Fourier-Mellin Transform: From Image Registration to Robot Vision

University of Chinese Academic of Sciences
ShanghaiTech University

Qingwen Xu

Jury:

Prof. Dr. Sören Schwertfeger, Supervisor
Prof. Dr. Laurent Kneip
(both ShanghaiTech University)
Prof. Dr. Jiamao Li
(University of Chinese Academy of Sciences)
Prof. Dr. Hong Lu
(Fudan University)
Prof. Dr. Huimin Lu
(National University of Defense Technology)
Prof. Dr. Xiaohua Tong
(Tongji University)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering Science

June 2021

Life is always tough, but wonderful.

Acknowledgements

Time flies, and turns to the sixth year of my PhD studies. In these six years, I have learned a lot and have grown a lot, which will become one of the most precious and most memorable journeys in my life. There are so many people to whom I would like to express my appreciation for helping me throughout my Ph.D. studies.

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Sören Schwertfeger. Prof. Sören Schwertfeger is not only my advisor, but also one of my friends. He gives me a lot of help on the research, answers my questions and helps me handle difficulties. He also has a big impact on my values. I learn a lot from him, for example, how to treat different opinions, cooperate with others and keep optimistic. I feel very lucky to be his first doctoral student, and complete this doctoral thesis under his supervision.

Next, I want to thank the members of my jury committee, Prof. Laurent Kneip, Prof. Hong Lu, Prof. Jiamao Li, Prof. Huimin Lu, and Prof. Xiaohua Tong for reading my thesis and providing constructive feedback. Their professional comments helped a lot to improve the quality and technical correctness of this thesis.

Furthermore, I would also like to thank Professor SörenSchwertfeger for giving me an opportunity to exchange abroad. During the visit to Jacobs University, I really appreciate Prof. Andreas Birk and Dr. Heiko Bülow's guidance. I am grateful for their supervision and enlightening advice in scientific research. Also, the topic of my thesis originates from this visit. In addition, I am also very grateful to my friends: Arturo Gomez Chavez, Christian A. Mueller and Tobias Doernbach, during this visit. They helped me to quickly adapt to the life in Bremen. Especially, Arturo also gave me a lot of help in my research.

My sincere thank then goes to Prof. Laurent Kneip. During his SLAM course, I learned basic knowledge of SLAM, which provides a strong basis for my further research. He

also gave me many suggestions on my research. In addition, I would also like to thank my laboratory friends. First of all, I would like to thank Yuning Jiang. Although our research directions are different, he often gave me new inspiration in scientific research, so that I have a new understanding of my research. Then, I would like to thank all the members in our group, especially Haofei Kuang, Xiaoling Long and Zhenpeng He. I could not conduct my experiments as soon as possible without their help. During the discussion with them, I often gain some new ideas. Moreover, I want to thank other colleagues at STAR center of ShanghaiTech University, especially Dandan Wang, He Wang, Xuyang Wu, Xuhui Feng, Jiahe Shi, and Zhihai Qu, for creating a relaxed atmosphere for the laboratory. Then, I would also like to thank my roommates in ShanghaiTech University: Xin Peng , Ke Xu and Xin Gao, for their kind care. Additionally, my research direction is similar to Peng Xin's, we thus often discuss research and help each other.

In addition, I would like to thank my other friends: my old roommates and my closest friend Zongyun Xie. They always support me when I am depressed. Also, my colleagues during my internship taught me a lot at work.

Last but not least, I would like to thank my parents for their constant encouragement and unwavering support throughout my past life. No words can adequately express my deepest gratitude to them. Finally, my special thanks go to my boyfriend, Naiyang Xu, for his encouragement, patience, and support during my Ph.D. studies. This thesis would not have been possible without him.

Shanghai, July 2021 Qingwen Xu

Abstract

With the development of visual odometry (VO) and related open-source algorithms, vision-based robot localization is becoming more and more popular. VO is applied in various scenarios, in which it may face challenges in certain environments, such as underwater turbidity, foggy weather with low visibility or feature-deprived settings. Traditional methods often fail here, due to the lack of clear textures. To overcome these challenges, we exploit the Fourier-Mellin Transform (FMT) to estimate the motion between images. FMT is a spectral method for image registration that is based on holistic descriptors and thus more robust than approaches using features or brightness consistency. But one of the drawbacks of FMT is that it requires all pixels in an image have the same distance to the imaging plane, i.e. single-depth images.

This thesis improves FMT in two major aspects and then applies it to visual odometry. One is to extract sub-images from original images, such that the sub-images meet the requirements of FMT. Another is to extend the application of FMT to multi-depth environments by rethinking its translation and zoom estimation. Concretely, the contributions of this work are summarized as follows:

- We propose omni-directional image matching based on FMT. For that, we first convert the omni-directional images from two consecutive frames to panorama images, from which we then extract sub-image sets. Afterwards, we apply FMT to calculate motion vectors from the corresponding sub-images of the two panorama images. Then we construct a motion flow field based on these motion vectors, i.e., omni-directional image matching. Additionally, this work utilizes these matched concordant points for omni-directional camera pose estimation. The experiments show the superior performance of our method compared to other feature-based approaches and optical flow by applying them to a pose estimation task.

- We propose a novel rotation estimation algorithm for omni-directional cameras based on the motion vectors calculated by FMT. Different from geometry methods like the five-point algorithm, this work models the rotation estimation of omni-directional cameras as sinusoidal fitting based on the properties of omni-directional cameras. The experiments show that the proposed method is more robust than the traditional geometry-based algorithms. The main reason for the robustness of the sinusoidal fitting approach owes to two points: one is that FMT provides accurate motion estimations on single-depth sub-images and another is that sinusoid fitting is very robust to outliers.
- We made a very interesting observation: There is a single peak in the phase shift diagram of FMT in single-depth scenarios. But in multi-depth environments, when the camera is translating, there are multiple high energy values lying in one line in the phase shift diagram of FMT. Based on this observation, we propose the extended Fourier-Mellin Transform (eFMT), that extends FMT to multi-depth scenarios. Specifically, eFMT finds the line with the maximum sum of energy, instead of a single peak, in the phase shift diagram. Since monocular VO algorithms like this one are up to an unknown scale factor, we need to re-scale between consecutive motion estimates. eFMT does this via pattern matching on the extracted energy vectors. Our experiments show that eFMT-VO is more robust than current popular visual odometry frameworks because eFMT maintains the superior robustness of FMT.

Keywords: Fourier-Mellin Transform, Visual Odometry, Pose Estimation, Omni-directional Vision, Sinusoidal Fitting

Contents

List of Figures

List of Tables

1 Introduction

With the rapid development of robotics technology, robots become more and more popular in all kinds of applications, such as industry 4.0 [54, 39], agriculture automation [4, 147] and smart home [135]. One of the important tasks for such mobile robots is navigation, i.e., how to move from pose A to pose B. For that, localization is the essential task, which tells where the robot is. These robots, sweeping robots [5], agricultural unmanned aerial vehicles [113, 114] and autonomous cars [89], need to first estimate their poses for path planning and navigation. For that, most robots utilize sensor fusion for localization. Moreover, different sensors are exploited based on the usage of robots. For instance, a global positioning system (GPS) and cameras are used for aerial vehicles; Inertial navigation system, Doppler velocity logs (DVLs) and digital compasses are usually utilized for underwater robots; Laser range finder (LRF), inertial measurement units (IMU), odometry and cameras are usually equipped on indoor robots.

These sensors have different advantages and disadvantages in the above applications. Specifically, the LRF and IMU with high precisions are usually very expensive; Differential GPS can achieve the precision of centimeter-level in open outdoor environments, but it cannot work well in indoor scenarios; Though the odometry calculates the robot pose easily, it will introduce accumulated error. In contrast, cameras are one of the most popular sensors owing to their low price, small size and low power consumption, which have been used in all kinds of robotic applications. Cameras were mostly used for monitoring in the very beginning. Later, they are used for robot localization with the development of vision-based localization. Additionally, omni-directional lenses and fisheye lenses are used for more robust pose estimation in several applications [9, 77].

In this thesis, we study how to use Fourier-Mellin Transform (FMT) for robot localization. Concretely, we estimate the robot poses from sequential images captured by

the cameras equipped on this robot, which is called Visual Odometry (VO). VO is an important tool to aid the localization of robots across various application domains including ground (e.g., [33, 57, 107, 128]), aerial (e.g., [11, 55, 145]), and underwater systems (e.g., [17, 35]). In this introduction, we will first recall the basic knowledge and related work for visual odometry in Sec. 1.1 and 1.2, respectively; Then we give a brief introduction of the FMT technology in Sec. 1.3 and discuss related work and applications in Sec. 1.4. FMT is a spectral method for image registration, which take the whole image into consideration instead of features or pixels, it is thus very robust and accurate; Afterwards, the challenges and difficulties of VO with FMT are summarized in Sec. 1.5; Finally, we introduce the structure and content of this thesis in Sec. 1.6.

1.1 Preliminary for Visual Odometry

In the past decades, there are many related works on visual odometry, which will be introduced in Sec. 1.2. In this section, we recall the basic knowledge for visual odometry based on the tutorial [119], including camera models, feature matching, motion estimation and back-end optimization.

1.1.1 Camera Models

In this section, we introduce three types of camera models that are used in this thesis, including pinhole model, catadioptric model and cylinder model. The pinhole model is usually used for normal perspective cameras, whereas catadioptric and cylinder models are applied on fisheye and omni-directional cameras.

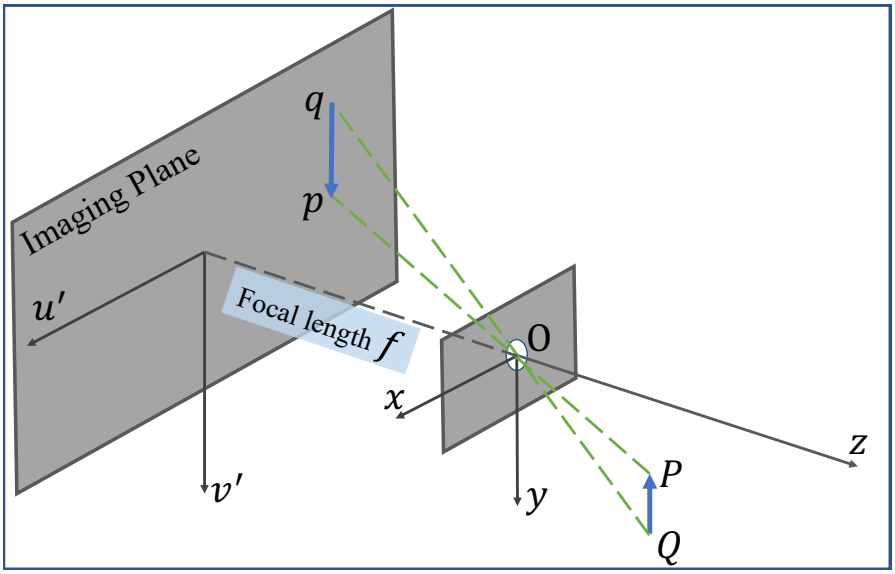


Figure 1.1: Pinhole camera model

Pinhole camera model: As shown in Fig. 1.1, the 3D ray PO through the pinhole O is projected on the 2D imaging plane at pixel p. Suppose the relationship between 3D point $P = [x, y, z]^\top$ and 2D pixel $p = [u, v]^\top$ in the camera's frame is described by

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \lambda \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{1.1}$$

where f_x and f_y are the focal lengths, and (c_x, c_y) is the image center. Note that there is only one focal length f in Fig. 1.1, whose metric is mm. It is different from the focal length f_x, f_y in Eq. (1.1), whose metric is pixel. These two focal lengths can be converted with a factor dp(mm/pixel). Additionally, the factor dp could be different in horizontal and vertical directions due to manufacturing. Thus, there are two focal lengths f_x, f_y in horizontal and vertical directions in Eq. (1.1).

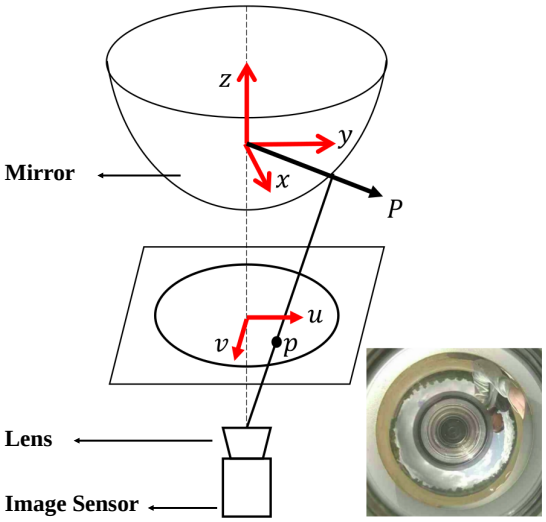


Figure 1.2: Omni-directional camera model according to [121]

Catadioptric camera model: The catadioptric camera model of [121] represents the relation between image pixels and camera rays as shown in Fig. 1.2. Two assumptions are made: (a) the center of the camera and the omni-directional lens are aligned and (b) the omni-directional lens/mirror rotates symmetrically. Suppose the coordinates of a pixel p in the omni image I_o are (u, v). When the origin is in the center of I_o, the camera ray that points from this pixel p in the direction of the corresponding real-world scene point is given by

$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \alpha u \\ \alpha v \\ f(u,v) \end{bmatrix} => \begin{bmatrix} u \\ v \\ f(u,v) \end{bmatrix} = \pi^{-1}(p) \,, \tag{1.2}$$

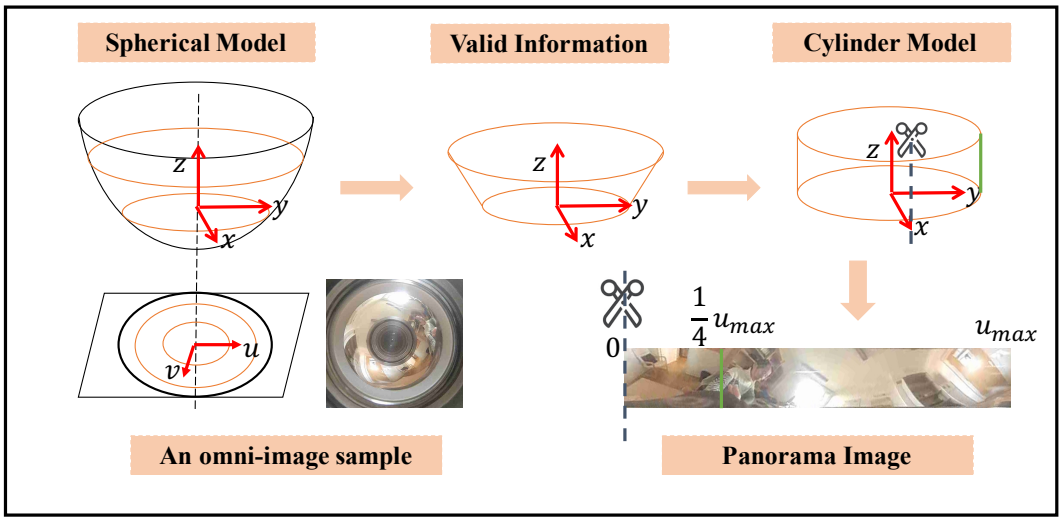


Figure 1.3: Cylindric camera model

where $\pi(\cdot)$ represents the projection function from P to p and $\pi^{-1}(\cdot)$ represents the inverse projection from p to P. Since the omni-directional lens/mirror is rotationally symmetric, $f(u,v)$ depends only on the distance $\rho = \sqrt{u^2+v^2}$ of a point to the image center. $f(\rho)$ is modeled as a high degree polynomial described by

$$f(u,v) = f(\rho) = \alpha_0 + \alpha_1\rho + \alpha_2\rho^2 + \alpha_3\rho^3 + ... \tag{1.3}$$

to represent different types of lenses. Finally, to account for possible errors in the hypothesis about the camera-lens alignment and the coordinates transformation of the omni-directional image I_o, an extra affine transformation is used as follows

$$\begin{bmatrix} u* \\ v* \end{bmatrix} = \begin{bmatrix} c & d \\ e & 1 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} x'_c \\ y'_c \end{bmatrix} \tag{1.4}$$

where (x'_c, y'_c) is the center of the omni-directional image and $(u*, v*)$ is the coordinate of p when the origin is in the upper left corner of I_o.

Cylinder camera model: As mentioned in [118], an image captured by a catadioptric omni-directional camera is usually unwrapped into a cylindric panorama image. Since the FMT requires undistorted sub-images, we also model the pose estimation problem based on the cylindric camera model in this work. Fig. 1.3 gives an intuitive description of the spherical to cylinder model transformation, which can be implemented by mapping and interpolation. The transformation between the cylinder and the panorama image can be described as

$$u = r \cdot \theta = r \cdot \arctan(\frac{y}{x}) \tag{1.5a}$$

$$v = \frac{H}{2} - z \tag{1.5b}$$

and

$$x = r \cdot \cos\theta = r \cdot \cos\frac{u}{r} \tag{1.6a}$$

$$y = r \cdot \sin\frac{u}{r} \tag{1.6b}$$

$$z = \frac{H}{2} - v \tag{1.6c}$$

When unwrapping omni-directional images to panorama images with Eq. (1.5), Eq. (1.6), we cannot make sure that each pixel is square, i.e. there may be a resolution inconsistency. In other words, the incident angle could be different with same pixels in $u-$ and $v-$axis. Thus we use a simple calibration method to find the ratio between angles per pixel in $u-$ and $v-$ direction. In detail, we can easily calculate the angles per pixel in $u-$axis based on the assumption that the pixels are distributed equally and the fact that the sum angles of the $u-$axis is $360°$. Then we calculate the vertical field of angles in contrast to square calibration patterns.

1.1.2 Feature Extraction and Matching

One very popular approach for VO is to use feature matching, but feature matching is not essential for visual odomtry, for example direct methods estimate camera motion directly base on brightness consistency, as will be introduced in Sec. 1.2. However, it is an essential step in feature-based visual odometry. This subsection gives a brief introduction to feature extraction and matching. Motion estimation based on brightness consistency will be discussed in the next paragraph.

Image features are usually classified into two types: corners and blobs. Common corners include Harris [38], Shi-Tomasi [130], SUSAN [132] and FAST [115], which are pixels in the image. In contrast, a blob is a local pattern in the image, whose intensity and texture are different from neighbors. Common blobs include SIFT [83, 84], SURF [15, 16], ORB [116], KAZE [7] and AKAZE [6]. Comparing these two kinds of features, the advantage of corners is the high precision of position, which corresponds to a single pixel, whereas the advantages of blobs are that they contain more information and have distinguished descriptors. Thus, corners can help find correspondences quickly when the motion is small while using blobs is more robust when the motion is big.

Feature matching should be conducted after feature extraction. When the difference between two images is small, we can use optical flow [64] to find the point x_2 of the second frame, which corresponds to the feature x_1 in the first frame. Optical flow is based on the assumption of brightness consistency, i.e., the pixel intensity is consistent during small intervals:

$$I(u,v,t) = I(u+\Delta u, v+\Delta v, t+\Delta t)\ . \tag{1.7}$$

However, the assumption does not always hold, especially when the motion is big. Thus, we usually utilize blobs to extract features when the motion is big and calculate a descriptor for each feature. The distance between two descriptors describes the similarity between two features. So a simple method for feature matching is using the brute force method based on the distance calculation. To decrease the computation, [94] proposed a method based on nearest neighbors to speed up matching.

1.1.3 Motion Estimation

Methods for motion estimation are divided into three types based on the dimensions of inputs, which are 2D-2D, 3D-2D and 3D-3D. The motion estimation methods for monocular VO usually include 2D-2D and 3D-2D whereas motion estimation based on 3D-3D is used for stereo VO or point cloud registration. Motion estimation based on 2D-2D correspondences is to estimate the transformation with corresponding pixels and that based on 3D-2D correspondences is the registration between 3D map points and 2D pixels. This subsection first introduces the 2D-2D motion estimation in feature-based and direct methods, and then gives a brief introduction on 3D-2D motion estimation.

Feature-based methods usually estimate pose transformation between two frames after feature matching by minimizing the reprojection error:

$${}_{k-1}^{k}T = \arg\min_{T} \sum_{i} \|u_i' - u_i\|_{\Sigma}^{2}\ , \tag{1.8}$$

where u_i and u_i' are the projection of 3D point P_i on the $k-1$ and k images, respectively. To solve Eq. (1.8), we first find the essential matrix based on geometry and then decompose rotation and translation from the essential matrix to construct ${}_{k-1}^{k}T$. Common methods for finding essential matrix include five-point [97] and eight-point [61] algorithms. Additionally, RANSAC is utilized to speed up the computation and increase robustness when there are many pairs of candidates.

Direct methods estimate transformation by minimizing photometric error instead of

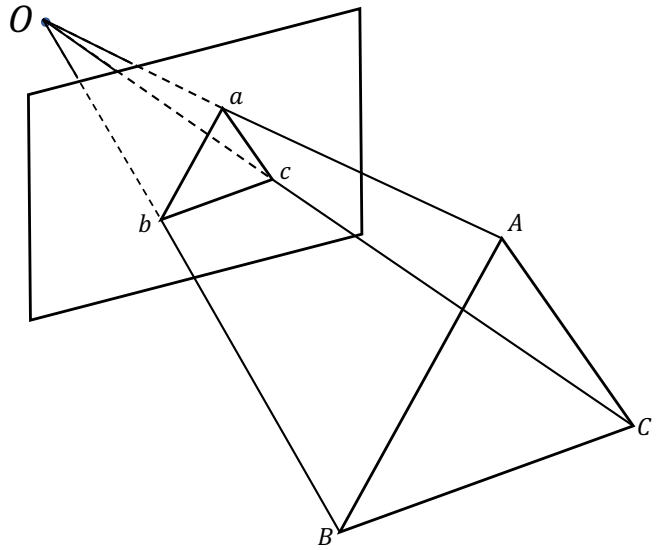


Figure 1.4: Example of a P3P problem [51]

feature detection:

$$ {}^{k}_{k-1}T = \arg\min_{T} \sum_{i} \| {}^{k}I(u'_i) - {}^{k-1}I(u_i) \|^2_{\sigma} \, , \tag{1.9} $$

where $I(\cdot)$ is the pixel intensity. Since direct methods do not extract features, we can not get corresponding u'_i and u_i directly. For that, we build the relationship function of u'_i and u_i based on the camera model and the probabilistic depth model. The function is

$$ u'_i = \pi(T \cdot \pi^{-1}(u_i) \cdot d) \, , $$

where $\pi(\cdot)$ is the projection from 3D to 2D provided by camera models, d is the pixel depth estimated by the probabilistic depth model.

If the 3D map is maintained when estimating camera poses, we can use the matched 3D map points and 2D pixels to estimate the transformation between the current camera pose and a reference frame. The transform can be estimated by minimizing reprojection error:

$$ {}^{k}_{k-1}T = \arg\min_{T} \sum_{i} \| u'_i - \pi(P_i) \|^2_{\Sigma} \, . \tag{1.10} $$

At least three pairs are needed for solving Eq. (1.10). Common methods include PnP [45], EPnP [78] and P3P[73]. Here, we take P3P as an example to explain how to estimate transformation from 3D-2D correspondences [51]. As shown in Fig. 1.4,

based on the law of cosines, we have

$$\begin{aligned} OA^2 + OB^2 - 2OA \cdot OB \cdot \cos < a, b >= AB^2 \ , \\ OB^2 + OC^2 - 2OB \cdot OC \cdot \cos < b, c >= BC^2 \ , \\ OA^2 + OC^2 - 2OA \cdot OC \cdot \cos < a, c >= AC^2 \ . \end{aligned} \tag{1.11}$$

Let $x = \frac{OA}{OC}$, $y = \frac{OB}{OC}$, $g = \frac{AB^2}{OC^2}$, then we can use $m \cdot g$ and $n \cdot g$ to denote $\frac{BC^2}{OC^2}$ and $\frac{AC^2}{OC^2}$, respectively. Thus, Eq. 1.11 becomes

$$\begin{aligned} x^2 + y^2 - 2xy \cos < a, b > -g = 0 \ , \\ y^2 + 1^2 - 2y \cos < b, c > -mg = 0 \ , \\ x^2 + 1^2 - 2x \cos < a, c > -ng = 0 \ , \end{aligned} \tag{1.12}$$

where only x and y are unknown parameters. These equations can be solved by the Wu-Ritt's zero decomposition method. Details are provided in [51].

1.1.4 Back-end Optimization

The first step of back-end optimization is loop detection. Back-end optimization is meaningful if and only if there is a loop. The core of loop detection is similarity calculation, which is similar to feature matching. In practice, we usually use bag of words [131] to implement loop detection to improve efficiency and robustness. There are two kinds of back-end optimization: bundle adjustment and pose optimization. The former optimizes camera poses and landmarks positions together whereas the latter only optimizes camera poses. Their objective function is

$$\arg\min_{^iC} \sum_i \sum_j \| ^iC - {}^i_jT \, {}^jC \|^2 \tag{1.13}$$

and

$$\arg\min_{P_i, {}^kC} \sum_i \| ^ku_i - \pi(P_i, {}^kC) \|^2 \ , \tag{1.14}$$

respectively. ^kC is the camera pose corresponding to the k^{th} frame. Open-source optimization toolboxes, such as g2o [56] and ceres [3], can be used for optimizing the objective function.

1.2 Related Work for Visual Odometry

This section recalls related work for VO. [137] gives a detailed tutorial on all kinds of camera models. In this section, we only introduce related work on omni-directional camera models since the pinhole model has already been well-known and well-studied.

Then we take feature-based and direct methods as representatives to recall related work for visual odometry. Finally, this section recalls the extension of VO for pinhole cameras to VO for omni-directional cameras.

1.2.1 Omni-directional Camera Model and Calibration

[69] summarizes that there are three types of omni-directional cameras, i.e., dioptric, catadioptric and polydioptric cameras, as shown in Fig. 1.5. The dioptric camera is also called the fisheye camera, whose field of view (FoV) can be up to $180°$. The structure of catadioptric cameras usually includes one common pinhole camera and a mirror. The shape of the mirror could be parabolic, hyperbolic or elliptical. The rays are reflected by the mirror and then reached the imaging plane via the optical center. The common horizontal FoV of a catadioptric camera is $360°$ and its vertical view could also achieve $100°$, which depends on the camera design. A polydioptric camera can be considered as a multi-camera system with several narrow view cameras. Both horizontal and vertical FoVs of a polydioptic camera are $360°$ because multiple cameras can cover all the environment. Additionally, several $360°$ omni-directional cameras are composed of two fisheye cameras, such as Ricoh Theta V and Insta360 One X.

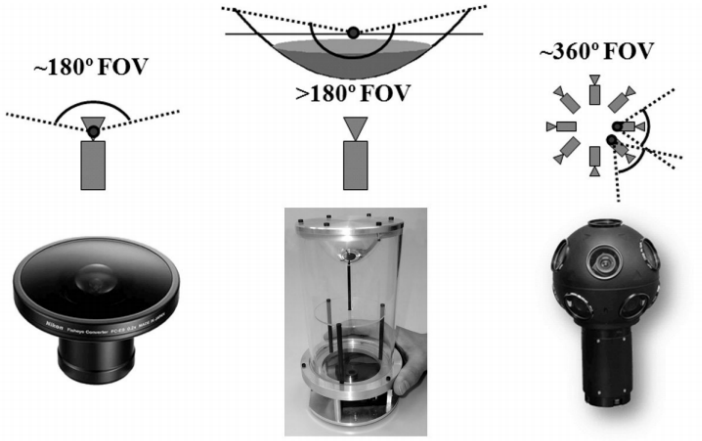


Figure 1.5: Left: Dioptric camera (e.g. fisheye); Middle: catadioptric camera; Right: an example polydioptric camera produced by Immersive Media.

[40] points out that the traditional pinhole camera model cannot describe these cameras with too large FoV. Then researchers propose specific models for omni-directional cameras. One type is the unifying model for catadioptric cameras, which is first proposed by [151] and then this model is improved by [13] with radial distortion. [151] discussed how to apply this model for fisheye cameras and showed that the calibration accuracy is not high enough for fisheye cameras. The main reason is that catadiop-

tric cameras can be described by accurate parametric models while fisheye cameras cannot. [91] extended the polynomial model to realize the calibration of fisheye cameras. Another type is the omni-directional camera model based on Taylor polynomials proposed in [121], which can be used for both catadioptric and fisheye cameras. For these two types of omni-directional camera calibration, there are different methods and toolboxes. For example, in addition to the calibration proposed in [52, 13], [41] exploited the geometry constraints based on the single projective center to improve the calibration accuracy for the unifying model. Mei et al. released a complete toolbox by increasing parameters to compensate for the error between real-world and theoretical cases [90]. For the second type, Scaramuzza et al. also released their corresponding MATLAB toolbox in [122], and then Urban et al. improved its robustness and accuracy by replacing the residual function and jointly optimizing all parameters [143].

All the above methods assume that the camera is central, i.e., all the rays refracted by the mirror interact at the same point. However, some omni-directional cameras are non-central due to the influence of manufacturing. For that, [136, 109] built the mapping between 2D pixels and 3D points for omni-directional cameras. The mapping can be calculated by the calibration grids with known geometry, which is useful for both central and non-central cameras.

1.2.2 Monocular VO

VO is usually mentioned with structure from motion (SFM) and visual simultaneous localization and mapping (VSLAM). They are considered as the same tasks, i.e., using the image sequences to estimate camera poses and reconstruct environments. VO usually focuses on the accuracy and computation efficiency of the pose estimation, whose inputs are ordered. SFM solves a more general problem, i.e., pose estimation and environment reconstruction no matter whether the input images are ordered or disordered. Also, offline optimization is usually used to increase the accuracy of reconstruction. VSLAM can be considered as the fusion of VO and SFM, since it needs to real-time estimate the pose and reconstruct 3D environments simultaneously, whereas its inputs are ordered. Since the focus of this thesis is VO, we mainly introduce the related work of VO in this section. Considering there is a big overlap between VO and VSLAM, some work about VSLAM will also be included.

The VO methods are divided into three types according to how they describe the environment: feature-based, appearance-based and hybrid methods [119]. Feature-based VO usually needs to extract distinct and repetitive regions from the image and build corresponding descriptors. Appearance-based methods do not need to extract features. They rely on the whole or partial images. Hybrid methods consider both

pixel consistency and features for pose estimation. There are many feature-based VO methods, such as [99, 120, 86, 12, 133]. [99] first implemented a large-scale and real-time monocular VO. It uses the five-point algorithm to estimate relative pose and the RANSAC [46] technology to sample from many correspondences, which speeds up the computation efficiency. Afterwards, many methods exploit the five-point algorithm for pose estimation, such as [100, 37]. Overall, the five-point algorithm is used for unconstrained 6-DoF motion in 3D space. However, the sensors' motion can be constrained because they are mounted on a mobile platform whose motion is in 2D, such as autonomous vehicles and sweeping robots. Based on these constraints, [120, 106] proposed to use fewer points for pose estimation. There are also several appearance-based VO approaches, such as [53, 92, 24]. Both [53] and [24] used FMT to calculate motion vectors between images. The former applies it for pose estimation and the latter implements underwater image mosaicking based on the motion vectors. [123] used the hybrid method for pose estimation, i.e., utilizing local appearance correlation for 2D rotation and estimating motion via ground features.

After the transformation is estimated between two frames, VO calculates the camera pose in the reference frame via chain rule. However, the accumulated error will get large after the camera moves for a long time. For that, some researches optimize the camera poses in a time interval via bundle adjustment, such that the accumulated error gets smaller [49, 138, 74]. In recent years, almost all VO/VSLAM methods include loop detection and pose optimization.

From the aspects of relative pose calculation, popular VO/VSLAM frameworks are classified into filtering-based, keyframe-based and direct methods. Davision et al. proposed the first monocular VSLAM: MonoSLAM [37]. The system uses the extracted features as landmarks and updates the pose with extended Kalman filter (EKF). Since EKF only updates the current pose without optimizing the previous status, it only needs few computation resources, but brings accumulated error. In contrast, keyframe-based methods [71, 79, 95] usually exploit bundle adjustment to provide accurate pose estimation in the long-term and large-scale environments. PTAM [71] first applied optimization in the back-end of VO and proposed the SLAM framework with two threads: localization and mapping. Afterwards, ORB-SLAM [95] improved feature extraction and applied loop detection based on PTAM, to increase the robustness and localization accuracy. Also, RDSLAM [139] improves the RANSAC technology of PTAM to implement robust localization in dynamic scenarios. VO based on the direct method estimates camera poses based on brightness consistency instead of feature extraction, it is thus more robust than feature-based methods in the feature-deprived environment. The first VSLAM based on the direct method is DTAM [96], which reconstructs a dense map. However, it needs GPU to meet the requirements of

large computation. Recent LSD-SLAM [44] only reconstructed semi-dense maps. The depth of each pixel can be calculated independently to achieve a high computation efficiency in LSD-SLAM. Moreover, the research group of LSD-SLAM also proposed a sparse VO based on direct method, i.e. DSO [43], which is more robust due to photometric calibration. Furthermore, both LSD-SLAM and DSO can run on the CPU since their computation resources is little. Another kind of VO is called the semi-direct method. One representative of semi-direct methods is SVO [48]. It uses the direct method in image registration but maintains the reprojection error minimization for pose estimation and bundle adjustment.

1.2.3 VO for Omni-directional Cameras

From the viewpoint of VO, omni-directional cameras have the advantage that they have a large FoV [88, 28, 50, 36, 81, 146]. Visual cues from panorama images help the robot to achieve homing in [9], which uses omni-directional cameras' advantage of the large FoV. [80] proposed local bundle adjustment for omni-directional cameras to recover camera trajectory and environmental map. Also, several meaningful applications with panorama cameras are mentioned in [18]. In this section, we mainly recall related work about VO for omni-directional cameras.

Similar to VO for pinhole cameras, VO for omni-directional cameras can be divided into pixel-based, feature-based and appearance-based methods. Pixel-based VO methods are based on brightness consistency. For example, [36] estimated the 2D motion for a planetary rover based on the optical flow calculated from the images captured by the catadioptric camera mounted on the rover. Popular SVO2 [47] also provided a omni-directional version. Most VO approaches for omni-directional cameras rely on features [112, 140, 123, 117]. In [77], panorama images are exploited to implement a bearings-only SLAM system, which can provide rich feature points. [140] compared FAST and SIFT, which showed that SIFT performs better in landmark detection. Also, long-term robust localization for omni-directional cameras is implemented with epipolar geometry and 3D map information, without bundle adjustment in [140]. Moreover, omni-directional and pinhole cameras are compared in [112], showing that omni-directional cameras can provide higher localization accuracy. Additionally, there are some specific researches, which are designed for feature matching for omni-directional cameras [59, 10]. Furthermore, some VO methods for omni-directional cameras are based on appearance. For instance, [105] evaluated different holistic descriptors for the localization of omni-directional cameras; The rotation estimation in [123] is based on local appearance.

Omni-directional cameras become more and more popular with the improvement

of manufacturing in recent years. Most popular VO/VSLAM frameworks support for omni-directional cameras. For example, ORB-SLAM provides fisheye model in its third version [27]. But it cannot be used for fisheye cameras with too large FoV because it still uses a parametric model. MultiCol SLAM [144] is also a variant of ORB-SLAM, which uses the omni-directional camera model in [122], such that it can support all kinds of omni-directional cameras. SVO2 [47] also uses the model in [122] and can be used for all kinds of omni-directional cameras. Additionally, the research group of DSO extended DSO for omni-directional cameras in [88] with the camera model provided by [52]. Additionally, some methods for omni-directional vision do not depend on a single omni-directional camera, but on a omni-directional vision system that is composed of multiple cameras. For example, MultiCol SLAM [144] as mentioned above is also designed for multi-camera systems. [129] proposed a robust VO system for an omni-directional visual system with four fisheye cameras. Most of the above algorithms implement VO for omni-directional cameras by replacing pinhole camera models with omni-directional cameras.

1.3 Preliminary for FMT

This section recaps the main idea of classic FMT [110]. Given two image signals ${}^1I, {}^2I$, the relationship between them is

$$\begin{aligned} {}^2I(x,y) = {}^1I(&zx\cos\theta_0 + zy\sin\theta_0 - x_0, \\ &-zx\sin\theta_0 + zy\cos\theta_0 - y_0) \end{aligned} \tag{1.15}$$

where z and θ_0 are constant and represent the zoom and rotation, respectively, and (x_0, y_0) is the translation between ${}^1I$ and ${}^2I$. The motion parameters (z, θ, x_0, y_0) can be estimated by FMT via the following steps:

- Fourier transform on the image signals from both sides of Eq. (1.15):

$$\begin{aligned} {}^2\mathcal{F}(\xi,\eta) = \quad & e^{-j2\pi(\xi x_0+\eta y_0)}z^{-2} \\ & {}^1\mathcal{F}(z^{-1}\xi\cos\theta_0 + z^{-1}\eta\sin\theta_0, \\ & -z^{-1}\xi\sin\theta_0 + z^{-1}\eta\cos\theta_0) \end{aligned} \tag{1.16}$$

- Convert the magnitude $\mathcal{M}$ of Eq. (1.16) in polar coordinates, ignoring the coefficients:

$${}^2\mathcal{M}(\rho,\theta) = {}^1\mathcal{M}(z^{-1}\rho, \theta-\theta_0)\ . \tag{1.17}$$

- Take the logarithm of ρ of Eq. (1.17):

$$^2\mathcal{M}(\xi,\theta) = {}^1\mathcal{M}(\xi - d, \theta - \theta_0) \ , \tag{1.18}$$

  where $\xi = \log\rho, d = \log z$.

- Obtain z and θ_0 from Eq. (1.18) based on the shift property of the Fourier Transform. Re-rotate and re-zoom ^2I to $^2I'$ so that

$$^2I'(x,y) = {}^1I(x - x_0, y - y_0) \ . \tag{1.19}$$

  Accordingly,

$$^2\mathcal{F}'(\xi,\eta) = e^{-j2\pi(\xi x_0 + \eta y_0)}{}^1\mathcal{F}(\xi,\eta) \ . \tag{1.20}$$

Thus, all the motion parameters (z, θ_0, x_0, y_0) can be calculated by conducting phase correlation on Eq. (1.18) and Eq. (1.19). Taking Eq. (1.19) as an example, we first calculate the cross-power spectrum by

$$Q = \frac{{}^1\mathcal{F}(\xi,\eta) \circ {}^2\mathcal{F}'^*(\xi,\eta)}{|{}^1\mathcal{F}(\xi,\eta) \circ {}^2\mathcal{F}'^*(\xi,\eta)|} \ , \tag{1.21}$$

where $\circ$ is the element-wise product and * represents the complex conjugate. By applying the inverse Fourier transform, we can obtain the normalized cross-correlation

$$q = \mathcal{F}^{-1}\{Q\} \ , \tag{1.22}$$

which is also called ***phase shift diagram (PSD)*** in this thesis. Then the translation (x_0, y_0) corresponds to the location of the highest peak in q:

$$(x_0, y_0) = \arg\max_{(x,y)}\{q\} \ . \tag{1.23}$$

In the implementation, the PSD is discretized into a grid of cells. Note that there exist partial non-corresponding regions between two frames due to the motion. Instead of contributing to the highest peak, these regions generate noise in the PSD. Since the energy of this noise is distributed over the PSD, it will not influence the detection and position of the highest peak when the overlap between the frames is big enough.

Classical FMT describes the transformation between two images, which corresponds to the 4-DoF motion of the camera, including 3-DoF translation (zoom is caused by the translation perpendicular to the imaging plane) and yaw (assume $z-$axis is perpendicular to the imaging plane). This indicates that FMT only works when the camera does not roll or pitch. Moreover, it is also limited to single-depth environments

because it assumes zoom z and translation (x_0, y_0) as consistent and unique, which does not hold in multi-depth environments.

1.4 Related Work for FMT

Spectral methods for image registration take whole images as input, which belongs to the appearance-based methods introduced in Sec. 1.2.2. More precisely, spectral methods do not only operate in local areas, but - as the name suggests - they take the full spectrum of the frequency domain into account, i.e., a continuous scale from small local patterns up to large distinctive structures covering the whole image in the frequency domain.

A core method for spectral registration is the Fourier Transform, or more precisely the Fast Fourier Transform (FFT) algorithm [34], which has been used since the early days of Computer Vision to determine the translation between images [8]. Using in addition the related Mellin transform, rotation and scale can also be determined [32, 111]. While there has been continuous research on using and improving image registration with FMT, it has recently received an increased amount of attention - see, e.g., the detailed survey of [142]. The main reason for this is that FMT is fast and very robust. As it takes the full range of image content from small local patches up to large structures into account, it is working quite well under challenging conditions, i.e., feature-deprived environments, under poor visibility conditions, with dynamics in the scene, etc. [142]. Additionally, FMT was shown to be more accurate and faster than SIFT in certain environments in [20]. The visual odometry based on FMT performs more accurately and robustly than that based on different features, like ORB and AKAZE, especially in feature-deprived environments [148].

Due to its robustness and high accuracy, FMT has been successfully applied in multiple applications, such as image registration [1, 58, 108], fingerprint image hashing [2], visual homing [29], point cloud registration [23], 3D modeling [26], remote sensing [102, 150], and localization and mapping [70, 31, 126]. However, it requires that the captured device doesn't roll or pitch and that the environment is planar and parallel to the imaging plane. There are already several efforts on solving the first restriction. For instance, Lucchese calculated the affine transform via optimization based on the affine FMT analysis [85]. In [22], the oversampling technology and Dirichlet-based phase filter were used to make FMT robust to some image skew. Moreover, the sub-image extraction strategy [63, 75, 104, 148] is popular in addressing the 3D motion problem. However, there are few researches on the second problem, which limits FMT's applications in VO.

