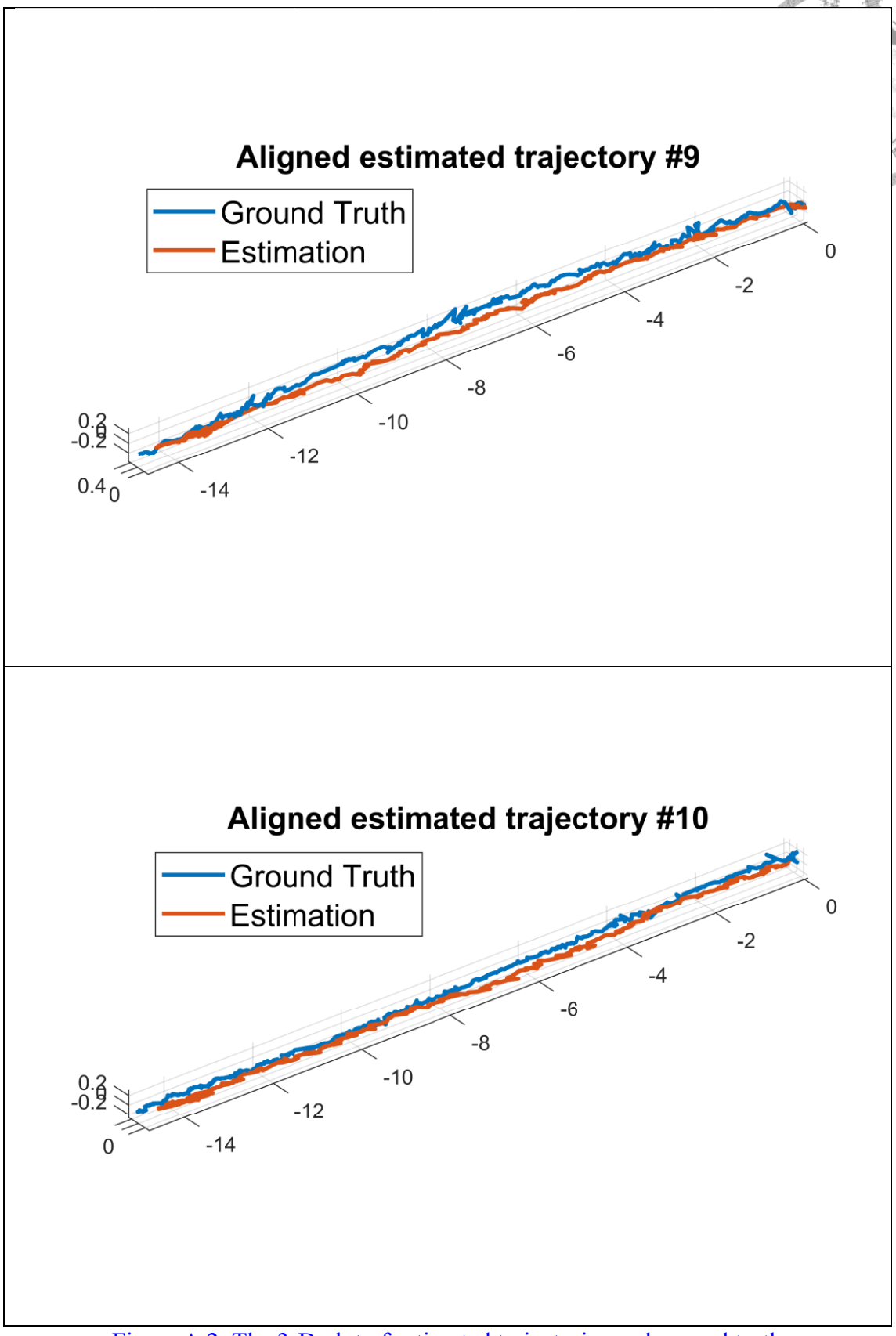


Figure A.2: The 3-D plot of estimated trajectories and ground truth

doi:10.6342/NTU202402036