



University of Pennsylvania
ScholarlyCommons


Publicly Accessible Penn Dissertations

2022

Visual-Inertial State Estimation With Information Deficiency

Wenxin Liu
University of Pennsylvania

Follow this and additional works at: <https://repository.upenn.edu/edissertations>

 Part of the [Artificial Intelligence and Robotics Commons](#), [Robotics Commons](#), and the [Systems Science Commons](#)

Recommended Citation

Liu, Wenxin, "Visual-Inertial State Estimation With Information Deficiency" (2022). *Publicly Accessible Penn Dissertations*. 5534.
<https://repository.upenn.edu/edissertations/5534>

This paper is posted at ScholarlyCommons. <https://repository.upenn.edu/edissertations/5534>
For more information, please contact repository@pobox.upenn.edu.

Visual-Inertial State Estimation With Information Deficiency

Abstract

State estimation is an essential part of intelligent navigation and mapping systems where tracking the location of a smartphone, car, robot, or a human-worn device is required. For autonomous systems such as micro aerial vehicles and self-driving cars, it is a prerequisite for control and motion planning. For AR/VR applications, it is the first step to image rendering. Visual-inertial odometry (VIO) is the de-facto standard algorithm for embedded platforms because it lends itself to lightweight sensors and processors, and maturity in research and industrial development. Various approaches have been proposed to achieve accurate real-time tracking, and numerous open-source software and datasets are available. However, errors and outliers are common due to the complexity of visual measurement processes and environmental changes, and in practice, estimation drift is inevitable.

In this thesis, we introduce the concept of information deficiency in state estimation and how to utilize this concept to develop and improve VIO systems. We look into the information deficiencies in visual-inertial state estimation, which are often present and ignored, causing system failures and drift. In particular, we investigate three critical cases of information deficiency in visual-inertial odometry: low texture environment with limited computation, monocular visual odometry, and inertial odometry. We consider these systems under three specific application settings: a lightweight quadrotor platform in autonomous flight, driving scenarios, and AR/VR headset for pedestrians. We address the challenges in each application setting and explore how the tight fusion of deep learning and model-based VIO can improve the state-of-the-art system performance and compensate for the lack of information in real-time. We identify deep learning as a key technology in tackling the information deficiencies in state estimation. We argue that developing hybrid frameworks that leverage its advantage and enable supervision for performance guarantee provides the most accurate and robust solution to state estimation.

Degree Type

Dissertation

Degree Name

Doctor of Philosophy (PhD)

Graduate Group

Computer and Information Science

First Advisor

Vijay Kumar

Second Advisor

Kostas Daniilidis

Keywords

Information deficiency, Micro aerial vehicles, Neural networks, State estimation, Uncertainty estimation, Visual-inertial odometry

Subject Categories

Artificial Intelligence and Robotics | Robotics | Systems Science

VISUAL-INERTIAL STATE ESTIMATION WITH INFORMATION DEFICIENCY

Wenxin Liu

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2022



Supervisor of Dissertation

Vijay Kumar

Professor, Nemirovsky Family Dean,
Computer and Information Science,
Electrical and Systems Engineering,
Mechanical Engineering and Applied
Mechanics



Co-Supervisor of Dissertation

Kostas Daniilidis

Ruth Yalom Stone Professor, Computer
and Information Science

Graduate Group Chairperson

Mayur Naik, Professor, Computer and Information Science

Dissertation Committee

Camillo Jose Taylor, Raymond S. Markowitz President's Distinguished Professor, Computer and
Information Science

Jianbo Shi, Professor, Computer and Information Science

Dinesh Jayaraman, Assistant Professor, Computer and Information Science, Electrical and Systems
Engineering

Davide Scaramuzza, Professor, Robotics and Perception Group, University of Zurich

Giuseppe Loianno, Assistant Professor, Agile Robotics and Perception Lab, New York University

VISUAL-INERTIAL STATE ESTIMATION WITH INFORMATION DEFICIENCY

COPYRIGHT

2022

Wenxin Liu

ACKNOWLEDGEMENT

I would like to first and foremost express my deepest appreciation to my advisors, Vijay Kumar and Kostas Daniilidis, for their unwavering support and constructive criticism through my research projects and the countless opportunities they have given me throughout my Ph.D. I would like to thank Giuseppe Loianno for guiding me on my first Ph.D. project in visual-inertial odometry and quadrotors. The challenges and the fun that I had led me to continue in this field. I would like to thank Jakob Engel, my manager during my internship, for supporting me on the TLIO project. The findings of this work inspired me to find my way to this thesis topic. I would like to thank all my collaborators who have provided amazing help and discussions on the projects. I learned tremendously from working with people with different expertise, and I want to acknowledge their contributions to the projects included in this thesis. Finally, I want to thank my lab mates, who have always been there to cheer me on and lend a helping hand when in need.

This thesis would not have been possible without the presence of these precious people in my academic life. I am incredibly fortunate to have the honor to work with so many talented and caring people. I am also truly blessed to have GRASP Lab as my home. It has been an extraordinary six-year journey, and I am deeply grateful for every help I have received.

ABSTRACT

VISUAL-INERTIAL STATE ESTIMATION WITH INFORMATION DEFICIENCY

Wenxin Liu

Vijay Kumar

Kostas Daniilidis

State estimation is an essential part of intelligent navigation and mapping systems where tracking the location of a smartphone, car, robot, or a human-worn device is required. For autonomous systems such as micro aerial vehicles and self-driving cars, it is a prerequisite for control and motion planning. For AR/VR applications, it is the first step to image rendering. Visual-inertial odometry (VIO) is the de-facto standard algorithm for embedded platforms because it lends itself to lightweight sensors and processors, and maturity in research and industrial development. Various approaches have been proposed to achieve accurate real-time tracking, and numerous open-source software and datasets are available. However, errors and outliers are common due to the complexity of visual measurement processes and environmental changes, and in practice, estimation drift is inevitable.

In this thesis, we introduce the concept of information deficiency in state estimation and how to utilize this concept to develop and improve VIO systems. We look into the information deficiencies in visual-inertial state estimation, which are often present and ignored, causing system failures and drift. In particular, we investigate three critical cases of information deficiency in visual-inertial odometry: low texture environment with limited computation, monocular visual odometry, and inertial odometry. We consider these systems under three specific application settings: a lightweight quadrotor platform in autonomous flight, driving scenarios, and AR/VR headset for pedestrians. We address the challenges in each application setting and explore how the tight fusion of deep learning and model-based VIO can improve the state-of-the-art system performance and compensate for the lack of information in real-time. We identify deep learning as a key technology in tackling the information deficiencies in state estimation. We argue that developing hybrid frameworks that leverage its

advantage and enable supervision for performance guarantee provides the most accurate and robust solution to state estimation.

TABLE OF CONTENTS

ACKNOWLEDGEMENT	iii
ABSTRACT	iv
LIST OF TABLES	viii
LIST OF ILLUSTRATIONS	x
CHAPTER 1 : Introduction	1
1.1 Visual-Inertial State Estimation	2
1.2 Information Deficiency and Machine Learning	9
CHAPTER 2 : Visual-Inertial State Estimation	15
2.1 Classifications and State of the Art	15
2.2 Basics	18
2.3 Deep Learning with Uncertainty Estimates	30
CHAPTER 3 : Semi-Dense VIO on Constrained Platforms	42
3.1 Introduction	43
3.2 Related Works	46
3.3 System Overview	47
3.4 Semi-Dense Direct Visual-Inertial Odometry	49
3.5 Implementation and Runtime Analysis	57
3.6 System Performance and Benchmark Evaluation	61
3.7 Low-Texture Scenarios	66
3.8 Summary	70
CHAPTER 4 : Learning Egomotion in a Hybrid Structure	74

4.1	Introduction	74
4.2	Related Works	76
4.3	Method	77
4.4	Performance Evaluation	83
4.5	Discussion and Insights	89
CHAPTER 5 : Tight-Learned Inertial Odometry		92
5.1	Introduction	92
5.2	Related Works	95
5.3	System Design	96
5.4	Neural Statistical Motion Model	98
5.5	Stochastic Cloning Extended Kalman Filter	100
5.6	Experimental Results	106
5.7	Components and Variation Studies	110
5.8	Conclusion and Insights	114
CHAPTER 6 : Conclusion		116
6.1	Research Summary	116
6.2	Contribution and Insights	117
6.3	Future Directions	121
APPENDIX		125
BIBLIOGRAPHY		128

LIST OF TABLES

TABLE 3.1	Time statistics on our platform with the selected settings for keyframes and non-keyframes, as well as the major threads in keyframe processing. Having the processing time below 100 ms allows us to run the system at 10 Hz in real time for flight. Note that the times for keyframe processing don't add up since they are running on separate threads.	61
TABLE 3.2	List of state-of-the-art stereo VIO algorithms we compare with on EuRoC Dataset and their categories.	62
TABLE 4.1	Comparison of model size.	83
TABLE 4.2	Evaluation against competing instantaneous unsupervised pose learning methods, as well as DVSO (Yang et al., 2018a), which uses a windowed optimization. Translation $t_{rel}(\%)$ and rotation $r_{rel}(^{\circ}/100\text{ m})$ RMSE on the 11 KITTI Odometry training sequences is presented. Bold indicates best instantaneous result for each column. Final means are computed over the training set (sequences 09 and 10) and all sequences, respectively. As SFM-Learner (Zhou et al., 2017) and Zhan et al. (2018) are monocular methods, their results have been corrected for scale. Only results for 09 and 10 are provided in Zhan et al. (2018) and Luo et al. (2018).	87
TABLE 4.3	Pose ablation study on the effect of the proposed refinement losses with models trained with and without the proposed refinement losses. Models were trained on sequences 00-08.	88
TABLE 4.4	Depth evaluation results on the Eigen (50 m cap) test split and KITTI stereo 2015. Left: lower is better, right: higher is better.	88
TABLE 4.5	Comparison and ablation of flow prediction on the KITTI Flow 2015 dataset. noc = occluded pixels removed. A pixel is considered an outlier if its EPE is higher than 3 pixels and 5% of its true magnitude.	89
TABLE 4.6	Evaluation of inlier quality with flow errors on the KITTI Flow 2015 dataset. Errors are computed only over inlier pixels, $\sim 21\%$ of pixels.	89