From the viewpoint of VO, FMT has the significant disadvantage that it is a 2.5D method

in the sense that it can only estimate 4-DoF, i.e., the 3-DoF of rigid motion in 2D plus 1-DoF scale. This can nevertheless be useful, e.g., for Unmanned Aerial Vehicles (UAV) or Unmanned Underwater Vehicles (UUV) with a down-looking camera or an imaging sonar under a flat world assumption [68, 67, 108, 66, 19, 24, 21]. [63] demonstrated that FMT can also be used to compute an optical flow by registering image patches, but they do this without recovering 3D poses. Furthermore, they used FMT as a dense method, i.e., as a substitute for standard simple correlation techniques in optical flow estimations. Due to the exhaustive nature of this approach, it is computationally very expensive [63]. [93] used this principle to implement a 2D visual compass. Using FMT for the registration of few sub-images, it can also be extended to estimate camera tilt including pitch and roll by combining the four-point-algorithm and a consistency criterion [104]. Nevertheless, it is not a full estimation of the 6-DoF rigid motion parameters in 3D.

A full 3D extension of FMT has been introduced by [23] in form of Fourier-Mellin-SOFT (FMS). FMS is a full 3D extension as it takes two 3D scans as input and it computes the 7-DoF transformation between the scans, i.e., the 6-DoF rigid motion transformation plus 1-DoF scale. The 3D scans can be point-clouds, e.g., from laser-range-finders (LRF) or from stereo vision, or volume data, e.g., from magnetic resonance tomography (MRT) or from computer tomography (CT). The handling of 3D rotations - which is the most challenging part - is tackled by a SO(3)-Fourier-Transform (SOFT) based on spherical harmonics. Note that the estimation of scale by FMS can be of interest when for example registering metric data with scale-free data, e.g., LRF scans with structure from motion (SfM) data, or when registering different object instances, e.g., MRT or CT data of bone structures from different patients with a template. But even when dealing with a fixed scale, i.e., when using FMS as a rigid 6-DoF registration method, it is very competitive due to its very high robustness and its fast computation speed [23]. Nevertheless, from the viewpoint of the work presented here, FMS has the disadvantage that it requires full 3D scans as input.

1.5 Research Challenges

With the development of VO and VSLAM, the visual localization technology attracts more attention, especially applying it for robot localization in different scenarios, such as indoor and outdoor localization for UAV, localization and navigation of service robots. However, these algorithms meet some challenges because there is a big gap between practical scenarios and public datasets. The main research challenges are summarized as follows:

1) The large FoV of omni-directional cameras causes low image resolution, which

decreases pixel precision. On the other hand, the low-cost omni-directional cameras are difficult to calibrate due to the error in manufacturing, such that accurate localization is difficult to achieve.

2) Current feature-based methods cannot conduct feature matching properly in some challenging scenarios, such as motion blur, turbid underwater and feature-deprived scenarios. Though direct methods can perform better than feature-based methods in feature-deprived cases, they cannot work well when there are few textures in the environments. Also, direct methods require accurate camera calibration, which does not support well for low-cost cameras.

3) Though FMT can achieve good results in challenging scenarios, it can only estimate 4-DoF camera motion, which largely limits its applications in VO. Also, FMT requires that single-depth environments, i.e., the depth of each pixel should be the same.

In the following, the thesis proposes some solutions to handle these challenges.

1.6 Outlines and Contributions of the Thesis

The thesis mainly focuses on extending FMT's application for VO considering FMT's robustness and limitation. The thesis has five chapters in total. The first chapter introduces basic knowledge and related work. The following chapters and contributions are structured as:

Chapter 2 exploits FMT for feature matching between omni-directional images. To meet the requirements of FMT, we propose recursive sub-image strategy to extract sub-images from omni-directional images. Then we use FMT to calculate motions between two frames to derive concordant points. To evaluate the performance of feature matching, we implement a simple VO based on the omni-directional camera models and the classic five-point algorithm. The performance of VO based on FMT, features and optical flow are compared on different datasets. The comparison and ablation study show that the recursive sub-image strategy and FMT give more robust matching results, especially in challenging scenarios. The majority of the content in Chapter 2 is based on the following papers:

- Q. Xu, A.G. Chavez, H. Bülow, A. Birk and S. Schwertfeger. Improved Fourier Mellin Invariant for Robust Rotation Estimation with Omni-Cameras[C]. In 2019 IEEE International Conference on Image Processing (ICIP) Taiwan, China, 320-324. September, 2019.

- Q. Xu, H. Bülow, A. Birk, and S.Schwertfeger. 3D Visual Odometry based on 2.5D Spectral Registration of Omnidirectional 2D Images[J]. Robotics and Autonomous Systems. (Under Review)

Chapter 3 proposes a novel method for rotation estimation of omni-directional cameras. We first present how to model the rotation estimation for omni-directional images to a sinusoidal fitting problem based on the property of omni-directional cameras, instead of simply replacing the camera models. However, we use the cylinder model during the modeling, which lacks consideration on multiple depths. Thus, we only evaluate rotation estimation in the experiments. Additionally, we analyze the influence of translation terms on pose estimation, which provides a potential extension in our future work. The majority of the content in Chapter 3 is based on the following papers:

- H. Kuang, Q. Xu, X. Long and S. Schwertfeger. Pose Estimation for Omni-directional Cameras using Sinusoid Fitting[C]. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 900-906. November, 2019.
- Q. Xu, X. Long, H. Kuang, and S. Schwertfeger, Rotation Estimation for Omni-directional Cameras using Sinusoid Fitting[J]. Journal of Intelligent & Robotic Systems. (Minor Revision)

Chapter 4 proposes the extended FMT (eFMT) to relax one of its limitations, the equidistant restriction. For that, we extend the zoom and translation estimation by revisiting the phase correlation in FMT. eFMT is based on the observation that there is a single peak on the phase shift diagram of FMT when the scenario is single depth and there are multiple high energy values when the scenario is multi-depth. During the extension, eFMT keeps the robustness of FMT with respect to feature-deprived scenarios. Also, we implement an eFMT-based visual odometry framework and test in different scenarios to compare the robustness and accuracy of eFMT with the state-of-the-art VO algorithms. The majority of the content in Chapter 4 is based on the following paper:

- Q. Xu, H. Kuang, L. Kneip and S. Schwertfeger. Rethinking the Fourier-Mellin Transform: Multiple Depths in the Camera's View. Remote Sensing 13, no. 5 (2021): 1000.

Chapter 5 concludes the main contributions of the thesis and proposes future directions.

2 Feature Matching for Omni-directional Cameras based on FMT

This chapter proposes using FMT to register omni-directional images. Different from traditional feature matching, we first divide omni-directional images into a few sub-images and then use FMT to calculate relative motion between each sub-image pair. To evaluate this matching, we perform the five-point algorithm on these motion pairs to estimate the relative pose between two frames. As the experiments later on show, through the use of a few sub-images and the option to recursively sub-divide them, robust results can be achieved. Furthermore, it is shown that the FMT based 3D motion estimation has an interesting advantage over state-of-the-art methods, as it tends to be more robust in poor visibility conditions, i.e., under motion blur, in feature-deprived environments, under the presence of fog and smoke, etc.

A recursive sub-division strategy for the sub-images is also introduced in this chapter (Sec. 2.3). This strategy uses the Signal-to-Noise-Ratio (SNR) of the PSD in the FMT registrations. The SNR indicates registration success or not. Multiple peaks in the phase shift diagram can indicate multiple apparent motions in a sub-image pair, which suggests that a further sub-division of this pair of sub-images should be done. This recursive sub-division strategy is an important aspect of our sparse approach. It leads to very competitive computation times, in contrast to the exhaustive use of FMT in for example [63], while providing a robust solution. This recursive sub-division strategy is hence of interest for the efficient use of spectral registration on sub-images in general. Possible additional use-cases include for example 2.5D mosaicking of non-flat terrain or structure from motion under non-standard projective functions, e.g., with imaging sonars.

The main contributions of this chapter are summarized as follows:

- We extend the FMT method to estimate motion between omni-directional images;
- We propose a motion model based on sub-image patches to compensate for the omni-directional images' non-linear distortions;
- Ablation studies regarding robustness against motion blur, noise and computation time are provided;
- This work also provides baseline comparisons against commonly used registration feature-based methods.

2.1 Overview of 3D Spectral Visual Odometry

A brief outline of our approach is shown in Fig. 2.1. First, the omni-directional images are transformed to panoramic images. Then, they are divided into a few pairs of co-located sub-images, i.e., each pair of image patches is taken from the same window of two consecutive video frames. Afterwards, the 2.5D transformations between the sub-image pairs are calculated using FMT, which then form a sparse motion flow field of concordant points. Then, we use the calibrated omni-directional camera model to recover the normalized camera rays of the concordant points in the omni-directional images. Finally, the five-point algorithm is used to estimate the relative pose between the two frames.

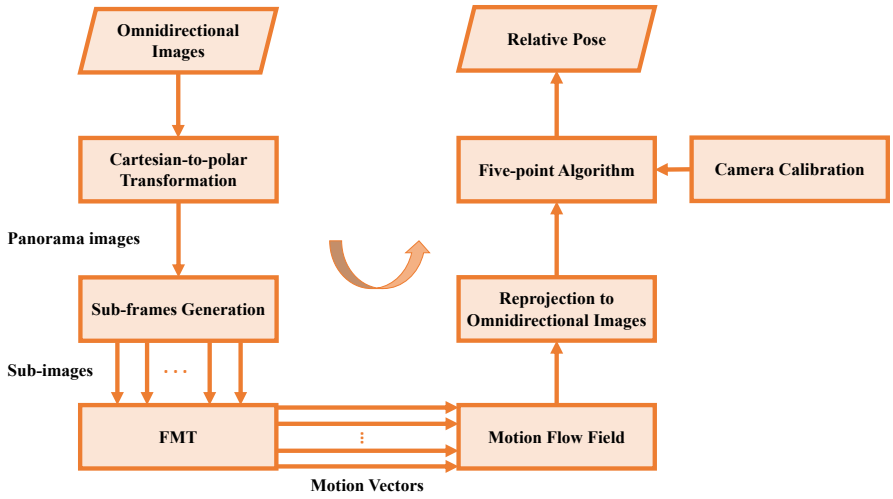


Figure 2.1: Overview of the processing pipeline of the proposed approach

We use the term *concordant* points here to stress the differences to appearance-based correspondences of visual features, i.e.:

- The two concordant points ${}^1p$ and ${}^2p$ represent two complete sub-images ${}^1a_i$ and

${}^2a_i$ that are taken from the same window w_i, i.e., the same set of coordinates, in two subsequent images of a video stream.

- The relative location of the concordant points ${}^1p$ and ${}^2p$ is not determined by their location with respect to the sub-images ${}^1a_i$ and ${}^2a_i$, but as a result of the 2.5D FMT registration of ${}^1a_i$ and ${}^2a_i$ and

- the subsequent projection of the apparent motion using the camera model.

The Pseudo-Code of 3D Spectral VO is shown in Alg. 1. The different steps therein are described in the following sections in more detail. Since the transformation between omni-directional images and panoramic images have already been introduced in Sec. 1.1.1, which will not be described again here.

Algorithm 1 3D Spectral VO with Omni-directional Images

1: **Input:** Omni-directional images ${}^1I_o$, ${}^2I_o$;
Noise filter thresholds th_{pr}, th_{pnr};
Sub-frame size threshold th_e
2: Obtain panorama images ${}^1I_p$, ${}^2I_p$ of size $W \times H$
by cartesian-to-polar transformation
3: Extract sub-image set ${}^1\mathbb{A}$, ${}^2\mathbb{A}$ from ${}^1I_p$ and ${}^2I_p$
4: **for all** sub-images ${}^1a_i \in {}^1\mathbb{A}$, ${}^2a_i \in {}^2\mathbb{A}$ **do**
5: Compute relative apparent motion
m_i = FMT(${}^1a_i$, ${}^2a_i$, th_{pr}, th_{pnr})= $[s\ \theta\ t_x\ t_y]^\top$
6: **if** $PR(m_i) < th_{PR}$ and $PNR(m_i) < th_{PNR}$ **then**
7: Select point ${}^1p_{a_i} = (c_x + \delta, c_y + \delta)$, where $\delta > 0$
8: Find concordant point $F_i = ({}^1p_{a_i}, {}^2p_{a_i})$ (Eq. (2.1))
9: Convert F_i to omni-directional image coordinates
polar-to-Cartesian(F_i)
10: Find camera ray pair $({}^1P_i, {}^2P_i) = \pi^{-1}(F_i)$ (Eq. (1.2))
11: Add $({}^1P_i, {}^2P_i)$ to concordances set $\mathbb{S}$
12: **else**
13: **while** size(${}^1a_i$) $< th_e$ **do**
14: Divide ${}^1a_i$, ${}^2a_i$ into four square sub-frames ${}^1a_{i.j}$, ${}^2a_{i.j}$ with $1/4$ size
15: Repeat line 5 to 17 on ${}^1a_{i.j}$, ${}^2a_{i.j}$
16: **end while**
17: **end if**
18: **end for**
19: Transformation T = RANSAC(Five-Points-Algorithm($\mathbb{S}$))
20: **Output:** T

2.2 Estimation of Concordant Points with FMT Registration

2.2.1 Selection of Sub-Images

Assume two images ${}^1I_o$ and ${}^2I_o$ are captured by the omni-directional camera in two different poses ${}^1C$ and ${}^2C$, so that the 6-DoF transformation ${}^1_2T$ describes the motion between the poses. Furthermore, the images ${}^1I_p$ and ${}^2I_p$ are related panorama images. We determine the transformation ${}^1_2T$ in two steps:

1. by using FMT registration (Sec. 2.2.2) to generate a motion flow field of concordant points (Sec. 2.2.3) that gets projected into calibrated views (Sec. 2.2.4)
2. and by estimating the transformation ${}^1_2T$ from the set of concordant points (Sec. 2.4).

Before addressing the different parts of the first step in the remainder of this section, the first question is how each of the pairs of sub-images ${}^1a_i$ and ${}^2a_i$ is selected from ${}^1I_p$ and ${}^2I_p$ for FMT registration. One option would be an exhaustive approach in the spirit of [63], who demonstrated that a dense motion flow can be computed with FMT. But even with their Kernel sizes of just 32×32 pixel, i.e., sub-images so small that the accuracy and robustness of FMT tends to suffer [127], the approach is computationally very expensive [63].

Note that there are two aspects in the selection of the sub-images. First, there is the size $N_a \times N_a$ of each sub-image, which directly relates to the robustness under challenging visibility conditions. This aspect is among others also discussed and evaluated in experiments in Sec. 2.5.3. To optimize robustness of FMT, large sub-images are taken in our approach. In the application oriented experiments in Sec. 2.5.3, we use for example the full height ${}^yN_p$ of the panorama images I_p, i.e., $N_a = {}^yN_p$. Second, not only the size, but also the number of sub-images matters. A dense selection of sub-images, i.e., the use of Kernels at all or many coordinates of I_p, is, as mentioned, computationally very expensive. Here, we demonstrate that a sparse approach, i.e., the use of only a few sub-images, is not only efficient but also sufficient for an accurate estimation. An important aspect to achieve this is the recursive sub-image strategy introduced in Sec. 2.3.

For the time being, we denote the two sub-images sets for FMT registration with ${}^1\mathbb{A} = \{{}^1a_1, {}^1a_2, {}^1a_3, ..., {}^1a_m\}$ and ${}^2\mathbb{A} = \{{}^2a_1, {}^2a_2, {}^2a_3, ..., {}^2a_n\}$, i.e., each ${}^1a_i$ is registered with its related ${}^2a_i$ ($i \in \{1, ..., m\}$). Each pair ${}^1a_i, {}^2a_i$ is selected from the same square window w_i, i.e., the same set of coordinates, in the two consecutive images ${}^1I_p$ and ${}^2I_p$ of the video stream. As discussed later on in more detail, the (few) windows w_i are placed in a fixed, regular pattern on I_p.

2.2.2 2.5D Apparent Motion Estimation between Sub-Images with FMT

The apparent 2.5D motion between the sub-images ${}^1a_i$ and ${}^2a_i$ is determined by FMT registration. FMT requires square images ${}^1a_i$ and ${}^2a_i$ of size $N_a \times N_a$ as input. When considering registration in general, images tend to be in a rectangular format. In that case, either a square cut-out can be used or simple Zero-padding allows using the full image content. In this work, we use the former one since the latter one will make the most region of the image be zeros if padding too much zeros. The classical FMT has previously been introduced in Sec. 1.4, which will be excluded in this chapter. In the work presented here, we use a variant dubbed FMT based on [22].

2.2.3 The FMT Motion Flow Field

To estimate the motion flow field $\mathbb{M}$, the concordant points of two panorama images ${}^1I_p, {}^2I_p$ need to be found, i.e., the concordant points of the sub-image sets ${}^1\mathbb{A}, {}^2\mathbb{A}$. First, the apparent motion m_i between corresponding sub-images ${}^1a_i, {}^2a_i$ is computed on the basis of 2.5D FMT registration (Sec. 2.2.2).

Let m_i be the result of the FMT registration of ${}^1a_i$ and ${}^2a_i$, i.e., the 4-DoF parameters of scale s, rotation θ and translation t_x, t_y between ${}^1a_i$ and ${}^2a_i$ are $m_i = [s\ \theta\ t_x\ t_y]^\top$. The pixel ${}^2p_{a_j} = (u'_2, v'_2)$ in ${}^2a_j$ that is concordant to pixel ${}^1p_{a_i} = (u'_1, v'_1)$ in ${}^1a_i$ under m_i is given by:

$$\begin{bmatrix} u'_1 \\ v'_1 \end{bmatrix} = \begin{bmatrix} u'_2\alpha - v'_2\beta + c_x(1-\alpha) + c_y\beta + t_x \\ u'_2\beta + v'_2\alpha - c_x\beta + c_y(1-\alpha) + t_y \end{bmatrix}, \tag{2.1}$$

where $\alpha = s\cos\theta$, $\beta = s\sin\theta$ and (c_x, c_y) are the center coordinates of the sub-window w_i used to determine ${}^1a_i, {}^2a_i$. Based on this, all concordant points for the pair of sub-image ${}^1a_j, {}^2a_j$ can be found via Eq. (2.1) - though it is sufficient to use a single pair of concordant points as they all represent the same apparent motion m_i. So, one arbitrary pixel ${}^kp_{a_i} = (c_x + \delta, c_y + \delta)$ with $\delta > 0$ is chosen of each sub-image ${}^ka_i$ to represent the motion flow. Note that δ should not be Zero to ensure that rotation and scaling are properly represented. An example of a motion flow field of concordant points derived from FMT registrations of sub-images is shown in Fig. 2.2c.

2.2.4 Projection of Concordant Points into Calibrated Views

The transformation between two camera poses ${}^1C$ and ${}^2C$ can be estimated on the basis of concordant points according to Eq. (2.1) using the Five-Points-Algorithm (Sec. 2.4). But this requires calibrated views, i.e., we have to project the concordant points into the according views. Based on the camera calibration (Sec. 1.1.1), we associate each pixel (u, v) in ${}^1I_o$ to the undistorted and normalized camera ray ${}^1P = [u, v, f(u, v)]^T$ in the ${}^1C$ coordinate frame using the omni-directional camera model. The same procedure

(a) First Panorama Image

(b) Second Panorama Image

(c) Motion Flow Field

Figure 2.2: An example of a motion flow field between two panorama images, which are horizontally divided into 10 sub-images. The red dot is point ${}^1p_{a_i}$ in the first sub-image while the green is the concordant point ${}^2p_{a_i}$ in the second sub-image. Each arrow represents the 2D component of the apparent motion between sub-images that is determined by FMT; the dots and arrows are scaled for sub-images 5 to 7.

is done for the image ${}^2I_o$.

Then, based on the pairs $({}^1p_{a_i}, {}^2p_{a_i})$ of concordant points that represent the motion flow field, the corresponding camera rays $({}^1P_i, {}^2P_i)$ are computed. It is important to note that for this, the concordant points $({}^1p_{a_i}, {}^2p_{a_i})$ are converted back from panorama image to omni-directional image coordinates. The relation between each 3D pair $({}^1P_i, {}^2P_i)$ can be described as

$$ {}^1P_i{}^T E {}^2P_i = \mathbf{0} \,, \tag{2.2}$$

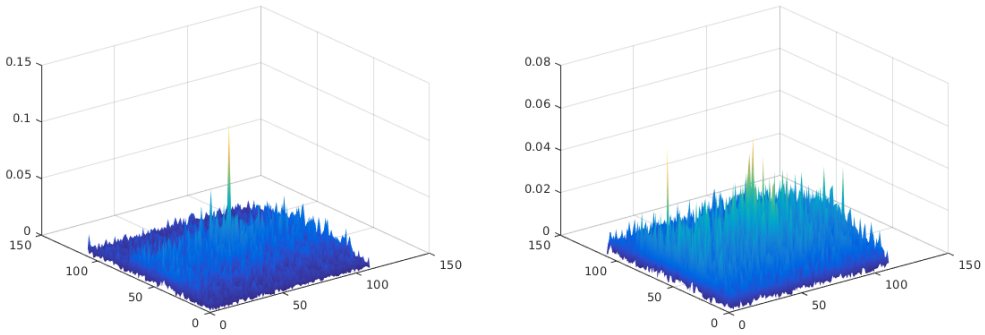
where the matrix E is the essential matrix based on epipolar geometry [60]. Finally, the rotation and translation between two camera poses can be extracted from the matrix E.

2.3 Detecting Registration Success and Recursive Sub-Images

2.3.1 The Signal-to-Noise-Ratio (SNR) in FMT

The final filter step in FMT estimates the translation part of the 2.5D motion. It yields in the ideal case a Dirac pulse that indicates the according 2-DoF translation parameters. In reality, there is noise in the sensor, a not perfectly static and flat world, and no perfect underlying 2.5D motion. Hence, the Dirac pulse becomes a flatter and broader peak and there is a substantial amount of noise in the parameter space.

Fig. 2.3 shows two examples of the PSDs during the translation calculation with FMT.



(a) High SNR (b) Low SNR

Figure 2.3: Examples of Phase Shift Diagrams (PSDs) with different Signal-to-Noise Ratios (SNR) in two different FMT registrations. With a high SNR, one distinct peak indicates the correct 2-DoF translation parameters in the final POMF in FMT. With a low SNR, the registration is likely unsuccessful; furthermore, multiple peaks can indicate multiple apparent motions that suggest a further sub-division of the sub-image.

In Fig. 2.3b, the peak is less distinct and the noise level is larger than in Fig. 2.3a, i.e., the two diagrams have different signal-to-noise-ratios (SNR). For FMT, the SNR is of great interest as it can indicate whether a registration is successful or not.

Two different options to calculate the SNR are considered here. The first one is the ratio PR between the peak energy and the energy of the second highest peak, i.e.:

$$PR = \frac{E_{1stpeak}}{E_{2ndpeak}} \,. \tag{2.3}$$

The second one is the ratio PNR between the peak energy to the sum of noise energy in the $\pm 10 \times \pm 10$ neighborhood of the peak, i.e.:

$$PNR = \frac{E_{peak}}{\sum_{i,j} E_{noise}(i,j)} \tag{2.4}$$

with $i \in \{Px-10, Px+10\}$, $j \in \{Py-10, Py+10\}$ and $peak = (Px, Py)$. The higher the SNR, the more reliable the estimated motion vector is.

As mentioned, successful registrations, i.e., correctly estimated motion parameters, and failed registrations, i.e., estimated motion parameters that are far from the correct ones due to too high noise and structural interferences in the two images, can usually be clearly distinguished as the two cases have significantly different SNR. Hence, a simple threshold can be used for this distinction. The two thresholds for the two ways to estimate the SNR are denoted with th_{PR} for PR and with th_{PNR} for PNR.

2.3.2 Further Sub-Division of Sub-Images

Suppose the SNR indicates that a registration of two sub-images ${}^1a_i$ and ${}^2a_i$ was unsuccessful. One option is to simply discard this registration attempt and to only use the successful ones for generating the set $\mathbb{S}$ of concordant points. But it can be that the two to be registered sub-images just violate the 2.5D motion assumption and that there are multiple apparent motions in ${}^1a_i$ and ${}^2a_i$. For example, the multiple peaks in Fig. 2.3b indicate that there are several different motion vectors between the two to be registered sub-images.

Therefore, it is a possible strategy to try to capture these different apparent motions instead of just discarding the information in ${}^1a_i$ and ${}^2a_i$. To this end, the window w_i in which the two sub-images ${}^1a_i$ and ${}^2a_i$ are located is further sub-divided into four equal parts (Alg. 1, line 6 & lines 13-17). More precisely, given the square window w_i of size N and with offset (x_o, y_o), i.e.:

$$\begin{aligned} &w_i \equiv \{(x,y)\} \text{ with} \\ &\quad x_o \leq x < x_o + N \wedge y_o \leq y < y_o + N \end{aligned} \tag{2.5}$$

The four subdivision windows $w_{i.j}$ are then given by

$$\begin{aligned} &j \in \{1,...,4\} : w_{i.j} \equiv \{(x,y)\} \text{ with} \\ &\quad x_o + (j-1)\cdot\frac{N}{4} \leq x < x_o + j\cdot\frac{N}{4} \\ &\quad y_o + (j-1)\cdot\frac{N}{4} \leq y < y_o + j\cdot\frac{N}{4} \end{aligned} \tag{2.6}$$

The related two times four further subdivided sub-images of ${}^1a_i$ and ${}^2a_i$ are then accordingly denoted with ${}^1a_{i.j}$ and ${}^2a_{i.j}$ with $j \in \{1,...,4\}$.

This strategy can of course be recursively applied to any of the windows $w_{i.j}$ in case there is an unsatisfactory FMT registration result for ${}^1a_{i.j}$ with ${}^2a_{i.j}$, i.e., the four further sub-windows $w_{i.j.k}$ ($k \in \{1,...,4\}$) can be generated to try the registration of the four ${}^1a_{i.j.k}$ with their related ${}^2a_{i.j.k}$.

There are two aspects that limit this recursion. First, the approach presented here benefits from the fact that the 2.5D registration of a few sub-images is sufficient to determine the 3D transformation T. Having additional registrations on further subdivided sub-images that may further confirm T, i.e., that produce more RANSAC inliers (Sec. 2.4), is nice but it comes at a computational overhead. Second, a core advantage of FMT is its robustness to poor visibility conditions, dynamics, etc., as it is a spectral method, i.e., it operates on the whole range of structures in the frequency domain. To

be able to do so, FMT requires proper images, i.e., a reasonable size of the input pixel arrays. An analysis by [127] suggests that FMT should preferably be used - with already a trade-off in accuracy - down to image sizes of 64×64 pixel, which puts a natural limit on the recursion.

These two aspects are captured in the threshold th_e on the size of the sub-windows, respectively sub-images (Alg. 1, line 13).

2.4 From Concordant Points to 3D Motion

Given the set $\mathbb{S}$ of back-projected concordant points, the last step (Alg. 1, line 19) is to use the Five-Point-Algorithm [97, 98] to determine the 6-DoF rigid transformation T between the two poses ${}^1C$ and ${}^2C$ from which the omni-directional images ${}^1I_o$ and ${}^2I_o$ are taken. For the implementation, the variant dubbed Stewenius-5-Point in OpenGV [72] is used, which is based on [134, 101].

The use of the five-point-algorithm is embedded into RANSAC [46] in a straightforward way. By randomly selecting different combinations of concordant point pairs, the transformation T with the most inliers is determined (Alg. 1, line 19).

2.5 Experiments and Results: Ablation Study

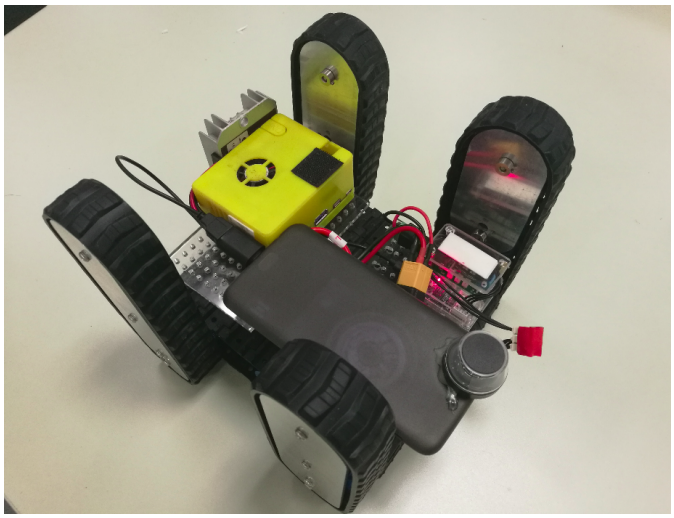
In this section, we evaluate the core parameters of the 3D spectral VO, including SNR,sub-image selection and sub-division strategy. The suitable parameters obtained in these experiments are used in Sec.9. for the comparison with other methods. In Sec.9, we evaluate these methods with geometric, rotation and translation errors. In addition, robustness is analyzed under degraded visibility cases.

2.5.1 The Datasets used in the Experiments

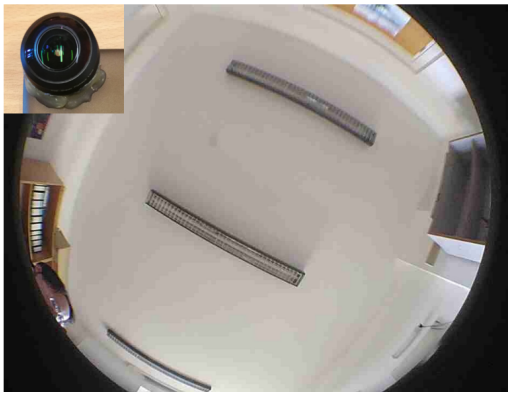
In the experiments presented in the following sections, four datasets are used. Examples of them are shown in Fig. 2.5. Two of them were generated by the authors and two of them are 3rd party datasets.

Fig. 2.4a shows the exploration robot which is used for our two datasets. It is a low-cost system for search and rescue operations. It has a smart phone with a low-cost omni-directional lens on top of the regular camera, which provides omni-directional images. The video streams from this robot are captured once in an indoor and once in an outdoor environment; the respective datasets with 500 images each are denoted as the *office* and the *lawn* datasets.

The third dataset, which is here referred to as the *CVLIBS* dataset, is based on [124,



(a) Low-cost exploratory robot



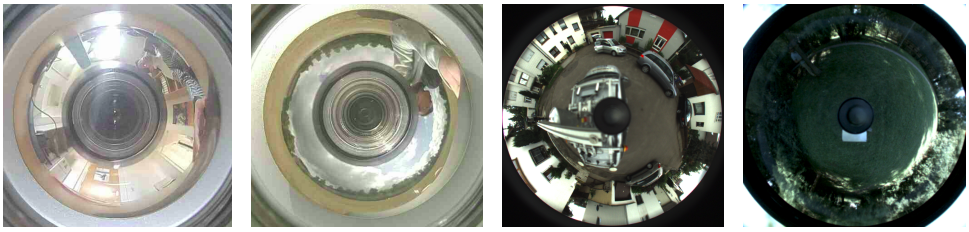
(b) Fisheye lens image

(c) 360° lens image

Figure 2.4: Two of the datasets used in the experiments are collected by a low-cost robot, which is controlled by a smartphone with a fisheye lens to generate omni-directional images.

125]. It contains 12607 omni-directional images from urban scenes collected from a driving platform. The first driving sequence of 200 frames with a focus on rotation is used for evaluation here.

The *OVMIS* dataset is the fourth one used in the experiments. It was generated by [93]. It includes several grass images captured by an omni-directional camera on a mobile robot.

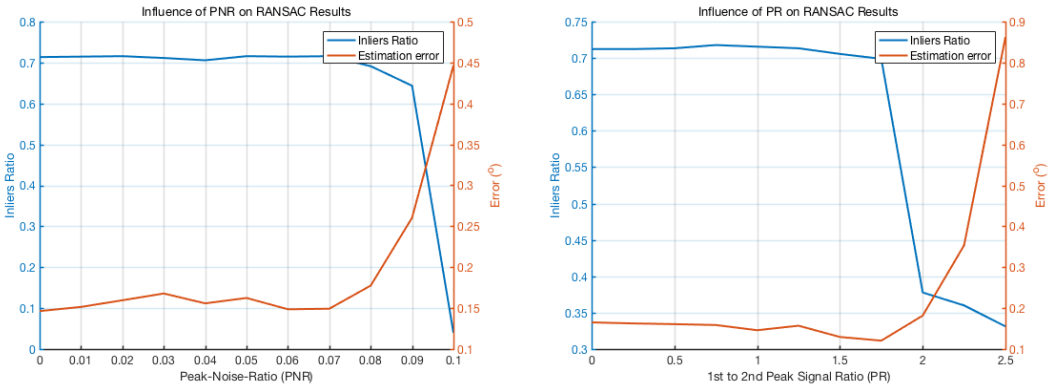


(a) *office* (b) *lawn* (c) *CVLIBS* (d) *OVMIS*

Figure 2.5: Example images from the four datasets

For all datasets, the measurements of the Inertial Navigation Systems (INS) of the respective platforms are also available as a comparison basis.

2.5.2 Evaluation of the Estimation of the Signal-to-Noise-Ratio (SNR)



(a) ratio between the peak energy and the neighborhood noise (PNR)

(b) ratio between the energy of the main peak and of the 2nd peak (PR)

Figure 2.6: Illustrative examples for the influence of the thresholds th_{pr} and th_{pnr} for the two methods PNR and PR to estimate the Signal-to-Noise-Ratio (SNR)to detect successful FMT registrations. It can be observed that both methods perform similarly and that over an extended range of the threshold settings, outliers get already excluded before the RANSAC step.

As discussed in Sec. 2.3.1, the signal-to-noise-ratio (SNR) in the last filter step of FMT indicates whether the registration was successful or not. Two methods are considered

to estimate the SNR in an efficient way, namely the ratio PR between the peak energy and the energy of the second highest peak and the ratio PNR between the peak energy to the sum of noise energy in the $\pm 10 \times \pm 10$ neighborhood of the peak. The respective thresholds to consider a registration successful or not are denoted with th_{pr} and th_{pnr}.

To illustrate the effects of the two methods, we evaluate them with respect to their effects on the use of RANSAC in the final estimation of the transformation T (Sec. 2.4). Concretely, when unsuccessful registrations get immediately rejected through the use of PR and/or PNR, i.e., the related point pairs are not part of the set $\mathbb{S}$ of concordant points used in RANSAC, the inliers' ratio should increase and the estimation error should decrease as a result.

Fig. 2.6 shows the inliers' ratio and the estimation error for PNR and PR over a range of thresholds th_{pr} and th_{pnr}, respectively. In this experiment, ten frames from the *office* dataset are used. The rotation between each two frames is five degrees in pitch. Each test is repeated ten times to obtain the average inliers ratio and error.

Two important observations can be made. First, the two methods PNR and PR to estimate the SNR behave quite similarly. Second, there is a range for both thresholds in which they lead to good results, i.e., only if they are set too high, a substantial amount of unsuccessful registrations get included in $\mathbb{S}$. This is in line with the general observation that the registration success of FMT tends to lead to clearly distinguishable SNR values.

Given these observations, we just use the PNR method. Furthermore, the related threshold th_{pnr} is set to 0.06 in all experiments.

2.5.3 Evaluation of the Sub-Image Selection

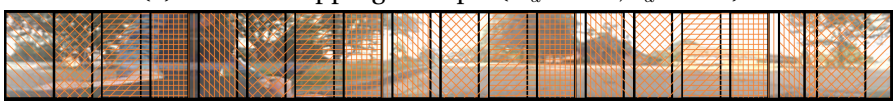
As discussed in Sec. 2.2.1, the panorama images ${}^1I_p$ and ${}^2I_p$ are divided into m sub-images ${}^1a_i$ and ${}^2a_i$ ($i \in \{1, ..., m\}$) with square size $N_a \times N_a$. Each pair ${}^1a_i$,${}^2a_i$ is taken from the same window w_i. The windows w_i are placed in a regular pattern over the images and potentially partitioned into four sub-windows.

For the windows w_i, there are hence two parameters of interest, namely the window size N_a and the stride s_a, which is used to determine the offset with which the next window w_{i+1} is placed in x-direction, respectively in y-direction for the next row of windows. Depending on the choice of the size N_a and of the stride s_a, windows can be overlapping or non-overlapping. For example, Fig. 2.7a shows the non-overlapping division with $N_a = 110$ and $s_a = 110$ whereas Fig. 2.7b presents an example of overlapping with $N_a = 110$ and $s_a = 60$.

In the following experiment, we illustrate the effects of the two parameters with respect to time and motion estimation errors. The aspects of the recursive sub-division strategy

are evaluated in additional experiments below.

(a) Non-overlapping example ($N_a = 110$, $s_a = 110$)



(b) Overlapping example ($N_a = 110$, $s_a = 60$)

Figure 2.7: Examples including non-overlapping and overlapping windows to generate the sub-images.

In this experiment, images from the *office* and the *lawn* datasets with ground truth poses are used for evaluation. Concretely, there are three sets of controlled motions with concurrent changes in roll, pitch and yaw at intervals of $2°$, $5°$ and $10°$ (Table 2.1).

The resolution of the omni-directional images is 1280×720 and the resolution of the related transformed panorama images is 1100×110.

The latter are divided horizontally and vertically into sub-images of three different sizes N_a: 110×110, 64×64 and 32×32. Furthermore, different strides for the placement are evaluated for each size. The different combinations are shown in Table 2.1.

Note that the maximum cardinality $\#_{\max}\mathbb{S}$ of the set $\mathbb{S}$ of concordant points is determined by the stride in this experiment in a straightforward manner. Given the image size ${}^yN \times {}^xN$ and a stride of s_a, there are $m = \lceil \frac{{}^yN}{s_a} \rceil \cdot \lceil \frac{{}^xN}{s_a} \rceil$ sub-windows w_i and hence also m FMT registration attempts between sub-images ${}^1a_i$ and ${}^2a_i$. The ratio of the image height, respectively width and the stride, i.e., $\frac{{}^yN}{s_a}$, respectively $\frac{{}^xN}{s_a}$, is not necessarily an integer. In that case, simple shifts to align right/down edges can be used to generate square sub-images for the FMT registration (Sec. 2.2.2). The maximum cardinality $\#_{\max}\mathbb{S}$ hence varies in this experiment from extremely sparse 10 with stride $s_a = 110$ to a much denser coverage of 315 with a stride $s_a = 10$ and sub-image size $N_a = 32$ (see the third column of Table. 2.1).

The window size N_a is related to the robustness of the FMT registration and its computation time. The larger N_a, the more robust FMT tends to be, but its computation time also increases.