TABLE 4.7	Evaluation of inlier quality with depth errors from the KITTI Stereo 2015 dataset, without the crop from Garg et al. (2016). The first row is computed over all pixels, while the last two are computed only over inlier pixels, $\sim 35\%$ of pixels.	91
-----------	---	----

LIST OF ILLUSTRATIONS

FIGURE 1.1	Illustration of a basic quadrotor navigation system and the relationships between system components. State estimation and mapping make use of sensory data to localize the robot and create a map of the environment. They are closely related to each other - the accuracy of the pose estimates depends on the accuracy of the map, and the map estimates are tied to the poses. The path planning component designs a trajectory to a goal location based on the estimated map and the robot pose, and the controller calculates motor commands that align the robot pose with the desired trajectory.	2
FIGURE 1.2	System diagram of ORB-SLAM3 (Campos et al., 2021).	8
FIGURE 2.1	Difference between a loosely-coupled (Liu et al., 2018) and a tightly-coupled (Liu et al., 2021b) VIO system.	16
FIGURE 3.1	The semi-dense map created by the vehicle while following a square trajectory. The picture on top shows a snapshot of the environment. The carpet on the ground, the chair, and the cone to the right can be clearly seen in the semi-dense map.	44
FIGURE 3.2	Factor graph of the system. The solid square represents the latest image keyframe, and the solid circle represents the frame adjacent to the current frame. The system solves for the state of the current frame, with an image alignment factor (red) to the keyframe and an IMU pre-integration factor (yellow) to the last frame.	49
FIGURE 3.3	The left image shows the grayscale image, and the right image shows only the high gradient pixels from feature extraction.	52
FIGURE 3.4	Coordinate system illustration. This figure shows the initial body frame as the world frame w , pose of the last keyframe kf and the current body frame b . Blue and green frames represent body and camera frames respectively, and they are rigidly attached with relative transformation bT_c in $SE(3)$	53

FIGURE 3.5	A 22 cm, 250 g quadrotor with a forward-facing stereo camera pair (Liu et al., 2021b).	58
FIGURE 3.6	Average and standard deviation of the processing time of keyframes and non-keyframes with different resolutions and grid sizes tested on a 2.60 GHz Intel i7-6600U laptop. The time required by processing a non-keyframe is approximately the same if we compare the two resolutions with similar number of high-gradient points (e.g. 320×240 with no grid vs. 640×480 with 2×2 grid size).	59
FIGURE 3.7	Average and standard deviation of the total processing time of a keyframe and its three parallel threads: feature extraction, disparity generation and tracking. The processing time for keyframe disparity generation and high-gradient point selection is proportional to the total number of pixels in an image, where the original resolution results in 4 times the computation than the downsampled resolution.	60
FIGURE 3.8	State estimates from direct VIO and the semi-dense map created concurrently on EuRoC dataset V1-02-medium. The 6DoF state estimates are represented by the axes and the map is represented by the point cloud. The computationally efficient block matching algorithm gives noisy disparity map but most of the noisy points are avoided as the algorithm selects only the high gradient pixels.	63
FIGURE 3.9	Average RMSE comparison on EuRoC Vicon room Dataset showing a reasonable performance on accuracy given the simplistic framework adopted in our approach.	64
FIGURE 3.10	Average CPU usage on EuRoC Vicon room Dataset for real time execution excluding the initialization stage. 100% indicates one full core on a 3.60 GHz Intel i9-9900K desktop CPU. Our method shows competitive efficiency in CPU usage which is suitable for on-board flight.	64
FIGURE 3.11	CPU usage visibly reduces by decreasing the number of feature points and window size in VI-Stereo-DSO optimization framework.	66
FIGURE 3.12	Square trajectory state estimates against VICON ground truth.	67

FIGURE 3.13	Position estimates during the square and tour trajectories plotted against time.	68
FIGURE 3.14	Forward facing camera view of the curtain area.	69
FIGURE 3.15	Position estimates in the low texture environment. The figure shows that feature-based methods lose track a short while after the camera sees the curtain, while semi-dense method keeps track with a reasonable small drift.	69
FIGURE 3.16	Improvement by photometric calibration on two other low texture datasets. <i>VIO-calib</i> shows the VIO output after compensating for the vignetting effect. Comparing to the original <i>VIO</i> , adding photometric calibration reduces error in tracking and can in some cases prevent the system from losing track, as is the case in the second illustration.	71
FIGURE 3.17	Average RMSE of our proposed semi-dense approach comparing to MSCKF and ROVIO. Photometric calibration visibly helps the direct semi-dense algorithm in such extreme low texture cases.	72
FIGURE 4.1	The pipeline given three images as inputs. Only forward and backward images are used as inputs, but a third image from the stereo is used for the disparity network and end-to-end training with RANSAC.	78
FIGURE 4.2	Using the RANSAC pose, we can use the predicted flow (top left) to estimate disparity (bottom left), and disparity (top right) to estimate flow (bottom right). Best viewed in color.	82
FIGURE 4.3	Selected trajectories of the models trained on KITTI raw before and after refinement, as well as our final model, against ground truth. Sequence numbers from top left to bottom right are 00, 01, 06, 07, 09, 10. Best viewed in color.	85
FIGURE 4.4	An example of a failure case from sequence 01. From left to right are flow, inlier mask and disparity respectively. The network consistently underestimates the flow over the textureless road section, causing RANSAC to select a large part of the sky as inliers. Best viewed in color.	85

FIGURE 4.5	Interesting qualitative results from our pipeline. Left to right: Grayscale image, inlier mask, predicted flow, predicted disparity. The inlier mask demonstrates RANSAC’s ability to remove areas over which the network has high uncertainty, such as vegetation and textureless regions such as the sky. In addition, it allows us to automatically remove independently moving objects from the scene. Note that in the fourth image, there is a person slight left of the center of the image. Best viewed in color.	90
FIGURE 5.1	An example 3D trajectory estimated by TLIO from a single MEMS IMU, showing the user repeatedly walking up and down a staircase. Our method estimates full orientation and translation in 3D: On the left, we show the estimated x,y,z position (blue) as well as associated uncertainties ($\pm 3\sigma$ in red). On the right, we show the resulting 3D trajectory. For both plots, we show the output of state-of-the-art visual-inertial odometry (green) for comparison.	93
FIGURE 5.2	System block diagram. The IMU buffer provides segments of gravity-aligned IMU measurements to the network, using rotation from the filter state. The network outputs displacement $\hat{\mathbf{d}}$ and uncertainty $\hat{\mathbf{u}}$ used as measurement update to the filter. The filter estimates rotation, velocity, position and IMU biases at IMU rate.	96
FIGURE 5.3	Comparing TLIO to 3D-RONIN on the error metrics with respect to the ground-truth. Each plot shows the cumulative density function of the chosen metric on the entire test set. The steeper the plots are the better the performance. TLIO shows significantly lower value than 3D-RONIN for all presented metrics.	107

FIGURE 5.4	Selection of trajectories. Each graph has x and y axes with unit in meters. 1.x and 2.x are the good and hard cases and 3.x show examples of failures. 2.a. shows a case with very wrong network outputs. 2.b shows a case of bad initialization in the filter leading to yaw drift. In 3.a and 3.b, unusual motions not present in the training set are performed: side-stepping repeatedly for 3 minutes and rolling on a chair.	108
FIGURE 5.5	Network variations with different past and total window sizes evaluated on the test set. Upper left: ATE of 3D-ROBIN concatenation result comparing to GT trajectories. Upper right: average MSE loss on the test set. Adding past data to the network input reduces average MSE loss over samples, but it does not imply lower ATE over trajectories.	111
FIGURE 5.6	Left: Auto-correlation function of the norm of the network displacement error over one sequence. The norms are computed at 20 Hz using overlapping windows of IMU data. Models with more input data have lower MSE but greater temporal correlation of errors.	111
FIGURE 5.7	Uncertainty returned by the network (standard-deviation σ) plotted against errors (m) on three axes x, y, z respectively. Points with larger errors have larger variance outputs, and over 99% of the points fall inside the 3σ cone region indicated by the dashed red line. We have 0.70% for x, y and 0.47% for z axis of error points outside 3σ bounds respectively. . . .	112
FIGURE 5.8	Sensitivity of the network prediction on bias and gravity direction noise. At test time the input IMU samples are perturbed for accelerometer bias, gyroscope bias, and gravity direction as described in Sec. 5.4.2 but with various ranges. The three models under comparison are trained with different data augmentation procedures: no perturbation (noptrb), perturbing only the bias (bias), and perturbing both the bias and the gravity direction (bias-grav). We observe that training with perturbation significantly improves robustness on accelerometer bias and gravity direction. . . .	113

FIGURE 5.9	Performance comparison showing the effectiveness of using a learned covariance. TLIO-mse and 3D-RONIN-mse use a network trained with only MSE loss. TLIO-fixcov and TLIO-mse use a hand-tuned constant covariance for all the measurements. 3D-RONIN and 3D-RONIN-mse concatenate displacement estimates directly without considering uncertainty. TLIO achieves the best performance with the covariance regressed from the network. Constant covariance approaches do not reach 100% for ATE and drift in this illustration due to a failure case.	114
FIGURE 5.10	Performance of different system configurations. Each group corresponds to a network model, as indicated by the x axis labels. Within each group, the colored boxplots differ by update frequency, and the grey ones show 3D-RONIN as baseline. The filtering approach constantly outperforms 3D-RONIN on all the experimented models. High frequency update yields the best ATE and drift in spite of temporal correlation of the measurements. RYE-1s increases with update frequency, indicating more jitter on the yaw estimates.	115

CHAPTER 1

INTRODUCTION

Humans possess various abilities to sense the world, such as eyesight, hearing, equilibrioception, proprioception, and touch. We rely on our senses to gain knowledge of our surroundings and decide how to act. Similar to humans, intelligent devices are carefully designed systems equipped with sensors to obtain information about the environment. The process of estimating internal system state using measurements is called *state estimation*.

State estimation holds fundamental importance to robotics and AR/VR applications - it answers the question of “where we are.” Figure 1.1 shows a block diagram of a typical robotic navigation system. In this context, the state often includes the *pose* (position and orientation), the velocity, and the internal system parameters such as calibration. The map is a representation of the observed environment, either in or a combination of metric, semantic, or topological forms. In autonomous systems, state estimation locates an agent within its environment, which is a prerequisite to the perception-action loop, providing necessary information for planning and control. In AR/VR applications, it registers the viewpoint of the device in the map and is the first step to image rendering. These applications also require state estimation to run in real-time, sometimes with computational constraints from a small size and payload, limited power hardware platform. In addition to the variety of scenarios that the system operates in, these factors pose significant challenges to this field of research.

The de-facto standard state estimation system in robotics and AR/VR is *Visual-Inertial Odometry* (VIO), using the combination of cameras and Inertial Measurement Units (IMUs). The methodologies of VIO systems are well-developed and categorized, with an abundance of open-source software available, some of which have industry-level accuracy. The problem is solved to a large degree. However, in practice, system breakdown occurs in a variety of scenarios, and system drift is inevitable in the most common settings. In an attempt to study the system where it fails, we formulate the causes of performance limitation as different types of information deficiency. In this thesis, we present our investigation of three cases of critical information deficiency in three application scenarios and

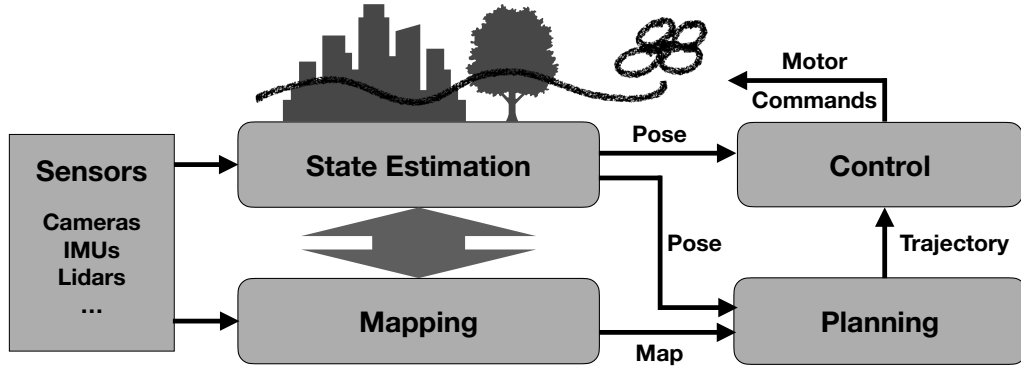


Figure 1.1: Illustration of a basic quadrotor navigation system and the relationships between system components. State estimation and mapping make use of sensory data to localize the robot and create a map of the environment. They are closely related to each other - the accuracy of the pose estimates depends on the accuracy of the map, and the map estimates are tied to the poses. The path planning component designs a trajectory to a goal location based on the estimated map and the robot pose, and the controller calculates motor commands that align the robot pose with the desired trajectory.

summarize our findings.

We start by briefly introducing visual-inertial state estimation from a historical point of view, how it works, and the characteristics of its current development in Section 1.1. Then in Section 1.2 we introduce the concept of information deficiency and its connection to machine learning in finding a solution. Next, Chapter 2 introduces the literature and basics for VIO that are relevant to our research, then the rest of the thesis includes the work and the conclusions.

1.1. Visual-Inertial State Estimation

1.1.1. From Visual Odometry to Visual-Inertial Odometry

The problem of reconstructing 3D structure and device motion from 2D camera images, also known as *Structure from Motion* (SfM), finds its origin in computer vision and photogrammetry in the 19th century. Various important techniques and foundations, including perspective projective geometry, epipolar geometry for stereo vision, and the trifocal tensor for multiple views, have been developed to provide elegant linear mathematical tools and solutions to the 3D modeling problem. However,

these approaches work best in wide-baseline settings and are sensitive to noise and erroneous measurements, and do not suit the needs of a real-time video system with consecutive small-baseline image frames. To work with the incremental and noisy information in those settings, nonlinear iterative approaches such as batch optimization through Levenberg-Marquardt minimization (Szeliski and Kang, 1994) and recursive approaches such as Extended Kalman Filter (Matthies et al., 1989) are developed. These developments form the basis for *visual odometry* (VO) solutions.

Visual odometry is a subset problem of SfM, where the system input is a stream of consecutive camera images, and the goal is to track the *pose* of the camera device, which includes the 3D position and 3D orientation. A VO system assumes static visual features are available with detectable correspondences across images. Using the fundamental principles of visual tracking, implementing a VO system involves multiple crucial steps to ensure accurate and robust real-time operation. This problem was first described in its application on the Mars rover by Moravec in 1980 (Moravec, 1980). This work presents the first complete VO pipeline that includes camera calibration, feature detection, feature matching using epipolar geometry and correlation with coarse-to-fine strategy (also called image pyramid), outlier removal, and least squares optimization for relative camera pose.

Three basic assumptions of VO

1. Sufficient texture for tracking. The environment has enough illumination and visual features to detect and match between frames.
2. Static scene assumption. The estimation of the relative pose between frames (*ego-motion*) assumes that the features being tracked stay static in the physical world. The system maintains a fixed inertial frame for all estimated poses.
3. The images have overlaps where the same features can be detected and matched between frames.

Over the years, visual odometry approaches evolve and mature, and developed into different sub-categories including stereo (Matthies, 1989) and monocular (Nistér et al., 2006) VO, feature-based (Nistér et al., 2004) and appearance-based (Scaramuzza and Siegwart, 2008) VO, and optimization-

based (Klein and Murray, 2007) and filter-based (Davison, 2003) VO. The methods of estimating motion between frames can also be categorized into 2D-2D using essential matrix (Longuet-Higgins, 1981; Nistér, 2004), 3D-3D by aligning feature points (Arun et al., 1987), and 2D-3D Perspective-n-Point (PnP) algorithm (Moreno-Noguer et al., 2007). Better feature detection and matching, as well as outlier rejection schemes such as RANSAC (Fischler and Bolles, 1981) were developed. The tutorial by Scaramuzza and Fraundorfer provides a good introduction to this part of the history and methods (Scaramuzza and Fraundorfer, 2011; Fraundorfer and Scaramuzza, 2012). While visual odometry is mainly concerned with the estimation of camera motion and saves a local map for windowed optimization, visual *simultaneous localization and mapping* (V-SLAM) is also concerned with reconstructing a consistent global map. The mapping problem is deeply intertwined with motion estimation in state estimation. Because the camera poses are relative to the map, the map quality directly affects tracking accuracy. As the map is anchored to the poses, its consistency is an indicator of low drift pose estimates.

Using visual information for state estimation has its unique challenges. Visual data is subject to environmental deviations from the ideal static world model, including lighting conditions, moving objects, occlusions, and the presence of reflections. Outliers are very common in visual feature and correspondence detection. If not detected and handled correctly, these situations can severely degrade system performance. Visual data processing is also one of the most computationally expensive tasks. With the presence of a large number of repetitive features and erroneous detections, if not initialized well, the system can take a long time to converge or converge to a suboptimal minimum, resulting in accumulated drifts that can only be corrected by global optimization for map consistency. There is no metric information for monocular VO algorithms which results in scale ambiguity. For these reasons, the fusion of camera and IMU sensor has gained research attention as it presents a more robust solution to state estimation, coined *visual-inertial odometry*.

This combination brings together the complementary benefits of the two sensors. The camera provides rich visual information at sub-pixel accuracy that works best for translation tracking. In contrast, the angular velocity information from the inertial sensor provides accurate rotation estimates,

complementing the advantage in vision. The high-frequency motion information from the IMU sensor also works well for short-term pose estimates, which complements visual tracking at larger baselines and aids convergence. The inertial sensor also provides scale information to monocular VO systems. The low cost and small form factor of this complementary sensor suite make VIO the de-facto standard state estimation system on small-scale robotics platforms and virtual reality devices.