Based on the results, the following two core observations can be made:

(1) There is a correlation between the sub-image size N_a and the accuracy, i.e., the inverse of the error, under large rotational changes. For example, the use of 32×32 sub-images yields substantial errors ($\approx 5°$) when the instantaneous rotations between two images are $\geq 5°$. It is substantially more robust for 64×64 sub-images when substantial

		yaw			pitch			roll		
size	stride	$\epsilon(^\circ)$	σ^2	$t(s)$	$\epsilon(^\circ)$	σ^2	$t(s)$	$\epsilon(^\circ)$	σ^2	$t(s)$
	110	0.046	0.0000	**0.110**	**0.010**	0.0015	0.122	0.083	0.0006	0.112
	60	0.048	0.0000	0.174	0.089	0.0000	0.180	0.226	0.0289	0.190
110	30	0.050	0.0000	0.298	0.079	0.0002	0.313	0.830	0.1156	0.301
	20	0.043	0.0000	0.434	0.126	0.0056	0.524	0.378	0.0556	0.439
	10	0.044	0.0000	0.842	0.191	0.0225	0.960	0.372	0.0752	0.822
	64	0.029	0.0000	0.114	0.057	0.0003	0.153	**0.044**	0.0002	**0.105**
	30	0.035	0.0000	0.269	0.043	0.0002	0.382	0.057	0.0001	0.255
64	20	**0.028**	0.0000	0.486	0.044	0.0000	0.716	0.051	0.0003	0.472
	10	0.039	0.0000	1.338	0.039	0.0001	1.344	**0.044**	0.0002	1.300
	32	0.102	0.0012	0.119	0.063	0.0005	**0.121**	0.051	0.0003	0.115
	20	0.105	0.0004	0.196	0.079	0.0006	0.191	0.076	0.0006	0.195
32	10	0.094	0.0007	0.588	0.058	0.0004	0.564	0.072	0.0011	0.562

(a) Motion Interval: 2°

		yaw			pitch			roll		
size	stride	$\epsilon(^\circ)$	σ^2	$t(s)$	$\epsilon(^\circ)$	σ^2	$t(s)$	$\epsilon(^\circ)$	σ^2	$t(s)$
	110	0.029	0.0000	**0.116**	0.149	0.0003	**0.118**	0.260	0.2138	**0.117**
	60	0.031	0.0000	0.203	0.312	0.0736	0.187	0.213	0.1386	0.185
110	30	**0.027**	0.0000	0.345	0.179	0.0226	0.336	0.105	0.0017	0.332
	20	0.028	0.0000	0.469	0.560	0.4221	0.479	0.153	0.0038	0.489
	10	**0.027**	0.0000	0.901	0.146	0.0088	0.907	0.136	0.0011	0.867
	64	0.076	0.0004	0.129	0.096	0.0011	0.130	0.157	0.0010	0.124
	30	0.052	0.0001	0.288	**0.077**	0.0003	0.303	0.098	0.0008	0.282
64	20	0.051	0.0001	0.526	0.106	0.0006	0.549	0.118	0.0005	0.521
	10	0.055	0.0002	1.430	0.078	0.0003	1.393	**0.093**	0.0001	1.385
	32	5.032	0.0828	0.129	0.138	0.0010	0.148	0.151	0.0015	0.130
	20	4.923	0.0912	0.205	0.112	0.0004	0.210	0.121	0.0009	0.201
32	10	4.935	0.0272	0.598	0.112	0.0008	0.761	0.131	0.0011	0.630

(b) Motion Interval: 5°

		yaw			pitch			roll		
size	stride	$\epsilon(^\circ)$	σ^2	$t(s)$	$\epsilon(^\circ)$	σ^2	$t(s)$	$\epsilon(^\circ)$	σ^2	$t(s)$
	110	0.095	0.0001	0.110	**0.169**	0.0034	**0.099**	0.542	0.0704	0.116
	60	0.068	0.0005	0.190	0.531	0.0120	0.153	0.421	0.0080	0.161
110	30	0.084	0.0003	0.325	0.709	0.2078	0.288	0.311	0.0122	0.289
	20	0.069	0.0005	0.437	0.611	0.9730	0.362	0.551	0.6600	0.410
	10	**0.052**	0.0001	0.793	0.481	0.1668	0.760	**0.300**	0.0455	0.754
	64	7.960	4.5076	**0.106**	0.273	0.0453	0.129	0.363	0.0842	0.113
	30	9.318	7.3541	0.241	0.518	0.1076	0.252	0.473	0.0797	0.251
64	20	7.355	3.5246	0.414	0.358	0.0356	0.465	0.644	0.1699	0.445
	10	5.410	8.6105	1.127	0.268	0.0415	1.238	0.590	0.1415	1.187
	32	9.997	0.0711	0.113	8.765	4.0711	0.114	7.593	3.2097	**0.111**
	20	10.089	0.1219	0.171	8.350	3.6449	0.175	8.673	2.7465	0.174
32	10	10.077	0.0731	0.488	7.875	1.8814	0.527	6.063	2.6723	0.508

(c) Motion Interval: 10°

Table 2.1: Rotation estimation error ϵ, covariance σ^2 and run-time t with different settings for the sub-image windows

errors start at larger rotations of 10° motion intervals. The largest sub-images with 110×110 pixels can in contrast also cope with the 10° motion intervals.

(2) Given a sub-image size N_a, there is not much influence of the overlap between sub-images, i.e., of the stride s_a, on accuracy. This can be seen by the similar error values in each block of Table 2.1. The average error difference is very small, namely $\approx 0.01^\circ$, for fixed N_a and it seems to be randomly distributed, i.e., smaller strides do not necessarily lead to smaller errors. But the stride s_a has a significant influence on computation time. Hence, large strides and hence very sparse sets of concordant points can already lead to accurate solutions in an efficient way.

For example, the stride of $s_a = 110$ leads to a set of just at most 10 concordant points in this experiment, which are sufficient to get accurate motion estimates with substantial instantaneous changes in roll, pitch, and yaw of 10°. In general, given a sufficiently large N_a, the motion estimation error is quite small with $\approx 0.1^\circ$.

2.5.4 Evaluation of the Sub-Division Strategy

As mentioned in the Sec. 2.3.2, the sub-division strategy is used in all experiments. The threshold $th_{pnr} = 0.06$ as determined in Sec. 8.2 is used for sub-divison. Furthermore, the minimum resolution to which we will sub-divide is 32. To better illustrate the advantages of the strategy, we perform an experiment to evaluate the performance of the sub-division strategy. Usually, the sub-division strategy is triggered when the images include objects of different distances or dynamic objects. For example, Fig. 2.8 shows an example pair of sub-images which includes dynamic objects (hand) and objects of different distances (wall and table). Thus the sub-division strategy is triggered when performing FMT on this pair.

Figure 2.8: An example of the sub-images pair where the sub-division strategy is triggered.

Then the comparison between using and not using the sub-division strategy on these image sets are performed on the *office* pitch dataset with sub-image size $N_a = 110$ and

stride $s_a = 110$. Fig. 2.9 shows that the sub-division strategy can improve performance slightly. In this dataset, the sub-division strategy is applied to only five percent of the pairs according to the FMT registration results.

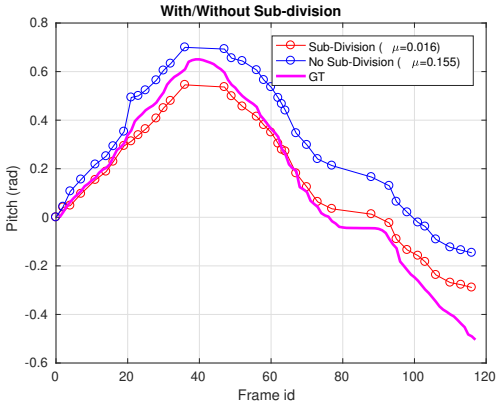


Figure 2.9: Performance on the algorithm with/without the sub-division strategy.

2.6 Experimental Comparisons to Other Methods

2.6.1 Alternative Methods used for Comparison

In the experiments presented in this section, we compare our 3D Spectral VO with several alternative methods. Concretely, two feature-based methods and one based on optical flow are used for comparison.

For the features, ORB [116] and AKAZE [7, 6] are chosen, as they cover two antithetical aspects of descriptors. ORB is known for its computational efficiency, while AKAZE is more robust, especially for images with large distortions due to their non-linearity [141]. For the VO based on ORB and AKAZE the following implementation is used. The feature matching is implemented with OpenCV[I] and the Stewenius-5-Points for the epipolar geometry is carried out by OpenGV[II]. The implementation of the optical flow method is similar, with OpenCV to calculate matched pixels and OpenGV for the epipolar geometry. We additionally compare our algorithm to the open source implementation of Semi-dense Visual Odometry (SVO2) [47].

[I] https://docs.opencv.org/
[II] https://laurentkneip.github.io/opengv/

2.6.2 The Reprojection/ Geometric Error and its Limitations

Given three omni-directional images ${}^1I_o$, ${}^2I_o$, ${}^3I_o$, with corresponding keypoints ${}^1\mathbb{K} = \{{}^1k_i\}$, ${}^2\mathbb{K} = \{{}^2k_i\}$, ${}^3\mathbb{K} = \{{}^3k_i\}$, the reprojection error ϵ_R is defined as

$$\epsilon_R = \frac{1}{L} \sum_{i \in L} \| {}^3k'_i - {}^3 k_i \| \ , \tag{2.7}$$

where L is the length of ${}^1\mathbb{K}$, ${}^3k'_i$ is the reprojected pixel in ${}^3I_o$, which is calculated by

$${}^3k'_i = K {}^1_3T P_i \ , \tag{2.8a}$$

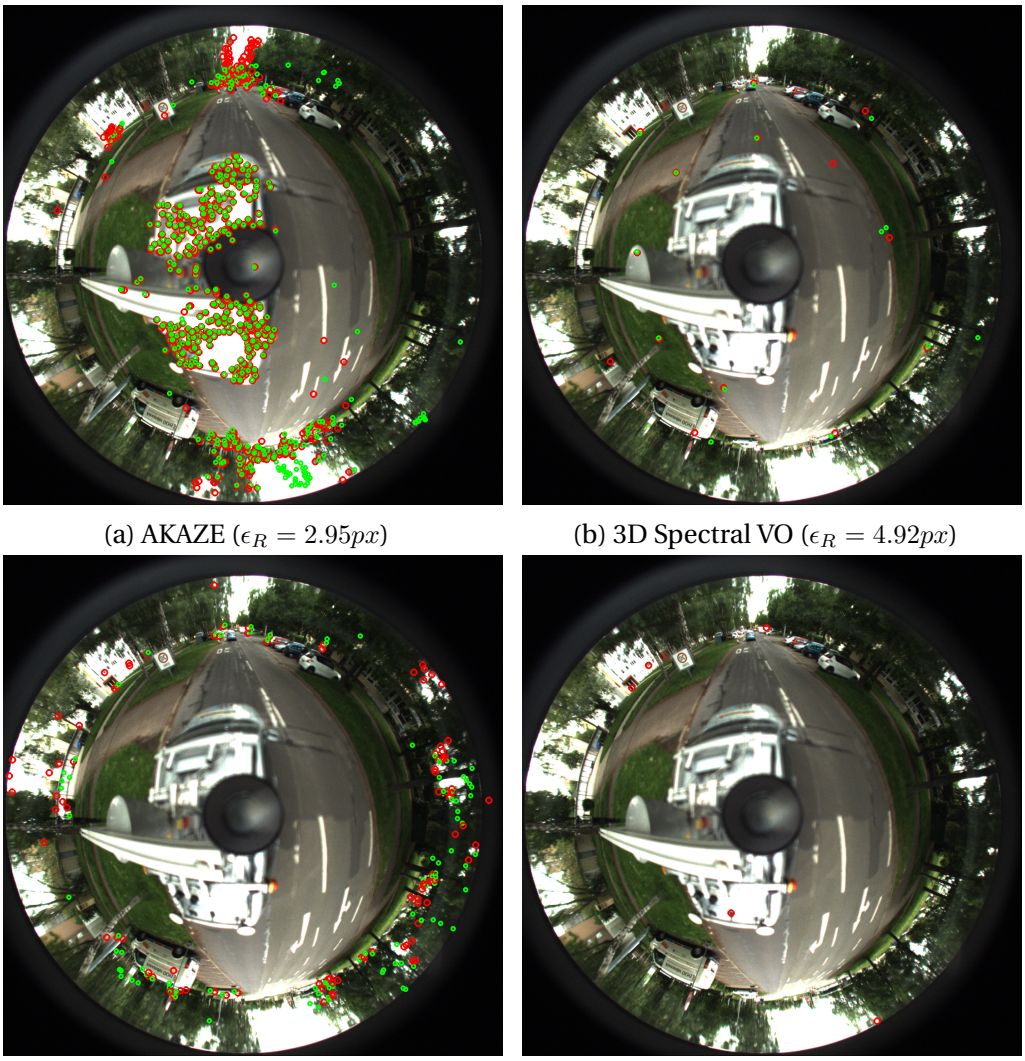
$$P_i = Triangulate({}^1k_i, {}^2 k_i, {}^1_2T) \ , \tag{2.8b}$$

where ${}^1_2T$ and ${}^1_3T$ is calculated by the transformation estimation algorithms.

Fig. 2.10 shows an illustrative example of the reprojection error of results of the four evaluated methods using the *CVLIBS* dataset. The keypoints are shown as circles; they are marked in red and the reprojected ones are marked in green. The reprojection error of the AKAZE based method is smallest with $\epsilon_R = 2.95px$. Our 3D Spectral VO follows with $\epsilon_R = 4.92px$. The optical flow method comes in 3rd with $\epsilon_R = 10.28px$. The ORB based method fails in this example.

It is important to note that there can be a substantial flaw in this form of evaluation. Most keypoints of the AKAZE based method, which appears to be performing best, are located on the autonomous car that is carrying the omni-directional camera (Fig. 2.10a). But the car is always in the center of the image at the same location, i.e., while the reprojection error is very small for these keypoints, they do not contribute to the correct estimation of the car's motion. In the contrary, they lead to the illusion that there is no motion at all. Of course, the image region of the car can be excluded from the VO. But this effect of a low reprojection error while there is a large motion estimation error can also occur for other dynamic objects in the scene. If the keypoints located on the car are not counted, the reprojection error of the AKAZE based method is $\epsilon_R = 8.64px$. Then the proposed 3D spectral VO performs best in this test.

We therefore lateron concentrate on the usage of the absolute error for comparisons using ground truth, where available, or from controlled motions based on inertial measurements, which at least for rotations provide a very good comparison basis.



(a) AKAZE ($\epsilon_R = 2.95px$)

(b) 3D Spectral VO ($\epsilon_R = 4.92px$)

(c) Optical Flow ($\epsilon_R = 10.28px$)

(d) ORB ($\epsilon_R = \infty px$)

Figure 2.10: Illustrative examples of the reprojection error in the *CVLIBS* dataset [125, 124]. Note that most keypoints in the AKAZE based method are located on the autonomous car, which is constantly fixed in the center of the image.

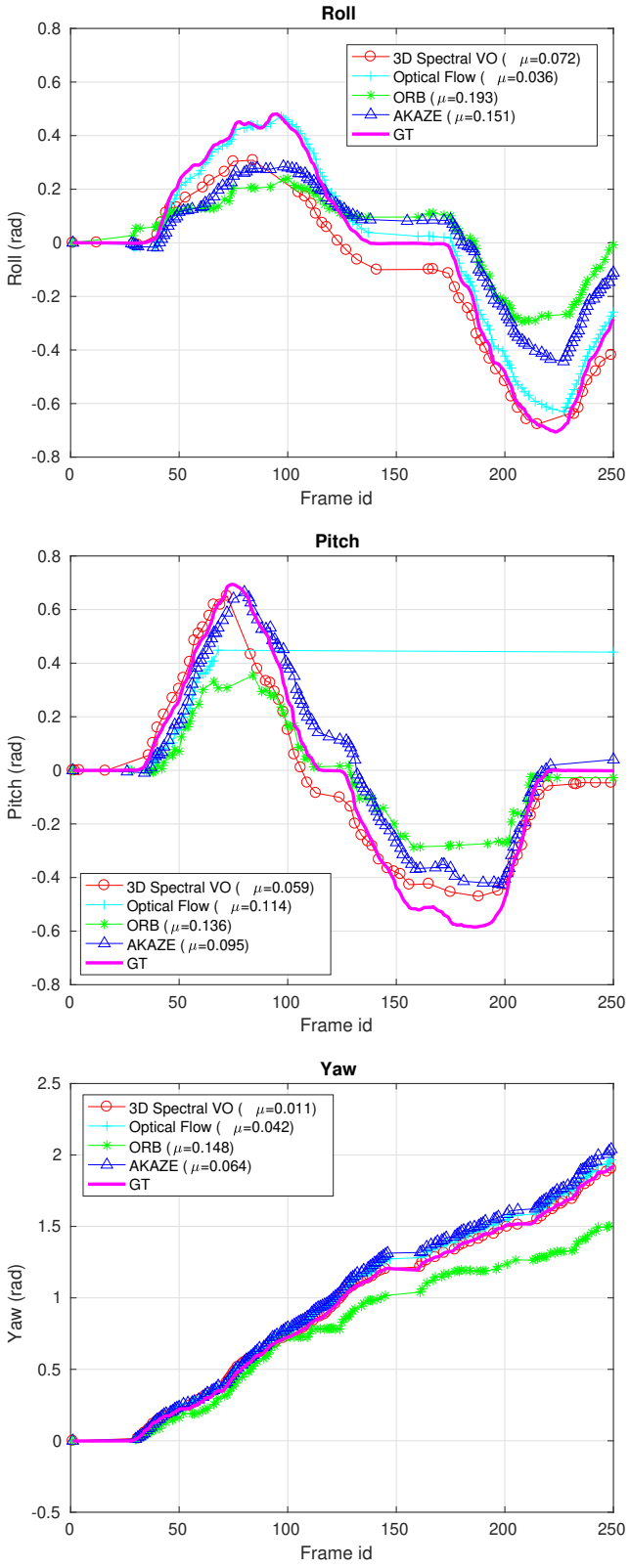


Figure 2.11: Rotation estimation on the *office* dataset and average error μ

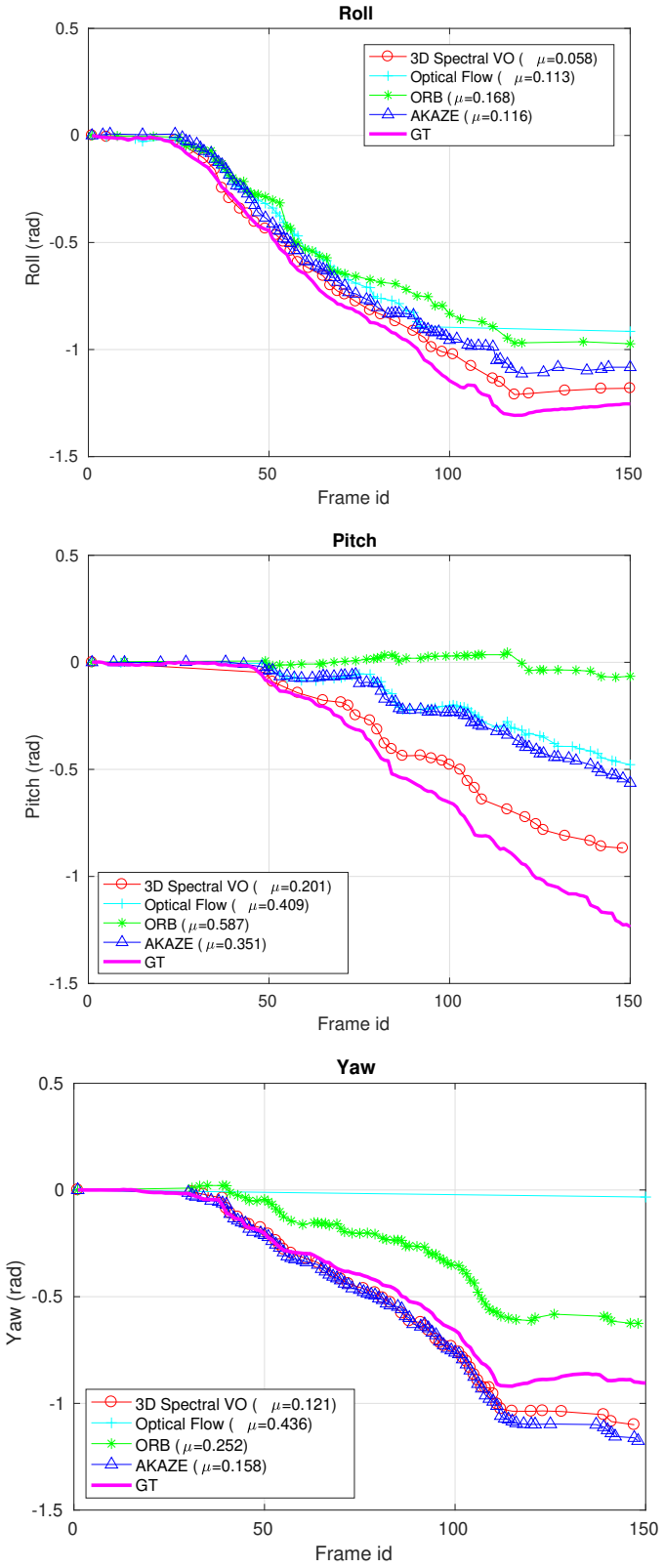


Figure 2.12: Rotation estimation on the *lawn* dataset and average error μ

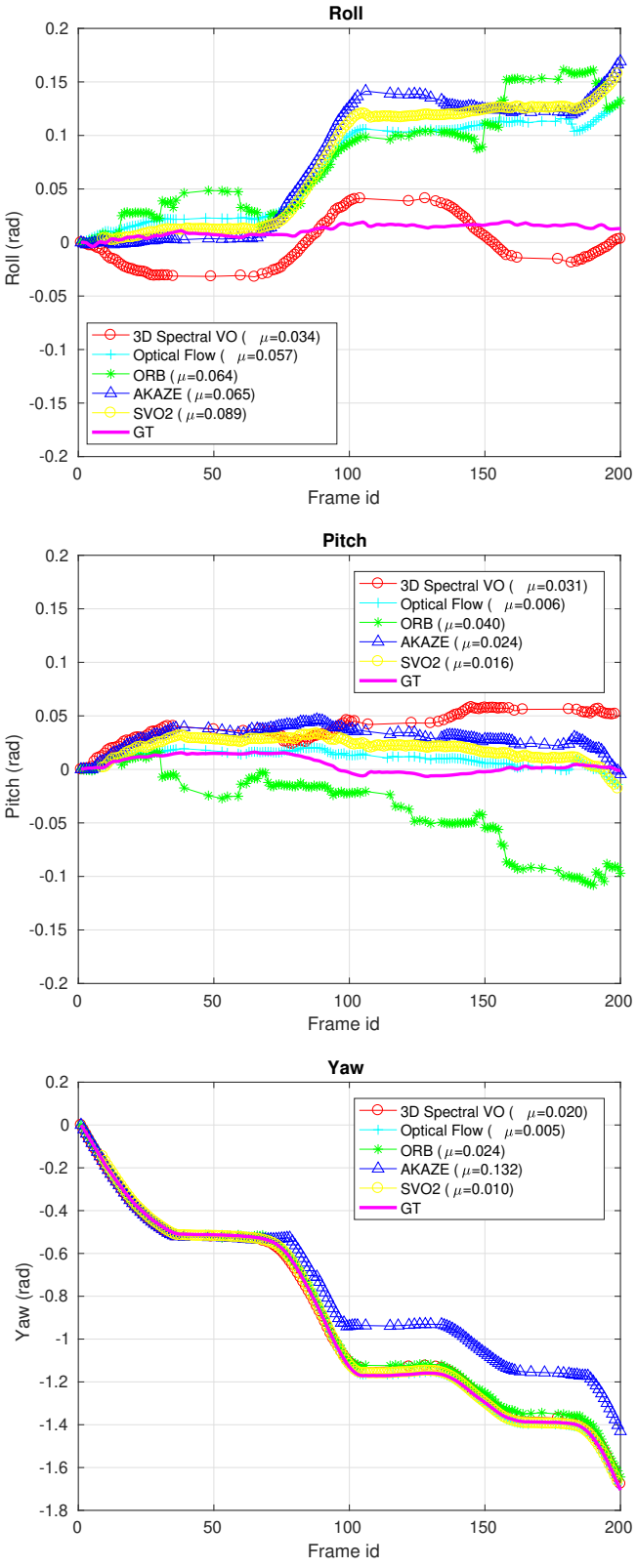


Figure 2.13: Rotation estimation on *CVLIBS* dataset [125, 124] and average error μ

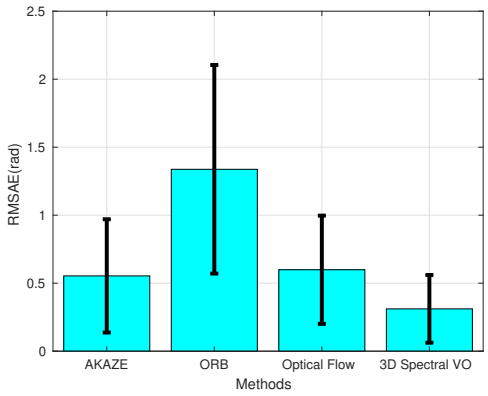


Figure 2.14: Translation error of the different methods on the *OVMIS* dataset [93].

2.6.3 Comparisons of the Different Methods on Multiple Datasets

As evaluated in Sec. 2.5.3, the 3D Spectral VO performs well for large sub-images that are sparsely placed. For the datasets with a resolution of the omni-directional images of 1280×720 and the related panorama images of 1100×110, the sub-image size N_a as well as the stride s_a are set to 110. For the *CVLIBS* dataset with 1400×1400 omni-directional images and 2670×450 panorama images, the sub-image size and the stride are set to $N_a = s_a = 256$.

For the comparison, we first evaluate the rotation estimation results on the three different datasets (*office*, *lawn*, *CVLIBS*). Fig. 2.11 depicts the results of the office scenario, which shows that the 3D Spectral VO and the optical flow method are almost always closest to the ground-truth, followed by AKAZE. In this environment, there are different objects with distinct features, e.g., books, shelves, chairs, etc.

However, for scenarios like the *lawn* dataset (Fig. 2.12), the images texture generates many ambiguous features which cannot be correctly matched. On the *lawn* dataset, AKAZE can extract around 520 features with about 365 correspondences found via feature matching. Then 190 correspondences are chosen as inliers with the five point algorithm. Whereas the total number of ORB features is 500 and 145 matches are used for estimate transformation with only about 95 inliers. For optical flow, the average total number of features is only 23 and about 17 of them are matched. Thus this dataset with ambiguous features is challenging for these methods. For example, Fig. 2.12 shows that the ORB based method is unable to track pitch motion and constantly underestimates the yaw; the optical flow fails to track the yaw motion. Our 3D Spectral VO offers the best results.

Finally, results on the *CVLIBS* dataset in Fig. 2.13 illustrate that our approach is accurate and robust in outdoor environments. In addition, only in this dataset SVO2 works with

a performance comparable to the other algorithms. This is because the phone with omni-lens cannot provide images with the high enough quality needed by SVO2, so it doesn't perform well on the *office* and *lawn* datasets.

Table 2.2 shows the root mean square error (RMSE) $\mu(\epsilon)$ and standard deviation of the rotation estimation on the three datasets. It should be noted that the RMSE of optical flow is small, because the failed cases are not included. As presented in the last row of Table 2.2, optical flow fails 28% among all the frames, which can also be found from Fig. 2.11 and 2.12.

The 3D Spectral VO performs very well. It especially keeps a robust performance in all types of environments, including the lawn environment with its ambiguous textures. Its accuracy is approximately three times larger than that of the ORB based method and two times larger than that of AKAZE.

When evaluating the rotation performance, we use RMSE between the estimated and ground truth rotation. However, the RMSE cannot be used for translation evaluation directly, because the estimated translation is up-to-scale. In addition, unlike rotations measurement, inertial motion estimates are usually quite imprecise and not well suited as comparison basis. Therefore, we use the angle of the estimated translation and groundtruth to describe the error of translation, which is abbreviated to RMSAE (root mean square angle error) in the following.

Fig. 2.14 compares the RMSAE and the standard derivations of the translation estimation for different approaches on the *OVMIS* dataset [93]. One image sequence of the outdoor scenario is used in this experiment, which contains 318 images for a 25.27 meters trajectory. To avoid the influence of accumulative error, we only calculate the pair-wise translation error between frames. The results of Fig. 2.14 show that our method provides the best performance on this dataset.

Table 2.2: Average error of Rotation estimation on all datasets

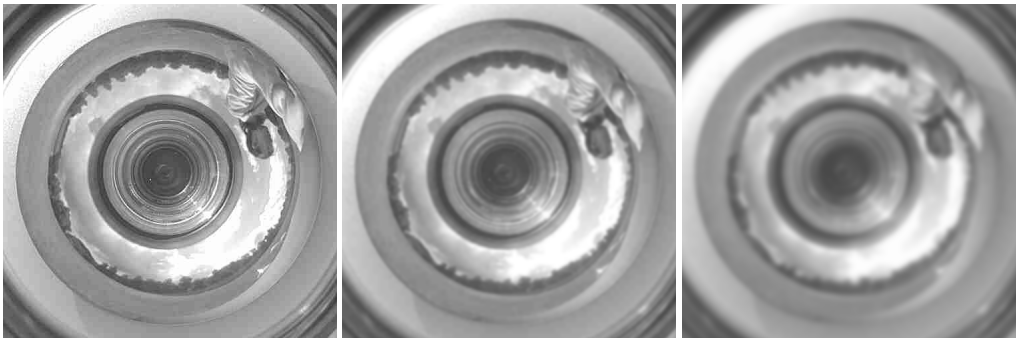
		3D Spectral VO	Optical Flow	ORB	AKAZE
roll	$\epsilon[rad]$	**0.061** $\pm$ 0.038	0.074 $\pm$ 0.045	0.163 $\pm$ 0.105	0.131 $\pm$ 0.075
pitch	$\epsilon[rad]$	**0.113** $\pm$ 0.090	~~0.240 $\pm$ 0.207~~*	0.293 $\pm$ 0.248	0.202 $\pm$ 0.164
yaw	$\epsilon[rad]$	**0.084** $\pm$ 0.065	~~0.057 $\pm$ 0.050~~*	0.218 $\pm$ 0.153	0.152 $\pm$ 0.097
$\mu(\epsilon)$	$[rad]$	**0.088** $\pm$ 0.068	~~0.136 $\pm$ 0.120~~ *	0.227 $\pm$ 0.174	0.163 $\pm$ 0.115
Fail	[%]	0	28	0	0

$*$ At certain frames optical flow failed to track pitch and yaw.

2.6.4 Robustness Tests under Degraded Visibility Conditions

As discussed in Sec. 1.4, spectral registration is performing very well under challenging visibility conditions such as motion blur, smoke, fog, or underwater turbidity. To evaluate this in a systematic way, we use image blur with a Gaussian kernel on the images from the *office*, *lawn* and *CVLIBS* datasets to simulate according conditions. Fig. 2.15 gives some examples of different blur levels from 0 to 20 pixels.

(a) Office scenario



(b) Lawn scenario

(c) *CVLIBS* dataset

Figure 2.15: Blur images with blur size 0, 10 and 20 pixels. Blur size i represents Gaussian noise with kernel size $(2i+1, 2i+1)$, standard deviation $i/5$ is added to the images.

For these experiments we selected 5 image pairs. For each pair one of the images

was blurred. We ran each test 10 times to account for random components in each algorithm (e.g. RANSAC). A total of 30 different blur levels were used in the experiments, testing the roll, pitch and yaw performance at $5°$. We compare our 3D Spectral VO against AKAZE, ORB, and optical flow. The pose estimation error and standard deviation are shown in Fig. 2.17.

The AKAZE (green) based method shows the most robust performance due to its Gaussian scale space. The error of the 3D Spectral VO (dark blue) and AKAZE (green) based methods are constantly smaller than the ORB based (light blue). Meanwhile, the performance of the optical flow-based method is not as robust as the 3D Spectral VO (dark blue) and AKAZE (green) based method.

Fig. 2.17 also shows the average run-time of all the approaches in the legend. Optical flow is about ten times faster than AKAZE, whereas ORB and 3D Spectral VO are about seven times faster than AKAZE.

Thus, our approach is almost as robust to blur as feature-based methods specifically designed for large distortions (AKAZE), but much faster. ORB is faster than our 3D Spectral VO, but has a high error for the VO experiments in this work. In addition, our method is more robust than the optical flow based method, though it is slightly worse in run-time. Though being out of the scope of this article, FMT itself as well as its use in regular patterns is especially well suited for acceleration by computations with a Graphics Processing Unit (GPU) [103] or a Digital Signal Processor (DSP), and even for hardware implementation with Field-Programmable-Gate-Arrays (FPGA) [82], with which several orders of magnitude of acceleration can be achieved.

Figure 2.16: An image with dynamic objects

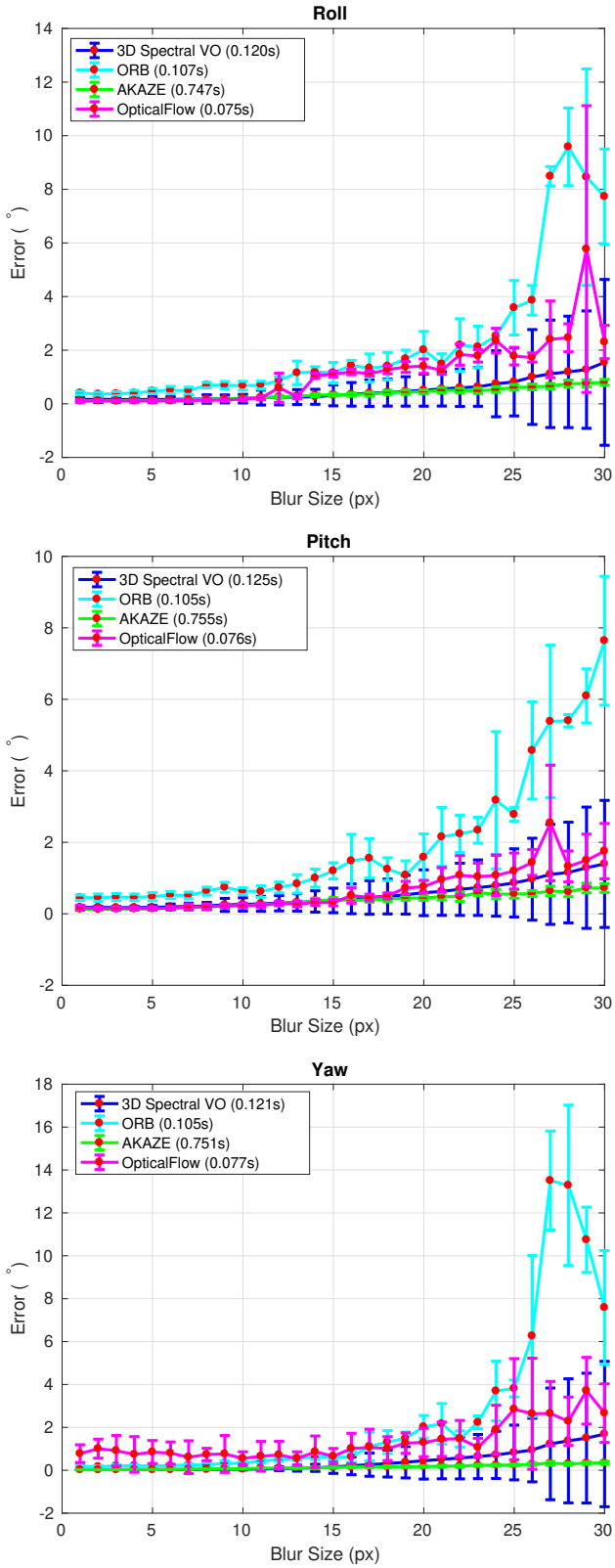


Figure 2.17: Estimation error and standard deviation of different algorithms on blurred images. Blur size i represents Gaussian noise with kernel size $(2i+1, 2i+1)$, standard deviation $i/5$ is added to the images.

2.6.5 Performance in Dynamic Scenes

For further robustness analysis, experiments in dynamic scenes are presented. For that we collected three datasets with roll, pitch and yaw, in which there is a dynamic object in every frame. One example image with a dynamic object is shown in Fig. 2.16, where the dynamic object is circled in red.

Fig. 2.18 shows the performance of the datasets with dynamic objects. It can be found that optical flow fails to track the motion when a roll or pitch happens, because the feature matches are decreased from about 30 to less than five. Although ORB performs more robust than optical flow, the accumulated error gets larger when the camera pitches. On this dataset, ORB can detect 500 features in each image and about 150 are matched. Then the five-point algorithm only considers half of the matches as inliers to estimate the motion. In addition, AKAZE detects about 750 features, of which about two-thirds are matches. Then more than half of the matches are chosen as inliers. Thus AKAZE performs better than ORB. Finally, we can see that our 3D Spectral VO gives the most robust and accurate results in such a dynamic environment.

2.7 Conclusion

In this chapter, we introduce 3D Spectral VO, a visual odometry method based on the 2.5D Fourier-Mellin-Transform (FMT) registration of a sparse set of sub-images. The method is demonstrated with the use of omni-directional images. As demonstrated in the experiments, the main potential of 3D Spectral VO is in challenging environments with, for example, featureless scenes, poor visibility conditions under, e.g., smoke, fog, motion blur, underwater turbidity, or the presence of dynamics.

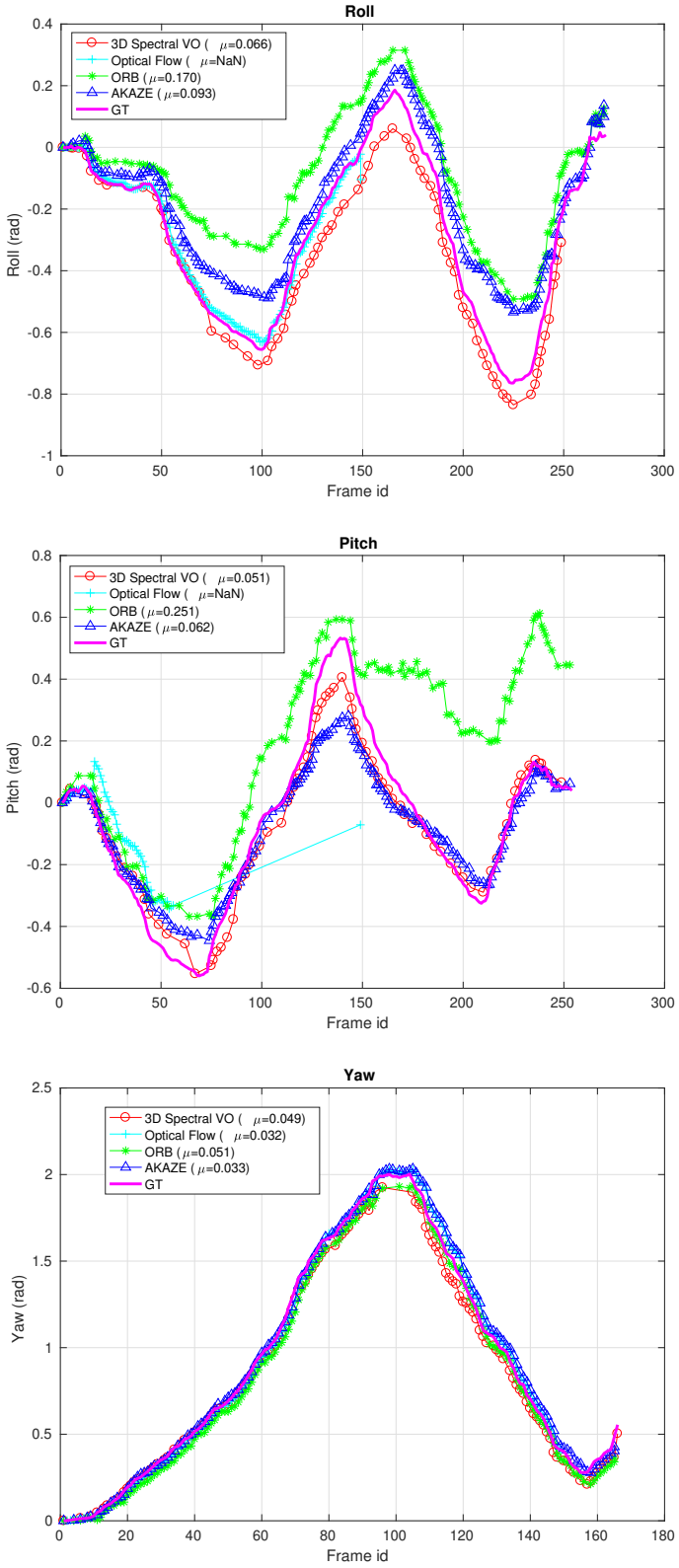


Figure 2.18: Performance evaluation on the dataset with dynamic objects and average error μ

3 Rotation Estimation for Omni-directional Cameras Using Sinusoid Fitting

Chapter 2 shows that the feature matching for omni-directional images based on FMT can achieve robust and accurate performance. Also, we use the traditional five-point algorithms to estimate the relative pose with the matching based on FMT, which requires high accuracy on camera calibration. This chapter proposes a rotation estimation method for omni-directional cameras based on sinusoidal fitting. This method only needs simple calibration, which is applicable for both low-cost and advanced cameras. We model the rotation estimation for omni-directional cameras as sinusoid fitting problems. In [76, 123], shifts in tangential and radial direction of the omni-images are exploited for a visual compass when the robot, which is equipped with an omni-directional camera, moves in the 2D plane. Inspired by this, we rethink the shifts when the camera moves in 3D space. The pixel shifts along the tangential and radial directions change depending on the radial angles (see Fig. 2.4c) when the camera rotates. Those tangential and radial shifts in the omni-image correspond to shifts along the u-axis (column) and v-axis (row) of the panorama image, respectively. Due to the properties of omni-directional geometry, the two shifts are actually sinusoid shaped, which will be discussed in detail in Sec. 3.1.2. Similar to the shifts, we can also estimate the rotation in certain regions of the panorama image, which also follows a sinusoid pattern. By fitting the estimated shifts and rotations to sinusoid functions we can thus directly estimate the 3-DoF rotation of the camera.

The sinusoid functions also have terms for the 3D translation of the camera. This work does not claim to estimate the translation of the camera because, as will be discussed in more detail in Sec. 3.1.2, the translation has several difficulties. But the experiments will show that keeping the translation terms in the sinusoid fitting functions improves the quality of the rotation estimates.

In addition, the pixel shifts and sub-image rotations should be reliable to ensure that the sinusoidal fitting can work properly. In this chapter, we will use FMT and optical flow to calculate pixel shifts and compare their performance.

The contributions in this work are summarized as:

- We propose a novel rotation estimation method based on geometric vision and fitting pixel displacement values to sinusoidal functions;
- This work exploits a 2D frequency-based algorithm as well as optical flow to estimate the 3D rotation of omni-directional cameras;
- Our algorithm is compared with commonly used epipolar geometry methods based on different feature matching methods, showing the advantages of our approach.

3.1 Methodology

3.1.1 Camera Model and Calibration

To simplify the calculation, we use the cylinder model for omni-directional cameras in this work, which is introduced in Sec. 1.1.1. Due to the property and manufacturing techniques of the cameras, we cannot ensure that each pixel is square. In other words, the same length with the same depth in the world may be projected to different pixels in $u-$ and $v-$ direction. In order to keep resolution consistent, we calibrate the pixel ratio between width and height in the beginning, thus guaranteeing square pixels. This model only has a few parameters, thus it is easy to calibrate, even for low-cost cameras with low resolution.

Note that in the panorama image there is a $u = 0$ associated with the positive x-axis of the camera (e.g. front of the robot) and another $u = \frac{1}{4}u_{max}$ pixels to the left, which is associated with the positive y-axis of the camera (e.g. left of the robot). The negative side of the axes is on the opposite side of the panorama image, that is $\frac{1}{2}u_{max}$ pixels away. Without loss of generality, in order to simplify the formulation, this work assumes that the x-axis will always locate at $u = 0$ (grey dash line in Fig. 3.1), and the y-axis is thus at $\frac{1}{4}u_{max}$ (green line in Fig. 3.1).

For the remainder of this work we assume a perfectly calibrated cylinder model with square pixels.

3.1.2 Motion Model of Panorama Images

In this subsection, it is mathematically shown that the pixel shifts in the panorama image actually follow the proposed sinusoid patterns. Sec. 3.1.3 continues then with the algorithm description regarding the motion vector extraction.

For the motion model, the following assumptions are made, which are usually satisfied in real scenarios:

1. The camera movement is small-enough. For example, the magnitude of rotation $\|R\| \leq 5°$.
2. Each pixel is square.

We model the motion of catadioptric omni-directional cameras as a sinusoidal curve. The overview can be summarized as follows:

$$y = A\sin(x + \phi) + B \tag{3.1}$$

where A, B, x and ϕ denote the amplitude, offset, phase and phase shift of the sinusoidal curve, respectively. The sinusoidal curve has a fixed frequency of 1. In this work, the phase x is given by the panorama image column index u_p and thus needs to be multiplied with the calibration parameter γ. Though the camera motion results in both column-wise and row-wise regular pixel movement, the variables of the sinosoid functions are different for column-wise and row-wise cases. The **core functions of this work** are thus:

$$\Delta v(u_p) = \gamma\|\mathrm{R_{xy}}\| \cdot \sin\left(\gamma u_p - \hat{\mathrm{R}}_{\mathrm{xy}}\right) + \lambda_i \mathrm{t}_z \tag{3.2a}$$

$$\Delta\theta(u_p) = \|\mathrm{R_{xy}}\| \cdot \sin\left(\gamma u_p - \hat{\mathrm{R}}_{\mathrm{xy}} + \frac{\pi}{2}\right) \tag{3.2b}$$

$$\Delta u(u_p) = \lambda_i\|\mathrm{t_{xy}}\| \cdot \sin\left(\gamma u_p + \hat{\mathrm{t}}_{xy}\right) + \gamma \mathrm{R_z} \tag{3.2c}$$

Note that these three sinusoid functions (Eq. 3.2a, Eq. 3.2b and Eq. 3.2c) include the 6-DoF parameters of camera transformation. Eq. 3.2b is similar to Eq. 3.2a except of phase shift $\frac{\pi}{2}$, the unit factor γ and the translation item $\lambda_i \mathrm{t}_z$, which means that the variables can be optimized together. Though the translation is also covered in these sinusoid functions, it is difficult to recover without known λ_i, i.e. parameters related to the depth of pixels. For rotation, the depth of the pixels is not important. Thus the translation item is only used as an auxiliary for the rotation estimation in this work. A

detailed discussion about this can be found in Sec. 3.3.2.

Intuitive Analysis

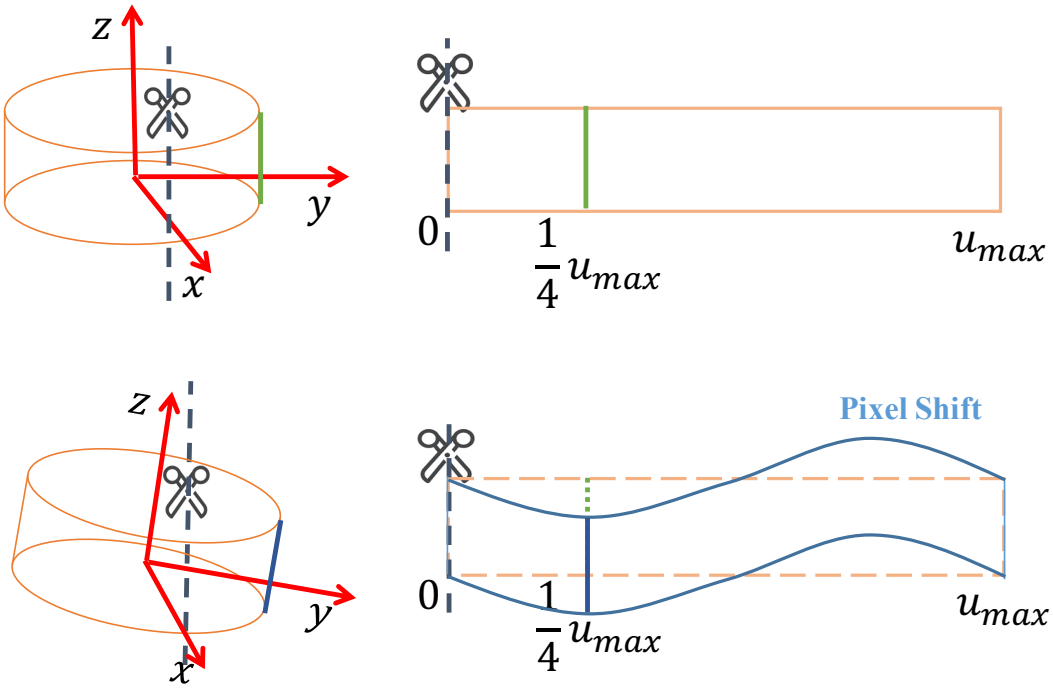


Figure 3.1: Intuitive demonstration of a rotation around the x-axis (roll)

Firstly, we analyze interpretively the motion patterns in four different cases to show how pixel motions actually follow the sinusoidal functions:

- **Rotation around *x*- or *y*-axis** (R_{xy}): (*small roll as example, see Fig. 3.1*) The closer column index u_p of the panorama image is to the (positive or negative) x-axis, the smaller the shift along the v-axis. The closer u_p is to the (positive or negative) y-axis, the larger the shift. See Eq. 3.2a.

  For the roll (rotation around the x-axis), the image rotation $\Delta\theta(u_p)$ from Eq. 3.2b is maximum (namely exactly the roll angle) at $u_p = 0$ and $u_p = \frac{1}{2}u_{max}$, because those correspond to the (positive or negative) x-axis of the camera. The closer u_p is to the (positive or negative) y-axis, the smaller the rotation.

- **Translation along *z*-axis** (t_z): In Eq. 3.2a the pixel shift is the one along v-axis of each column index u_p, which is the same for all u_p, so it is an offset.

- **Rotation around *z*-axis** (R_z): In Eq. 3.2c this yaw is the shift along u-axis in each column u_p, which is the same for all u_p, so it is an offset.

- **Translation along *x*- and *y*-axis (t_{xy})**: (*using translation along x-axis as example*) Similar to the rotation case, the closer column index u_p to the (positive or negative) x-axis is, the smaller shift along u-axis; the closer to the (positive or negative) y-axis, the larger shift is. See Eq. 3.2c.

In the following, we discuss the mathematical explanation and present the detailed derivation of the sinusoid model for camera rotation. Suppose two images ${}^1I_p$ and ${}^2I_p$ and a transform ${}^1_2T = [{}^1_2R, {}^1_2t]$ between them. Assume there is an arbitrary point ${}^2p = (u, v)^T$ in the second panorama image ${}^2I_p$ and its cylinder coordinate ${}^2P$ is

$$ {}^2P = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} r\cos\frac{u}{r} \\ r\sin\frac{u}{r} \\ \frac{H}{2} - v \end{bmatrix}. \tag{3.3} $$

Then we analyze the shifts of each row or column when the camera moves like the above two cases. Firstly, the 3D point ${}^2P$ is transformed to ${}^1P = [x_1, y_1, z_1]^T$ with specified transformation matrix ${}^1_2T$; secondly, we find the intersection ${}^1\bar{P} = [\bar{x}_1, \bar{y}_1, \bar{z}_1]^T$ between the line segment ${}^1PO$ and the cylinder $\{C : x^2 + y^2 = r^2\}$, which is then unwrapped into the point ${}^1p$ in the panorama image ${}^1I_p$; finally, we calculated the shift between ${}^1p$ and ${}^2p$ row-wise and column-wise, respectively.

Mathematical Analysis

Rotation around *x*-axis The transformation matrix T is:

$$ {}^1_2T = \left[\begin{array}{ccc:c} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x & 0 \\ 0 & \sin\theta_x & \cos\theta_x & 0 \end{array}\right], \tag{3.4a} $$

the transformed point ${}^1P$ is

$$ {}^1P = \begin{bmatrix} r\cos\frac{u}{r} \\ r\sin\frac{u}{r}\cos\theta_x - (\frac{H}{2} - v)\sin\theta_x \\ r\sin\frac{u}{r}\sin\theta_x + (\frac{H}{2} - v)\cos\theta_x \end{bmatrix}, \tag{3.4b} $$

the intersection ${}^1\bar{P}$ is

$$ {}^1\bar{P} = \begin{bmatrix} \frac{r}{\sqrt{1+k^2}} \\ \frac{kr}{\sqrt{1+k^2}} \\ \frac{r\sin\frac{u}{r}\sin\theta_x + (\frac{H}{2} - v)\cos\theta_x}{r\cos\frac{u}{r}}\frac{r}{\sqrt{1+k^2}} \end{bmatrix}, \tag{3.4c} $$

where $k = \frac{r \sin\frac{u}{r} \cos\theta_x - (\frac{H}{2} - v)\sin\theta_x}{r\cos\frac{u}{r}}$; finally we get the shift Δv in column direction:

$$\begin{aligned}
\Delta v =& (\frac{H}{2} - z_2) - (\frac{H}{2} - z_1') \\
=& \bar{z}_1 - z_2 \\
=& \frac{r(r\sin\frac{u}{r}\sin\theta_x + (\frac{H}{2} - v)\cos\theta_x)}{\sqrt{(r\cos\frac{u}{r})^2 + (r\sin\frac{u}{r}\cos\theta_x - (\frac{H}{2} - v)\sin\theta_x)^2}} \\
& - (\frac{H}{2} - v) \\
\approx& \frac{r(r\sin\frac{u}{r}\theta_x + (\frac{H}{2} - v))}{\sqrt{r^2 + r\sin\frac{u}{r}(\frac{H}{2} - v)\sin 2\theta_x + (\frac{H}{2} - v)^2}} \\
& - (\frac{H}{2} - v) \\
=& \theta_x r \sin\frac{u}{r} \, .
\end{aligned} \tag{3.4d}$$

The approximately equal holds only if the small motion assumption holds, which leads

$$\sin\theta_x \approx \theta_x \quad \text{and} \quad \cos\theta_x \approx 1 \, .$$

The last equal holds because the middle of rows is chosen, which means $v_1 = \frac{H}{2}$. Similarly, the shift Δu is

$$\begin{aligned}
\Delta u &= r\arctan\frac{y_2}{x_2} - r\arctan\frac{\bar{y}_1}{\bar{x}_1} \\
&= r\arctan(\tan\frac{u}{r}) - r\arctan k \\
&= u - r\arctan\frac{r\sin\frac{u}{r}\cos\theta_x - (\frac{H}{2} - v)\sin\theta_x}{r\cos\frac{u}{r}} \\
&\approx u - r\arctan\frac{r\sin\frac{u}{r} - (\frac{H}{2} - v)\theta_x}{r\cos\frac{u}{r}} \\
&= 0 \, .
\end{aligned} \tag{3.4e}$$

The derivation holds under same condition with Δv.

Rotation around *z*-axis The transformation matrix T is:

$${}^1_2T = \left[\begin{array}{ccc:c} \cos\theta_z & -\sin\theta_z & 0 & 0 \\ \sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right] , \tag{3.5a}$$

the transformed point P^1 is

$$ {}^1P = \begin{bmatrix} r\cos(\frac{u}{r}+\theta_z) \\ r\sin(\frac{u}{r}+\theta_z) \\ \frac{H}{2}-v \end{bmatrix}, \tag{3.5b} $$

the intersection ${}^1\bar{P}$ remains same:

$$ {}^1\bar{P} = {}^1P = \begin{bmatrix} r\cos(\frac{u}{r}+\theta_z) \\ r\sin(\frac{u}{r}+\theta_z) \\ \frac{H}{2}-v \end{bmatrix}. \tag{3.5c} $$

Finally we get the shift Δu in row direction:

$$ \Delta u = r\arctan\frac{y_2}{x_2} - r\arctan\frac{\bar{y}_1}{\bar{x}_1} = -r\theta_z \,. \tag{3.5d} $$

Moreover, Δv equals to zero, due to no difference between ${}^2P$ and ${}^1\bar{P}$ in z coordinate.

Hybrid rotation For the hybrid rotation case, the transformation matrix is the combination of roll, pitch and yaw:

$$ {}^1_2T = \left[\begin{array}{ccc:c} czcy & czsysx-szcx & czsycx+szsx & 0 \\ szcy & szsysx+czcx & szsycx-czsx & 0 \\ -sy & cysx & cycx & 0 \end{array}\right], \tag{3.6a} $$

where $sx = \sin\theta_x$, $cx = \cos\theta_x$, $sy = \sin\theta_y$, $cy = \cos\theta_y$, $sz = \sin\theta_z$ and $cz = \cos\theta_z$ for simplification. After applying the transformation on point ${}^2P$. The resulting point ${}^1P$ would be

$$ {}^1P = \begin{bmatrix} r\cos(\frac{u}{r})czcy + r\sin(\frac{u}{r})(czsysx-szcx) + f_x \\ r\cos(\frac{u}{r})szcy + r\sin(\frac{u}{r})(szsysx-czcx) + f_y \\ -r\cos(\frac{u}{r})sy + r\sin(\frac{u}{r})cysx + (\frac{H}{2}-v)cycx \end{bmatrix}, \tag{3.6b} $$

where $f_x = (\frac{H}{2}-v)(czsycx+szsx)$ and $f_y = (\frac{H}{2}-v)(szsycx-czsx)$, which contain the $\frac{H}{2}-v$ item. Therefore, they would be zero at some point. For simplification, they are replaced by f_i. Then we project the point onto cylinder model to retrieve ${}^1\bar{P}$:

$$ {}^1\bar{P} = \begin{bmatrix} \frac{r}{\sqrt{1+k^2}} \\ \frac{kr}{\sqrt{1+k^2}} \\ \frac{-r\cos(\frac{u}{r})sy + r\sin(\frac{u}{r})cysx + (\frac{H}{2}-v)cycx}{r\cos(\frac{u}{r})czcy + r\sin(\frac{u}{r})(czsysx-szcx)+f_x}\frac{r}{\sqrt{1+k^2}} \end{bmatrix} \tag{3.6c} $$

where $k = \frac{r\cos(\frac{u}{r})szcy + r\sin(\frac{u}{r})(szsysx - czcx) + f_y}{r\cos(\frac{u}{r})czcy + r\sin(\frac{u}{r})(czsysx - szcx) + f_x}$. Then the shift can be obtained after applying the camera model and approximation based on reasonable assumptions. Concretely,

$$\begin{aligned}
\Delta u &= r\arctan\frac{y_2}{x_2} - r\arctan\frac{\bar{y}_1}{\bar{x}_1} \\
&= r\arctan\frac{\sin\frac{u}{r}}{\cos\frac{u}{r}} - r\arctan k \\
&= r\arctan\frac{(\frac{\sin\frac{u}{r}}{\cos\frac{u}{r}}) - k}{1 + k\frac{\sin\frac{u}{r}}{\cos\frac{u}{r}}} \\
&\quad\vdots \\
&\approx r\arctan\frac{r\sin^2\frac{u}{r}\theta_y\theta_x - r\theta_z - \frac{r}{2}\sin\frac{2u}{r}\theta_x\theta_y\theta_z}{r + \frac{r}{2}\sin\frac{2u}{r}\theta_x\theta_y + r\sin^2\frac{u}{r}\theta_z\theta_y\theta_x} \\
&\approx r\arctan(-\theta_z) \\
&\approx -r\theta_z\ ;
\end{aligned} \tag{3.6d}$$

$$\begin{aligned}
\Delta v &= (\frac{H}{2} - z_2) - (\frac{H}{2} - \bar{z}_1) \\
&= \bar{z}_1 - z_2 \\
&= \frac{-r\cos(\frac{u}{r})sy + r\sin(\frac{u}{r})cysx}{r\cos(\frac{u}{r})czcy + r\sin(\frac{u}{r})(czsysx - szcx)}\frac{r}{\sqrt{1+k^2}} \\
&\quad\vdots \\
&\approx \frac{-r\cos\frac{u}{r}\theta_y + r\sin\frac{u}{r}\theta_x}{\sqrt{(1+\theta_z^2)(1+\sin^2\frac{u}{r}\theta_y^2\theta_x^2 + 2\cos\frac{u}{r}\sin\frac{u}{r}\theta_y\theta_x)}} \\
&\approx -r\cos\frac{u}{r}\theta_y + r\sin\frac{u}{r}\theta_x \\
&= \sqrt{\theta_y^2 + \theta_x^2}r\sin(\frac{u}{r} - \arctan\frac{\theta_y}{\theta_x})\ .
\end{aligned} \tag{3.6e}$$

We omitted several simple multiplication and approximation steps to save space.

Image rotation For image rotation $\Delta\theta$ (Eq. 3.2b), considering only one pixel in the image is not enough for derivation. Instead, we take the direction vector of every column line as a reference. In order to extract the image rotation from the camera movement, we project the rotated direction vector into the tangent plane of the reference column index.

Assuming the camera is standing upright, the direction vector of each column is the

same as $\mathbf{r_2} = (\mathbf{0}, \mathbf{0}, \mathbf{1})^{\mathbf{T}}$ in 3D space. The image rotation is independent of the yaw and translational movement. Thus we only take the roll and pitch into consideration during derivation for simplification. Thus the transformation matrix is:

$$ {}^{1}_{2}R = \begin{bmatrix} cy & sysx & sycx \\ 0 & cx & -sx \\ -sy & cysx & cycx \end{bmatrix} . \tag{3.7} $$

Then the rotated direction vector r_1 is

$$ \begin{aligned} \mathbf{r_1} = {}^{\mathbf{1}}_{\mathbf{2}}\mathbf{R}\mathbf{r_2} &= \begin{bmatrix} cy & sysx & sycx \\ 0 & cx & -sx \\ -sy & cysx & cycx \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \sin\theta_y \cos\theta_x \\ -\sin\theta_x \\ \cos\theta_y \cos\theta_x \end{bmatrix} \end{aligned} \tag{3.8} $$

For the following derivation, we first recall the property of projections of lines on planes. Let $\mathbf{A}$ be any line and $\mathbf{B}$ be the normal vector of a plane in 3D space. The projection of line $\mathbf{A}$ on plane $\mathbf{B}$ is

$$ \mathbf{A}\|\mathbf{B} = \mathbf{B} \times (\mathbf{A} \times \mathbf{B}) \tag{3.9} $$

or

$$ \mathbf{A}\|\mathbf{B} = (\mathbf{B} \times \mathbf{A}) \times \mathbf{B} \, . \tag{3.10} $$

The symbol "$\times$" above denotes the cross product of the two vectors. Then, for the column index u_p of the image, the normal vector of the tangent plane with respect to the current column is $\mathbf{n} = (\cos(\gamma \mathbf{u_p}), \sin(\gamma \mathbf{u_p}), \mathbf{0})^{\mathbf{T}}$. We substitute the direction vector and the normal vector of the plane into Eq. 3.9 and get the projected direction vector on the tangent plane as

$$ \begin{aligned} \hat{\mathbf{r}}_\mathbf{1} &= \mathbf{n} \times (\mathbf{r_1} \times \mathbf{n}) \\ &= \begin{bmatrix} \sin\theta_x \cos\gamma u_p \sin\gamma u_p + \sin\theta_y \cos\theta_x \sin^2\gamma u_p \\ -\sin\theta_x \cos^2\gamma u_p - \sin\theta_y \cos\theta_x \cos\gamma u_p \sin\gamma u_p \\ \cos\theta_y \cos\theta_x \end{bmatrix} \end{aligned} \tag{3.11} $$

Therefore, the angle of image rotation for a column index can be extracted as follows.

$$\begin{aligned}\cos\Delta\theta &= \frac{\mathbf{r_2}\cdot\hat{\mathbf{r}}_\mathbf{1}}{\|\mathbf{r_2}\|_\mathbf{2}\|\hat{\mathbf{r}}_\mathbf{1}\|_\mathbf{2}} \\ &\approx \frac{1}{\sqrt{(\theta_x\cos\gamma u_p+\theta_y\sin\gamma u_p)^2+1}}\end{aligned} \tag{3.12}$$

The approximate equal holds for small camera rotations. Then, we can get following

$$\begin{aligned}\sin\Delta\theta &= \frac{\theta_x\cos\gamma u_p+\theta_y\sin\gamma u_p}{\sqrt{(\theta_x\cos\gamma u_p+\theta_y\sin\gamma u_p)^2+1}} \\ &= \frac{\sqrt{(\theta_x^2+\theta_y^2)}\sin(\gamma u_p+\arctan\frac{\theta_x}{\theta_y})}{\sqrt{(\sqrt{(\theta_x^2+\theta_y^2)}\sin(\gamma u_p+\arctan\frac{\theta_x}{\theta_y}))^2+1}} \\ &= \frac{\sqrt{(\theta_x^2+\theta_y^2)}\sin(\gamma u_p+\arctan\frac{\theta_x}{\theta_y})}{\sqrt{(\theta_x^2+\theta_y^2)}\sin(\gamma u_p+\arctan\frac{\theta_x}{\theta_y})+1} \\ &\approx \sqrt{(\theta_x^2+\theta_y^2)}\sin(\gamma u_p+\arctan\frac{\theta_x}{\theta_y})\end{aligned} \tag{3.13}$$

The last approximate equal holds when the 3D rotation is small enough. Finally we can obtain the result

$$\Delta\theta \approx \sqrt{(\theta_x^2+\theta_y^2)}\sin(\gamma u_p+\arctan\frac{\theta_x}{\theta_y})\ . \tag{3.14}$$

3.1.3 Motion Vector Extraction

From the derivation, the camera pose can be extracted from the shifts of pixels. In order to retrieve the camera pose, motion vector extraction and sinusoid curve fitting are essential processes. In this section, we will briefly introduce two methods, optical flow and FMT, to calculate pixel movements. Then the details about sinusoid curve fitting will be presented in Sec. 3.1.4.

Optical flow [14] is a classical method used for object tracking. Assume that the illumination is constant during a time interval, the pixel intensity should stay the same. Based on this, the consistency of pixel intensity between two frames can be exploited to calculate the pixel movements in $u-$ and $v-$ direction, i.e. Δu and Δv. Since the rotation $\Delta\theta$ between two frames cannot be extracted from optical flow directly, Eq. 3.2b is excluded in this work when using optical flow in the experiments.

The FMT algorithm [19] has been successfully applied in feature-deprived environments, like underwater. It first estimates rotation $\Delta\theta$ and scaling between two images

by transforming images to the frequency domain and then sampling the images of the frequency domain to log-polar coordinates. With the estimated rotation and scaling we can then re-rotate and re-scale the second image. Then phase correlation is used to find the translation $\Delta u, \Delta v$ between the two images. However, the FMT algorithm usually works in the 2D plane and it requires that the scenario should be planar. In [149], the omni-directional images are divided into several sub-images with a sliding window so that the sub-images can be considered as planar and FMT can be used to calculate transformation between two sub-images. To use FMT to calculate the motion vector $(\Delta u, \Delta v, \Delta\theta)$, a similar sub-image strategy is used in this work.

It should be noted that though FMT can also estimate scaling between images, scaling will not be used in this work. In our future work, we plan to use it for translation estimation.

3.1.4 Fitting Algorithm

For curve fitting, there are some unknown parameters $\Phi = \{A, \phi, B\}$ in Eq. 3.1 to estimate. The easiest way to estimate yaw is to average the Δu in Eq. 3.2c even if the translation in $x-y$ plane exists. However, we cannot use a similar approach to estimate roll and pitch, that is R_{xy} in Eq. 3.2a because the $z-$ axis translation would negatively affect the amplitude, i.e. $\|R_{xy}\|$. That is to say, the offset term ($z-$ axis translation) of Eq. 3.2a is necessary for roll and pitch estimation, which will be discussed in Sec. 3.3.2. Although yaw can be estimated via average, we still use the optimization methods to estimate the unknown parameters Φ by modeling them as nonlinear least-squared problems. Then the rotation R and translation t can be calculated from Φ.

In order to find the corresponding parameters $\Phi_v = \{\|\mathrm{R_{xy}}\|, \hat{\mathrm{R}}_{\mathrm{xy}}, \mathrm{t}_z\}$, $\Phi_u = \{\|\mathrm{t_{xy}}\|, \hat{\mathrm{t}}_{xy}, \mathrm{R_z}\}$ in Eq. 3.2a, Eq. 3.2b and Eq. 3.2c, we build the following two objective functions:

$$r_v(u_p, \Phi_v) = \Delta v(u_p; \Phi_v) - y_v \tag{3.15a}$$

$$r_\theta(u_p, \Phi_v) = \Delta\theta(u_p; \Phi_v) - y_\theta \tag{3.15b}$$

$$r_u(u_p, \Phi_u) = \Delta u(u_p; \Phi_u) - y_u \tag{3.15c}$$

$$\min_{\Phi_v} L_v(u_p; \Phi_v) = \min_{\Phi_v} L_v^1(u_p; \Phi_v) + \eta L_v^2(u_p; \Phi_v) \tag{3.15d}$$

$$= \min_{\Phi_v} \frac{1}{2}(\|r_v(u_p, \Phi_v)\|_2^2 + \eta \|r_\theta(u_p, \Phi_v)\|_2^2) \tag{3.15e}$$

$$\min_{\Phi_u} L_u(u_p; \Phi_u) = \min_{\Phi_u} \frac{1}{2} \|r_u(u_p, \Phi_u)\|_2^2 \tag{3.15f}$$

where η is the weight coefficient for optimization; r_v, r_θ and r_u are the residuals of the

shifts; y_θ is the image rotation, y_v and y_u are the measured shifts in column and row direction extracted by either the FMT algorithm [19] or optical flow. L_v and L_u are loss functions in the standard least-squared form. Afterwards, the nonlinear optimization Levenberg-Marquardt algorithm [87] is used to minimize the objective functions, which could also be replaced with other optimization methods.

$$L_v^1(u_p;\Phi_v,\delta) = \delta \cdot \left(\sqrt{1+\left(\frac{r_v(u_p,\Phi_v)}{\delta}\right)^2} - 1\right) \tag{3.16a}$$

$$L_v^2(u_p;\Phi_v,\delta) = \delta \cdot \left(\sqrt{1+\left(\frac{r_\theta(u_p,\Phi_v)}{\delta}\right)^2} - 1\right) \tag{3.16b}$$

$$L_u(u_p;\Phi_u,\delta) = \delta \cdot \left(\sqrt{1+\left(\frac{r_u(u_p,\Phi_u)}{\delta}\right)^2} - 1\right) \tag{3.16c}$$

We choose a robust loss function to handle the outliers problem. In this work, the Huber loss function is used to reduce the influence of outliers, which was proven to be less sensitive to outliers in data than the L2 loss [65]. The Pseudo-Huber loss function [30] combines the best properties of L2 loss and Huber loss, which is strongly convex when close to the minimum and less steep for outliers. Thus we use it to replace the standard least-squared loss form from (Eq. 3.15d, Eq. 3.15f) with Eq. 3.16a, Eq. 3.16b and Eq. 3.16c.

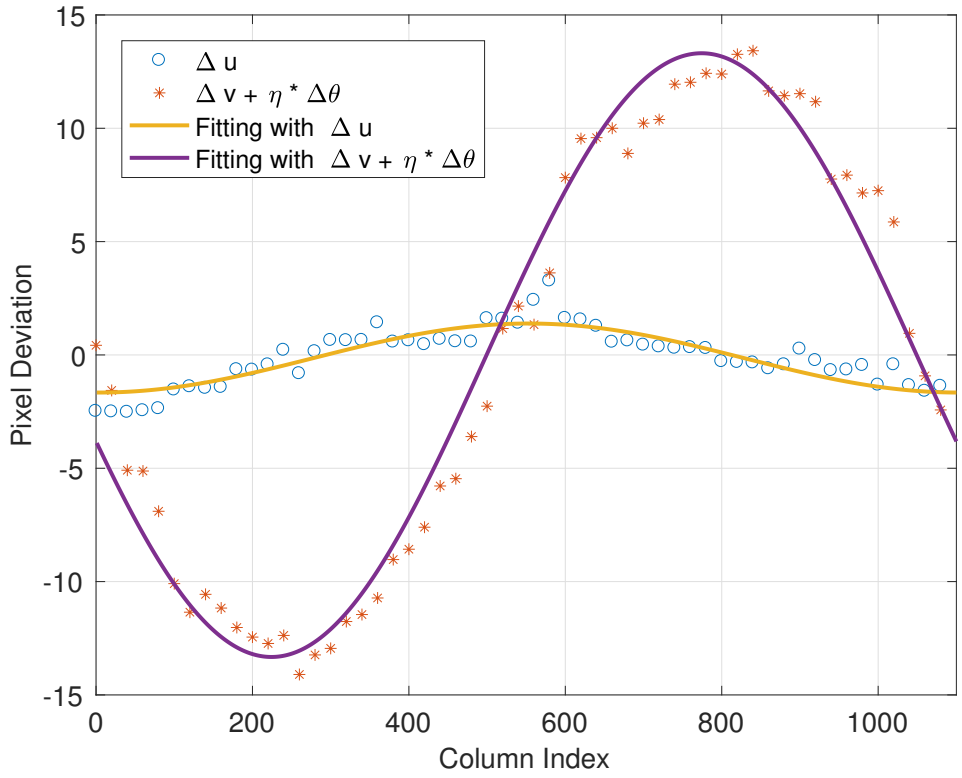
3.2 Implementation

The implementation of our method is described in Algorithm 2, where W, L, H and d depend on the different datasets.

A square window is used to slide along the $u-$direction on the panorama images ${}^1I_p$ and ${}^2I_p$. For each window pair, we use the FMT method to find the 2D motion. Then we get the set of shifts Δu, Δv and $\Delta\theta$ versus column index $u_p = \frac{1}{2}L + k \times d$. Similarly, optical flow can also be used to find the 2D motion, but only with shifts Δu and Δv. Afterwards, we fit the values with the sinusoidal function to estimate parameter Φ, as line 7 of Algorithm 2 describes. $\delta = 0.5$ of Eq 3.16 is selected in our implementation. Fig. 3.2c displays an example of the curve fitting between two frames. The optimization algorithm fits the sinusoids effectively. In Fig. 3.2c we see maxima in the $\Delta v + \eta\Delta\theta$ curve at around $\frac{1}{4}u_{max}$ and $\frac{3}{4}u_{max}$. Following the definition from above, that the x-axis is at $u = 0$, this means there is a big motion in the image where the y-axis is. It thus follows that there has been a rotation around the x-axis of the camera (roll).

(a) Frame 1

(b) Frame 2



(c) Fitting Results

Figure 3.2: Δu and Δv is shift of u and v direction, respectively, which are measured using FMT. The results of function fitting by non-linear least squared method through the Δu and Δv values are shown. An example motion of roll here (column 0 is the $x-$axis)

Algorithm 2 Proposed rotation estimation for omni-cameras

Input: Omni images ${}^1I_o$, ${}^2I_o$;
Sliding window size $L \times L$ and step d
Obtain panorama images ${}^1I_p$, ${}^2I_p$ of size $W \times H$
by cartesian-to-polar transformation
while $L + k \times d \leq W, k \in \mathbb{N}$ **do**
    Compute the scaling, rotation and translation t_{uv}
for k^{th} window between I_p^1 and I_p^2
by FMT or optical flow
    Push t_{uv} to motion set $\mathbb{M}$
end while
Estimate parameters Φ_v and Φ_u by sinusoid fitting
with Levenberg-Marquardt on $\mathbb{M}$ (Eq. 3.15)
Calculate transformation T from Φ_v, Φ_u (Eq. 3.1)
Output: T

3.3 Experiments

To evaluate the performance of the proposed method, we perform different experiments to analyze the robustness, speed and accuracy. Firstly, we present the advantage of additionally using the rotation $\Delta\theta$ of the panorama sub-images, estimated by FMT, for joint optimization. Secondly, we show that the proposed methods are robust to the translation disturbance. Moreover, we also use two types of geometry-based methods in comparison, i.e. the STEWENIUS five-point algorithm and the n-point approach, where the latter is used for pure rotation [72]. Both geometry-based methods are implemented using the OpenGV[I] Library. Thirdly, the proposed sinusoid fitting method is compared with the STEWENIUS five-point and n-point algorithms with different features and the direct methods w.r.t. accuracy and speed. In addition, this comparison is performed on different datasets, to evaluate the robustness of each method.

The datasets used in the experiments range from indoor to outdoor scenarios, as well as another dataset with ambiguous features. In addition to the public datasets *OVMIS* [93] and *CVLIBS*[124, 125], we also capture images by ourselves using a phone (Oneplus 5) covered with a low-cost omni-lens (Kogeto Dot Lens), to increase the abundance of the camera motion. We collect images in similar scenarios to the public datasets, because these public datasets are usually captured with high-end sensors, so that the images are with high resolution, whereas our images have a low resolution and poor quality (see Fig. 3.3), which corresponds to the proposed application scenario of a low cost robot in the future, as introduced in Fig. 2.4a.

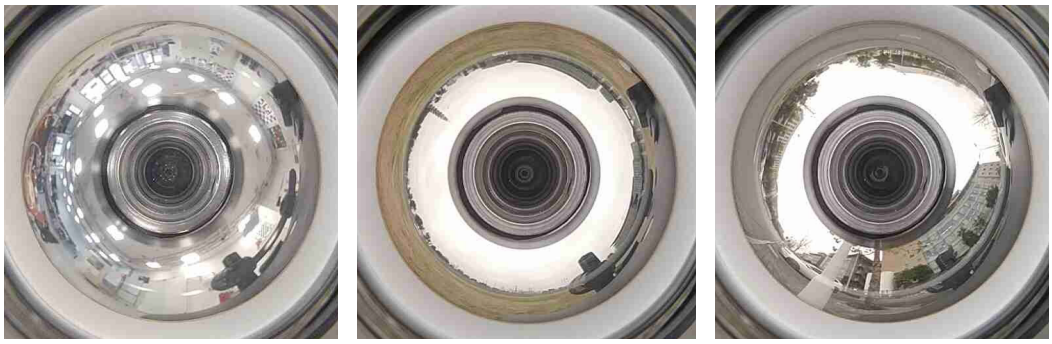
When collecting our datasets, the phone is fixed at a position and then rotated on a gimbal to get the sequential of images. The phone and its mount take a considerable

[1]https://github.com/laurentkneip/opengv

space in the panorama image. Nevertheless we will see, that our approach is very robust against this deceptive data.

Table 3.1 gives an overview of the datasets and in which experiments they are used. The ways to obtain ground truth are also shown in this table. We use the IMU of the phone as the ground truth source for the rotations in our own datasets. We actually measured the average error of the IMU with an OptiTrack tracking system and found that the error is smaller than $0.5°$, which is good enough for the following evaluations. Fig. 3.3 and Fig. 3.4 give examples from different image sequences from our datasets and the public ones, respectively.

In the following experiments except in Sec. 3.3.2, we use the absolute root mean square error (RMSE) of the roll, pitch and yaw as metric, where the error is calculated by difference between ground truth values and the estimated rotations for each frame with enough disparity. That is to say, the euler angles of ground truth and estimations will be transformed to rotation matrix and then we calculate relative transformation in the type of rotation matrix. Afterwards, the relative rotation matrix is converted to the axis-angle type. The magnitude of the angle is the error used to estimate the performance. For a visual odometry system, the real-time accuracy of localization is more interesting than the error between frames, because an instant big error may break the system. Thus the absolute RMSE is used as the metric instead of relative RMSE in this work. In Sec. 3.3.2, we calculated the relative error between several pairs of frames. Since the reference frame is fixed, there is no difference between absolute and relative errors in this case. More details will be explained there.



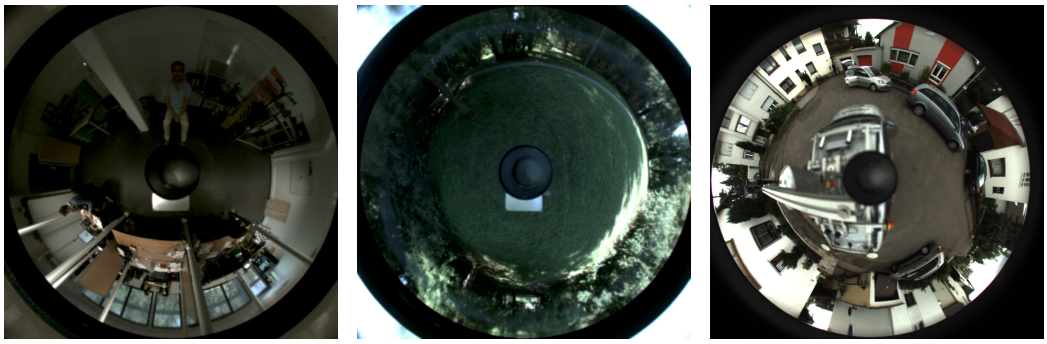
(a) Indoor (b) Grass (c) Street

Figure 3.3: Image examples captured by Oneplus 5

Additionally, different approaches are compared in the experiments, which are classified into two types: the proposed sinusoid fitting method and the geometry approach. The former is combined with optical flow and FMT under different settings, which will be introduced in detail in the following experiments. Moreover, the latter, including

Table 3.1: Datasets Overview

Scenarios	Motion	Name	#images	Sections	Ground truth
		Ours			
Indoor	roll	indoor_single_roll	155	5.3 Evaluation of Single Rotation	IMU of the phone
Grass	roll	grass_single_roll	158	5.3 Evaluation of Single Rotation	IMU of the phone
Street	roll	street_single_roll	160	5.3 Evaluation of Single Rotation	IMU of the phone
Indoor	pitch	indoor_single_pitch	116	5.3 Evaluation of Single Rotation	IMU of the phone
Grass	pitch	grass_single_pitch	178	5.3 Evaluation of Single Rotation	IMU of the phone
Street	pitch	street_single_pitch	174	5.3 Evaluation of Single Rotation	IMU of the phone
Indoor	yaw	indoor_single_yaw	200	5.3 Evaluation of Single Rotation	IMU of the phone
Grass	yaw	grass_single_yaw	200	5.3 Evaluation of Single Rotation	IMU of the phone
Street	yaw	street_single_yaw	200	5.3 Evaluation of Single Rotation	IMU of the phone
Indoor	roll & pitch & yaw	indoor_rpy	200	5.1 Rotation as Joint Optimization; 5.4 Comparison on Different Datasets	IMU of the phone
Grass	roll & pitch & yaw	grass_rpy	183	5.1 Rotation as Joint Optimization; 5.4 Comparison on Different Datasets	IMU of the phone
Street	roll & pitch & yaw	street_rpy	191	5.1 Rotation as Joint Optimization; 5.4 Comparison on Different Datasets	IMU of the phone
Indoor	roll & pitch & yaw & z	office_zrpy	11	5.2 Fitting With Translation	IMU of the phone
		OVMIS[93]			
Indoor	roll & pitch & yaw	OVMIS_1	160	5.4 Comparison on Different Datasets	Robotic Platform (ARIA Library)
Grass	yaw	OVMIS_2	156	5.4 Comparison on Different Datasets	Robotic Platform (ARIA Library)
Grass	yaw & x & y	OVMIS_3	200	5.2 Fitting With Translation	Robotic Platform (ARIA Library)
		CVLIBS[**schonbein2014omnidirectional**]			
Street	yaw & x & y	CVLIBS	200	5.4 Comparison on Different Datasets	IMU/GPS



(a) OVMIS_1 (b) OVMIS_2 (c) CVLIBS

Figure 3.4: Image examples from public datasets

$n-$point and five-point, are implemented with ORB, AKAZE, optical flow and FMT. ORB and AKAZE are chosen as the representatives of feature-based methods to find matches, because ORB is one of the fastest detectors and AKAZE is designed for detecting features in non-linear space according to [141]. Moreover, optical flow and FMT are two kinds of direct methods, especially FMT also stands for the frequency-based methods. The preliminary results of the five-point algorithm with FMT can be found in chapter 2.

We run the sinusoid fitting approaches ("ours") and the FMT geometry approaches on the extracted panorama images, because we require flat images for FMT and Δu, Δv and $\Delta\theta$ parameters from the panorama images for sinusoid fitting. The other approaches (geometry approaches except when FMT is used) use the raw omni-images to avoid additional errors from image transformation.

All the experimental computations are conducted on a PC with an Intel Core i7-4790 CPU and 16 GB memory with single-threaded C++ programs.

3.3.1 Evaluation of using Rotation as Joint Optimization

In this work, 3-DoF motion vectors (Δu, Δv and $\Delta\theta$) are exploited for the input of sinusoidal fitting if we use FMT for the motion vectors calculation. 2-DoF motion vectors (Δu and Δv) are utilized if they are estimated by optical flow. As analyzed in Sec. 3.1.4, the rotation of each sub-region versus its column index also follows the sinusoid curve, similar to the pixel deviation. Thus we can jointly optimize these two items. We set η, the weight coefficient for $\Delta\theta$ of the joint optimization from Eq. 3.15, to 0.1, based on a small experiment regarding the precision of the estimation of the image rotation $\Delta\theta$. This value is also used in all further experiments that use $\Delta\theta$.