Contributions of the inertial sensor to visual tracking

An inertial sensor in a VIO system typically serves some or all of the following purposes:

1. Provides a better initialization for visual tracking, making the system more robust to aggressive movements and converging faster
2. Adds an inertial factor to the optimization framework to improve accuracy
3. Provides metric scale information to monocular VO systems
4. Provides gravity direction
5. Supports high-frequency state estimation required by the control system of agile robots

In a *model-based* framework (in contrast to data-driven methods with machine learning), there are two ways to fuse inertial information with visual tracking: *propagation* and *preintegration*. Propagation is the simple application of the IMU motion model to the state, which is often used in the prediction step of a Kalman filter. With noise modeling from raw data, the state covariance can be propagated linearly in each step. IMU preintegration technique (Lupton and Sukkarieh, 2011; Forster et al., 2017a) is developed based on the propagation model to form a relative pose and velocity constraint integrated over multiple IMU samples independent of the initial state. In pose-graph optimization, this is needed to construct an inertial factor where multiple IMU samples are present between image frames. Model-based VIO approaches make use of one or both methods in their optimization frameworks, and in Chapter 5 we introduce a third approach using deep learning to form a statistical constraint learned from data. IMU bias is typically modeled as a random walk process and is added to the state. With the filter measurement update or windowed optimization over multiple images, the bias estimate can be calibrated and further reduce drift in IMU integration. It is worth

noting that the benefit of adding an inertial sensor to visual tracking also comes with additional effort. Additional calibration between the IMU and the camera sensors is required, as well as proper handling of timestamp alignment between sensory data and delays.

Other sensors for state estimation

Other than IMUs and cameras, more common sensors include global navigation satellite system (GNSS) or GPS for outdoor positioning, and distance sensors such as lidars. GNSS provides position and time information of the receiver registered in a global frame. The signals are transmitted from the satellites, which can be blocked by high buildings and trees. Lidars detect the distance to the obstacles through laser light. These sensors work well in the general indoor environment, but the light travels through glass, making such obstacles difficult to detect. They also cannot work in fog and rain conditions due to reflections. There are different types of lidars. The most common choice is spinning lidar, which is safer and provides a 360-degree view. However, new technologies continuously push the frontier to smaller and faster hardware, such as solid-state and flash lidars. There are also ultrasonic sensors, wireless sensors, Bluetooth, and barometers. Wheel odometry on rovers and cars and the rotor speed measurements from the Electronic Speed Controller (ESC) connected to propellers on aerial platforms can also provide information to the system, aiding the state estimates. Newly developed sensors such as event-based cameras are susceptible to illumination changes in the dark. With measurements at over 10,000 fps, such sensors provide new solutions to low-light high-speed tracking in a continuous-time optimization framework.

1.1.2. Improving Visual-Inertial Tracking

With sufficient engineering effort and suitable system expansions, a VIO system can increase the robustness of the optimization algorithm over various unfavorable and common scenarios and improve estimation accuracy. We list below, not exhaustively, some of the major ways to improve a model-based visual-inertial tracking system performance by large margins:

1. Use an adaptive keyframe selection scheme to maintain tracking quality across different move-

ment speeds and viewing angles (Piao and Kim, 2019)

2. Refine feature detection and matching pipeline and associated data structure to maintain a robust map representation that can reject moving objects and reduce misalignment and duplicates (Engel et al., 2018; Mur-Artal et al., 2015a)
3. Make use of the local map for failure recovery (*relocalization*) (Williams et al., 2011; Qin et al., 2018a)
4. Make use of a global map and a library of past observations for drift correction when revisiting previous places (*loop closure*) (Qin et al., 2018a; Mur-Artal et al., 2015a)
5. Add observable calibration parameters to the state to correct sensory and mechanical changes online (Bergmann et al., 2017; Huang et al., 2020)
6. Have exposure control or photometric compensation to improve feature tracking under illumination inconsistencies (Bergmann et al., 2017; Engel et al., 2016)
7. Compensate for motion blur with trajectory tracking and image formation model (Liu et al., 2021a)
8. Incorporate semantics information into the framework (Shan et al., 2020)

We can observe the development of visual-inertial tracking algorithms in pursuit of higher accuracy and robustness have the following traits. A large portion of the improvement considerations overlaps with visual tracking since the challenges are shared. Improvements can be attempted from the following aspects, and we put the labels of the corresponding examples above in brackets: designing better heuristics for more adaptive and precise supervision of the optimization (1); using more comprehensive feature descriptors to improve visual tracking accuracy and reduce erroneous detections (2); estimating more system parameters online to account for calibration changes and biases (5); making use of the mapping counterpart or outlier rejection schemes to correct drift and errors (2,3,4); incorporating information from existing or novel models to compensate for the predictable errors (5,6,7) or form new constraints (8). We have not yet considered the challenges in multi-agent

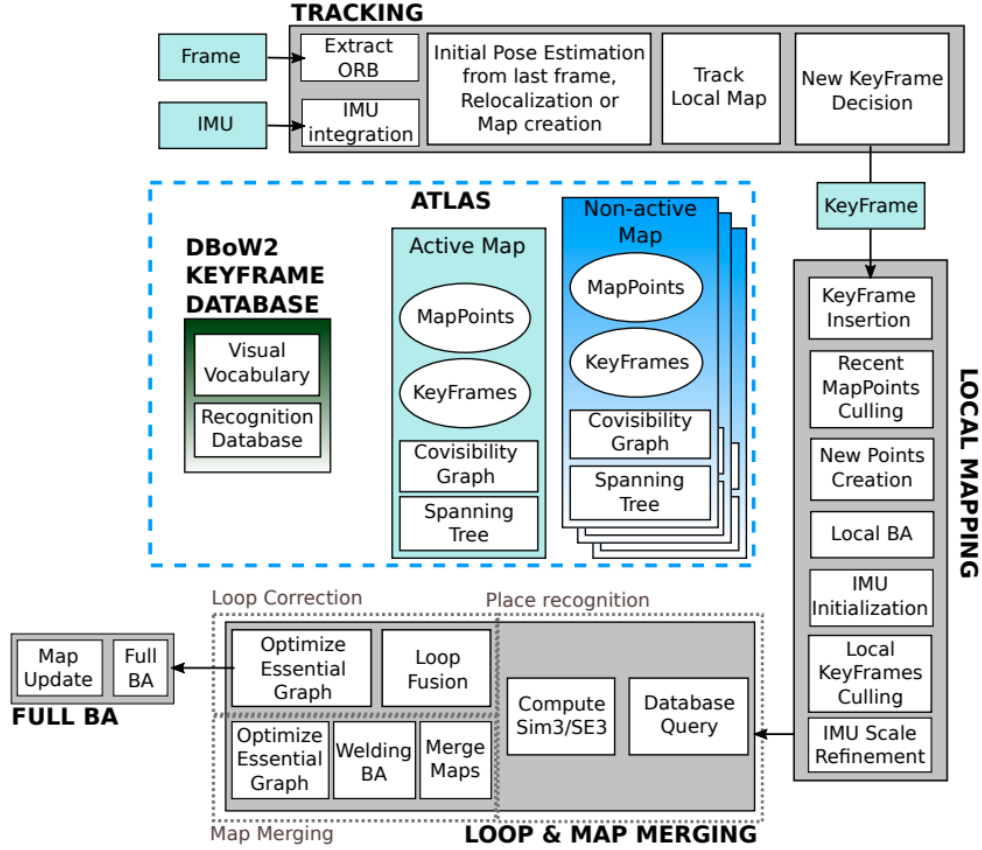


Figure 1.2: System diagram of ORB-SLAM3 (Campos et al., 2021).

systems or additional sensory inputs.

It is essential to recognize the growing complexity of a visual-inertial state estimation system and its demands in engineering work to increase system robustness. We can also see the increasing importance of an accurate map as a tool to correct drift as the state estimate improves. A VIO system becomes a VI-SLAM system as the consistency requirement increases. Figure 1.2 shows the system diagram of ORB-SLAM3 (Campos et al., 2021), a state-of-the-art visual-inertial SLAM algorithm that manages map points carefully in the process. For a VIO system to work across an extending set of scenarios, acquiring the knowledge of predictable patterns and error sources is equally important to design a robust overall general system. For example, the visual features do not move in an elevator, but inertial data indicates otherwise. In this scenario, the static scene assumption is broken in visual tracking, and a detection mechanism can be applied to handle such situations. The prior knowledge

that the possibility of entering an elevator is present leads to the system adaptation that avoids such failures. There is a need for a general framework and guideline that comprehensively describes the VIO system, covering different edge cases and explaining the resulting complex engineering structure. In the next section, we introduce the concept of information deficiency to understand and formulate the source of errors in VIO systems, providing a new framework to argue about errors and the potential for improvement.

1.2. Information Deficiency and Machine Learning

1.2.1. Information Deficiency in State Estimation

We have learned that improving a visual-inertial state estimation system involves considering various aspects. As system designers and developers, we need a way of prioritizing such tasks to reduce errors and drifts effectively. Therefore, we propose to use information deficiency as a tool to help identify the source of errors and aid the decision-making process in the system of visual-inertial state estimation.

The concept of information deficiency stems from management and information science, where it is used as an indicator of how crucial a category is relative to other categories Srinivasan and Kaiser (1981). In visual-inertial state estimation, we describe *information deficiency* to consider the following aspects: the knowledge of different error source categories and their importance or contributions, the availability and usability of source information, and the implementation compatibility with the system requirements.

Information deficiency in state estimation includes the unknown source of errors, unavailable information for error prediction, and incompatible model with the system pipeline to provide the necessary information. Based on the contribution of an error source category to the system performance, information deficiency can be identified with different levels of significance.

To explain this concept further, we look back to the summary from Section 1.1.2, and we can differentiate various methods of improvement as either system design improvements, error corrections, or

new information utilization. Designing better heuristics and feature descriptors are modifications to the details in system design that determine the baseline performance. Outlier rejection schemes are detections and corrections to the failures and errors in the baseline system, which we can interpret as a result of a lack of information. Photometric calibration, motion blur compensation, semantics incorporation, and estimating additional calibration parameters, including the elevator example, are all methods that develop models to use available information to reduce estimation errors. To develop a VIO system, we first identify different sources of errors and evaluate the severity and possibility of addressing these errors in terms of information deficiency to inform a decision.