Though we exploit different methods to find correspondences, FMT can also provide the rotation and scaling information between images in addition to pixel shifts in $u-$

and $v-$ direction. Note that the rotation between images and that between camera poses are different, we use $\Delta\theta$ and Θ as symbols to distinguish these two. When the camera orientation Θ changes, $\Delta\theta$ of each sub-image pair follows a sinusoid curve, which we thus include in the joint optimization.

Concretely, we conduct experiments with two different settings: with and without rotation $\Delta\theta$ for joint optimization in sinusoid fitting, that is whether to set $\eta \neq 0$ in Eq. 3.15d. Moreover, each setting is combined with and without translation, which will be discussed in Sec. 3.3.2. Thus we totally compare four different methods in this section, written as `ours_fmt_wotrans_wrot`, `ours_fmt_wotrans_worot`, `ours_fmt_wtrans_wrot`, `ours_fmt_wtrans_worot`. Optical flow is excluded in this experiment, since it cannot directly estimate image rotations.

Table 3.2: RMSE (rad) of with/without rotation for joint optimization

	indoor_rpy	grass_rpy	street_rpy
`ours_fmt_wotrans_wrot`	**0.212**	**0.314**	**0.499**
`ours_fmt_wotrans_worot`	0.214	0.336	0.522
`ours_fmt_wtrans_wrot`	**0.204**	**0.567**	**0.504**
`ours_fmt_wtrans_worot`	0.217	0.592	0.533

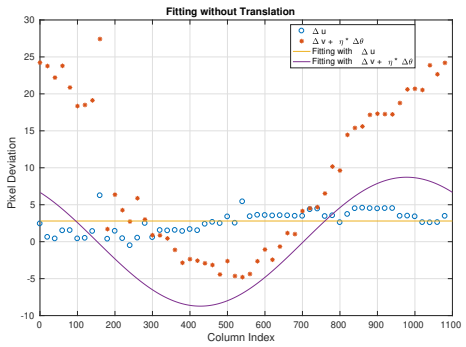
All the methods are applied to three different datasets: indoor_rpy, grass_rpy and street_rpy. The results are shown in Table 3.2, which indicate that using rotation for joint optimization improves the performance to some degree. Thus only the joint optimization methods, `ours_fmt_wtrans_wrot` and `ours_fmt_wotrans_wrot`, are used for comparison in the following sections.

3.3.2 Robustness Test with Translation

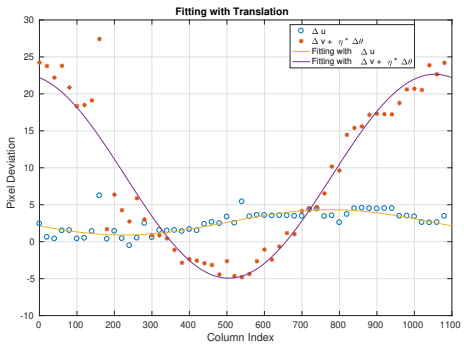
In Sec. 3.1.2, we formulated the rotation estimation model with the translation term. Here we show experiments that explore fitting with the translation term. Intuitively, there will be big errors on the fitting results if the translation λt_z is excluded when the translation in $z-$axis and roll/ pitch happens together (see Eq. 3.2a). The similar estimation issue happens for yaw in Eq. 3.2c. Thus, experiments with and without the translation term are compared here. We take two images with rotation and translation in z direction as examples, which are shown in Fig. 3.5a and 3.5b. Fig. 3.5c and Fig. 3.5d present the fitting results without and with the translation term, respectively. It can be seen that the purple curve cannot fit the shift $\Delta v + \eta\Delta\theta$ (red dots) well when fitting without the translation term in Fig. 3.5c. In contrast, the purple curve fits the $\Delta v + \eta\Delta\theta$ (red dots) much better when fitting with the translation term, as shown in Fig. 3.5d. Therefore, fitting with translation performs better than that without translation when

(a) Input Image 1

(b) Input Image 2



(c) Fitting without Translation



(d) Fitting with Translation

Figure 3.5: Sinusoid fitting with/without translation on the office_zrpy dataset

there are big translations with rotations.

Based on such a preliminary test, we perform a small experiment with 11 images (office_zrpy) to show the performance of $n-$point and five-point algorithms quantitatively when there are big translations and rotations. We use one image as the reference frame and register the rest images to the reference image. The camera motion between each rest image and the reference one includes the same z translation and different rotations. The proposed sinusoid fitting method based on FMT and optical flow are evaluated in this experiment together with the geometry-based methods. For the two geometry-based methods, $n-$point and five-point methods, the similar case exists that the $n-$point algorithm is announced as only working for the rotation estimation when there are no or small translations, whereas the five-point algorithm should work in the general cases.

The results are demonstrated in the box plot in Fig. 3.6, where the bottom and top edges of each blue box show the $25th$ and $75th$ percentiles, the central red lines represent the median, the minimum and maximum values are shown as short black lines and the outliers as red $+$. It can be found that the the five-point algorithm performs best with the camera translation, the proposed methods come second and the $n-$point works worst, as indicated by the median values (central red marks).

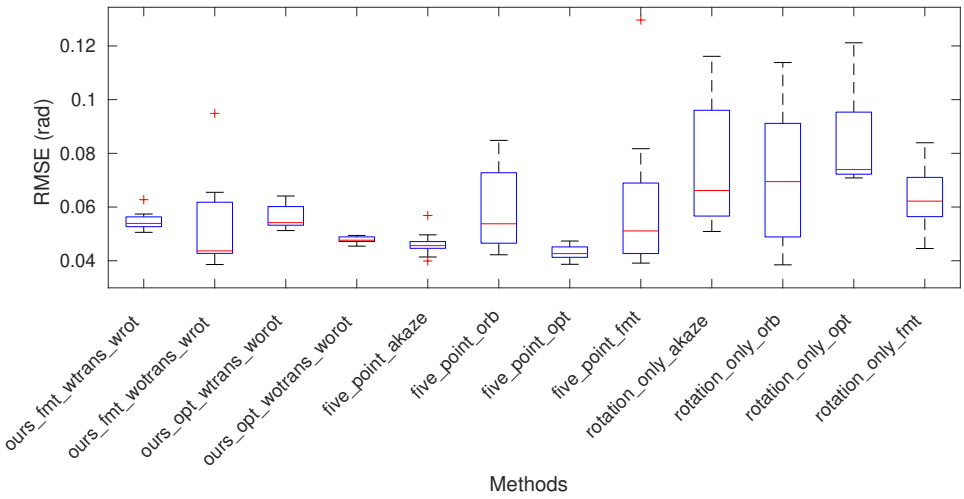


Figure 3.6: Performance of the geometry methods on rotation estimation testing on the office_zrpy dataset

One interesting result is that better fitting does not ensure better estimation results, which is indicated by the RMSE in Fig. 3.6. From the two settings of the proposed method based on optical flow, i.e. `ours_opt_wtrans_worot` and `ours_opt_wotrans_worot`, the setting "without translation" works better though its fitting performance is worse. The accompanying video shows all frames from all experiments with their fitting results. One reason may be that the rotation is small, so that there are chances that "without translation" perform better than "with translation". Moreover, since the assumption of our model is that the rotation should be less than $5°$, the experiments on large rotation will be more uncontrollable.

To sum up, compared to the other methods, the proposed sinusoid fitting with or without the translation term can both achieve good results. For the geometry methods, the five-point works better with respect to the translation. Since there are few translations in the following sections: Sec. 3.3.3 and Sec. 3.3.4, all the four methods will be evaluated.

3.3.3 Evaluation of Single Rotation

In this section, we evaluate the performance on the same twelve methods as above in Sec. 3.3.2, on single rotation, that is the camera is only rotated around one axis ($x-$, $y-$ and $z-$). With this we want to show the basic capabilities of the rotation estimation, before looking at more complex scenarios in the next section. The 12 methods are: $n-$point with AKAZE, ORB, optical flow and FMT (first four rows in Fig. 3.8), five-point with the same four features (middle four rows) and the proposed method with FMT

and optical flow (last four rows). These experiments are conducted on nine datasets collected by the phone with omni-directional lens including indoor, grass and street scenarios, as shown in Table 3.1.

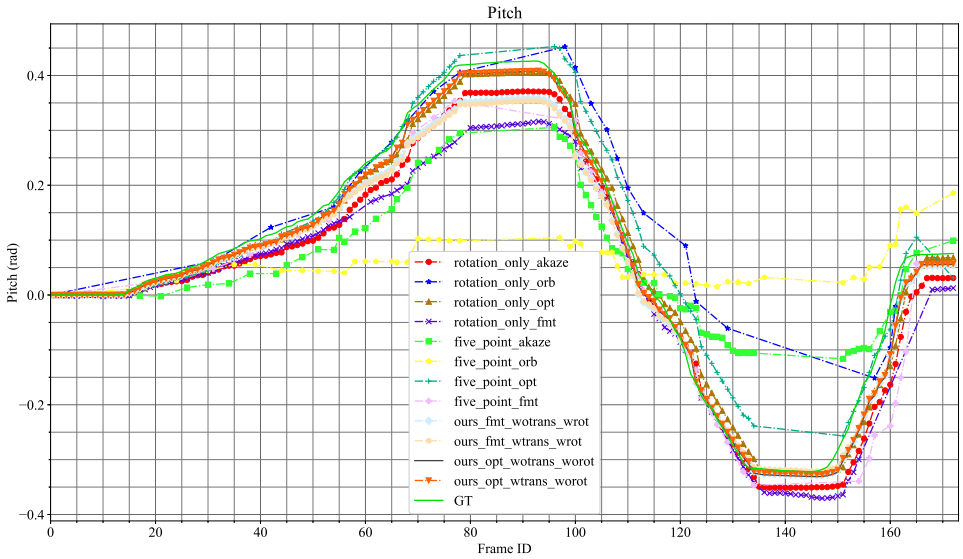


Figure 3.7: Example qualitative results for single rotation estimation on the street_single_pitch dataset

Fig. 3.7 shows an example of qualitative results on the street_single_pitch dataset. We can see that the rotation estimated by almost all methods is close to the ground truth (green solid). The results on the other datasets are shown in the Appendix.

Fig. 3.8 presents the quantitative results for each single rotation of our datasets. The intensity of the color represents the error: the darker the color is, the larger the error is. As described above, instead of relative RMSE, we use absolute RMSE here, since we treat the tracking process as a visual odometry system. Thus the RMSE values shown in Fig. 3.9 seem quite large. To avoid the confusion on these values, we give a simple worst case example here: if the estimated error for each frame is $+1^\circ$ and there are 100 frames in total, then the accumulated error will be 5050° and thus the average error is about 50°, i.e. $0.87rad$. With this example, we can have an intuitive feeling about the performance of each method based on the RMSE error. In addition, when evaluating the performance, we only need to focus on which method has a smaller RMSE, instead of the values themselves. In Sec. 3.3.4, the evaluation is the same.

We can see that almost every method achieves acceptable results in the street scenarios. In this scenario the image texture usually includes rich features, which is helpful for correspondences finding, such that both sinusoid fitting and geometry-based methods work well for rotation estimation. The indoor scenario also provides several features, so all methods achieve quite good results on pitch and roll. However, different

performances are presented when it comes to the lawn scenario, where the features are too similar to correctly match, especially for geometry methods. For instance, the error of the geometry method based on AKAZE is quite large on the grass_yaw dataset, as is the error for `rotation_only_orb`. We can see that `rotation_only_opt` achieves the most robust and accurate results of the rotation only approaches in the first four rows of Fig. 3.8. The `five_point_opt` is the best among the five-point algorithm. The proposed methods under different settings with FMT and optical flow have similar performance. Generally speaking, the sinusoid fitting performs better than the geometry method, as indicated by the light color distribution in Fig. 3.8.

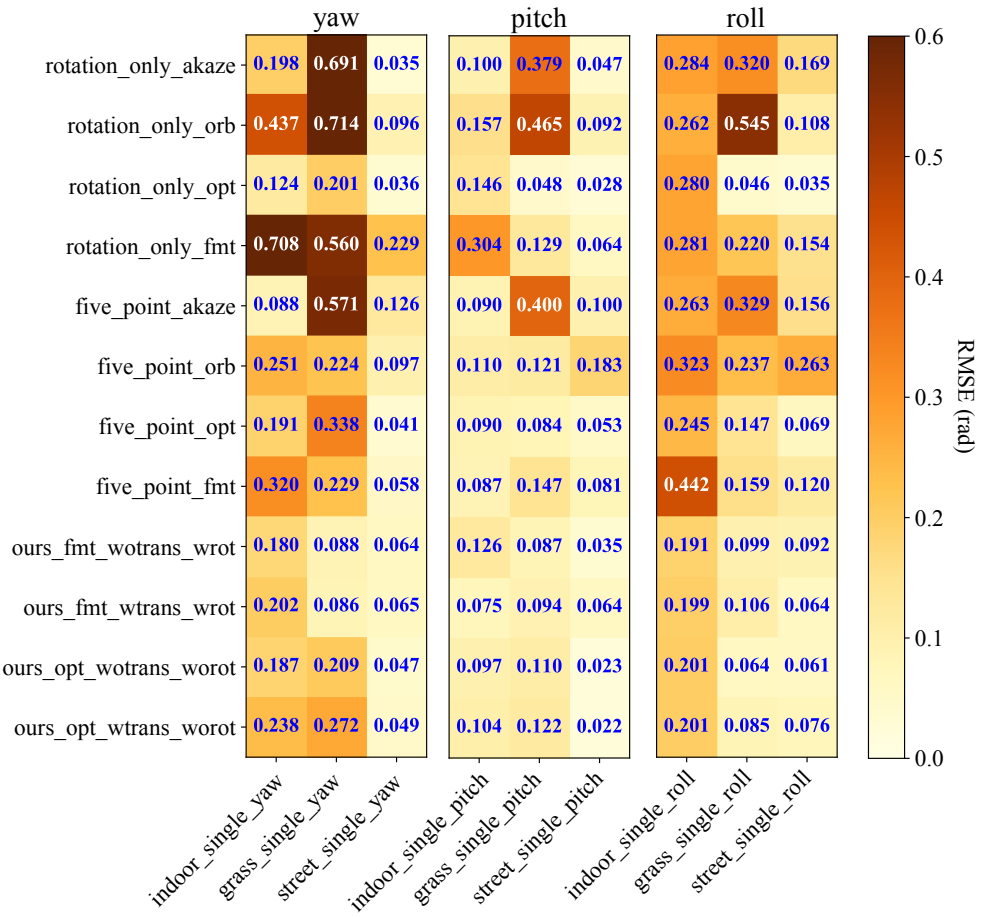


Figure 3.8: Quantitative results for single rotation estimation on different datasets

3.3.4 Comparison on Multiple Datasets

In this section we look at complex experiments, where the camera rotates around more than one axis. We use datasets collected by the phone including indoor_rpy, grass_rpy and street_rpy. Additionally we exploit two public datasets. *OVMIS*[93] provides indoor

(OVMIS_1) and lawn (OVMIS_2) scenarios, whereas the images of *CVLIBS* [124, 125] are captured on the street. So both ours and the public datasets cover indoor, grass and street scenarios.

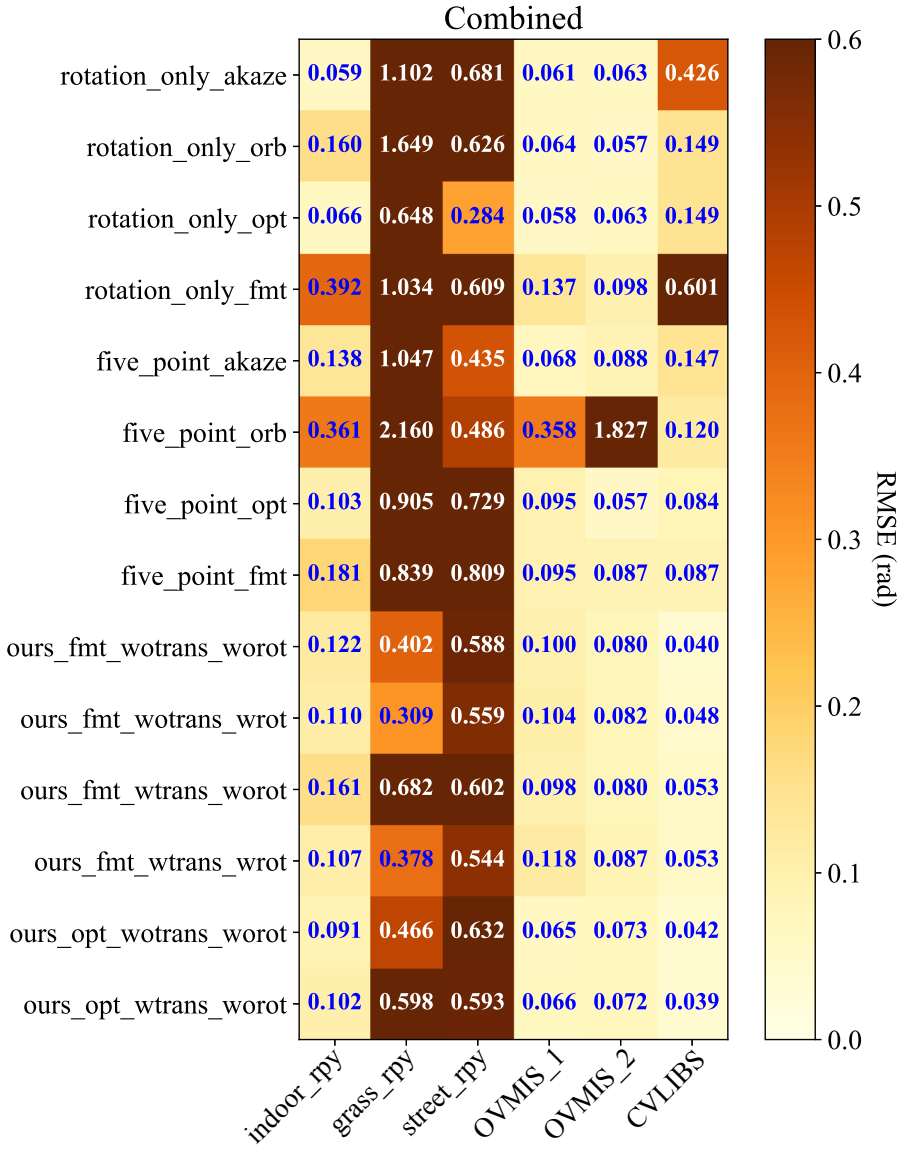


Figure 3.9: Different methods evaluation on multiple datasets

Fig. 3.9 demonstrates the rotation estimation performance on our (first three columns) and the public datasets (last three columns) and the qualitative results are shown in the Appendix. Again, darker colors represent larger RMSE. On our own datasets, it shows that all the approaches perform best in the indoor scenario. Their performance decreases on the grass and street dataset. From the point of different methods, we can only find that `five_point_orb` and `rotation_only_fmt` perform worst.

On the public datasets, the input images are high-quality omni-directional ones. All the methods achieve good results and the proposed sinusoid fitting methods (last six rows) are more robust. Moreover, each method performs better on the public datasets than on our own datasets. The reasons are twofold: 1) the cheap device that we used to capture data is a rolling shutter and with low resolution; 2) the cheap omni-lens is not well produced.

Furthermore, the proposed sinusoid fitting methods (last six rows) give the most robust and accurate performance considering the overall performance of all datasets. Though the sinusoid fitting is not always the best, it achieves good results on all the datasets except street_rpy under each setting. However, the geometry method depends on the features and fails to track the rotation from time to time, such as `five_point_orb` on the grass_rpy dataset. When both the sinusoid fitting and the geometry method are implemented with optical flow, `rotation_only_opt` gives the best performance, `ours_opt_wotrans_worot` comes next. Also, `ours_fmt_wotrans_wrot` is the best when both are with FMT.

3.3.5 Run-time analysis

To analyze which step requires high-cost run-time, we test the run-time of two steps of rotation estimation process: finding correspondences and rotation calculation. The first step is characterized by finding matches when it comes to AKAZE, ORB and optical flow and manifested as calculating motion vectors (including pixel deviation $\Delta u, \Delta v$ and rotation $\Delta\theta$) for FMT. The second step, estimation on camera orientation Θ are the $n-$point and five-point algorithms and sinusoid fitting, respectively.

Fig. 3.10 shows the time consumption of the two steps. Finding correspondences takes longer time than the second part for each method. Optical flow and ORB are much faster than FMT and AKAZE, which can be seen from the left part of Fig. 3.10. Under our settings, the number of correspondences for sinusoid fitting based on FMT is fixed whereas that for the geometry methods depends on the appearance richness of images except of five_point_fmt. The number of correspondences for sinusoid fitting based on optical flow is not fixed because we mixed good features and the center points for tracking, thus it also relates to the scenarios. Note that the optical flow in the proposed method is faster than that in the geometry approach because the latter track features on the raw image while the former tracks the features in the panorama image. Also, the latter can be improved with image masks. It can also be seen that the $n-$point is the fastest and then the sinusoid fitting comes next. The comparison between five_point_fmt and sinusoid fitting with FMT (first four rows) indicates that the sinusoid fitting method is faster than the five-point algorithm. Last but not least,

the sinusoid fitting methods based on optical flow (ours_opt_wotrans_worot and ours_opt_wtrans_worot) are the fastest algorithm among the experiments.

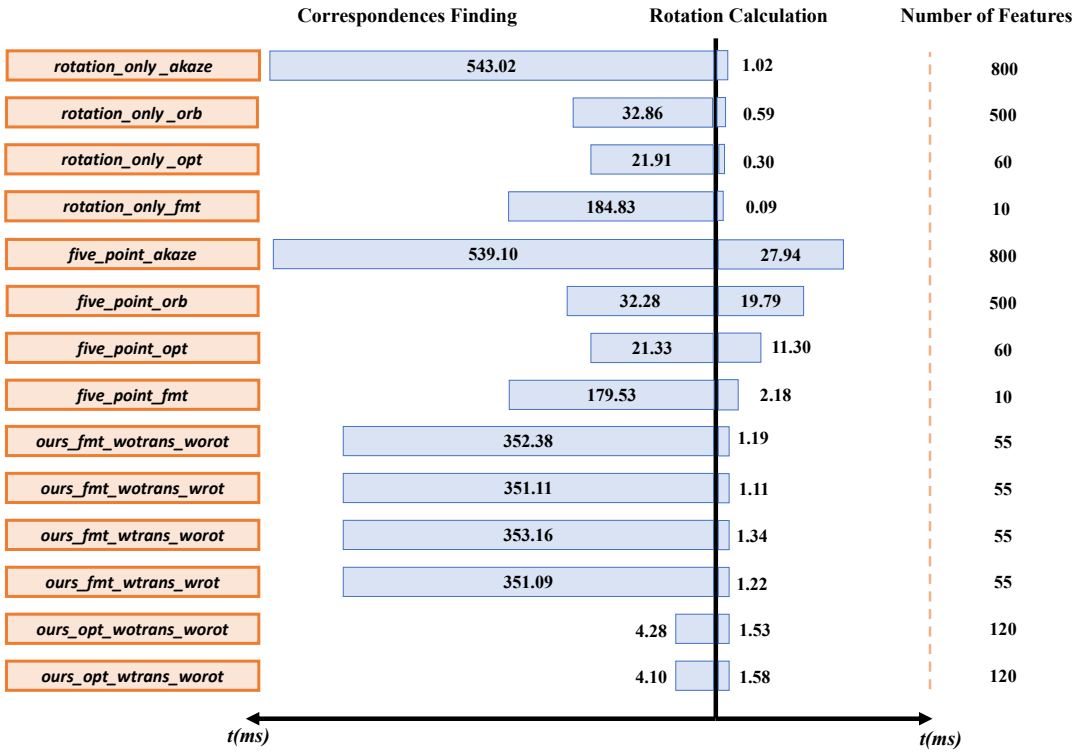


Figure 3.10: Run-time analysis per frame

3.4 Conclusions

In this chapter, we proposed a novel rotation estimation method based on sinusoid fitting of pixel displacements in panoramic images captured by omni-directional cameras. In addition to pixel shifts, we also incorporate the rotation of the sliding window in our model to get better and more robust results.

Our approach supports both FMT and optical flow to calculate the pixel deviation. Additionally, FMT also gives the rotation $\Delta\theta$ between two sub-images, which is exploited for joint optimization. These motion vectors are fitted to two sinusoidal curves to estimate the camera rotation.

In extensive experiments we compared the proposed sinusoid fitting methods and geometry approaches based on different methods. Two geometry methods are used in comparison, i.e. $n-$point for pure rotation and five-point based on epipolar geometry. Experiments on different datasets presented the robustness of the algorithm, which showed good results thanks to both correspondence finding and sinusoid fitting.

Finding correspondences removes unmatched features and the fitting algorithm can exclude outliers of wrong pixel deviation. In addition, these results also demonstrate the accuracy of the proposed method that the sinusoid fitting method is as good as the geometry-based ones. Moreover, it shows better performance when it comes to images with low resolution and difficult scenarios, like grass. Comparing to the five-point algorithm with FMT proposed in Chapter 2, the sinusoid fitting method is more robust and accurate for rotation estimation in our tests.

The run-time analysis of all approaches demonstrated that the finding of correspondences takes most of the run-time, with optical flow being the fastest one. Both sinusoid fitting and geometry-based methods are really fast for rotation calculation, the $n-$point algorithm is the fastest, the sinusoid fitting comes next and five-point is the last. Our sinusoid fitting approach with optical flow has the fastest run-time of all tested algorithms while also showing very good results regarding accuracy and robustness. FMT showed its strengths in the feature-deprived grass datasets.

The disadvantage of the proposed method is lack of translation estimation, so that the 6-DoF transformation estimation cannot be implemented. Also, as mentioned in Chapter 1, FMT requires single-depth scenarios, thus it cannot provide correction motion estimation for sub-images in multi-depth environments. This will also brings difficulties for the 6-DoF transformation estimation. For that, we rethink the FMT algorithm in multi-depth environments and propose an extension of FMT in Chapter 4. In our future work we plan to solve the translation of FMT with deep consideration and integrate our method into a full omni-directional visual odometry and SLAM framework.

4 Extending Fourier-Mellin Transform in Multi-depth Scenarios

FMT is used for motion estimation in Chapter 2 and 3, but faces the challenge that FMT cannot work properly in multi-depth environments. For that, Chapter 2 exploits the recursive sub-image strategy and Chapter 3 uses sinusoidal fitting to remove outliers. In this chapter, we propose the extended Fourier-Mellin transform (eFMT) for pinhole cameras to relax the constraints of equidistance and planar environments. If the depths of objects are different, the pixels' motion will be different even though the camera's motions is the same, which is due to the perspective projection. Since FMT can only give the image motion of the dominant depth, the camera's speed cannot be correctly inferred from the FMT's results when the dominant plane changes. Thus, an FMT-based visual odometry cannot work in multi-depth scenarios.

To overcome the drawback of FMT, this work presents the extended Fourier-Mellin Transform: eFMT. It extends FMT's 3D translation (translation and zoom), while keeping the original rotation estimation, because multiple depths result in multiple zooms and translations, which will be discussed in detail in Sec. 4.1. Since FMT has already been used in all kinds of applications, such as remote sensing, image registration, localization and mapping, 3D modeling, visual homing, etc. (see above), we see great potential for eFMT further enlarging the application scenarios of FMT. In this work, we proceed to a highly practically relevant application of our proposed eFMT odometry algorithm, which is the motion estimation with a down-looking camera.

As we will also show in this work, in contrast to FMT, feature based and direct visual odometry frameworks usually do not perform well in challenging environments, such as low-texture surfaces (e.g. lawn, asphalt), underwater and fog scenarios [25]. The main advantage of FMT over other approaches—its robustness—is preserved in eFMT. To maximize the robustness and accuracy of FMT, we use the implementation of the improved FMT in [20, 25] in the comparison and build eFMT upon it in this work.

Our main contributions are summarized as follows:

- To the best of our knowledge, we are the first to apply the zoom and translation estimation in FMT to multi-depth environments. Our method is more general than FMT but maintains its strengths;

- We implement an eFMT-based VO framework for one potential use-case of eFMT;

- We provide benchmarks in multi-depth environments between the proposed eFMT, the improved FMT [20, 25], and popular VO approaches. The state-of-the-art VO methods, ORB-SLAM3 [27], SVO [47] and DSO [42], are chosen as comparisons because they are the representative of feature-based, semi-direct and direct VO methods, respectively [62].

4.1 Problem Formulation

This section formulates the general image transformation with the 4-DoF camera motion in multi-depth scenarios.

Given a pixel $p = [x, y]^\top$ of image ${}^1I$, it is normalized to

$$\bar{p} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{f_x}(x - c_x) \\ \frac{1}{f_y}(y - c_y) \\ 1 \end{bmatrix}$$

with the focal length f_x, f_y and image center (c_x, c_y). Assume the pixel p corresponds to the 3D point P with depth δ, then the coordinate of P in the image ${}^1I$'s frame is

$$P = \begin{bmatrix} \frac{\delta}{f_x}(x - c_x) \\ \frac{\delta}{f_y}(y - c_y) \\ \delta \end{bmatrix}.$$

Suppose the transformation between the camera poses of ${}^1I$ and ${}^2I$ is a 4-DoF motion with the rotation around the camera principal axis, i.e., yaw θ, the 2D translation in the imaging plane $(\Delta x, \Delta y)$, and the translation perpendicular to the imaging plane $\Delta\delta$, then P in the ${}^1I$'s frame is projected to ${}^2I$ at point p':

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} P + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\delta \end{bmatrix},$$

that is

$$p' = \begin{bmatrix} \frac{\delta}{\delta+\Delta\delta}(x\cos\theta - y\sin\theta) \\ +\frac{1}{\delta+\Delta\delta}(-\delta c_x\cos\theta + \delta c_y\sin\theta + f_x\Delta x) + c_x \\ \\ \frac{\delta}{\delta+\Delta\delta}(x\sin\theta + y\cos\theta) \\ +\frac{1}{\delta+\Delta\delta}(-\delta c_x\sin\theta - \delta c_y\cos\theta + f_y\Delta y) + c_y \end{bmatrix} .$$

Thus we can derive a general equation

$$\begin{aligned} {}^2I(x,y) \quad &= {}^1I(z_\delta(x\cos\theta_0 - y\sin\theta_0) + x_\delta, \\ &\qquad z_\delta(x\sin\theta_0 + y\cos\theta_0) + y_\delta) \end{aligned} \tag{4.1}$$

to describe the pixel transformation between ${}^1I$ and ${}^2I$, where $\theta_0 = \theta$,

$$z_\delta = \frac{\delta}{\delta + \Delta\delta} , \tag{4.2}$$

$$x_\delta = \frac{1}{\delta+\Delta\delta}(-\delta c_x\cos\theta + \delta c_y\sin\theta + f_x\Delta x) + c_x \tag{4.3}$$

and

$$y_\delta = \frac{1}{\delta+\Delta\delta}(-\delta c_x\sin\theta - \delta c_y\cos\theta + f_y\Delta y) + c_y . \tag{4.4}$$

It can be found that a zoom z_δ and a translation (x_δ, y_δ) of a pixel depend on its depth δ, while rotation θ_0 is independent. Eq. (1.15) is a simplification of Eq. (4.1) under the condition that the depth δ of each pixel is the same. For I_1 and I_2 in a multi-depth scenario, there will be multiple solutions to Eq. (4.1)-(4.4), depending on the depth of the individual pixel, so there are multiple zooms and translations. The energy of the cells in the PSD is positively correlated to the number of pixels with depth δ for which (x_δ, y_δ) falls in that cell. Since FMT assumes an equidistant environment, the depth δ is considered constant for every pixel. That is, FMT supposes that the translation (x_δ, y_δ) and zoom z_δ is the same for all pixels p. Thus for FMT all (x_δ, y_δ) fall in a singe cell, forming a peak.

In this work, we propose eFMT that relaxes the equidistance constraint by solving (4.1) with different depths z_δ to estimate camera poses.

4.2 Methodology

In this section, we first solve (4.1) with the translation-only case and zoom-only case, respectively. Then we present how to handle the general case with a 4-DoF motion. Since the absolute magnitude of the monocular camera's poses cannot be found,

the re-scaling for translation and zoom is also discussed to estimate the up-to-scale transformation.

Without loss of generality, we use frame indices 1, 2 and 3 for any three consecutive frames in the following.

4.2.1 Translation-only case

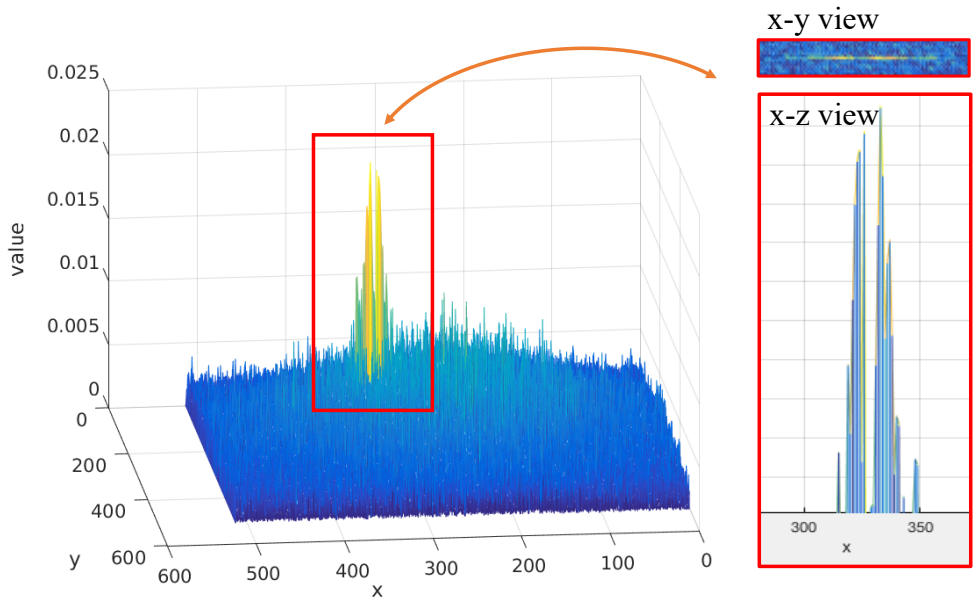


Figure 4.1: An example of a translation phase shift diagram in a multi-depth (more than two depths) environment.

FMT decouples the translation estimation from rotation and zoom calculation. Thus we only consider that the camera moves in the $x-y$ plane in the translation-only case. Then Eq. (4.1) is simplified to

$$^{2}I(x,y) = {}^{1}I(x+x_{\delta}, y+y_{\delta}) \quad . \tag{4.5}$$

As indicated by Eq. (4.3) and Eq. (4.4), translation (x_{δ}, y_{δ}) is not a single energy peak in the PSD like in Eq. (1.23), due to the multi-depth environment. Fig. 4.1 shows a translation PSD in a multi-depth environment. It can be seen that there are multiple peaks in the PSD and the $x-y$ view shows that these high peaks lie on one line. The collinear property is derived from the definition of x_{δ} and y_{δ}. In the translation-only

case, Eq. (4.3) and Eq. (4.4) are reduced to

$$x_\delta = \frac{f_x \Delta x}{\delta} \, , y_\delta = \frac{f_y \Delta y}{\delta} \, .$$

It can be found that the direction of each translation (x_δ, y_δ) is the same, i.e.:

$$(\frac{f_x \Delta x}{\sqrt{(f_x \Delta x)^2 + (f_y \Delta y)^2}}, \frac{f_y \Delta y}{\sqrt{(f_x \Delta x)^2 + (f_y \Delta y)^2}}) \, ,$$

which is independent on the pixel depth δ. Also, the translation (x_δ, y_δ) lies in the line:

$$f_y \Delta y \cdot x - f_x \Delta x \cdot y = 0 \, .$$

Thus, the peaks with high values lie in a line across the center of the PSD. Additionally, pixels cannot move in the opposite direction. So the peaks lie in a line that starts from the center. The extreme case is a slanted plane in the camera's view. Then there are not distinguishable peaks, but a continuous line segment in the PSD. To keep it general and not rely on peak detection, this work proposes the following way to estimate the translation.

Independent of their depth, with a given camera translation, all pixels will move with collinear translation vectors - the magnitude of this translation depends on their depth and the magnitude of the camera translation. Thus we can treat the translation estimation in a novel way different from finding only the highest peak. Concretely, starting from the center of the PSD, which represents the no-translation case, we perform a polar search for the sector r_{max} that sums up the most energy. This sector now represents the direction of the translation vector, abbreviated as t. We have no concrete estimate for the magnitude of the motion, which would be anyways up to the unknown scale factor, therefore the estimated translation vector t is a unit vector, which is called ***unit translation vector*** in this work.

As introduced in Sec. 1.4, one weakness of FMT is that it does not consider the scale consistency for visual odometry, where the estimated translation between images ${}^1I$ and ${}^2I$ has to be re-scaled to be in the same unit as the one between ${}^2I$ and ${}^3I$. To overcome this drawback, eFMT calculates the re-scaling factor on the r_{max} sector. For that, we sample a translation energy vector $\mathbb{V}_t$ from the r_{max} sector of the PSD. With a given camera translation, regions with different depths correspond to different indices in the translation energy vector. The more pixels correspond to a region, the higher the energy. Assume the translation energy vector between ${}^1I$ and ${}^2I$ is ${}^2_1\mathbb{V}_t$ and that between ${}^2I$ and ${}^3I$ is ${}^3_2\mathbb{V}_t$. The second image ${}^2I$ is shared between both translations, thus the depths of the regions are the same for both translations. Any difference between the

translation energy vectors ${}^{2}_{1}\mathbb{V}_t$ and ${}^{3}_{2}\mathbb{V}_t$ must thus come from different magnitudes of translation, independently from the direction of that translation. In fact, the vectors are simply scaled by the ratio of the translation magnitudes, which then also maintains the correspondence of the regions and their size/ energy values in the vectors. Thus, the re-scaling factor ${}^{2\to1}_{3\to2}s_t$ can be calculated via pattern matching on ${}^{2}_{1}\mathbb{V}_t$ and ${}^{3}_{2}\mathbb{V}_t$ by

$$ {}^{2\to1}_{3\to2}s_t = \arg\min_{s} ||{}^{2}_{1}\mathbb{V}_t - f({}^{3}_{2}\mathbb{V}_t, s)||^2_2 \,, \tag{4.6} $$

where $f(\cdot)$ uses s to scale the vector ${}^{3}_{2}\mathbb{V}_t$ in length and value. Details are presented in Sec. 4.3.

Differences in the regions from changing occlusions and field of views add noise to the PSD but can be ignored in most cases, analogous to the image overlap requirement in the classical FMT [127].

4.2.2 Zoom-only Case

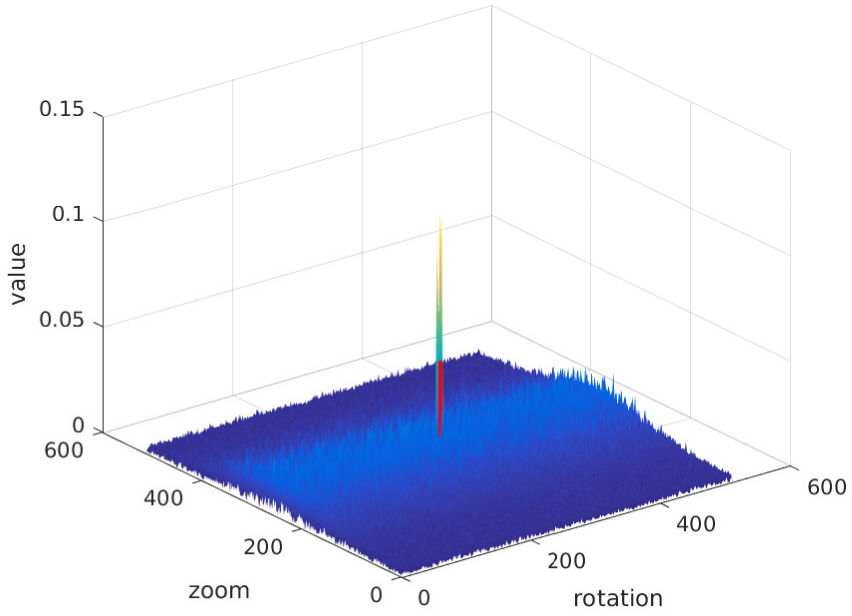


Figure 4.2: An example of rotation and zoom phase shift diagram.

As implied in Eq. (1.18), rotation and zoom share the same PSD (see Fig. 4.2). Also, the rotation is depth-independent and the same for all pixels, as shown in Eq. (4.1). Thus, eFMT calculates rotation in the same way as FMT does. In this section, we just consider the zoom-only case, i.e., the camera moves perpendicular to the imaging

plane. In this case, the Eq. (4.1) is simplified to

$$^2I(x,y) = {}^1I(z_\delta x, z_\delta y) \ . \tag{4.7}$$

Meanwhile, Eq. (1.18) becomes

$$^2M(\xi,\theta) = {}^1M(\xi - d_\delta, \theta) \ , \tag{4.8}$$

where $d_\delta = \log z_\delta$. Therefore, the multiple peaks of zoom lie in one column in the rotation and zoom PSD, because all the zoom peaks correspond to one rotation, i.e. the same column index. Note that these zoom peaks are sometimes continuous in real applications due to the continuous depth change, then these zoom peaks become high values in the PSD. For that, we no longer search for multiple peaks. Instead, a set of multi-zoom values $\mathbb{Z} = \{z_\delta\}$ is uniformly sampled between the maximum zoom z_{max} and minimum zoom z_{min} estimated from the column $\mathbb{C}_z^*$ with maximum sum energy. $\mathbb{C}_z^*$ can be found by

$$\mathbb{C}_z^* = \arg\max_{\mathbb{C}_z} \{\mathbb{C}_z \in q_z\} \ , \tag{4.9}$$

where q_z is the rotation and zoom PSD. In addition, as derived in Sec. 4.1, the zoom z_δ is described by Eq. (4.2), which is inversely proportional to the depth δ. Thus, the minimum and maximum zooms, estimated from the PSD, indicate the maximum and minimum pixel depths, respectively. Since the energy in the translation PSD also relates to the pixel depths, we can build correspondences between zoom energy and translation energy, which will be discussed in the next section.

Additionally, re-scaling for zoom is also essential in the zoom-only case for visual odometry. For that, a zoom energy vector $\mathbb{V}_z$ is extracted from $\mathbb{C}_z^*$. $\mathbb{V}_z$ is half of $\mathbb{C}_z^*$ with higher energy. This is based on the prior knowledge that all regions should consistently either zoom in or out. Suppose the zoom energy vector between ^1I and ^2I is ${}^2_1\mathbb{V}_z$ and that between ^2I and ^3I is ${}^3_2\mathbb{V}_z$. The re-scaling factor ${}^{2\to1}_{3\to2}s_z$ between ${}^2_1\mathbb{V}_z$ and ${}^3_2\mathbb{V}_z$ is found by

$${}^{2\to1}_{3\to2}s_z = \arg\min_s ||{}^2_1\mathbb{V}_z - g({}^3_2\mathbb{V}_z, s)||^2_2 \ ,$$

where $g(\cdot)$ is the function of shifting the vector ${}^3_2\mathbb{V}_z$. It is a variant of the pattern matching used in translation re-scaling. The only difference is that, while the translation energy vectors above are matched via scaling, the zoom energy vectors have to be matched via shifting. Both algorithms will be shown in Sec. 4.3.

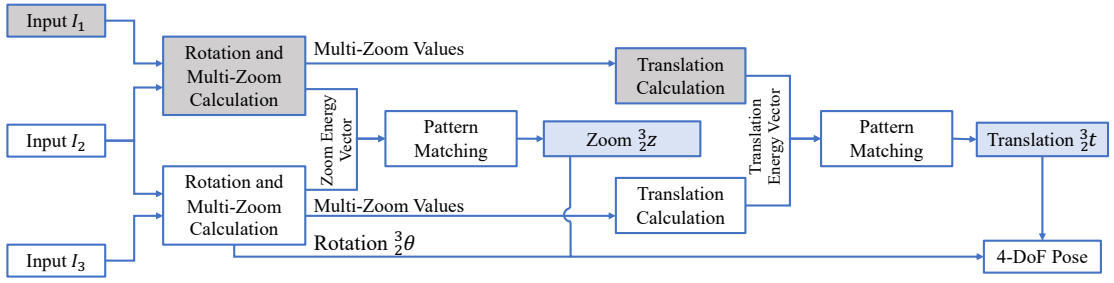


Figure 4.3: Pipeline of eFMT. Note that the output is the 4-DoF transformation between frame ${}^2I$ and ${}^3I$, but using 3 frames to estimate the re-scaling factor. The gray boxes indicate that the computation results of the previous iteration are reused.

4.2.3 General 4-DoF Motion

When the 4-DoF motion of the camera happens, the transformation between two poses is estimated following the scheme of the FMT. Our eFMT pipeline is shown in Fig. 4.3. Since monocular visual odometry algorithms are up-to-scale [99], we use three frames to calculate the up-to-scale transformation.

Similar to the FMT pipeline [110], we first calculate the rotation and zoom between two frames. Instead of searching for the highest peak value on the rotation and zoom PSD, we exploit all the information of half a column of the PSD in eFMT, yielding multi-zoom values $\mathbb{Z} = \{z_\delta\}$ and the zoom energy vector $\mathbb{V}_z$, as introduced in Sec. 4.2.2. In addition, the multi-zoom values $\mathbb{Z} = \{z_\delta\}$ are uniformly sampled between the minimum and maximum high zoom values of the PSD, which takes the energy instead of peaks into consideration. Thus it is robust to the continuous energy in the PSD. Afterwards, we obtain translation PSDs for the rotation θ_0 and each zoom value z_δ, by first re-rotating and re-zooming the second image:

$$ {}^2I' = {}^2I(z_\delta x \cos\theta_0 - z_\delta y \sin\theta_0, z_\delta x \sin\theta_0 + z_\delta y \cos\theta_0) \ , $$

and then performing phase correlation on image ${}^1I$ and ${}^2I'$ with Eq. (1.21). With the method introduced in Sec. 4.2.1, the translation energy vector $\mathbb{V}_{t,z_\delta}$ is extracted from the translation PSD. Then these multiple translation energy vectors are combined according to the weight of the zoom energy:

$$ {}^2_1\mathbb{V}_t = \sum_{z_\delta \in \mathbb{Z}} \frac{\mathbb{V}_z[z_\delta]}{U} * {}^2_1\mathbb{V}_{t,z_\delta} \ , \tag{4.10} $$

where $\mathbb{V}_z[\cdot]$ is the function to find the energy corresponding to the zoom z_δ and $U = \sum_{z_\delta \in \mathbb{Z}} \mathbb{V}_z[z_\delta]$. Since the higher the zoom value is, the more pixels correspond to the zoom, the corresponding translation energy vector should get the higher weight

accordingly. Thus the Eq. (4.10) holds.

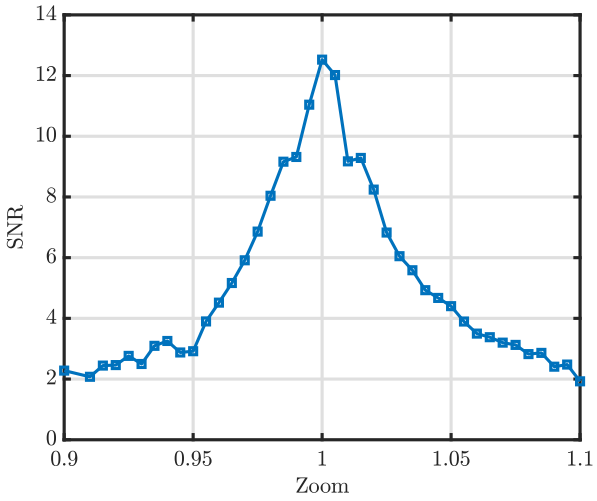
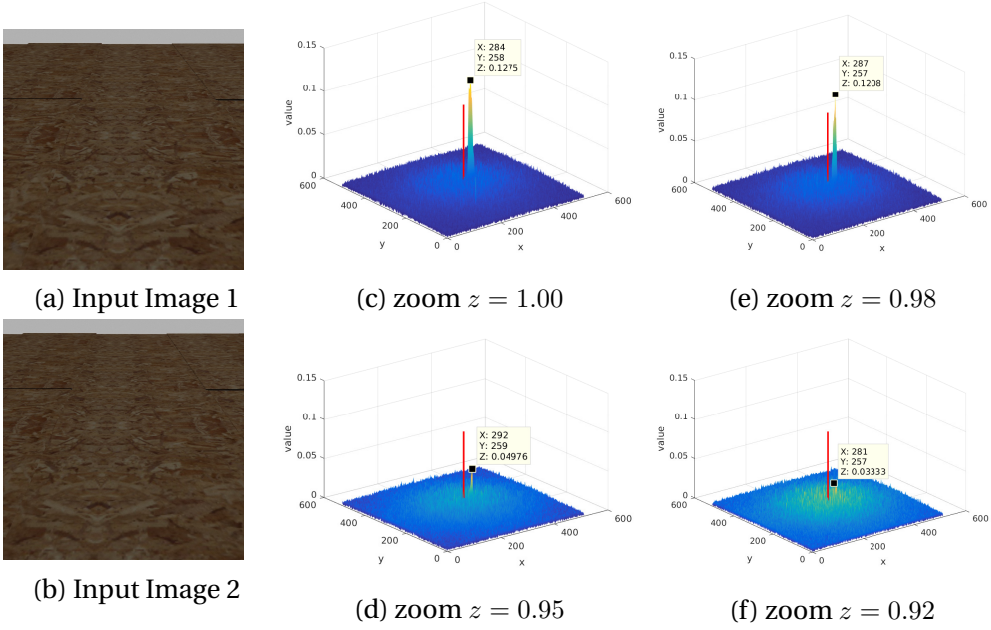


Figure 4.4: Signal-to-Noise ratio with different zoom. (Correct zoom is 1.)



(a) Input Image 1

(c) zoom $z = 1.00$

(e) zoom $z = 0.98$

(b) Input Image 2

(d) zoom $z = 0.95$

(f) zoom $z = 0.92$

Figure 4.5: Translation PSDs with different zoom values. The groundtruth zoom between the two input images is 1. To test the influence of zoom on the translation phase shift diagram, we set zoom value from 1.00 to 0.94 manually to re-zoom the second image and then perform phase correlation.

4.2.4 Tidbit on General 4-DoF Motion

Classical FMT decouples rotation and zoom from the translation. For eFMT this is not as simple: as the camera moves along the z-axis (perpendicular to the image plane), objects of different depths are zoomed (scaled) by different amounts. In a combined zoom and translation case, the apparent motion of a pixel depends on its depth, zoom and translation. But in order for the pattern matching of the translation energy vectors (Eq. (4.6)) to be based just on a simple scaling, the energy in the pixel motions has to be based just on the pixel depth and translation speed, so they have to be independent of the zoom. As described above, eFMT will calculate translation energy vectors $\mathbb{V}_{t,z_\delta}$ for different zoom values. This means that in multi-depth images there will be parts of the image that are zoomed with the incorrect zoom value but are then used as input in Eq. (1.21) and ultimately combined into the translation energy vector from Eq. (4.10).

One could assume that those incorrectly zoomed image parts lead to wrong pixel translation estimations, thus leading to a compromised translation energy vector. But this is not the case: The phase correlation (Eq. (1.21)) is sensitive to the zoom! It will only pick up on signals that are in the same zoom (scale) - other parts will just be noise. This is because with a wrong zoom Eq. (4.5) does not hold. Fig. 4.5 shows how a wrong zoom will influence the translation PSD. It can be found that wrong zoom decreases the energy of the correct translation and distributes the energy over the PSD. Also, slight difference does not change the translation PSD too much whereas big difference may destroy the result. To give a better explanation, we also demonstrate the signal-to-noise ratio (SNR) of the translation PSD with different fixed zoom values in Fig. 4.4. The SNR value is calculated by the ratio between the mean of the high values from the translation energy vector and the mean of the remaining values in the PSD. Fig. 4.4 and 4.5 show that a deviation of zoom of only 0.08 will lead to a SNR below 2.6, which is very noisy already. Thus when eFMT is iterating through the different multi-zoom values in $\mathbb{Z}$, the translation energy vectors $\mathbb{V}_{t,z\delta}$ will just pick up on the pixels that are correctly zoomed, thus leading to a correct $\mathbb{V}_t$.

4.2.5 Practical Consideration - Visual Odometry

We demonstrate the advantage of eFMT over FMT on camera pose estimation, i.e. visual odomtry. The main considerations in visual odometry are how to put translation and zoom in the same metric, i.e., translation and zoom consistency.

For that, we analyze the relationship between image transformation and camera motion again. As shown in Fig. 4.6, assume the objects with size l_i and depth δ_i are in the FoV of the camera C in Pose 1. The camera moves to Pose 2 with the motion $(\Delta x, \Delta y)$ in the $x-y$ plane and $\Delta\delta$ along the z direction.

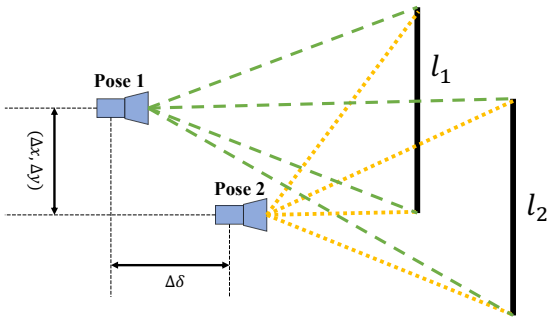


Figure 4.6: Objects of different depths in the FoV of the camera. When the camera moves from Pose 1 to Pose 2, the pixel motions of ${}^1l$ and ${}^2l$ are inversely proportional to the depth.

According to the basic properties of pinhole cameras, the zoom between the two frames captured in Pose 1 and 2 is $z_{\delta_i} = \frac{\delta_i}{\delta_i + \Delta\delta}$. Similarly, we can derive the translation between two frames of different depths δ_j. We are using j here, because in the algorithm, zoom and translation are calculated independently. The pixel translation between Pose 1 and 2 are $(\frac{f\Delta u_j}{\delta_j}, \frac{f\Delta v_j}{\delta_j})$, where f is the focal length of the camera. Then the ratio between the translation perpendicular to the imaging plane and that in the $x-y$ plane can calculated by

$$\frac{(\frac{1}{z_{\delta_i}} - 1)f}{\|(\Delta u_j, \Delta v_j)\|}, \tag{4.11}$$

if and only if $i = j$, meaning that the same object distance $\delta = \delta_i = \delta_j$ is used.

We can use pattern matching between the zoom energy vector and the translation energy vector to find the corresponding i and j. For simplicity, in this work we use maximum energy finding to determine the zoom z_{peak} with the highest peak in the zoom energy vector $\mathbb{V}_z$ (this corresponds to l_i with depth δ_i). In the translation energy vector for z_{peak} we then find the peak translation vector $(\Delta u', \Delta v')$ (l_j, which actually is l_i). This holds for all pixels with the same depth without the limitation of lying in one continuous plane.

Then we can get the 3D translation t between the camera poses:

$$t = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\delta \end{bmatrix} = \begin{bmatrix} \Delta u \\ \Delta v \\ \frac{(\frac{1}{z_{peak}} - 1)f}{\|(\Delta u', \Delta v')\|} \end{bmatrix}, \tag{4.12}$$

where $(\Delta u, \Delta v)$ is the unit translation vector.

4.2.6 Summary of Key Ideas

The key ideas of eFMT are outlined as follows:

- Observation that multiple depths will lead to multiple strong energies in the PSDs for zoom and translation, and that these signals are collinear.
- Instead of finding one maximum peak, as the classical FMT is doing, we represent the translation in a one-dimensional translation energy vector that encodes the number of pixels with certain amounts of motion, which correspond to certain depths. We treat the orientation and the magnitude independently. The orientation from the center of the PSD, from which the translation energy vector was sampled, is the direction of the motion, represented as a unit translation vector. The zoom is represented analogously. Thus, eFMT keeps the accuracy and robustness of FMT w.r.t features and direct methods, and improves the scale consistency of FMT.
- We put the zoom and translation in the same reference frame by finding the correspondence between zoom and translation based on pattern matching.
- Finally, we assign a magnitude to the second of the two found unit translation vectors of three consecutive frames by estimating a re-scaling factor between the translation energy vectors via pattern matching. The re-scaling for zoom is estimated analogously.

4.3 Implementation

This section introduces the implementation of a visual odometry framework based on eFMT. We first present this framework and then discuss in detail how to implement re-scaling for translation and zoom.

Algorithm 3 demonstrates the implementation of the eFMT-based visual odometry. FMT is directly applied for the first two frames to estimate rotation θ_0, zoom z and unit translation vector t. Additionally, zoom and translation energy vectors $\mathbb{V}_z$ and $\mathbb{V}_t$, used for pattern matching in the next iteration, are generated from the corresponding PSDs, respectively. For the following frames, eFMT is performed to calculate re-scaled zoom and translation, so that the 4-DoF motion between frames can be estimated. Moreover, the trajectory of the camera is generated via the chain rule.

As described above, for translation calculation, we find the sector with maximum sum energy r_{max} instead of the highest peak. Concretely, the PSD is divided into n sectors from the center b. Then we sum up the energy of the cells in each sector within a

Algorithm 3 eFMT-based Visual Odometry

Input: $\mathbb{I} = \{{}^{i}I | i \in \mathbb{N} \wedge 0 \leq i < \#\text{ frames}\}$
for i in $[1..len(\mathbb{I})]$ **do**
  if $i = 1$ **then** ▷ Similar to FMT
    Estimate rotation ${}^{1}_{0}\theta_0$, zoom ${}^{1}_{0}z$ and translation ${}^{1}_{0}t$
    Generate ${}^{1}_{0}\mathbb{V}_z$ from the rotation and zoom PSD
    Generate ${}^{1}_{0}\mathbb{V}_t$ from the translation PSD
  else ▷ Multi-zoom and Multi-translation
    Calculate the rotation and zoom PSD between ${}^{i-1}I$ and ${}^{i}I$
    Estimate the rotation ${}_{i-1}{}^{i}\theta_0$ and zoom values vector ${}_{i-1}{}^{i}\mathbb{Z}$ from the PSD
    Generate ${}_{i-1}{}^{i}\mathbb{V}_z$ from the PSD
    for j in $[0..len({}_{i-1}{}^{i}\mathbb{Z})]$ **do**
      Get translation energy vector ${}_{i-1}{}^{i}\mathbb{V}_{t,j}$ and unit translation vector ${}_{i-1}{}^{i}t_j$
    end for
    Combine translation energy vector to ${}_{i-1}{}^{i}\mathbb{V}_t$ via (4.10)
    Estimate 3D translation introduced in Sec. 4.2.5
    Estimate re-scaling factor between ${}^{i-1}_{i-2}\mathbb{V}_z$ and ${}_{i-1}{}^{i}\mathbb{V}_z$ via pattern matching
    Estimate re-scaling factor between ${}^{i-1}_{i-2}\mathbb{V}_t$ and ${}_{i-1}{}^{i}\mathbb{V}_t$ via pattern matching
    Update zoom and translation
    Perform chain rule on the 4-DoF transformation
  end if
end for
Output: camera poses corresponding to $\mathbb{I}$

Algorithm 4 Re-scaling for Translation

Input: ${}^{2}_{1}\mathbb{V}_t$ and ${}^{3}_{2}\mathbb{V}_t$
Initialize distance d with infinity
for $s = 0.1 : 0.002 : 10.0$ **do**
  Scale ${}^{3}_{2}\mathbb{V}_t$ to ${}^{3}_{2}\mathbb{V}'_t$ with s
  Calculate Euclidean distance d_s between ${}^{2}_{1}\mathbb{V}_t$ and ${}^{3}_{2}\mathbb{V}'_t$
  if $d_s < d$ **then**
    $d \leftarrow d_s$
    ${}^{2\rightarrow1}_{3\rightarrow2}s_t \leftarrow s$
  end if
end for
Output: rescaling factor ${}^{2\rightarrow1}_{3\rightarrow2}s_t$

certain opening angle o, e.g. 2°, to find the r_{max}. Afterwards, the direction from the center b to the highest value of the sector r_{max} is considered as the translation direction, i.e. unit translation vector ${}^2_1t_i$. Furthermore, we represent the values of the maximum sector ${}^2_1r_{max}$ as the 1D energy vector ${}^2_1\mathbb{V}_{t,z_\delta}$. We sample the energy in the maximum sector r_{max} at uniform distances to fill ${}^2_1\mathbb{V}_{t,z_\delta}$. Then the translation energy vectors are combined to ${}^2_1\mathbb{V}_t$ with Eq. (4.10).

Moreover, the pattern matching algorithms, used in the re-scaling for translation and zoom, are shown in Algorithm 4 and 5, respectively. There are several methods to handle pattern matching, for example phase correlation, search algorithms and dynamic programming. Considering the robustness on outliers of the PSD signals, we use a search algorithm in this work.

Algorithm 5 Re-scaling for Zoom

Input: ${}^2_1\mathbb{V}_z$ and ${}^3_2\mathbb{V}_z$
Initialize distance d with infinity
for $\Delta = -r : 1 : r$ **do** ▷ r is the length of ${}^2_1\mathbb{V}_z$
    Shift ${}^3_2\mathbb{V}_z$ to ${}^3_2\mathbb{V}'_z$ with Δ
    Calculate Euclidean distance d_s between ${}^2_1\mathbb{V}_z$ and ${}^3_2\mathbb{V}'_z$
    if $d_s < d$ **then**
        $d \leftarrow d_s$
        ${}^{2\rightarrow1}_{3\rightarrow2}s_z \leftarrow$ shift_to_scale{Δ}
    end if
end for
Output: rescaling factor ${}^{2\rightarrow1}_{3\rightarrow2}s_z$

4.4 Experiments and Analysis

In this section, we evaluate the proposed eFMT algorithm in both simulated and real-world multi-depth environments. Note again that there are multiple variants of FMT, we use the improved one from [20, 25] for better robustness and accuracy. Since all the FMT implementations only search for one peak in the PSDs, they will meet difficulties in multi-depth environments, no matter which implementation is used.

We first present basic experiments about the zoom and translation re-scaling in the simulation test. The scenario only includes two planes with different depths to show the basic effectiveness of eFMT. Then eFMT is compared with FMT and the state-of-the-art VO methods, ORB-SLAM3 [27], SVO [47] and DSO [42], in real-world environments. The three state-of-the-art VO methods that do not rely on FMT are the most popular and representative monocular ones, as pointed out in [62]. The tests in the real-world environments include two parts: one toy example with two wooden boards and a large-

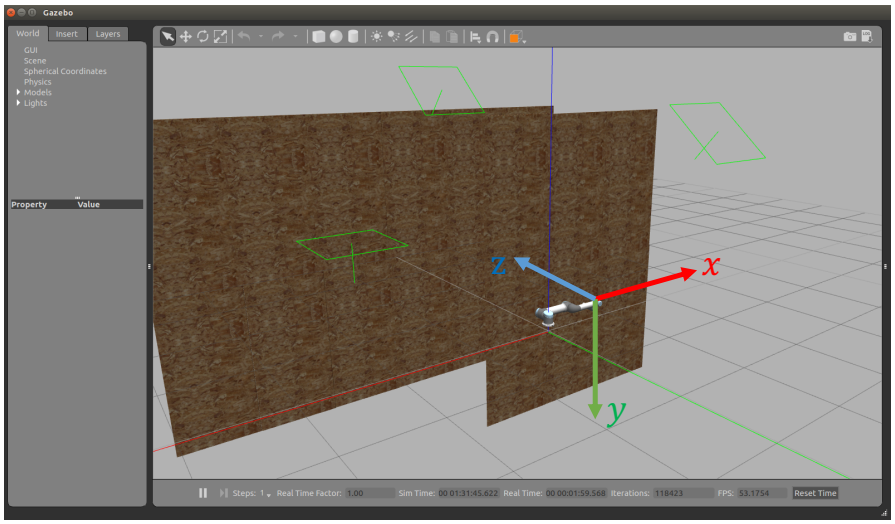


Figure 4.7: Simulated environment (y points down).The camera is equipped on the end-effector of a robot arm such that we can control the robot arm to move the camera.

scale UAV dataset[I]. The toy example is similar to the simulation environment. Since the features are very similar on the wooden board, the scenario is more difficult than general indoor environments, even though there are only two planes. To evaluate the eFMT algorithm in a more general case and provide a potential use-case of eFMT, we proceed the second test with a down-looking camera mounted on a UAV. The scenario includes many different elements, such as building roofs, grass and rivers. Since there are many different depths in the view, especially that the building will be a slanted plane due to the perspective projection, it is thus challenging for FMT. In addition, the feature-deprived road surface and grass would be a big challenge for classic VO methods. We will show that eFMT can handle both difficulties.

All experiments are conducted with an Intel Core i7-4790 CPU@3.6GHz and 16GB Memory without GPU. The algorithm is implemented in C++ using a single thread.

4.4.1 Experiments on the Simulated Datasets

In this test, images are collected in the Gazebo simulation for accurate ground truth. As shown in Fig. 4.7, the camera is equipped on the end-effector of a robot arm such that we can control the robot arm to move the camera.

Zoom Re-scaling

In this case, we move the robot arm along the $z-$axis to generate three simulated images with two planes in different depths. As shown in Fig. 4.8b, 4.8c, 4.8d, they

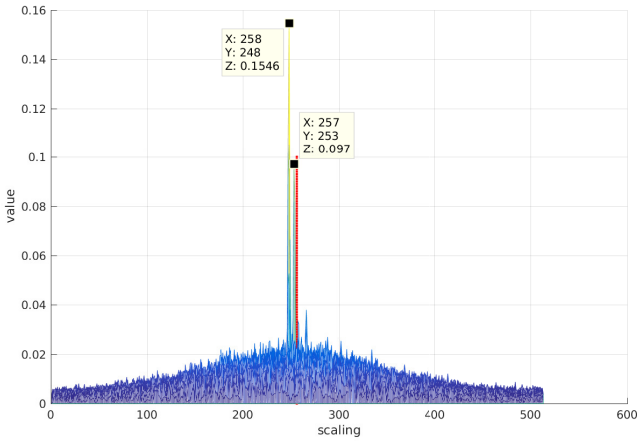
[I]https://robotics.shanghaitech.edu.cn/static/datasets/eFMT/ShanghaiTech_Campus.zip

(a) Visual Aid Input Image 0

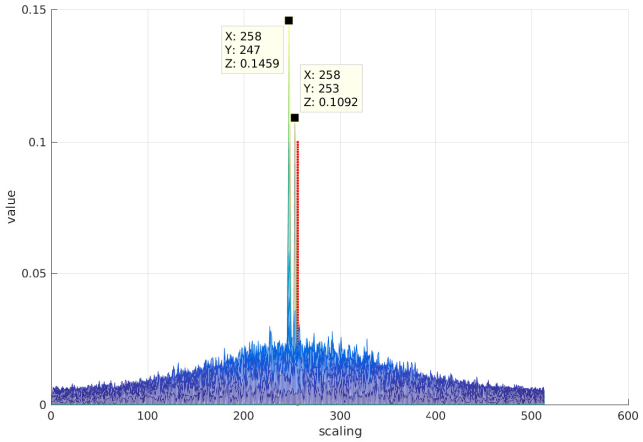
(b) Input Image 0

(c) Input Image 1

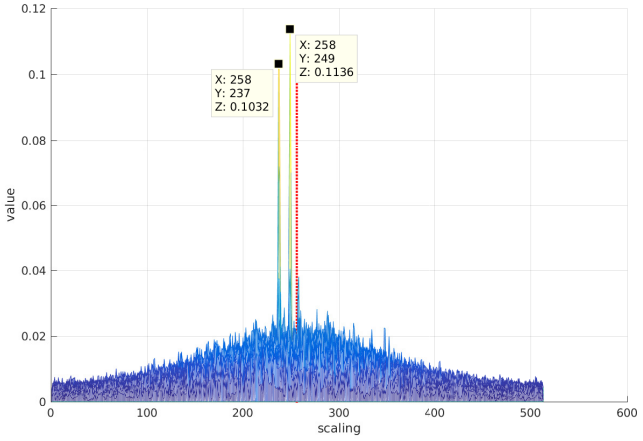
(d) Input Image 2



(e) PSD between Image 0 and 1



(f) PSD between Image 1 and 2



(g) PSD between Image 0 and 2

Figure 4.8: Three rotation and zoom phase shift diagrams (PSD) with multiple zooms. In the first row, the left parts of the input images are further than the right parts w.r.t the camera. Thus there are two peaks in the rotation and zoom PSD.

are zoomed in from top to down. In each image, the left half is further whereas the right half is closer. Then the rotation and zoom PSDs are shown in the second row of Fig. 4.8. It can be seen that each diagram has two peaks, which indicates two different depths in the view. Moreover, the higher peak is not always in the left, which implies the majority depth in the view changes, which destroys the scale consistency of the FMT. Traditional FMT only uses the highest peak. Instead, the proposed eFMT takes the zoom energy vector into consideration and puts all zoom values into the same scale through re-scaling - up to one unknown scale factor.

Table 4.1: Loop Closure for Zoom Estimation

	${}^1_0z$	${}^2_1z$	${}^2_0z$	$\|\|{}^1_0z * {}^2_1z/{}^2_0z\|\|$
eFMT	0.889	0.889	0.768	1.029
FMT [20, 25]	0.889	0.881	0.902	0.868

Here we show that the eFMT outperforms FMT by using the three images as a small loop closure. The zoom ${}^2_0z$ between image 0 and 2 should equal to the product of the zoom ${}^1_0z$ between image 0 and 1 and ${}^2_1z$ between image 1 and 2. The result in Table 4.1 shows that eFMT estimates the zoom correctly, so that the zoom loop holds, i.e. $||{}^1_0z * {}^2_1z/{}^2_0z|| \approx 1$. However, FMT only tracks the highest peak. The plane that the highest peak in Fig. 4.8g corresponds to is different from that in Fig. 4.8e and Fig. 4.8f, so ${}^2_0z$ and ${}^2_1z$ are calculated based on different planes with different depths. Thus $||{}^1_0z * {}^2_1z/{}^2_0z||$ is further away from 1.

Visual Odometry in Simulated Scenario

In this case, the simulated robot arm moves in the $x-z$ plane to generate images with combined translation and zoom. Here, we compare the visual odometry based on eFMT and FMT on this dataset. Fig. 4.9 shows that eFMT tracks the correct re-scaling factor to the end while the FMT fails at about $z = -0.5m$, which indicates that eFMT also works better than FMT with zoom and translation. This benefits from the re-scaling based on pattern matching, as introduced in Sec. 4.2.

4.4.2 Experiments on Real Datasets

After the preliminary tests in the simulated environment, we evaluate the performance of eFMT by comparing it with FMT and other state-of-the-art VO methods in real-world scenarios. The first example is similar to the simulation setting with two wooden boards in the camera's view, as shown in Fig. 4.10a. The ground truth is provided by a tracking system. In the second example, we collect a dataset with an unmanned aerial

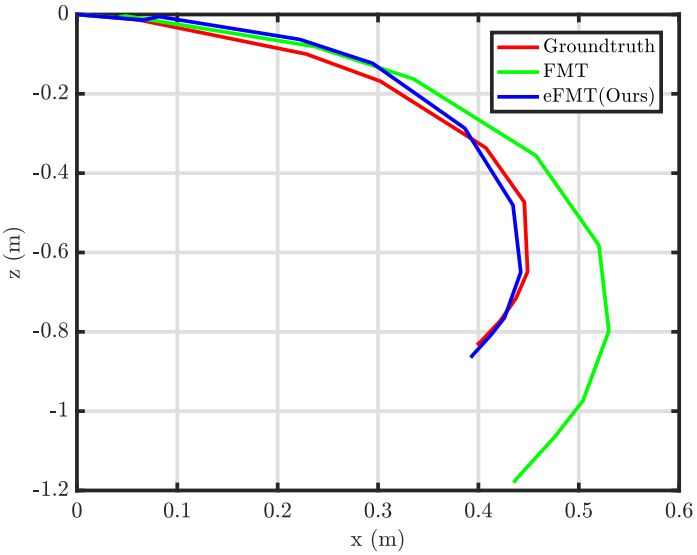


Figure 4.9: Visual odometry comparison in a simulated scenario. The data is collected with the setting in Fig. 4.7.

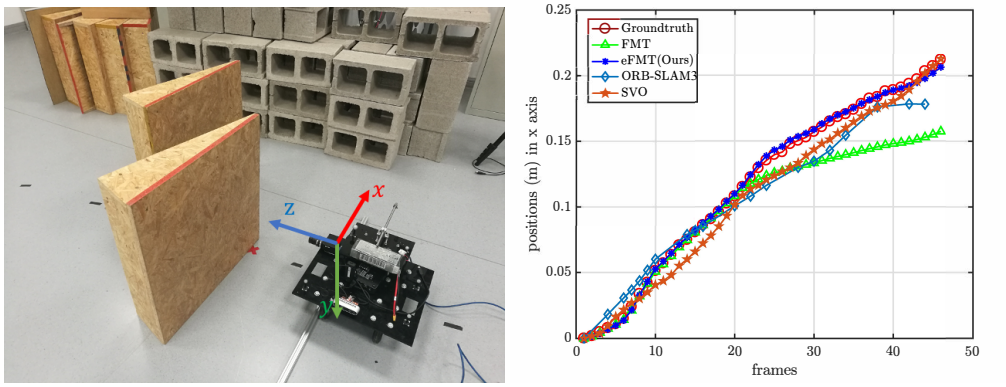
vehicle flying over our campus. More details are introduced in the following.

A toy example

In this case, we evaluate the visual odometry with only translation along the $x-$axis (see in Fig. 4.10a) with two different depths. Similar to the simulation, the wooden board with smaller depth first goes into the camera's view, then both boards are in the view, finally only the wooden board with larger depth is observed.

Fig. 4.10b compares the localization results with different methods, including FMT (green triangle), eFMT (blue star), SVO (blue triangle) and ORB-SLAM3 (brown star). The results of DSO are omitted here because it fails tracking in this scenario. To compensate for the unknown scale factor, the estimated results are aligned to the ground-truth (via a tracking system) by manual re-scaling. Since the camera only moves in the x direction, we only show the positions in x axis versus frames. The absolute error (Table. 4.2) will include errors in both x and y direction.

We can see that FMT begins to suffer from scale drift approximately from the 20^{th} frame, where FMT changes the tracked panel, because the new panel now is bigger in the view and thus has a higher peak. That new panel is further away, thus the pixels move slower, thus FMT underestimates the motion compared to previous frames. In contrast, the proposed eFMT maintains the correct scale till the last frame, because our pattern matching re-scales all unit translation vectors correctly. Compared with SVO and ORB-



(a) Real environment (y points down) (b) Visual Odometry Comparison

Figure 4.10: A visual odometry example in a real-world environment. DSO fails in this test, thus it is ignored in this figure.

SLAM3, eFMT tracks each frame more accurately. The absolute trajectory errors in Table. 4.2, including mean, max and median of errors, also shows that eFMT achieves the smallest error, followed by SVO and ORB-SLAM3. Concretely, the mean error of eFMT is approximately $1/6$ of that of ORB-SLAM3, and about $1/8$ against FMT or SVO. This test shows that eFMT outperforms the popular visual odometry algorithms in this challenging environment, thanks to the robustness of the spectral-based registration.

Table 4.2: Absolute trajectory error comparison

Methods	Mean (mm)	Max (mm)	Median (mm)
FMT [20, 25]	17.1	54.7	10.1
eFMT	**2.1**	**6.0**	**1.8**
SVO [47]	17.0	38.0	17.5
ORB-SLAM3 [27]	13.0	21.6	13.6
DSO [42]	/	/	/

The UAV dataset

In addition to the above toy examples, we compare the proposed eFMT with FMT, ORB-SLAM3, SVO and DSO on a bigger dataset. The dataset is collected by a down-looking camera equipped on a DJI Matrice-300 RTK. The flying speed is set to $2m/s$ and the image capture frequency is 0.5 Hz. The path of the drone over our campus is shown in Fig. 4.11. The DJI aerial vehicle collected 350 frames on a trajectory of about $1,400$ meters. The height above ground is about 80 meters, which is approximately 20 meters higher than the highest building. As we mentioned in the beginning of the experiment,

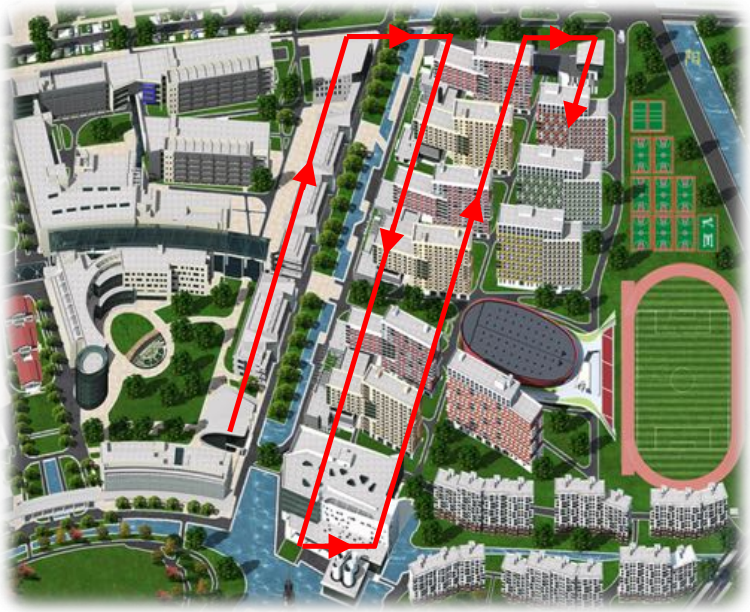


Figure 4.11: A UAV's flying trajectory over a campus. A down-looking camera is equipped on it.

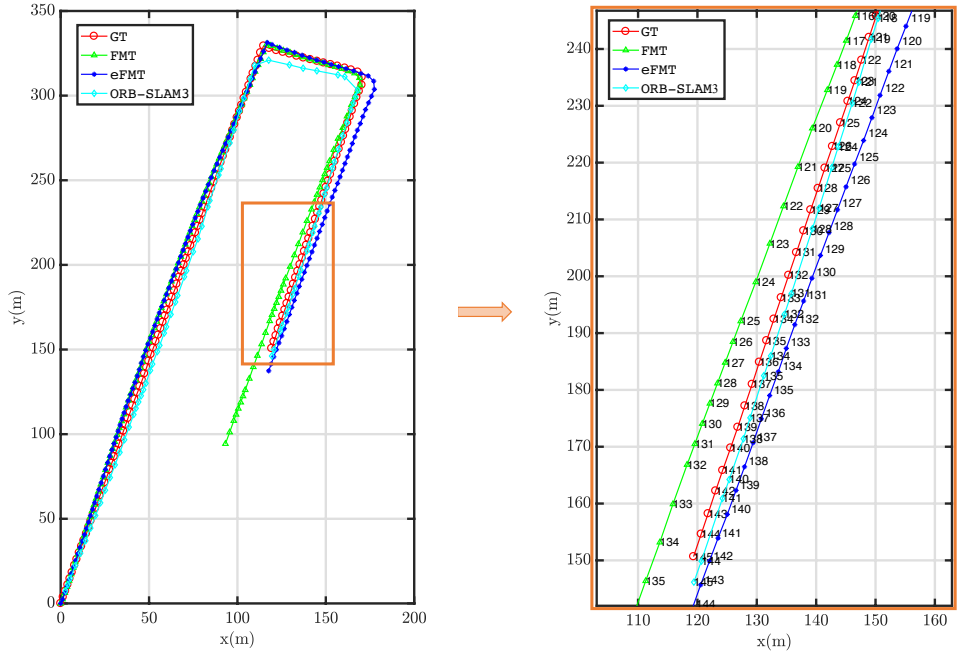


Figure 4.12: Estimated trajectories on the UAV dataset. SVO and DSO fail to track the images.

this dataset contains all kinds of different elements. These include roofs, road surfaces, a river and grass, where some of them are challenging for the classic VO methods that are not based on FMT. Furthermore, the multiple depths increase the difficulty for FMT. In this case, we will show that the eFMT not only keeps the robustness of FMT but also overcomes its single-depth limitation.

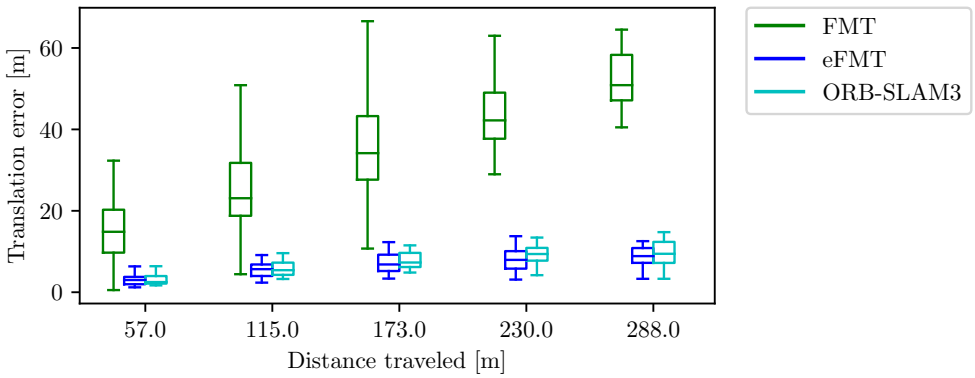
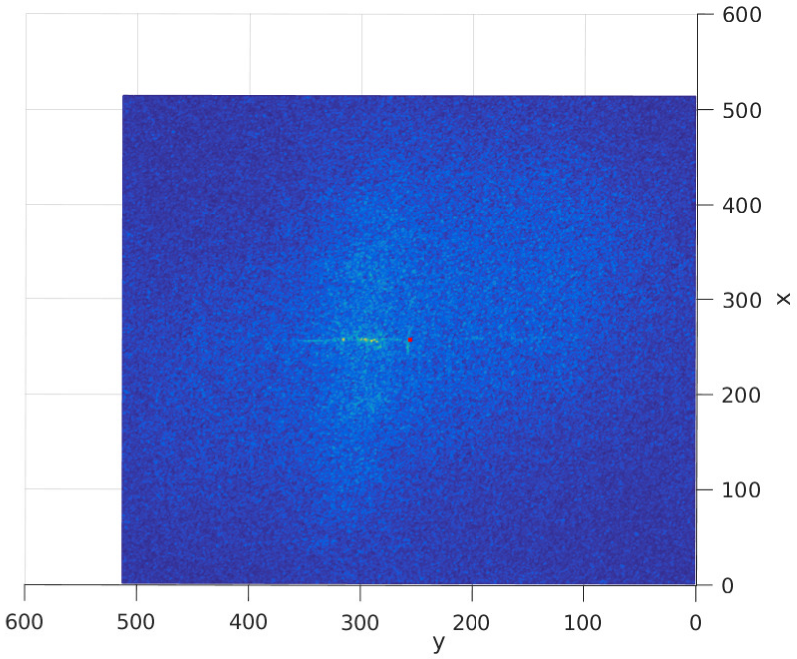


Figure 4.13: Absolute translation error on the UAV dataset. SVO and DSO fail to track the images.