Let us take occlusion as an example to illustrate the idea. Occlusion is a common occurrence in vision - a moving viewpoint will reveal previously unknown appearances behind objects and lose sight of some previous features. These can result in errors in feature tracking. One can use RANSAC or a robust estimator to remove or suppress these errors, regarding them as outliers. However, these errors can also be considered as a result of the system not knowing of such occlusions, in other words, a type of information deficiency. There are numerous sources of error in a visual-inertial state estimation system. If a specific error source is unknown, then this type of error can only be corrected with outlier rejection or robust estimators. If an error source is known and the information to predict the error is available, a solution is possible by developing methods to utilize the information. Suppose an error source is known, but the information to address the error is unavailable when it is needed in the processing pipeline. In that case, in addition to using robust estimators, we argue with research that machine learning is an effective tool to bring prior information into the system as an approach to address this type of information deficiency.

Visual front-end in a state estimation system is often tricky because of the wide variety of corner cases and high probability of outlier occurrence. However, instead of perceiving them as erroneous estimates, they can also be viewed as information deficiency. Viewing errors and outliers as a consequence of unavailable information instead of an inherent characteristic of image processing helps break down a complicated problem into solvable components scientifically. Identifying the error sources and the availability of relevant information provides a systematic and theoretical framework

for understanding state estimation problems.

1.2.2. Three Cases of Critical Information Deficiency

In this thesis, we present our research on three critical information deficiency cases:

- low texture scenarios with limited computation,
- monocular visual odometry, and
- inertial odometry.

We explore the boundary of visual-inertial state estimation systems from both a model-based perspective and the combination with deep learning regime.

Chapter 3 presents a visual-inertial odometry system operating in low-texture visual environment on a resource-constrained platform (Liu et al., 2021b). In particular, we are looking at direct semi-dense VIO that enables autonomous flight of a small-size quadrotor equipped with an ARM board. We use a typical optimization framework whose computation requirement directly conflicts with the hardware limitation. We propose a simplified framework that balances the trade-off between computation and accuracy and discuss the complications of making such reductions.

Chapter 4 presents a monocular visual odometry system that recovers scale information without an IMU sensor (Zhu et al., 2018). We propose a hybrid structure that combines state-of-the-art depth and flow estimation using deep learning and least square optimization with RANSAC outlier rejection for ego-motion estimation. We show how to use a learned prior to make metric scale observable within a robust optimization framework.

Finally, Chapter 5 presents an inertial-only navigation system with a single IMU sensor when vision information is entirely unavailable (Liu et al., 2020). We propose a solution by tightly fusing a deep learning module into a probabilistic state estimation framework, namely a multi-state extended Kalman filter. The aim is to have a system that works when none of the visual tracking assumptions hold. We show how deep learning provides a new solution to the drift problem for IMU sensors that renders sensor biases observable and enables accurate long-term tracking for pedestrians.

These three scenarios are considered critical cases of information deficiency because the missing information is of fundamental importance to the system. Some of these problems can be considered ill-posed in a model-based framework. In the first case, the low texture scenario presents very little visual feature information for tracking. Utilizing as much information as possible from image gradients meets the computation ceiling on the small quadrotor platform. We present a semi-dense solution with comparisons to the state of the arts and a detailed analysis of the choice of algorithms and parameters. The second case is the widely studied monocular visual odometry, where drift in scale cannot be corrected as it is not observable without the IMU sensor. We consider the problem in driving scenarios and make use of unsupervised learning to improve accuracy while compensating for the scale information. The last case considers only an inexpensive IMU sensor famous for significant drifts within seconds with double integration in its kinematic motion model. We propose using supervised learning as an additional statistical model in an Extended Kalman Filter (EKF) framework and show performance comparable to a state-of-the-art VIO system for pedestrian VR/AR headset applications.

Why do we care about such severe information deficiency cases? One reason is that we encounter these situations more often than we expect. Walking into a dark room, facing a white wall, or having no image overlap due to sensor failures, latency, or environmental shifts are all examples that break visual tracking assumptions. They happen easily in practice, but we often do not see them in the standard datasets or benchmarks. A second reason is for high-level system design. For example, power-saving strategy or privacy concerns can influence the decision on the sparsity of images taken, even to turn off the camera completely.

If we do not have the information, how is it possible to make it work? We argue that the benefit of data-driven approaches stands out in addressing such challenges. Since a historical expertise-based approach is often better understood and well functioning, many deep learning approaches achieve high performances yet are not accepted into practical safety-critical applications. However, this is no longer the case with incomplete information. The lack of information comes with the opportunity to fuse the knowledge learned from past data into the online framework. There are numerous deep

learning approaches in visual-inertial tracking, and we summarize some representative developments in Section 2.1. The following section summarizes our approaches to fusing deep learning into visual-inertial state estimation frameworks and draws the main conclusions to this thesis.

1.2.3. Fusion with Learning

In this thesis, we present two approaches in Chapter 4 and 5 that incorporates deep learning into the state estimation framework. We are interested in combining data-driven and model-based methods, particularly in using the data-driven approach as a module within a model-based framework. We prefer this hybrid approach over end-to-end network frameworks because state estimation is the fundamental component of intelligent systems, and its reliability is crucial. We require this part of the system to be dependable, not just to have superior performance on public datasets. In the VO framework (Chapter 4), we design a system that enforces the geometric principles in the optimization, providing supervision to the deep learning module. In the IO framework (Chapter 5), we use a deep learning module in a filtering framework to constrain the growing errors in the IMU kinematic model, while on the other hand, the probabilistic framework provides statistical methods such as χ^2 tests for outlier rejection. We also estimate statistically consistent uncertainty from supervised learning, which allows us to contain errors effectively in the learning module.

From these case studies with information deficiency, we derive two important conclusions to this thesis:

- Deep learning is a key technology in addressing information deficiencies in state estimation;
- The tight fusion of deep learning and model-based state estimation approaches provide the most accurate and robust system performance.

A deep learning model can compensate for the lack of information online by learning a prior from training data, or regressing a more accurate statistical model from limited information. Experimental results back the performance boost of a hybrid system. Looking deeper into the fusion of learning and model-based methods, it is essentially the combination of the statistical model and the physics model: the statistical model provides high accuracy numerical approximations and learned priors, while the

physics model provides geometric and dynamical constraints. Developing hybrid frameworks that utilize the advantages of both models and enable cross-supervision for performance guarantees can effectively improve system performance.

In a word, we are looking for ways to provide a performance guarantee while incorporating deep learning methods into the state estimation system. We focus on the cross-supervision relationship between the data-driven and model-based aspects of the system. We show with system comparisons that these methods outperform both purely learned and model-based approaches while providing a practical solution to reducing errors with information deficiency.

In addition, uncertainty estimation with deep learning plays a vital role in facilitating the tight fusion of data-driven and model-based approaches in a probabilistic framework. The accurate measure of uncertainty derived from data noise and the capability to argue about out-of-distribution data are necessary for developing performance guarantees. For this reason, we extend the scope of this thesis to include a survey in Section 2.3 on uncertainty estimation with neural networks. From studying existing probabilistic algorithms and literature review on the practical approaches and existing benchmarks, we identify that there is still a gap between neural network approaches for uncertainty estimation and Bayesian modeling, which explains the relationship between different types of uncertainties. The non-Bayesian methods perform the best on the existing benchmarks. However, more work still needs to be done to develop an effective way to argue about out-of-distribution data.

CHAPTER 2

VISUAL-INERTIAL STATE ESTIMATION

This chapter introduces visual-inertial state estimation, summarizing different approaches and the current state of the arts in Section 2.1 and the preliminaries in Section 2.2. Then as an extension in pursuit of fusing deep learning tools into a model-based framework with a performance guarantee, we survey practical uncertainty estimation approaches and benchmarks with neural networks and report insights in Section 2.3.

2.1. Classifications and State of the Art

VIO systems have been extensively studied in the literature. Various review papers and benchmarks are available in terms of method comparisons (Huang, 2019), system evaluations (Delmerico and Scaramuzza, 2018), and applications (Scaramuzza and Zhang, 2019). This section briefly introduces the important classifications and available resources in state-of-the-art and discusses the development trend and some deep learning applications in this field.

There are different ways to classify VIO algorithms. Depending on the solver type, there are *filtering* approaches and *optimization* approaches (Gui et al., 2015). A successful representative approach in filtering is MSCKF (Mourikis and Roumeliotis, 2007). In contrast to classic EKF-based approaches, which include a small number of features in the state (Davison et al., 2007; Bloesch et al., 2015), feature measurements in MSCKF are marginalized, resulting in a more scalable solution having linear complexity in the number of features instead of quadratic. There are various extensions to the MSCKF framework, such as improving estimation inconsistency issues by reducing linearization errors with iterative EKF (Bloesch et al., 2017), explicitly constraining the observable space with OC-MSCKF (Li and Mourikis, 2012; Hesch et al., 2013), and adopting robocentric models (Huai and Huang, 2018). In addition, square root inverse formulation (Wu et al., 2015) is proposed to improve efficiency and numerical stability.