The overall trajectories of different approaches are shown in Fig. 4.15. The trajectories are aligned with a scale a rotation calculated from the poses of the $0th$ frame and the $80th$ frame. Since SVO and DSO fail to estimate the camera poses, the trajectories of them are not included in this figure. Also, it can be found that the ORB-SLAM3 fails to track several times, as indicated by the red stars. After each failure, the trajectory of ORB-SLAM3 is realigned. Both FMT and eFMT succeed in estimating the camera poses till the end of the dataset, though the translation has some drift. To evaluate the performance of FMT, eFMT and ORB-SLAM3, we compare these methods only up to the frame that ORB-SLAM3 fails. The performances of different approaches are shown in Fig. 4.12. From the right local enlarged figure, we can find that the estimated speeds of eFMT and ORB-SLAM3 are almost constant, as indicated by the equal distances between the frames. This is consistent with the centimeter-grade RTK GPS ground truth. However, the estimated speed of FMT changes according to the view. For instance, the speed is faster from frame 125 to 132 than that from frame 132 to 138, because the dominant plane is ground in the former case whereas the dominant plane changes to the roof in the latter case. In addition, Fig. 4.13 displays the absolute translation error versus distances with the evaluation tool from [152]. If only comparing the performance when all three approaches are tracking successfully, the performance of eFMT is on a par with ORB-SLAM3 and both of them are better than FMT, because FMT suffers from different depths.

(a) Frame 143 (b) Frame 144



(c) Translation PSD

Figure 4.14: Example line segment in the translation phase shift diagram (PSD).

Note that there are continuous line segments in the translation PSD when there are slanted planes in the view. As shown in Fig. 4.14, the buildings in Image 1 and 2 become inclined due to the perspective projection, which yields the line segments (left to the red center) in the translation PSD below. In the UAV dataset, such inclined planes are common, thus pattern matching is necessary for re-scaling. Moreover, the estimated trajectory shown in Fig. 4.15 demonstrates that eFMT can handle such slanted planes issues.

Thus, this experiment shows that eFMT has two advantages: 1) it successfully extends FMT to multi-depth environments, that is, no matter the multiple depths are continuous (e.g., slanted plane) or discrete (e.g. roofs and ground), eFMT can track the camera motion; 2) it keeps the robustness of FMT that it can still track the camera motion in the feature-deprived scenarios, such as building roofs, whereas the classic VO methods may fail tracking. The drawback of eFMT is that it may suffer from the accumulated errors as the camera moves for a long time. To overcome this, we will introduce pose optimization in our future work.

4.4.3 Computation Analysis

[127] pointed out that image resolution has a big impact on the FMT algorithm and image down-sampling does not hurt FMT performance. In our preliminary test, we find that this still holds for eFMT. The run-time of eFMT is about 0.7 seconds per frame with image resolution 512×512 and about 0.2 seconds per frame with a resolution of 256×256 based on the single-threaded C++ implementation, which is about two times slower than that of FMT. In addition, the multi-translation calculations with multi-zoom values are independent from each other, thus we can compute these in parallel to speed up the algorithm. Thus, eFMT could run as fast as FMT.

4.5 Conclusions

This chapter extends the classical FMT algorithm to be able to handle zoom and translation in multi-depth scenes. We present a detailed problem formulation and algorithm. Experiments show the clear benefit of our proper re-scaling for Visual Odometry in scenes with more than one depth and compare it to FMT, which indicates that eFMT inherits the advantages of FMT and extends its application scenarios. Moreover, eFMT performs better than the popular VO methods ORB-SLAM3, DSO and SVO in all our experiments, performed on our datasets collected in challenging scenarios. Since FMT has already been used in all kinds of applications, we will consider applying eFMT in the use-cases that FMT has been exploited, for example remote sensing, image registration and localization. In addition, we plan to employ eFMT in further experiments

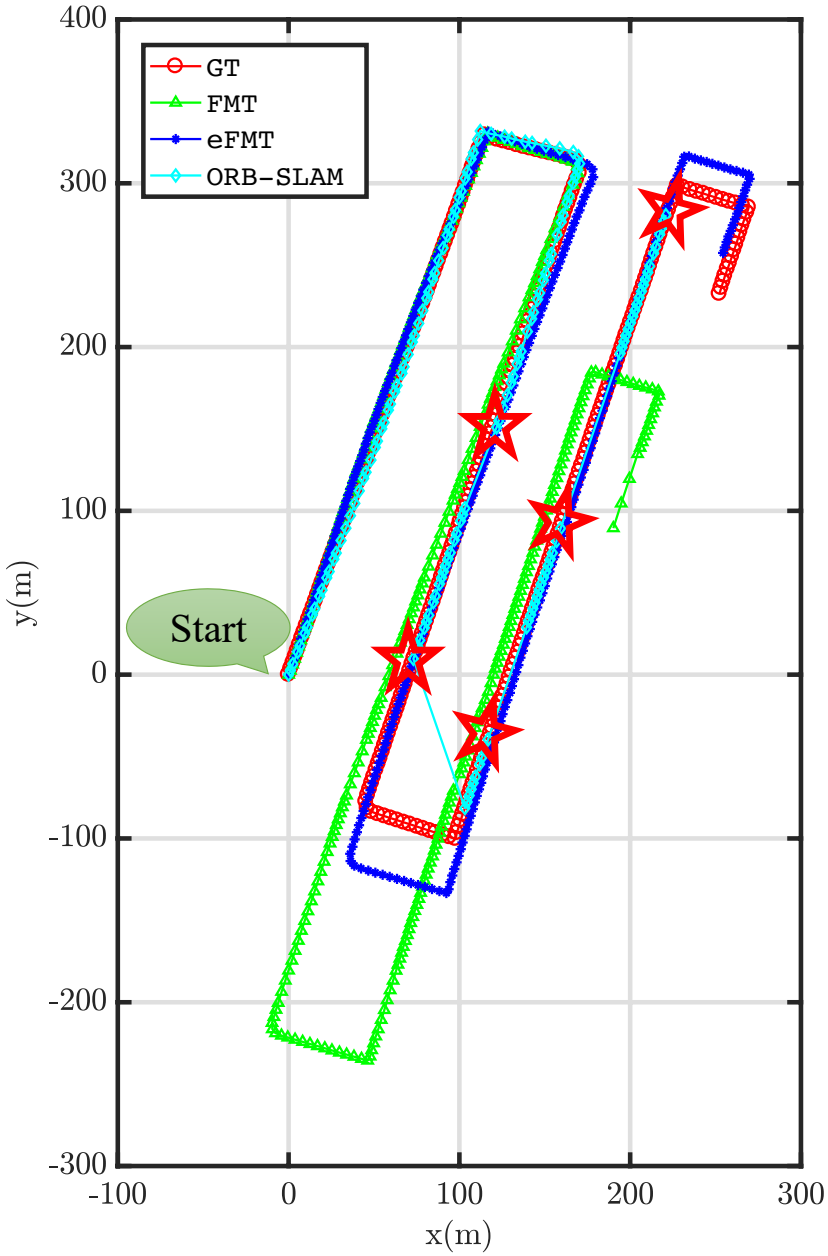


Figure 4.15: Overall trajectories of different methods on the UAV dataset. ORB-SLAM3 fails several times, as indicated by the red stars. These sub trajectories of ORB-SLAM3 are aligned manually for visualization. SVO and DSO fail to track the images.

in feature-deprived environments with different depths, such as underwater and foggy scenarios, because eFMT is also more robust than the classical VO approaches in feature-deprived environments. Also, with the help of an IMU, eFMT will be able to estimate the 6-DoF motion. Furthermore we plan to add loop closing, uncertainty estimation and graph optimization to our approach to develop a fully fledged SLAM system.

5 Conclusions and Outlook

5.1 Conclusions

With more released VO/VSLAM frameworks, more and more robots use vision-based localization. However, current VO/VSLAM algorithms may have difficulties when the scenarios are challenging, such as feature-deprived, blurry and turbid underwater environments. The thesis exploits FMT to estimate motion between images, such that the robustness and accuracy of VO are improved. However, FMT can only be used for pinhole camera's 2D motion and requires single-depth scenarios. For that, the thesis proposes eFMT for multi-depth scenarios. Overall, this thesis studies the application of FMT in visual localization, including the following aspects.

Firstly, Chapter 2 uses FMT for omni-directional image matching and implements a simple VO based on that. Since FMT can only estimate motion in single-depth scenarios, we first convert omni-directional images to panorama images and then extract single-depth sub-images via recursive sub-image strategy. The SNR ratio of the phase shift diagram (PSD) is used to determine whether the recursion is needed. Moreover, the experiments show that using FMT for finding concordant points is more robust than other features. Thus, the concordant points calculated by FMT can provide more robust results for omni-directional camera pose estimation, especially in feature-deprived, blurry and dynamic environments.

Secondly, we propose a novel rotation estimation for omni-directional cameras based on the motion vectors calculated by FMT in Chapter 3. The rotation estimation of omni-directional cameras is modeled as sinusoidal fitting. In this chapter, we first divide panorama images into several sub-images in the horizontal direction and apply FMT on these sub-image pairs. Afterwards, we find that the motion vectors of sub-image pairs are sinusoidal shapes based on mathematical derivation. Thus, rotation can be estimated with the parameters from sinusoidal fitting. Compared with geometry methods, the proposed sinusoidal fitting method achieves higher accuracy

and robustness. Also, the sinusoid fitting method with FMT is more robust and accurate than the the method proposed in Chapter 2: five-point algorithm with FMT, in rotation estimation for omni-directional cameras. Though we considered translation of cameras in the sinusoidal model, it is difficult to estimate the translation because we assume that the scenario is single-depth, which is difficult to meet. In addition, multi-depth will also influence the performance of FMT, which is compensated by sinusoidal fitting in this chapter.

Thirdly, in Chapter 4, we the propose extended FMT to solve the multi-depth issue mentioned in the above two chapters. eFMT always extracts the line with maximum sum energy, i.e., energy vector, instead of finding the highest peak in the PSD. So it keeps the motion information for multi-depth. Additionally, eFMT calculates the re-scaling factor by pattern matching on energy vectors. Finally, we implement an eFMT-based VO for perspective images and compare with popular VO methods. The experiments show that the accuracy of eFMT-based VO is in a par with ORB-SLAM3, and eFMT is more robust than ORB-SLAM3 and other VO methods. However, the disadvantage of the eFMT-based VO is obvious that it can be only used for 4-DoF motion estimation, the same as FMT.

5.2 Outlook

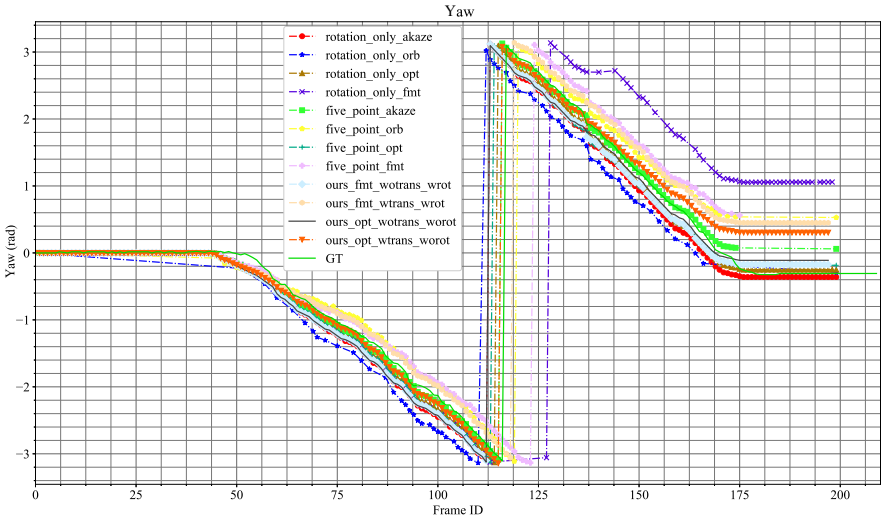
As mentioned in the conclusions, there exist obvious disadvantages of the proposed algorithms in this thesis. For example, eFMT can only estimate 4-DoF motion; the sinusoidal fitting cannot estimate the translation. To overcome these drawbacks, we propose potential research for future work.

- To estimate the camera tilt with eFMT, we consider two possible solutions. One is to borrow the technology of fractional FFT, which is used for tilt estimation in the improved FMT. Another is to fuse with other sensors, such as IMU and accelerators, which can provide information of camera tilt. However, it does not mean that eFMT can estimate 6-DoF motion with these sensors because there may be limited overlap after transforming the image with tilt angles.
- We will try to extend the sinusoidal fitting for translation estimation. Based on the observations in Chapter 4, there will be multiple sinusoidal curves when the scenario is multi-depth. Moreover, only the amplitude of these curves is different when the camera moves in the imaging plane and only the offset is different when the camera moves perpendicularly to the imaging plane. Thus we can model translation estimation into a multi-sinusoidal fitting problem.
- We consider building 3D maps based on eFMT. Based on the analysis in Chapter 4

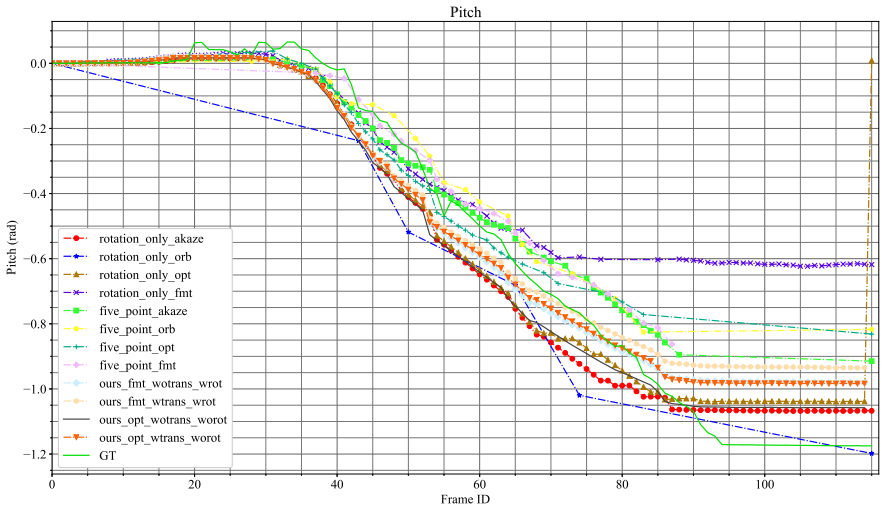
that different depths will cause different positions of high values, which lies in one line. Thus, the energy vector contains depth information. We plan to use multiple energy vectors to recover depth maps and reconstruct 3D maps in pinhole images.

- In addition, we will further evaluate the robustness of pose estimation based on eFMT in our future work.

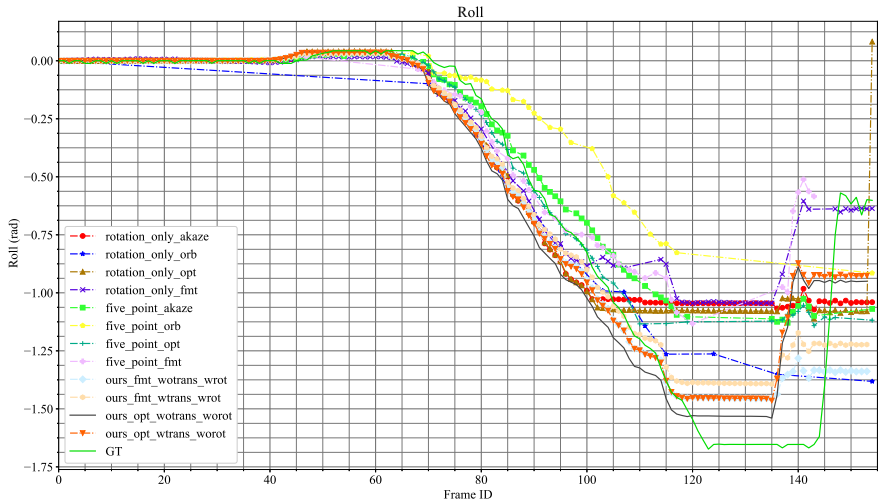
A Comparison between Sinusoidal Fitting and Geometry Methods



(a) Yaw

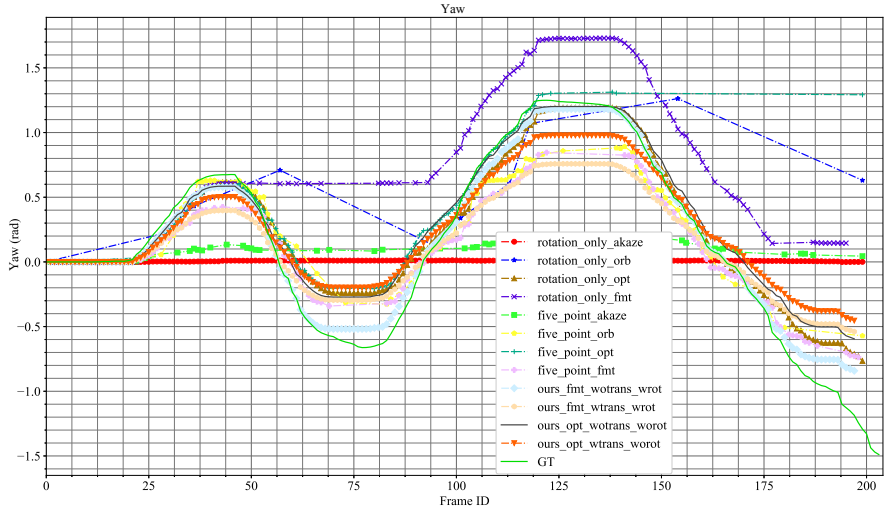


(b) Pitch

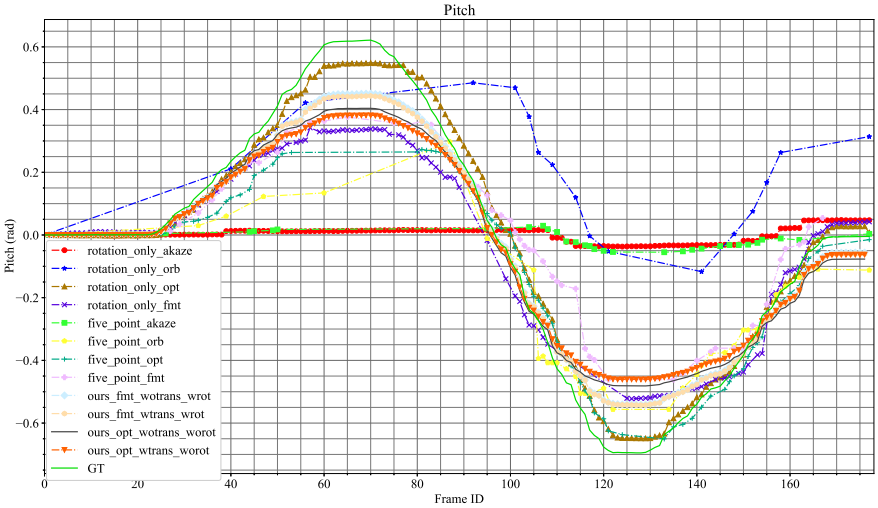


(c) Roll

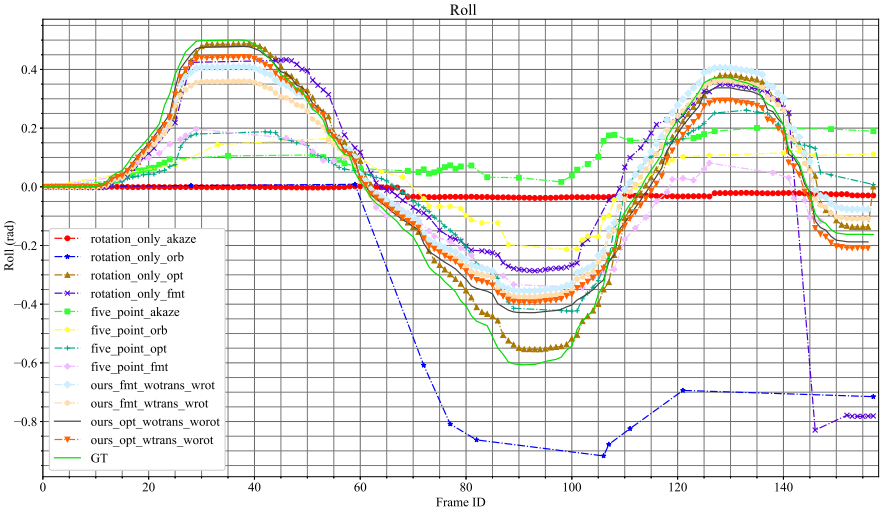
Figure A.1: Qualitative results for single rotation estimation on indoor_single_yaw, indoor_single_pitch and indoor_single_roll datasets



(a) Yaw

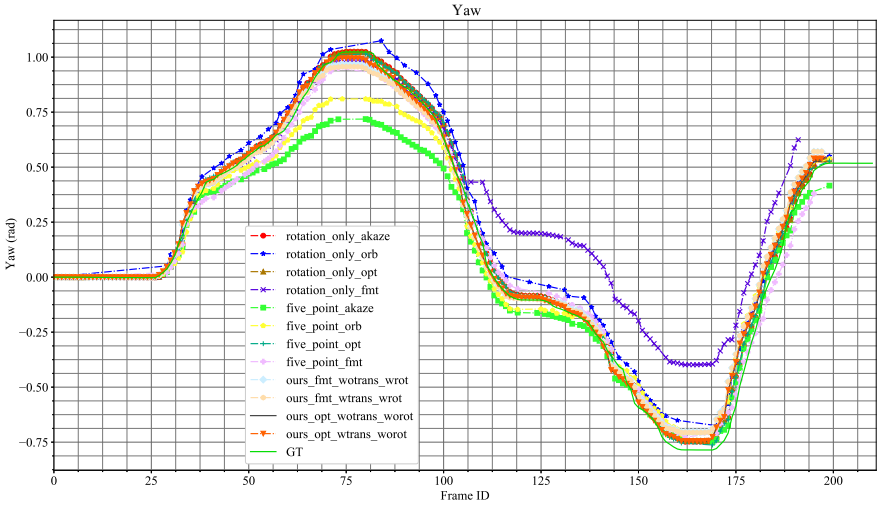


(b) Pitch

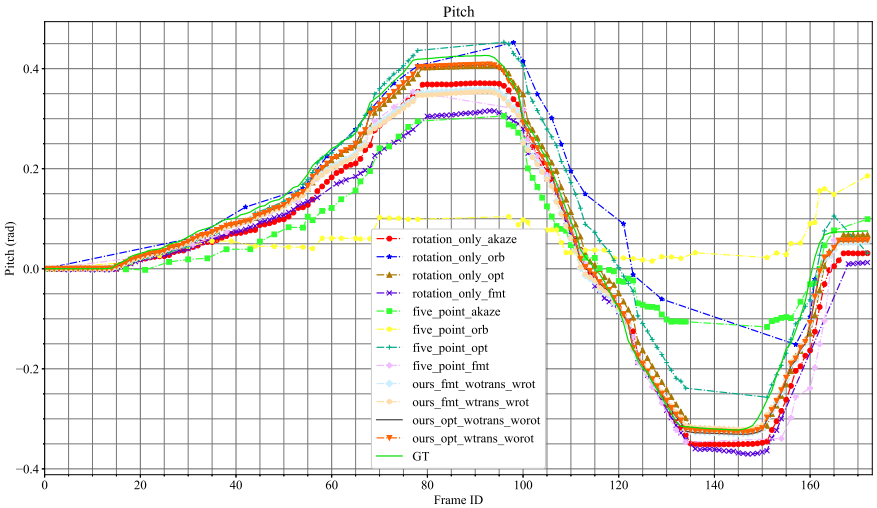


(c) Roll

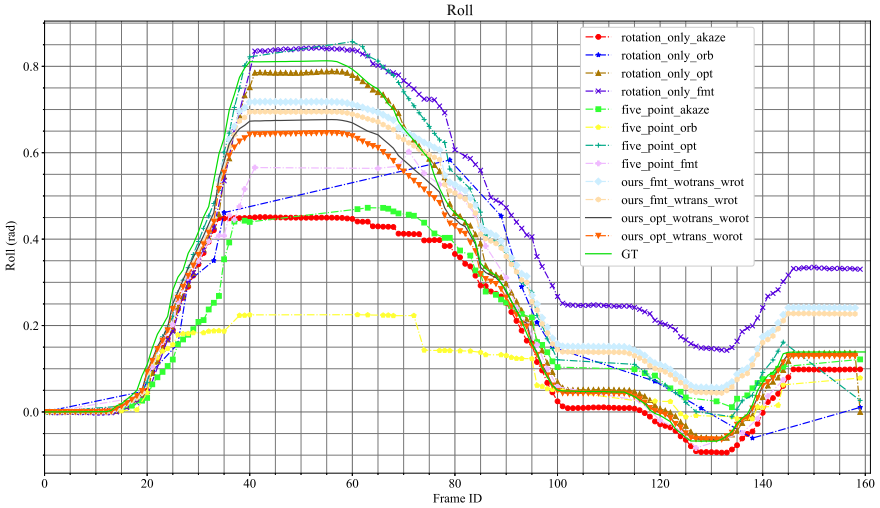
Figure A.2: Qualitative results for single rotation estimation on grass_single_yaw, grass_single_pitch and grass_single_roll datasets



(a) Yaw

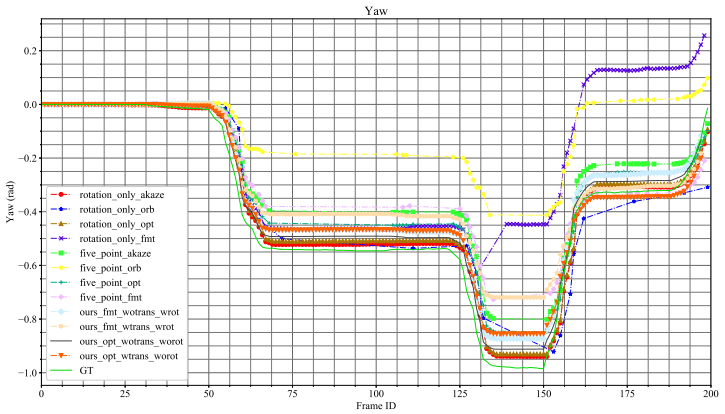


(b) Pitch

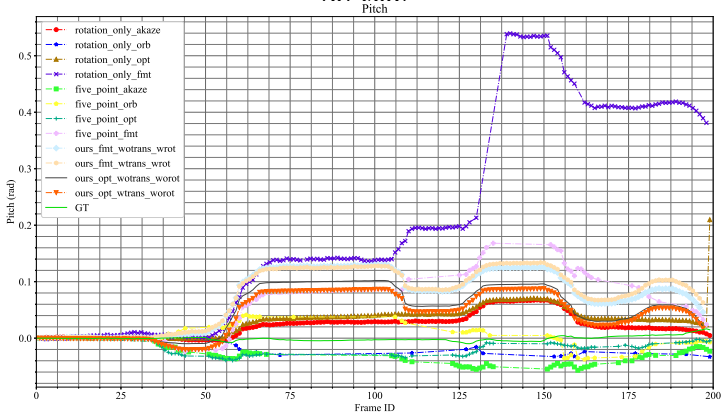


(c) Roll

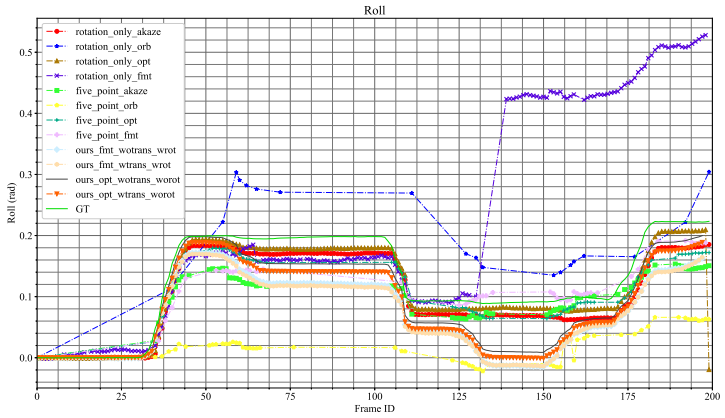
Figure A.3: Qualitative results for single rotation estimation on street_single_yaw, street_single_pitch and street_single_roll datasets.



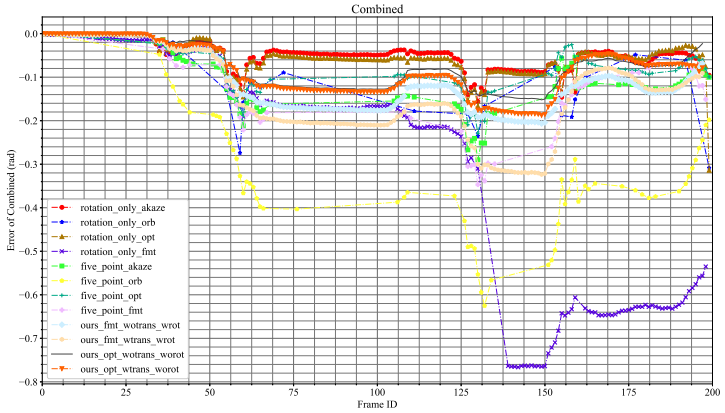
(a) Yaw



(b) Pitch

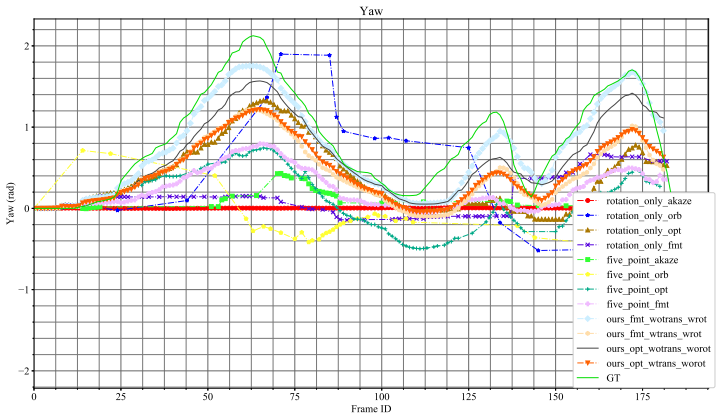


(c) Roll

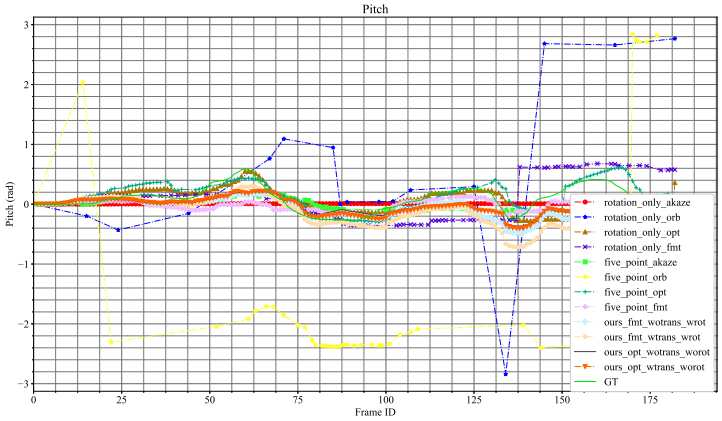


(d) Combined

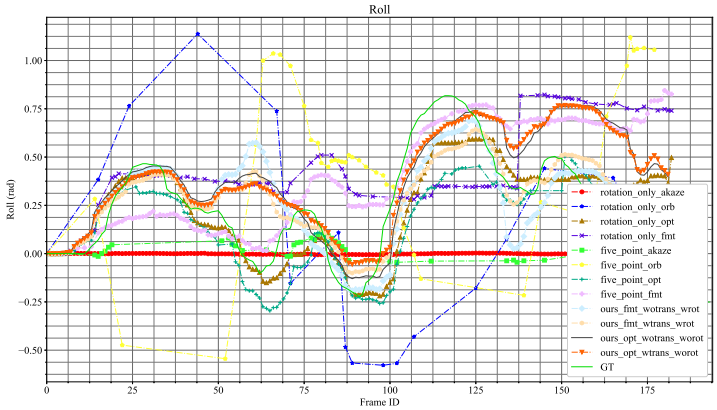
Figure A.4: Hybrid rotation estimation experiments on our datasets: indoor_rpy



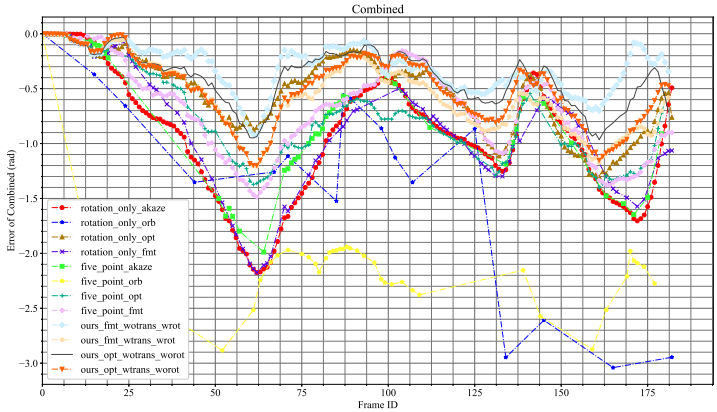
(a) Yaw



(b) Pitch

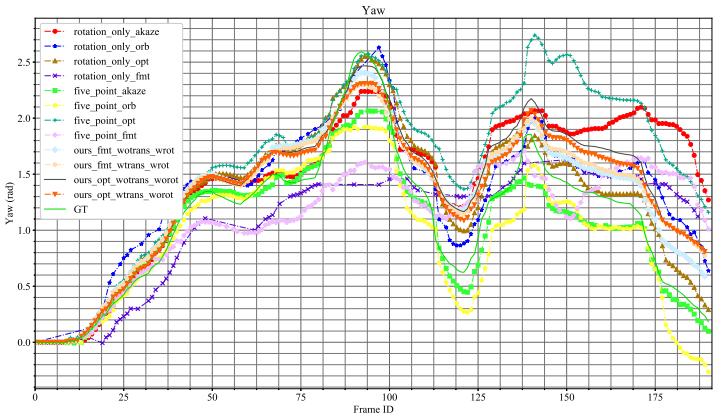


(c) Roll

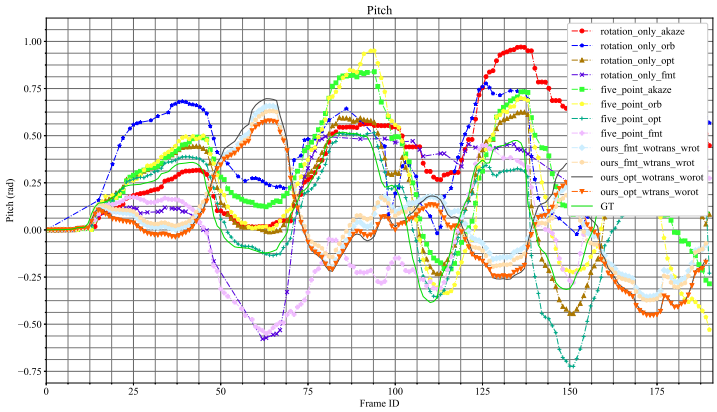


(d) Combined

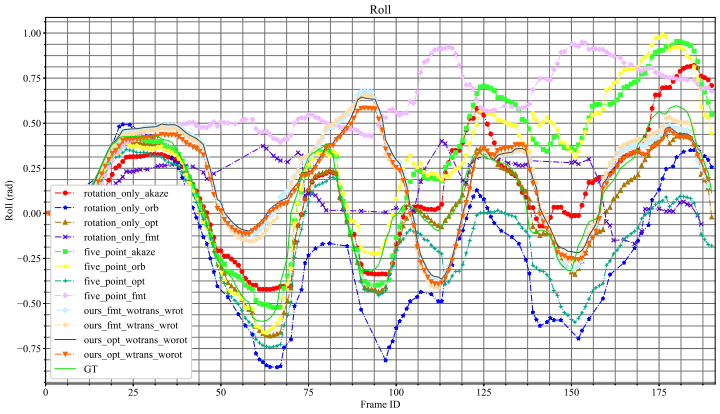
Figure A.5: Hybrid rotation estimation experiments on our datasets: grass_rpy



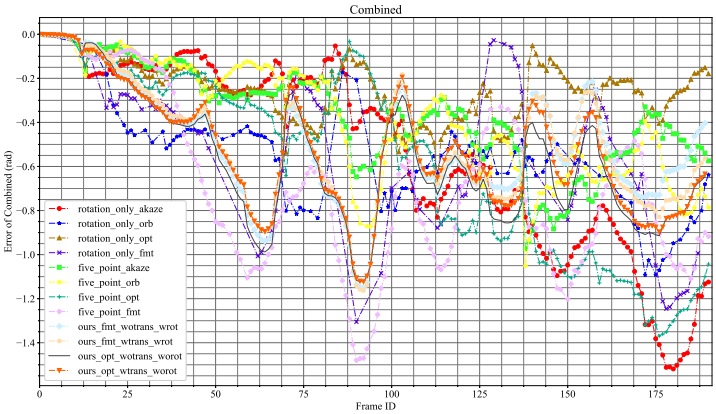
(a) Yaw



(b) Pitch

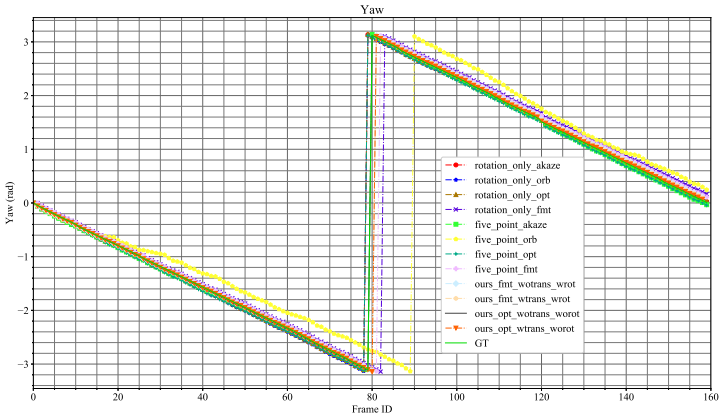


(c) Roll

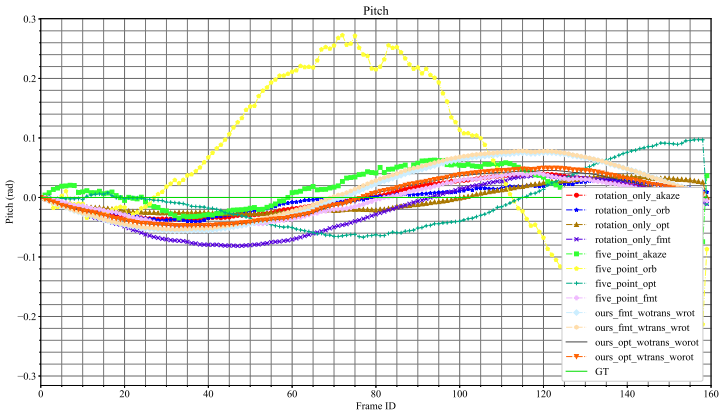


(d) Combined

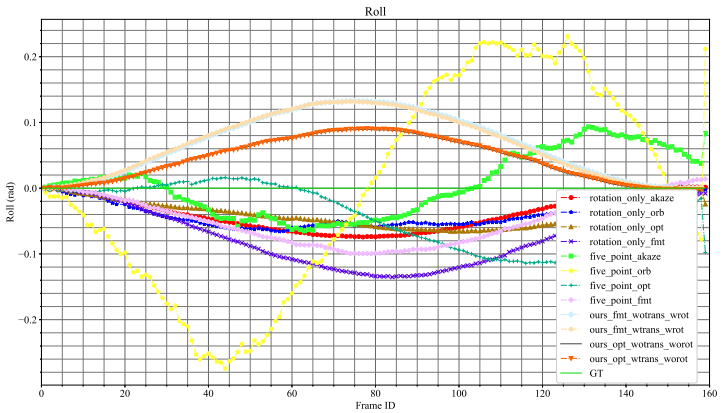
Figure A.6: Hybrid rotation estimation experiments on our datasets: street_rpy