Optimization approaches solve a non-linear least square problem using Bundle Adjustment (Triggs et al., 1999). A major difference from filtering approaches is that an information matrix is used

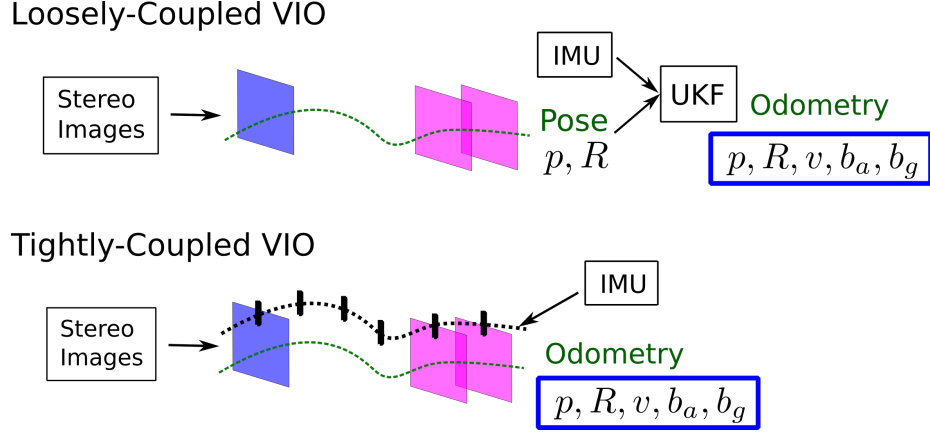


Figure 2.1: Difference between a loosely-coupled (Liu et al., 2018) and a tightly-coupled (Liu et al., 2021b) VIO system.

instead of the covariance matrix, derived from the maximum a posteriori (MAP) estimate formulation. Because of the costly matrix inversion operation, the sparsity structure of the Hessian matrix is exploited to ensure computational efficiency (Leutenegger et al., 2013). However, batch optimization still takes more computation than filtering approaches because the full state is optimized for all camera poses in the framework. A successful representative approach is VINS-MONO (Qin et al., 2018b), which includes loop closure. In addition, incremental smoothing technique (Kaess et al., 2008, 2012) has been developed to exploit the sparsity of information matrix in factor graph formulation and calculate updates incrementally to boost computational efficiency.

There are two different ways to fuse IMU and vision measurements: *tightly-coupled* and *loosely-coupled* (Corke et al., 2007). Loosely-coupled approach (Klein and Murray, 2007) fuses IMU data with the result of visual odometry as an independent process, while the tightly-coupled approach (Leutenegger et al., 2013; Mourikis and Roumeliotis, 2007) processes image and IMU measurements together to obtain a final estimate. Figure 2.1 shows an example noting the difference between these two types of systems. There are two types of image measurement constraints: *geometric* and *photometric*, also known as *indirect* and *direct*, or *feature-based* and *appearance-based* methods. Indirect methods (Leutenegger et al., 2013; Mourikis and Roumeliotis, 2007; Mur-Artal et al., 2015a) extract and match image features and minimize reprojection errors for the optimal pose, while direct methods (Engel et al., 2018, 2014) make use of the image gradient and minimize intensity errors.

Based on the number of image measurements processed, there are *sparse*, *semi-dense*, and *dense* approaches. Indirect methods most often optimize only sparse feature points (Mourikis and Roumeliotis, 2007), while direct methods are more suitable for optimizing semi-dense (Engel et al., 2014) and dense (Newcombe et al., 2011) measurements using high gradient pixels and full images. The algorithms are often developed for different camera setups including *monocular*, *stereo* and *rgbd*.

Many existing open-source datasets are available, and one of the most popular for comparison is the EuRoC (Burri et al., 2016) Dataset, which consists of 11 motion sequences. In addition, various open-source implementations of VIO algorithms of different approaches have been developed, and the paper of ORB-SLAM3 (Campos et al., 2021) provides a summary of the representative algorithms and their code availability.

The performance of VIO systems is subject to various non-algorithmic factors, including the hardware. For example, different embedded platforms with various clock speeds can affect system performance. A benchmark comparison of representative approaches on different platforms is introduced in (Delmerico and Scaramuzza, 2018). System efficiency can also be boosted with hardware/software collaborative design, leveraging the acceleration of visual front-end pipeline with FPGA hardware (Mandal et al., 2019; Lentaris et al., 2015). The emergence of event-based cameras further opens up research opportunities for a novel VIO framework to process asynchronous visual measurements (Vidal et al., 2018; Mueggler et al., 2018).

With the rapid development of deep learning techniques in the vision community, many data-driven solutions have been proposed. Using image warping and photoconsistency constraints, image depth (Garg et al., 2016; Godard et al., 2017), flow (Jason et al., 2016; Meister et al., 2017), and 3D camera motion can be learned in an unsupervised fashion (Zhou et al., 2017; Zhan et al., 2018; Li et al., 2018; Wang et al., 2018b). This approach assumes that the scene is static and is corrupted by independently moving objects. Some works try to remove these objects by predicting invalid pixel masks (Zhou et al., 2017; Vijayanarasimhan et al., 2017) or finding mismatches in the predictions from motion constraints (Ranjan et al., 2018; Luo et al., 2018; Yang et al., 2018b). Some works make use of the model-based methods as supervision (Wang et al., 2018a), or to use deep learning to complement

existing optimization frameworks for monocular depth, moving object mask, or optimization initialization (Yang et al., 2018a, 2020). Deep learning has outperformed the model-based depth and flow estimation methods, and the combination of learning and the existing optimization framework has shown great potential for improvement. In addition, some works incorporate uncertainty estimation in the learning framework (Ilg et al., 2018), which can provide valuable information to probabilistic state estimation frameworks. We will discuss in detail these approaches in Section 2.3. Apart from vision, deep learning on IMU data also shows the benefit of correcting drift (Yan et al., 2020). End-to-end training for visual-inertial odometry has also been proposed (Clark et al., 2017). However, these pure network-based approaches have not reached the same accuracy as model-based methods.

2.2. Basics

2.2.1. 3D Pose Representation

In state estimation, we need to represent the 3D rigid body pose estimates in an orthonormal reference frame. Since rotation is not defined in the Euclidean space, we use Lie group representation to denote poses and the corresponding Lie algebra for a minimum representation of rotation.

Given two frames A and B , the rotation from frame A to B can be denoted with a 3×3 rotation matrix \mathbf{R}_A^B whose columns are the basis vectors of A represented in B . A rotation matrix is orthogonal and has three degrees of freedom (DoF), and can be composed by matrix multiplication:

$$\mathbf{R}_A^C = \mathbf{R}_B^C \mathbf{R}_A^B.$$

The *Special Orthogonal Group* is a smooth manifold defined as $SO(3) \doteq \{\mathbf{R} \in \mathbb{R}^{3 \times 3} : \mathbf{R}^T \mathbf{R} = \mathbf{I}, \det(\mathbf{R}) = 1\}$, and each element in $SO(3)$ represents a 3D rotation matrix. The tangent space of the $SO(3)$ manifold at the identity is the corresponding *Lie algebra* and is denoted as $\mathfrak{so}(3)$, and each element in $\mathfrak{so}(3)$ is a skew symmetric matrix. A skew symmetric matrix $[\phi]_{\times}$ can be obtained from its vector form ϕ in \mathbb{R}^3 .

Elements in Lie group and Lie algebra can be mapped to each other. The map from the vector form of $\mathfrak{so}(3)$ to $SO(3)$ is called the *exponential map*: $\text{Exp}: \mathbb{R}^3 \rightarrow SO(3)$ defined as $\phi \mapsto \exp([\phi]_{\times})$,

and the inverse operation to the exponential map is the *logarithm map* Log .

For rotations in state estimation, we optimize for the state increment $\delta\phi$ instead of the parameters of the rotation matrix in $SO(3)$ directly in order to stay on the rotation manifold using minimum parametrization. The exponential map can be computed using the Rodrigues' formula:

$$\text{Exp}(\phi) = \mathbf{I} + \frac{\sin(\|\phi\|)}{\|\phi\|} [\phi]_{\times} + \frac{1 - \cos(\|\phi\|)}{\|\phi\|^2} ([\phi]_{\times})^2, \quad (2.1)$$

and its first order approximation gives a useful property

$$\text{Exp}(\phi) \approx \mathbf{I} + [\phi]_{\times}. \quad (2.2)$$

For linearization during optimization, we often make use of the first-order approximations of the exponential and logarithm maps on the right hand side:

$$\text{Exp}(\phi + \delta\phi) \approx \text{Exp}(\phi) \text{Exp}(\mathbf{J}_r(\phi) \delta\phi) \quad (2.3)$$

$$\text{Log}(\text{Exp}(\phi) \text{Exp}(\delta\phi)) \approx \phi + \mathbf{J}_r^{-1}(\phi) \delta\phi \quad (2.4)$$

where $\mathbf{J}_r(\phi)$ is the *right Jacobian* of $SO(3)$.

Another useful property is the adjoint which transforms between tangent spaces on $SO(3)$ for any rotation matrix $\mathbf{R} \in SO(3)$:

$$\mathbf{R} \cdot \text{Exp}(\phi) = \text{Exp}(\mathbf{R}\phi) \cdot \mathbf{R}. \quad (2.5)$$

For 6 DoF relative pose representation, we use *Special Euclidean Group* $SE(3)$ which includes a 3 DoF rotation in $SO(3)$ and a translation vector in \mathbb{R}^3 . It is defined as:

$$SE(3) = \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \mid \mathbf{R} \in SO(3), \mathbf{t} \in \mathbb{R}^3 \right\}. \quad (2.6)$$

In matrix homogeneous representation, transformation composition can be achieved by matrix mul-

multiplication, similar to rotation:

$$\mathbf{T}_A^C = \mathbf{T}_B^C \mathbf{T}_B^A, \quad (2.7)$$

and a transformation matrix T is invertible and its inverse can be written as

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{R}^\top & -\mathbf{R}^\top \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}. \quad (2.8)$$

2.2.2. Image Formulation and Computer Vision

A digital camera collects the rays of light reflected from objects in the scene. The properties of the optics can cause misfocus, aberrations, and vignetting effect. After passing through the lens, the light rays reach the digital image sensor, and its energy is captured within exposure time and converted to voltage signals. The raw signals then go through post-processing steps to generate the image in a compressed form such as JPEG.

Camera Models, Distortions, and Stereo Rectification

A camera model describes the projection geometry that connects spatial coordinates in 3D with pixel coordinates on the image plane. One option is to treat the camera lens as an ideal *pinhole*. In a pinhole camera model (Ma et al., 2012), the projection of a point $(x_c, y_c, z_c)^\top$ from the camera frame to the homogeneous pixel coordinates $(u, v, 1)^\top$ is captured by the camera intrinsic matrix \mathbf{K} :

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \frac{1}{z_c} \mathbf{K} p_c = \frac{1}{z_c} \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} \quad (2.9)$$

f_x and f_y are focal lengths expressed in units of pixels, obtained by dividing the width and height of the pixel sensor. The s value encodes the skew, which is often set to 0 in practice. c_x and c_y denotes the offsets from the principal point to the upper left corner of the image, which is the origin of the pixel coordinates.

The geometry of the lens can cause distortions as a form of optical aberration. For example, straight

lines can appear curved either outward away from the center or inward towards the center, which is often observable near the edge of the image. This is called *radial distortion*, and it can be modeled as:

$$x_{dist} = x(1 + k_1r^2 + k_2r^4) \quad (2.10)$$

$$y_{dist} = y(1 + k_1r^2 + k_2r^4) \quad (2.11)$$

where x and y are the original image coordinates with origin at the radial distortion center. k_1 and k_2 are the radial distortion parameters and $r^2 = x^2 + y^2$. In addition to radial distortion, there is also the *tangential distortion*, which is caused by a tilted image plane not perpendicular to the optical axis. The tangential distortion can be modeled as:

$$x_{dist} = x + (2p_1xy + p_2(r^2 + 2x^2)) \quad (2.12)$$

$$y_{dist} = y + (p_1(r^2 + 2y^2) + 2p_2xy) \quad (2.13)$$

Summarizing the distortion parameters, the *distortion coefficients* is given by (k_1, k_2, p_1, p_2) , which is part of the camera calibration parameters. There are also models that include higher-order terms. For further details and model derivations, we refer the reader to the book by Szeliski (2010).

In addition to the pinhole camera model, the fisheye camera model has a much wider field of view but more significant distortions. Therefore a different model is needed to describe the camera projection (Scaramuzza et al., 2006).

Stereo cameras offer depth information. To obtain depth, we can rectify the camera images to make the epipolar lines horizontal in both images. Then depth estimation becomes a more straightforward procedure by scanning along the epipolar line for the best pixel match and computing from their disparity and a known camera baseline (Loop and Zhang, 1999). Additional calibration parameters of the relative transformation between the two cameras are needed for stereo rectification.

Camera Calibration

Accurate calibration is crucial to the precision of visual tracking. A standard calibration procedure to obtain the calibration parameters is by using a chessboard pattern (Bradski and Kaehler, 2008). The pattern on the board has known dimensions and the corners are easy to detect. By collecting enough images viewing the chessboard from different angles at different portions of the image, optimizing the re-projection errors can give us the calibration estimates offline.

When using direct methods which assume illumination consistency, photometric calibration and exposure control become critical to tracking accuracy. An important aspect of photometric calibration is to remove the *vignetting effect* on pixel intensities. Vignetting effect causes the pixels away from the center of the image to have lower intensity. This can be caused by lens geometry, angular sensitivity of the sensor, or blockage of light. The vignetting effect can be modeled as a scaling factor V to the original intensity values I_0 at pixel location x : $I(x) = V(x)I_0(x)$, and approximated with a sixth-order polynomial:

$$V(x) = 1 + \alpha_1 r(x)^2 + \alpha_2 r(x)^4 + \alpha_3 r(x)^6 \quad (2.14)$$

where $r(x)$ represents the radius of pixel location x to the image center (Goldman, 2010). The exposure time adds to the existing vignetting model as another scaling factor t for intensity: $I(x) = tV(x)I_0(x)$. Both vignetting model parameters and exposure changes can be estimated with selected pixel measurements (Bergmann et al., 2017).

Motion Field Equation

Optical flow is defined as the pixel motion between successive images. Assuming that the brightness of a pixel does not change, we can write down the brightness consistency constraint for pixel velocity on the image plane (v_x, v_y) :

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) \quad (2.15)$$

$$\frac{\partial I}{\partial x} \frac{dx}{dt} + \frac{\partial I}{\partial y} \frac{dy}{dt} + \frac{\partial I}{\partial t} = I_x v_x + I_y v_y + I_t = 0 \quad (2.16)$$

For image flow that results from motion in the 3D space, we can derive the velocity on the image plane by differentiating with respect to time given the relation between a point in 3D $P = [X, Y, Z]^T$ and in 2D $p = P/Z = [x, y, 1]^T$:

$$\dot{p} = \frac{\dot{P}}{Z} - \frac{\dot{Z}}{Z^2}P = \frac{\dot{P}}{Z} - \frac{\dot{Z}}{Z}p \quad (2.17)$$

The time derivative of the 3D point P can be written with the angular and linear velocity Ω and V of the camera:

$$\dot{P} = -\Omega P - V \quad (2.18)$$

Combining with Eq 2.17, we can obtain

$$\dot{p} = \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} -1 & 0 & x \\ 0 & -1 & y \end{bmatrix} V + \begin{bmatrix} xy & -(1+x^2) & y \\ (1+y^2) & -xy & -x \end{bmatrix} \Omega \quad (2.19)$$

This equation connects the pixel depth Z , optical flow \dot{p} , and the 3D motion of the camera in terms of linear velocity V and angular velocity Ω . It also separates optical flow into translational and rotational components.

Neural network methods have emerged as the state-of-the-art high performance optical flow estimation technique. Together with estimated image depth, we can solve for 3D camera motion using least squares. There are 6 degrees of freedom in camera motion, and we need only 3 points to have a fully constrained linear system. Therefore we can use 3-point RANSAC algorithm to find inliers. This approach is adopted in Chapter 4 and details of the RANSAC algorithm is introduced in Section 2.2.4.

2.2.3. Inertial Measurement Unit

The inertial measurement unit usually includes accelerometers, gyroscopes, and magnetometers. The accelerometer and gyroscope provide motion information for the device. They are less affected by the environment, which is the primary source of information for visual-inertial navigation.

The accelerometer measures the acceleration relative to the free-fall state. Therefore the sensor measures any acceleration relative to the earth inertial frame W plus gravity. The sensory data are represented in the body frame B , and the measurement model is written as:

$$\mathbf{a} = \mathbf{R}_W^B(\mathbf{a}_{true}^W - \mathbf{g}^W) + \mathbf{b}_a + \mathbf{n}_a \quad (2.20)$$

The sign before gravity is negative because the gravity vector is pointing downward. We assume the world frame W has its z axis pointing upwards. We also assume the sensor noise $\mathbf{n}_a \sim \mathcal{N}(0, \mathbf{N}_a)$ follows a zero mean Gaussian distribution in continuous-time. \mathbf{b}_a is accelerometer bias, modeled as an additive random walk process. The spectral noise $\boldsymbol{\eta}_a \sim \mathcal{N}(0, \mathbf{N}_{ba})$ is modeled as a zero-mean Gaussian. When used in discrete-time, the covariance of the noises is subject to the sampling rate and can be derived as:

$$\mathbf{N}_{ad} = \frac{1}{\Delta t} \mathbf{N}_a \quad (2.21)$$

$$\mathbf{N}_{bad} = \Delta t \mathbf{N}_{ba} \quad (2.22)$$

The gyroscope measures the rotational velocity, which can be transformed directly from the world frame by rotation. Similar to the accelerometer, the sensor data is corrupted by noise and bias:

$$\boldsymbol{\omega} = \boldsymbol{\omega}_{true} + \mathbf{b}_g + \mathbf{n}_g \quad (2.23)$$

where the sensor noise $\mathbf{n}_g \sim \mathcal{N}(0, \mathbf{N}_g)$ and $\boldsymbol{\eta}_g \sim \mathcal{N}(0, \mathbf{N}_{bg})$ for the random walk bias \mathbf{b}_g . The discrete formulation follows Eq. 2.21 and 2.22.