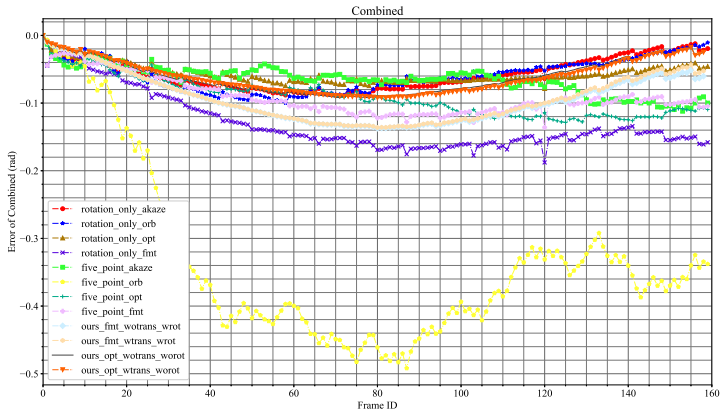
(a) Yaw



(b) Pitch

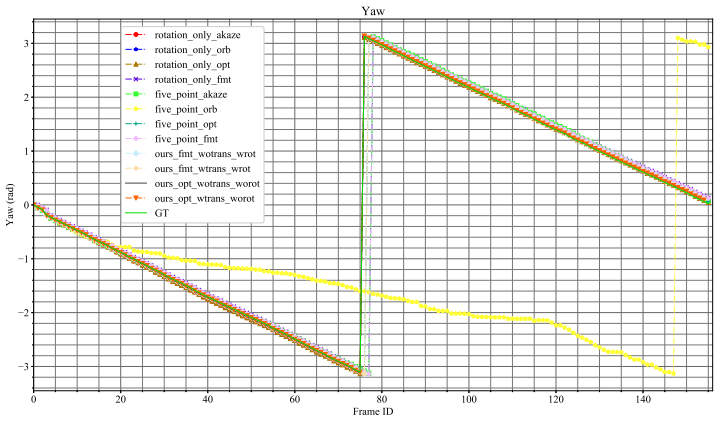


(c) Roll

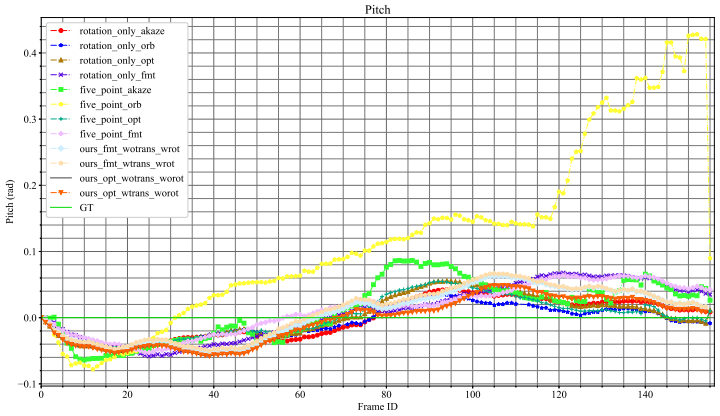


(d) Combined

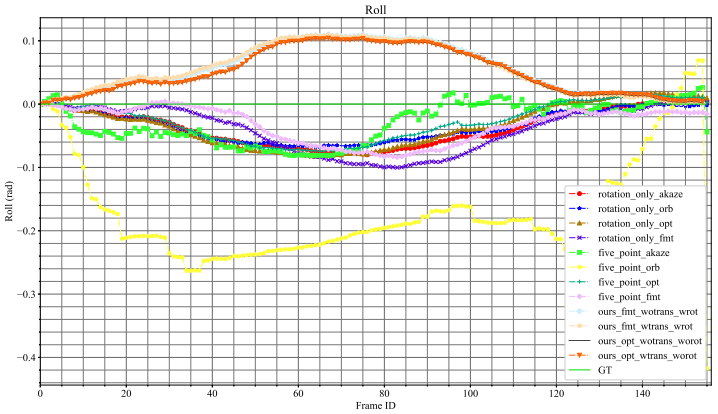
Figure A.7: Hybrid rotation estimation experiments on public datasets: *OVMIS*_1



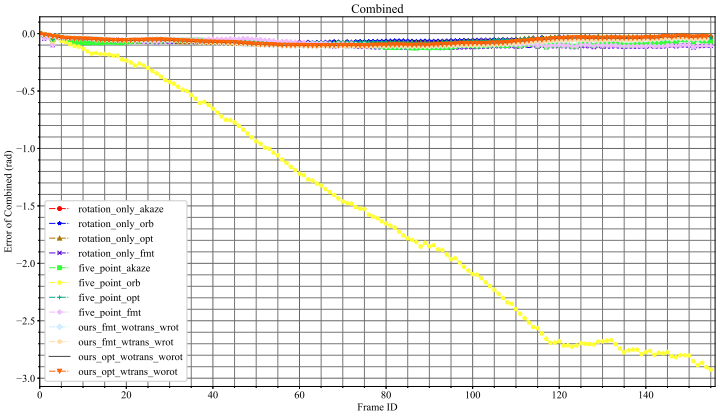
(a) Yaw



(b) Pitch

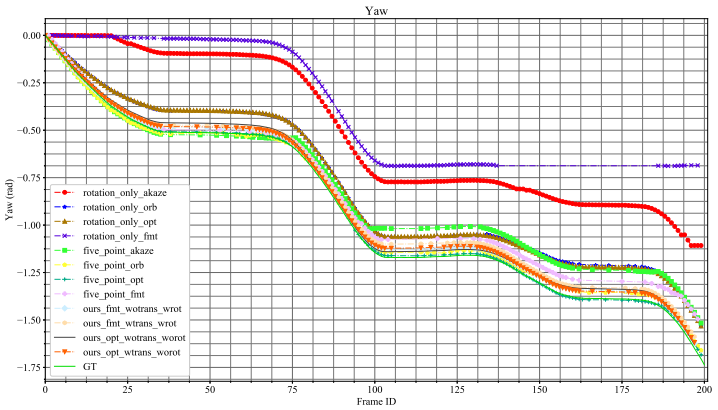


(c) Roll

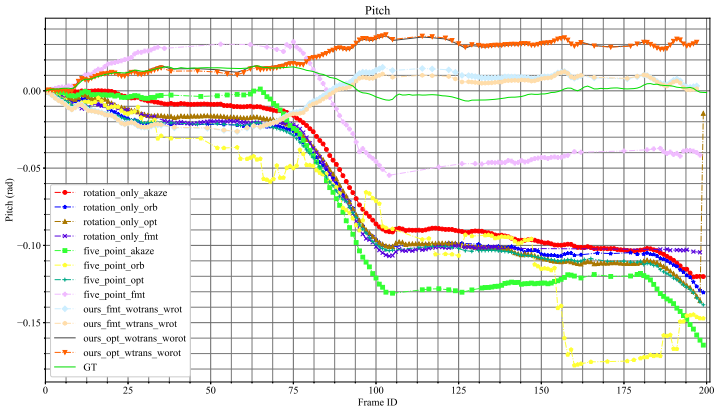


(d) Combined

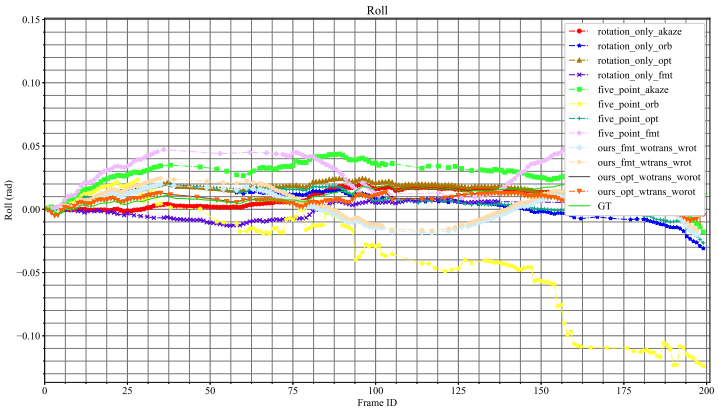
Figure A.8: Hybrid rotation estimation experiments on public datasets: *OVMIS_2*



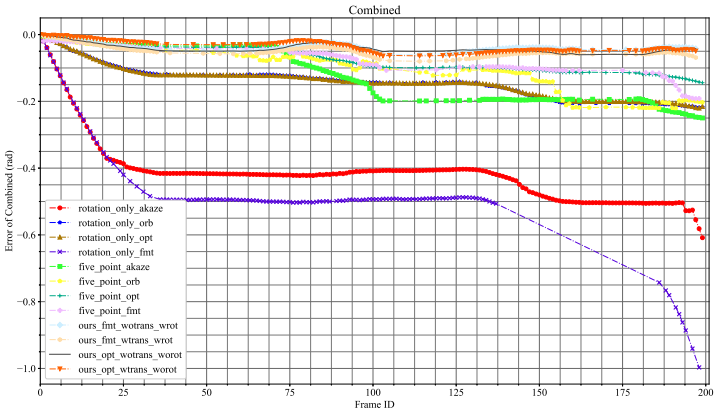
(a) Yaw



(b) Pitch



(c) Roll



(d) Combined

Figure A.9: Hybrid rotation estimation experiments on public datasets: *CVLIBS*

Bibliography

[1] R Abdelfattah and JM Nicolas. “InSAR image co-registration using the Fourier–Mellin transform”. In: *International Journal of Remote Sensing* 26.13 (2005), pp. 2865–2876.

[2] Sani M Abdullahi and Hongxia Wang. “Fourier-Mellin Transform and Fractal Coding for Secure and Robust Fingerprint Image Hashing”. In: *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE. 2018, pp. 1–7.

[3] Sameer Agarwal, Keir Mierle, and Others. *Ceres Solver*. http://ceres-solver.org.

[4] Alireza Ahmadi et al. “Visual Servoing-based Navigation for Monitoring Row-Crop Fields”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 4920–4926.

[5] Mazda Ahmadi and Peter Stone. “A multi-robot system for continuous area sweeping tasks”. In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE. 2006, pp. 1724–1729.

[6] Pablo F Alcantarilla, Jesùs Nuevo, and Adrien Bartoli. “Fast explicit diffusion for accelerated features in nonlinear scale spaces”. In: *IEEE Trans. Patt. Anal. Mach. Intell* 34.7 (2011), pp. 1281–1298.

[7] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J Davison. “KAZE features”. In: *European Conference on Computer Vision*. Springer. 2012, pp. 214–227.

[8] P. E. Anuta. “Spatial Registration of Multispectral and Multitemporal Digital Imagery Using Fast Fourier Transform Techniques”. In: *IEEE Transactions on Geoscience Electronics* 8.4 (1970), pp. 353–368. ISSN: 0018-9413. DOI: 10.1109/TGE.1970.271435.

[9] Antonis A Argyros et al. “Robot homing by exploiting panoramic vision”. In: *Autonomous Robots* 19.1 (2005), pp. 7–25.

[10] Zafer Arican and Pascal Frossard. "OmniSIFT: Scale invariant features in omnidirectional images". In: *2010 IEEE International Conference on Image Processing*. IEEE. 2010, pp. 3505–3508.

[11] Abraham Bachrach et al. "Estimation, planning, and mapping for autonomous flight using an RGB-D camera in GPS-denied environments". In: *The International Journal of Robotics Research* 31.11 (2012), pp. 1320–1343. DOI: 10.1177/0278364912455256. URL: https://journals.sagepub.com/doi/abs/10.1177/0278364912455256.

[12] Hernán Badino, Akihiro Yamamoto, and Takeo Kanade. "Visual odometry by multi-frame feature integration". In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2013, pp. 222–229.

[13] Joao P Barreto. "A unifying geometric representation for central projection systems". In: *Computer Vision and Image Understanding* 103.3 (2006), pp. 208–217.

[14] John L Barron, David J Fleet, and Steven S Beauchemin. "Performance of optical flow techniques". In: *International journal of computer vision* 12.1 (1994), pp. 43–77.

[15] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "SURF: Speeded Up Robust Features". In: *European Conference on Computer Vision (ECCV)*. Ed. by Ales Leonardis, Horst Bischof, and Axel Pinz. Vol. 3951. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2006, pp. 404–417. ISBN: 978-3-540-33832-1. DOI: 10.1007/11744023_32. URL: http://dx.doi.org/10.1007/11744023_32.

[16] Herbert Bay et al. "Speeded-Up Robust Features (SURF)". In: *Computer Vision and Image Understanding* 110.3 (2008), pp. 346–359. ISSN: 1077-3142. DOI: 10.1016/j.cviu.2007.09.014. URL: http://www.sciencedirect.com/science/article/pii/S1077314207001555.

[17] Fabio Bellavia, Marco Fanfani, and Carlo Colombo. "Selective visual odometry for accurate AUV localization". In: *Autonomous Robots* 41.1 (2017), pp. 133–143. ISSN: 1573-7527. DOI: 10.1007/s10514-015-9541-1. URL: https://doi.org/10.1007/s10514-015-9541-1.

[18] Ryad Benosman, S Kang, and Olivier Faugeras. *Panoramic vision*. Springer-Verlag New York, Berlin, Heidelberg, 2000.

[19] Andreas Birk et al. "Safety, Security, and Rescue Missions with an Unmanned Aerial Vehicle (UAV): Aerial Mosaicking and Autonomous Flight at the 2009 European Land Robots Trials (ELROB) and the 2010 Response Robot Evaluation Exercises (RREE)". In: *Journal of Intelligent and Robotic Systems* 64.1 (2011), pp. 57–76.

[20] Heiko Bülow and Andreas Birk. “Fast and robust photomapping with an unmanned aerial vehicle (uav)”. In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2009, pp. 3368–3373.

[21] Heiko Bülow and Andreas Birk. “Fast and Robust Photomapping with an Unmanned Aerial Vehicle (UAV)”. In: *International Conference on Intelligent Robots and Systems (IROS)*. IEEE Press, 2009.

[22] Heiko Bülow and Andreas Birk. “Large-scale Fourier-Mellin Invariant Registration Using a Dirichlet Based Phase Filter”. In: *under review* (2019). URL: http://robotics.jacobs-university.de/TMP/Dirichlet-FMI.pdf.

[23] Heiko Bülow and Andreas Birk. “Scale-Free Registrations in 3D: 7 Degrees of Freedom with Fourier-Mellin-SOFT transforms”. In: *International Journal of Computer Vision (IJCV)* 126.7 (2018), pp. 731–750. DOI: 10.1007/s11263-018-1067-5.2018.

[24] Heiko Bülow, Andreas Birk, and Vikram Unnithan. “Online Generation of an Underwater Photo Map with Improved Fourier Mellin based Registration”. In: *IEEE OCEANS*. IEEE Press, 2009.

[25] Heiko Bülow, Andreas Birk, and Vikram Unnithan. “Online generation of an underwater photo map with improved fourier mellin based registration”. In: *OCEANS 2009-EUROPE*. IEEE. 2009, pp. 1–6.

[26] Heiko Bülow et al. “A Divide and Conquer Method for 3D Registration of Inhomogeneous, Partially Overlapping Scans with Fourier Mellin SOFT (FMS)”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 8594–8601.

[27] Carlos Campos et al. “ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM”. In: *arXiv preprint arXiv:2007.11898* (2020).

[28] David Caruso, Jakob Engel, and Daniel Cremers. “Large-scale direct slam for omnidirectional cameras”. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 141–148.

[29] Riccardo Cassinis et al. “Unsupervised matching of visual landmarks for robotic homing using Fourier–Mellin transform”. In: *Robotics and Autonomous Systems* 40.2-3 (2002), pp. 131–138.

[30] Pierre Charbonnier et al. “Deterministic edge-preserving regularization in computed imaging”. In: *IEEE Transactions on image processing* 6.2 (1997), pp. 298–311.

[31] Paul Checchin et al. “Radar scan matching SLAM using the Fourier-Mellin transform”. In: *Field and Service Robotics*. Springer. 2010, pp. 151–161.

[32] Qin-sheng Chen, Michel Defrise, and Frank Deconinck. “Symmetric phase-only matched filtering of Fourier-Mellin transforms for image registration and recognition”. In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 12 (1994), pp. 1156–1168.

[33] Yang Cheng, M.W. Maimone, and L. Matthies. “Visual odometry on the Mars exploration rovers - a tool to ensure accurate driving and science imaging”. In: *Robotics and Automation Magazine, IEEE* 13.2 (2006), pp. 54–62. ISSN: 1070-9932 VO - 13.

[34] James W. Cooley and John W. Tukey. “An Algorithm for the Machine Calculation of Complex Fourier Series”. In: *Mathematics of Computation* 19.90 (1965), pp. 297–301. ISSN: 00255718, 10886842. DOI: 10.2307/2003354. URL: www.jstor.org/stable/2003354.

[35] P. Corke et al. “Experiments with Underwater Robot Localization and Tracking”. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. 2007, pp. 4556–4561. ISBN: 1050-4729. DOI: 10.1109/robot.2007.364181.

[36] Peter Corke, Dennis Strelow, and Sanjiv Singh. “Omnidirectional visual odometry for a planetary rover”. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*. Vol. 4. IEEE. 2004, pp. 4007–4012.

[37] Andrew J Davison et al. “MonoSLAM: Real-time single camera SLAM”. In: *IEEE transactions on pattern analysis and machine intelligence* 29.6 (2007), pp. 1052–1067.

[38] Konstantinos G Derpanis. “The harris corner detector”. In: *York University* 2 (2004).

[39] T Dhanabalan and A Sathish. “Transforming Indian industries through artificial intelligence and robotics in industry 4.0”. In: *International Journal of Mechanical Engineering and Technology* 9.10 (2018), pp. 835–845.

[40] Aubrey K Dunne, John Mallon, and Paul F Whelan. “A comparison of new generic camera calibration with the standard parametric approach”. In: (2007).

[41] Aubrey K Dunne, John Mallon, and Paul F Whelan. “Efficient Generic Calibration Method for General Cameras with Single Centre of Projection”. In: *2007 IEEE 11th International Conference on Computer Vision*. IEEE. 2007, pp. 1–8.

[42] J. Engel, V. Koltun, and D. Cremers. “Direct Sparse Odometry”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (Mar. 2018).

[43] Jakob Engel, Vladlen Koltun, and Daniel Cremers. “Direct sparse odometry”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.3 (2017), pp. 611–625.

[44] Jakob Engel, Thomas Schöps, and Daniel Cremers. “LSD-SLAM: Large-scale direct monocular SLAM”. In: *European Conference on Computer Vision*. Springer. 2014, pp. 834–849.

[45] Martin A Fischler and Robert C Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[46] Martin A. Fischler and Robert C. Bolles. *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography*. Vol. 24. Association for Computing Machinery, 1981, 381–395. ISBN: 0001-0782. DOI: 10.1145/358669.358692. URL: https://doi.org/10.1145/358669.358692.

[47] C. Forster et al. “SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems”. In: *IEEE Transactions on Robotics* 33.2 (Apr. 2017), pp. 249–265. ISSN: 1941-0468. DOI: 10.1109/TRO.2016.2623335.

[48] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. “SVO: Fast semi-direct monocular visual odometry”. In: *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2014, pp. 15–22.

[49] Friedrich Fraundorfer, Davide Scaramuzza, and Marc Pollefeys. “A constricted bundle adjustment parameterization for relative scale estimation in visual odometry”. In: *2010 IEEE International Conference on Robotics and Automation*. IEEE. 2010, pp. 1899–1904.

[50] Tarak Gandhi and Mohan Trivedi. “Parametric ego-motion estimation for vehicle surround analysis using an omnidirectional camera”. In: *Machine Vision and Applications* 16.2 (2005), pp. 85–95.

[51] Xiao-Shan Gao et al. “Complete solution classification for the perspective-three-point problem”. In: *IEEE transactions on pattern analysis and machine intelligence* 25.8 (2003), pp. 930–943.

[52] Christopher Geyer and Kostas Daniilidis. “A unifying theory for central panoramic systems and practical implications”. In: *European conference on computer vision*. Springer. 2000, pp. 445–461.

[53] Roland Goecke et al. “Visual vehicle egomotion estimation using the fourier-mellin transform”. In: *2007 IEEE Intelligent Vehicles Symposium*. IEEE. 2007, pp. 450–455.

[54] Ruchi Goel and Pooja Gupta. "Robotics and industry 4.0". In: *A Roadmap to Industry 4.0: Smart Production, Sharp Business and Sustainable Development*. Springer, 2020, pp. 157–169.

[55] Volker Grabe et al. "Nonlinear ego-motion estimation from optical flow for online control of a quadrotor UAV". In: *The International Journal of Robotics Research* 34.8 (2015), pp. 1114–1135. DOI: 10.1177/0278364915578646. URL: https://journals.sagepub.com/doi/abs/10.1177/0278364915578646.

[56] Giorgio Grisetti et al. "g2o: A general framework for (hyper) graph optimization". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China*. 2011, pp. 9–13.

[57] Vitor Guizilini and Fabio Ramos. "Semi-parametric learning for visual odometry". In: *The International Journal of Robotics Research* 32.5 (2013), pp. 526–546. DOI: 10.1177/0278364912472245. URL: https://journals.sagepub.com/doi/abs/10.1177/0278364912472245.

[58] Xiaoxin Guo et al. "An application of Fourier-Mellin transform in image registration". In: *The Fifth International Conference on Computer and Information Technology (CIT'05)*. IEEE. 2005, pp. 619–623.

[59] Peter Hansen, Peter Corke, and Wageeh Boles. "Wide-angle visual feature matching for outdoor localization". In: *The International Journal of Robotics Research* 29.2-3 (2010), pp. 267–297.

[60] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[61] Richard I Hartley. "In defense of the eight-point algorithm". In: *IEEE Transactions on pattern analysis and machine intelligence* 19.6 (1997), pp. 580–593.

[62] Ming He et al. "A review of monocular visual odometry". In: *The Visual Computer* 36.5 (2020), pp. 1053–1065.

[63] Huy Tho Ho and Roland Goecke. "Optical flow estimation using fourier mellin transform". In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2008, pp. 1–8.

[64] Berthold KP Horn and Brian G Schunck. "Determining optical flow". In: *Artificial intelligence* 17.1-3 (1981), pp. 185–203.

[65] Peter J Huber. "Robust estimation of a location parameter". In: *Breakthroughs in statistics*. Springer, 1992, pp. 492–518.

[66] N. Hurtos et al. "Evaluation of registration methods on two-dimensional forward-looking sonar imagery". In: *MTS/IEEE OCEANS*. 2013, pp. 1–8. DOI: 10.1109/OCEANS-Bergen.2013.6608124.

[67] N. Hurtos et al. “Real-time mosaicing with two-dimensional forward-looking sonar”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 601–606. ISBN: 1050-4729. DOI: 10.1109/icra.2014.6906916.

[68] Natalia Hurtos et al. “Fourier-based Registration for Robust Forward-looking Sonar Mosaicing in Low-visibility Underwater Environments”. In: *Journal of Field Robotics* 32.1 (2015), pp. 123–151. DOI: doi:10.1002/rob.21516. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21516.

[69] Katsushi Ikeuchi. *Computer vision: A reference guide*. Springer Publishing Company, Incorporated, 2014.

[70] Tim Kazik and Ali Haydar Göktoğan. “Visual odometry based on the Fourier-Mellin transform for a rover using a monocular ground-facing camera”. In: *2011 IEEE International Conference on Mechatronics*. IEEE. 2011, pp. 469–474.

[71] Georg Klein and David Murray. “Parallel tracking and mapping for small AR workspaces”. In: *2007 6th IEEE and ACM international symposium on mixed and augmented reality*. IEEE. 2007, pp. 225–234.

[72] Laurent Kneip and Paul Furgale. “OpenGV: A unified and generalized approach to real-time calibrated geometric vision”. In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE. 2014, pp. 1–8.

[73] Laurent Kneip, Davide Scaramuzza, and Roland Siegwart. “A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation”. In: *CVPR 2011*. IEEE. 2011, pp. 2969–2976.

[74] Kurt Konolige, Motilal Agrawal, and Joan Sola. “Large-scale visual odometry for rough terrain”. In: *Robotics research*. Springer, 2010, pp. 201–212.

[75] Haofei Kuang et al. “Pose Estimation for Omni-directional Cameras using Sinusoid Fitting”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Nov. 2019, pp. 900–906. DOI: 10.1109/IROS40897.2019.8968087.

[76] Frédéric Labrosse. “The visual compass: Performance and limitations of an appearance-based method”. In: *Journal of Field Robotics* 23.10 (2006), pp. 913–941.

[77] Thomas Lemaire and Simon Lacroix. “SLAM with panoramic vision”. In: *Journal of Field Robotics* 24.1-2 (2007), pp. 91–111.

[78] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. “Epnp: An accurate o (n) solution to the pnp problem”. In: *International journal of computer vision* 81.2 (2009), p. 155.

[79] Stefan Leutenegger et al. “Keyframe-based visual–inertial odometry using nonlinear optimization”. In: *The International Journal of Robotics Research* 34.3 (2015), pp. 314–334.

[80] Maxime Lhuillier. “Automatic structure and motion using a catadioptric camera”. In: *Proceedings of the 6th Workshop on Omnidirectional Vision, Camera Networks and Non-Classical Cameras*. 2005.

[81] Shih-Schon Lin and Ruzena Bajcsy. “High resolution catadioptric omni-directional stereo sensor for robot vision”. In: *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*. Vol. 2. IEEE. 2003, pp. 1694–1699.

[82] Almudena Lindoso et al. “Correlation-based fingerprint matching using FPGAs”. In: *Proceedings. 2005 IEEE International Conference on Field-Programmable Technology, 2005*. IEEE. 2005, pp. 87–94.

[83] David G. Lowe. “Object recognition from local scale-invariant features”. In: *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*. Vol. 2. 1999, 1150–1157 vol.2. DOI: 10.1109/iccv.1999.790410.

[84] D.G. Lowe. “Local feature view clustering for 3D object recognition”. In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Vol. 1. 2001, I–682–I–688 vol.1. ISBN: 1063-6919 VO - 1. URL: 10.1109/CVPR.2001.990541.

[85] Luca Lucchese. “Estimating affine transformations in the frequency domain”. In: *Proceedings 2001 International Conference on Image Processing (Cat. No. 01CH37205)*. Vol. 2. IEEE. 2001, pp. 909–912.

[86] Mark Maimone, Yang Cheng, and Larry Matthies. “Two years of visual odometry on the mars exploration rovers”. In: *Journal of Field Robotics* 24.3 (2007), pp. 169–186.

[87] Donald W Marquardt. “An algorithm for least-squares estimation of nonlinear parameters”. In: *Journal of the society for Industrial and Applied Mathematics* 11.2 (1963), pp. 431–441.

[88] Hidenobu Matsuki et al. “Omnidirectional DSO: Direct sparse odometry with fisheye cameras”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3693–3700.

[89] Markus Maurer et al. *Autonomous driving: technical, legal and social aspects*. Springer Nature, 2016.

[90] Christopher Mei and Patrick Rives. “Single view point omnidirectional camera calibration from planar grids”. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE. 2007, pp. 3945–3950.

[91] Branislav Micusik and Tomas Pajdla. “Structure from motion with wide circular field of view cameras”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.7 (2006), pp. 1135–1149.

[92] Michael J Milford and Gordon F Wyeth. “Single camera vision-only SLAM on a suburban road network”. In: *2008 IEEE International Conference on Robotics and Automation*. IEEE. 2008, pp. 3684–3689.

[93] Fabio Morbidi and Guillaume Caron. “Phase Correlation for Dense Visual Compass from Omnidirectional Camera-Robot Images”. In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 688–695.

[94] Marius Muja and David G Lowe. “Fast approximate nearest neighbors with automatic algorithm configuration.” In: *VISAPP (1)* 2.331-340 (2009), p. 2.

[95] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. “ORB-SLAM: a versatile and accurate monocular SLAM system”. In: *IEEE transactions on robotics* 31.5 (2015), pp. 1147–1163.

[96] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. “DTAM: Dense tracking and mapping in real-time”. In: *2011 international conference on computer vision*. IEEE. 2011, pp. 2320–2327.

[97] D. Nister. “An efficient solution to the five-point relative pose problem”. In: *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.* Vol. 2. 2003, pp. II–195. ISBN: 1063-6919. DOI: 10.1109/CVPR.2003.1211470.

[98] D. Nister. “An efficient solution to the five-point relative pose problem”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.6 (2004), pp. 756–770. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2004.17.

[99] D. Nister, O. Naroditsky, and J. Bergen. “Visual odometry”. In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Vol. 1. 2004, pp. I–I. ISBN: 1063-6919. DOI: 10.1109/CVPR.2004.1315094.

[100] David Nistér, Oleg Naroditsky, and James Bergen. “Visual odometry for ground vehicle applications”. In: *Journal of Field Robotics* 23.1 (2006), pp. 3–20.

[101] David Nister and Henrik Stewenius. “Using Algebraic Geometry for Solving Polynomial Problems in Computer Vision”. In: *tutorial, IEEE International Conference on Computer Vision (ICCV)*. 2005, p. 283.

[102] Alvaro Ordonez, Francisco Argüello, and Dora B Heras. “Fourier–Mellin registration of two hyperspectral images”. In: *International Journal of Remote Sensing* 38.11 (2017), pp. 3253–3273.

[103] Álvaro Ordóñez, Francisco Argüello, and Dora B Heras. "GPU accelerated FFT-based registration of hyperspectral scenes". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 10.11 (2017), pp. 4869–4878.

[104] Kaustubh Pathak et al. "Robust estimation of camera-tilt for iFMI based underwater photo-mapping using a calibrated monocular camera". In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE. 2013, pp. 5111–5116.

[105] Luis Payá et al. "Using Omnidirectional Vision to Create a Model of the Environment: A Comparative Evaluation of Global-Appearance Descriptors". In: *Journal of Sensors* 2016 (2016), pp. 1–21.

[106] Xin Peng, Jiadi Cui, and Laurent Kneip. "Articulated multi-perspective cameras and their application to truck motion estimation". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 2052–2059.

[107] Valentin Peretroukhin, Lee Clement, and Jonathan Kelly. "Inferring sun direction to improve visual odometry: A deep learning approach". In: *The International Journal of Robotics Research* 37.9 (2018), pp. 996–1016. DOI: 10.1177/0278364917749732. URL: https://journals.sagepub.com/doi/abs/10.1177/0278364917749732.

[108] M. Pfingsthorn et al. "Large-Scale Mosaicking with Spectral Registration based Simultaneous Localization and Mapping (iFMI-SLAM) in the Ligurian Sea". In: *IEEE Oceans*. IEEE Press, 2013.

[109] Srikumar Ramalingam and Peter Sturm. "A unifying model for camera calibration". In: *IEEE transactions on pattern analysis and machine intelligence* 39.7 (2016), pp. 1309–1319.

[110] B Srinivasa Reddy and Biswanath N Chatterji. "An FFT-based technique for translation, rotation, and scale-invariant image registration". In: *IEEE transactions on image processing* 5.8 (1996), pp. 1266–1271.

[111] B.S. Reddy and B.N. Chatterji. "An FFT-based technique for translation, rotation, and scale-invariant image registration". In: *Image Processing, IEEE Transactions on* 5.8 (1996), pp. 1266–1271. ISSN: 1057-7149 VO - 5.

[112] Alejandro Rituerto, Luis Puig, and José Jesús Guerrero. "Visual slam with an omnidirectional camera". In: *2010 20th International Conference on Pattern Recognition*. IEEE. 2010, pp. 348–351.

[113] Maria Romero et al. “Vineyard water status estimation using multispectral imagery from an UAV platform and machine learning algorithms for irrigation scheduling management”. In: *Computers and electronics in agriculture* 147 (2018), pp. 109–117.

[114] Cristina Romero-Trigueros et al. “Effects of saline reclaimed waters and deficit irrigation on Citrus physiology assessed by UAV remote sensing”. In: *Agricultural water management* 183 (2017), pp. 60–69.

[115] Edward Rosten and Tom Drummond. “Machine learning for high-speed corner detection”. In: *European conference on computer vision*. Springer. 2006, pp. 430–443.

[116] Ethan Rublee et al. “ORB: An efficient alternative to SIFT or SURF”. In: *Computer Vision (ICCV), 2011 IEEE international conference on*. IEEE. 2011, pp. 2564–2571.

[117] D. Scaramuzza and F. Fraundorfer. “Visual Odometry - Part I: The First 30 Years and Fundamentals”. In: *Robotics and Automation Magazine (RAM), IEEE* 18.4 (2011), pp. 80–92. ISSN: 1070-9932. DOI: 10.1109/mra.2011.943233.

[118] Davide Scaramuzza. “Omnidirectional camera”. In: *Computer Vision: A Reference Guide* (2014), pp. 552–560.

[119] Davide Scaramuzza and Friedrich Fraundorfer. “Visual odometry [tutorial]”. In: *IEEE robotics & automation magazine* 18.4 (2011), pp. 80–92.

[120] Davide Scaramuzza, Friedrich Fraundorfer, and Roland Siegwart. “Real-time monocular visual odometry for on-road vehicles with 1-point ransac”. In: *2009 IEEE International conference on robotics and automation*. Ieee. 2009, pp. 4293–4299.

[121] Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart. “A flexible technique for accurate omnidirectional camera calibration and structure from motion”. In: *Computer Vision Systems, 2006 ICVS'06. IEEE International Conference on*. IEEE. 2006, pp. 45–45.

[122] Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart. “A toolbox for easily calibrating omnidirectional cameras”. In: *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE. 2006, pp. 5695–5701.

[123] Davide Scaramuzza and Roland Siegwart. “Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles”. In: *IEEE transactions on robotics* 24.5 (2008), pp. 1015–1026.

[124] Miriam Schönbein and Andreas Geiger. “Omnidirectional 3D Reconstruction in Augmented Manhattan Worlds”. In: *International Conference on Intelligent Robots and Systems (IROS)*. 2014.

[125] Miriam Schönbein, Tobias Strauss, and Andreas Geiger. "Calibrating and Centering Quasi-Central Catadioptric Cameras". In: *International Conference on Robotics and Automation (ICRA)*. 2014.

[126] Sören Schwertfeger, Andreas Birk, and Heiko Bülow. "Using iFMI spectral registration for video stabilization and motion detection by an Unmanned Aerial Vehicle (UAV)". In: *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*. IEEE. 2011, pp. 61–67.

[127] Sören Schwertfeger, Heiko Bülow, and Andreas Birk. "On the effects of sampling resolution in improved fourier mellin based registration for underwater mapping". In: *IFAC Proceedings Volumes* 43.16 (2010), pp. 617–622.

[128] Stephen Se, David Lowe, and Jim Little. "Vision-based mobile robot localization and mapping using scale-invariant features". In: *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*. Vol. 2. IEEE. 2001, pp. 2051–2058.

[129] Hochang Seok and Jongwoo Lim. "Rovo: Robust omnidirectional visual odometry for wide-baseline wide-fov camera systems". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 6344–6350.

[130] Jianbo Shi et al. "Good features to track". In: *1994 Proceedings of IEEE conference on computer vision and pattern recognition*. IEEE. 1994, pp. 593–600.

[131] Josef Sivic and Andrew Zisserman. "Video Google: A text retrieval approach to object matching in videos". In: *Computer Vision, IEEE International Conference on*. Vol. 3. IEEE Computer Society. 2003, pp. 1470–1470.

[132] Stephen M. Smith and J. Michael Brady. "SUSAN - A New Approach to Low Level Image Processing". In: *International Journal of Computer Vision* 23.1 (1997), pp. 45–78. ISSN: 1573-1405. DOI: 10.1023/a:1007963824710. URL: http://dx.doi.org/10.1023/A:1007963824710.

[133] Shiyu Song, Manmohan Chandraker, and Clark C Guest. "Parallel, real-time monocular visual odometry". In: *2013 ieee international conference on robotics and automation*. IEEE. 2013, pp. 4698–4705.

[134] Henrik Stewénius et al. "A minimal solution for relative pose with unknown focal length". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 2. IEEE. 2005, pp. 789–794.

[135] Biljana L Risteska Stojkoska and Kire V Trivodaliev. "A review of Internet of Things for smart home: Challenges and solutions". In: *Journal of Cleaner Production* 140 (2017), pp. 1454–1464.

[136] Peter Sturm and Srikumar Ramalingam. "A generic concept for camera calibration". In: *European Conference on Computer Vision*. Springer. 2004, pp. 1–13.

[137] Peter Sturm and Srikumar Ramalingam. *Camera models and fundamental concepts used in geometric computer vision*. Now Publishers Inc, 2011.

[138] Niko Sünderhauf et al. "Visual odometry using sparse bundle adjustment on an autonomous outdoor vehicle". In: *Autonome Mobile Systeme 2005*. Springer, 2006, pp. 157–163.

[139] Wei Tan et al. "Robust monocular SLAM in dynamic environments". In: *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE. 2013, pp. 209–218.

[140] Jean-Philippe Tardif, Yanis Pavlidis, and Kostas Daniilidis. "Monocular visual odometry in urban environments using an omnidirectional camera". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2008, pp. 2531–2538.

[141] Shaharyar Ahmed Khan Tareen and Zahra Saleem. "A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK". In: *Computing, Mathematics and Engineering Technologies (iCoMET), 2018 International Conference on*. IEEE. 2018, pp. 1–10.

[142] X. Tong et al. "Image Registration With Fourier-Based Image Correlation: A Comprehensive Review of Developments and Applications". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 12.10 (2019), pp. 4062–4081. ISSN: 2151-1535. DOI: 10.1109/JSTARS.2019.2937690.

[143] Steffen Urban, Jens Leitloff, and Stefan Hinz. "Improved wide-angle, fisheye and omnidirectional camera calibration". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 108 (2015), pp. 72–79.

[144] Steffen Urban et al. "MultiCol Bundle Adjustment: A Generic Method for Pose Estimation, Simultaneous Self-Calibration and Reconstruction for Arbitrary Multi-Camera Systems". In: *International Journal of Computer Vision* (2016), pp. 1–19.

[145] Michael Warren, Peter Corke, and Ben Upcroft. "Long-range stereo visual odometry for extended altitude flight of unmanned aerial vehicles". In: *The International Journal of Robotics Research* 35.4 (2016), pp. 381–403. DOI: 10.1177/0278364915581194. URL: https://journals.sagepub.com/doi/abs/10.1177/0278364915581194.

[146] Niall Winters et al. “Omni-directional vision for robot navigation”. In: *Omni-directional Vision, 2000. Proceedings. IEEE Workshop on*. IEEE. 2000, pp. 21–28.

[147] Xiaolong Wu et al. “Robotic weed control using automated weed and crop classification”. In: *Journal of Field Robotics* 37.2 (2020), pp. 322–340.

[148] Qingwen Xu et al. “Improved Fourier Mellin Invariant for Robust Rotation Estimation with Omni-cameras”. In: *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2019, pp. 320–324.

[149] Qingwen Xu et al. “Improved Fourier Mellin Invariant for Robust Rotation Estimation with Omni-Cameras”. In: *2019 IEEE International Conference on Image Processing (ICIP)* (Sept. 2019). DOI: 10.1109/icip.2019.8802933. URL: http://dx.doi.org/10.1109/ICIP.2019.8802933.

[150] Z Ye et al. “Precise Disparity Estimation for Narrow Baseline Stereo Based on Multiscale Superpixels and Phase Correlation.” In: *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences* (2019).

[151] Xianghua Ying and Zhanyi Hu. “Can we consider central catadioptric cameras and fisheye cameras within a unified imaging model”. In: *European Conference on Computer Vision*. Springer. 2004, pp. 442–455.

[152] Zichao Zhang and Davide Scaramuzza. “A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry”. In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2018.

Curriculum Vitae

Education

Southeast University — Aug. 2011 - Jul. 2015
B.E., Information Engineering

ShanghaiTech University — Sep. 2015 - Jul. 2021
University of Chinese Academy of Sciences — GPA: 3.77/4
Ph.D., Information Engineering

List of Publications

Journal Paper

1. **Q. Xu**, H. Kuang, L. Kneip, and S.Schwertfeger. Rethinking the Fourier-Mellin Transform: Multiple Depths in the Camera's View. *Remote Sensing* 13, no. 5 (2021): 1000.
2. **Q. Xu**, H. Blow, A. Birk, and S.Schwertfeger. 3D Visual Odometry based on 2.5D Spectral Registration of Omnidirectional 2D Images. *Robotics and Autonomous Systems*, under review.
3. **Q. Xu**, X. Long, H. Kuang, and S. Schwertfeger. Rotation Estimation for Omnidirectional Cameras using Sinusoid Fitting. *Journal of Intelligent & Robotic Systems*, minor revision.

Conference Paper

†: equal contribution

1. Y.Yuan, **Q. Xu**, and S. Schwertfeger. Configuration-Space Flipper Planning on 3D Terrain. *IEEE International Symposium on Safety, Security, Rescue Robotics (SSRR)*, Khalifa University, Abu Dhabi, UAE (Virtual), pp. 318-325. 2020.
2. **Q. Xu**†, Z. He†, Z. Chen, and Y. Jiang. An Optical Flow Based Multi-Object

Tracking Approach Using Sequential Convex Programming. *16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, Shenzhen, China (Virtual), pp. 1216-1221. 2020.

3. X. Long†, **Q. Xu**†, Y. Yuan, Z. He, and S. Schwertfeger. Improved Visual-Inertial Localization for Low-cost Rescue Robots. *21st World Congress of the International Federation of Automatic Control (IFAC)*, Berlin, Germany (Virtual). 2020.
4. H. Kuang, **Q. Xu**, X. Long, and S. Schwertfeger. Pose Estimation for Omni-directional Cameras using Sinusoid Fitting. *EEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Macau, China. 2019: 900–906.
5. A. G. Chavez, **Q. Xu**, C. A. Mueller, S. Schwertfeger, and A. Birk. Adaptive Navigation Scheme for Optimal Deep-Sea Localization Using Multimodal Perception Cues. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Macau, China. 2019: 7211–7218.
6. **Q. Xu**, A. G. Chavez, H. Blow, A. Birk, and S.Schwertfeger. Improved Fourier Mellin Invariant for Robust Rotation Estimation with Omni-cameras. *2019 26th IEEE International Conference on Image Processing (ICIP)*, Taiwan, China. 2019: 320–324.

Patents

1. S. Schwertfeger, **Q. Xu**, X. Long, and H. Kuang. Rotation Estimation for Omni-directional Cameras Using Sinusoid Fitting. CN111354044A, in process.
2. **Q. Xu**, and S. Schwertfeger. An Extended Fourier-Mellin Transform Method for Multi-depth Scenarios. CN111951318A, in process.