In addition to the bias and error modeling, the scale factor \mathbf{T}_a and \mathbf{T}_g and g-sensitivity \mathbf{T}_s can also be added to the model. The scale factor calibrates the non-orthogonality and scaling of the sensor,

and the g-sensitivity calibrates the cross influence from acceleration under gravity:

$$\mathbf{a} = \mathbf{T}_a \mathbf{R}_W^B (\mathbf{a}_{true}^W - \mathbf{g}^W) + \mathbf{b}_a + \mathbf{n}_a \quad (2.24)$$

$$\boldsymbol{\omega} = \mathbf{T}_g \boldsymbol{\omega}_{true}^B + \mathbf{T}_s \mathbf{a}_{true}^B + \mathbf{b}_g + \mathbf{n}_g \quad (2.25)$$

2.2.4. Optimization Techniques

Nonlinear Least Squares

In a VIO system, the visual front-end includes operations such as extracting features, obtaining pixel coordinates and intensities, and calculating gradient and depth. In the optimization back-end, a cost function is formulated as photometric or geometric errors using the information, and the objective is often to minimize a cost function formulated with these error residuals. The cost function is usually expressed in a least-squares form, and minimizing the squared errors is analytically equivalent to optimizing for the maximum a posteriori (MAP) solution under the assumption of white Gaussian noise:

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} (E(\mathbf{x})) = \arg \min_{\mathbf{x}} \left(\sum_i \|\mathbf{r}_i(\mathbf{x})\|_{\boldsymbol{\Sigma}_r^{-1}}^2 \right) \quad (2.26)$$

$E(\mathbf{x})$ is the cost function, which is the summation of all residuals \mathbf{r}_i for each feature point with covariance $\boldsymbol{\Sigma}_r$. The energy function can be written as the error vector weighted by the inverse covariance matrix $\boldsymbol{\Sigma}_r^{-1}$:

$$\|\mathbf{r}(\mathbf{x})\|_{\boldsymbol{\Sigma}_r^{-1}}^2 = \mathbf{r}(\mathbf{x})^\top \boldsymbol{\Sigma}_r^{-1} \mathbf{r}(\mathbf{x}) \quad (2.27)$$

Visual observations often have a large number of error terms. To keep a sparse structure of the matrix for computational efficiency of the matrix inverse operation, $\boldsymbol{\Sigma}_r$ is often considered as diagonal. For inertial observations, the full covariance can be estimated with IMU integration using the IMU measurement noise model.

The cost function is often non-convex and nonlinear. This is particularly the case for direct methods,

where the local image gradient limits the area of convergence. Assuming a good initialization, we find the local minimum by iteratively linearizing the error term at the current state estimate using Taylor expansion.

$$E(\mathbf{x} \oplus \boldsymbol{\delta}) = \|\mathbf{r}(\mathbf{x} \oplus \boldsymbol{\delta})\|_{\boldsymbol{\Sigma}_r^{-1}}^2 \approx \|\mathbf{r}(\mathbf{x}) + \mathbf{J}_r \boldsymbol{\delta}\|_{\boldsymbol{\Sigma}_r^{-1}}^2 \quad (2.28)$$

$$= \mathbf{r}^\top(\mathbf{x}) \boldsymbol{\Sigma}_r^{-1} \mathbf{r}(\mathbf{x}) + 2\boldsymbol{\delta}^\top \mathbf{J}_r^\top \boldsymbol{\Sigma}_r^{-1} \mathbf{r}(\mathbf{x}) + \boldsymbol{\delta}^\top \mathbf{J}_r^\top \boldsymbol{\Sigma}_r^{-1} \mathbf{J}_r \boldsymbol{\delta} \quad (2.29)$$

$$= \mathbf{r}^\top \boldsymbol{\Sigma}_r^{-1} \mathbf{r} + 2\boldsymbol{\delta}^\top \mathbf{b} + \boldsymbol{\delta}^\top \mathbf{H} \boldsymbol{\delta} \quad (2.30)$$

We omit (\mathbf{x}) in the last equation for simplicity, and $\boldsymbol{\delta}$ is the small increment of state that we want to estimate. Here the Jacobian is the derivative of error with respect to the state evaluated at the current estimate \mathbf{x}_0 :

$$\mathbf{J}_r = \left. \frac{d\mathbf{r}(\mathbf{x})}{d\mathbf{x}} \right|_{\mathbf{x}_0} \quad (2.31)$$

The resulting approximate cost function is in the quadratic form. Setting the derivative to zero gives:

$$\mathbf{H} \boldsymbol{\delta} + \mathbf{b} = 0 \quad (2.32)$$

Then we can solve for the state increment $\boldsymbol{\delta}$ as:

$$\boldsymbol{\delta} = -\mathbf{H}^{-1} \mathbf{b} = -(\mathbf{J}_r^\top \boldsymbol{\Sigma}_r^{-1} \mathbf{J}_r)^{-1} \mathbf{J}_r^\top \boldsymbol{\Sigma}_r^{-1} \mathbf{r} \quad (2.33)$$

Finally, the state is updated with the optimized increment:

$$\mathbf{x} \leftarrow \mathbf{x} \oplus \boldsymbol{\delta} \quad (2.34)$$

Note that we use \oplus instead of $+$ because the state is not on the Euclidean space, but rather on the $SE(3)$ manifold. In order to keep a minimum representation of the pose and avoid singularities, $\boldsymbol{\delta}$ has 6 degrees of freedom and is expressed on the linear algebra ($\mathfrak{se}(3)$ and $\mathfrak{so}(3)$). The state update

with δ requires first calculating its exponential map, then doing matrix operations on the lie group:

$$\text{Exp}(\mathbf{x}) \leftarrow \text{Exp}(\mathbf{x}) \cdot \text{Exp}(\delta) \quad (2.35)$$

Levenberg-Marquardt Algorithm

Some techniques can further improve the optimization process. The quadratic approximation in the Gauss-Newton algorithm generally works well when the state is well initialized around the local minimum. However, it may become unstable and unable to find the minimum due to uncontrollable step sizes. Adding a damping parameter λ to the optimization helps control the step sizes and more reliably converge to the local minimum. The increment is now calculated as:

$$\delta = -(\mathbf{H} + \lambda \text{diag}(\mathbf{H}))^{-1} \mathbf{b} \quad (2.36)$$

or simply

$$\delta = -(\mathbf{H} + \lambda \mathbf{I})^{-1} \mathbf{b} \quad (2.37)$$

The value of λ is adjusted based on the error changes. It is decreased when error is reduced, making the next iteration closer to the Gauss-Newton quadratic approximation. It is increased when the step is not effectively reducing errors, balancing towards a simple gradient descent. In Chapter 3, we choose an upscale factor of 10 and a downscale factor of 2 for λ as our heuristic choice.

Robust Estimators

The error squares, or $L2$ norms, are dominated by large errors. The $L1$ norms take absolute values and scale linearly with error values, but are not differentiable. To keep the optimization more robust to outliers, robust estimators are often used to down-weight the large error terms (Huber, 2011). They are called M-estimators, which define new norm functions that are less sensitive to outliers and differentiable at the same time. Instead of minimizing the sum of squared errors, the cost function

is now written as:

$$E(\mathbf{x}) = \sum_i \rho(r_i(\mathbf{x})) \quad (2.38)$$

where ρ is the robust norm function. Optimizing this new cost function is equivalent to optimizing *weighted* least square errors, and the weights are functions of r_i . The most frequently used M-estimators include Huber, Cauchy, and Tucky estimators. In Chapter 3 we implemented the Huber's loss function in the optimization process. In Huber estimator, there is a threshold k for the absolute error terms. When $|r_i| < k$, the weight is 1. When $|r_i| \geq k$, the weight is $k/|r_i|$. k is commonly chosen to be 1.345σ , where σ is the standard deviation of all errors.

Random Sample Consensus

In addition to adjusting the weights placed on the residual terms, one can also choose to identify and reject outliers in the image processing step before optimization. A representative approach is RANSAC (Fischler and Bolles, 1981). We consider a point an outlier if it does not fit the probabilistic model. Often cases, we design an error threshold to determine if an observation is an inlier or outlier during RANSAC procedures.

In RANSAC, we first identify the minimum number of samples needed to deterministically solve for system parameters. Denoting this as the size of the minimum sample set M , we then find the number of sample sets k we need to have at least one sample set that has no outliers with probability p . If the probability of a sample being inlier is ϵ , we can derive k by:

$$k = \frac{\log(1 - p)}{\log(1 - \epsilon^M)} \quad (2.39)$$

We set k to be the maximum number of iterations in the RANSAC sampling process, and repeat the following steps: randomly select a minimum sample set; derive a solution using this set of samples, and count the number of inliers for this set; keep the set with a maximum number of inliers. We set a threshold for the sufficient number of inliers to stop the sampling process or repeat until k iterations have been executed. Once the sample set with the maximum number of inliers is found, optimize

using all the inliers with this sample set to obtain the final solution. RANSAC approach is used in Chapter 4 for outlier rejection from the network flow and depth outputs, where the optimization system is based on the motion field equation.

2.2.5. Neural Networks

Neural network is a statistical tool that can define a complex model and efficiently find the model parameters that best fit to the data through gradient descent. Consider linear regression which fits a linear model $f(\mathbf{x}) = \mathbf{x}\mathbf{W}$ to N input-output pairs $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i \in [1, N]}$. The parameters in this model are included in the weight matrix \mathbf{W} . Given the full range of data, the desired parameters can be solved using least-squares technique, which gives us analytical solution $\mathbf{W} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}$, where \mathbf{X} is of dimension $N \times |\mathbf{x}|$ and \mathbf{Y} is of dimension $N \times |\mathbf{y}|$. However, for highly non-linear functions, a closed-form solution may not be available. Iterative approaches are adopted, and neural network techniques are developed for modeling complex functions that are scalable to large datasets to suit the needs of modern computer vision tasks.

In a feed-forward neural network with a single hidden layer and an activation function $\sigma(\cdot)$, the network outputs

$$\hat{\mathbf{y}} = \sigma(\mathbf{x}\mathbf{W}_1 + \mathbf{b})\mathbf{W}_2 \quad (2.40)$$

given input \mathbf{x} . \mathbf{W}_1 and \mathbf{W}_2 are the weight matrices that scale the data between layer units, and \mathbf{b} is the bias on the hidden layer. In a fully-connected network, each layer operation is linear plus a nonlinear activation function. With multiple layers stacking together, the network can model highly nonlinear and complex functions with these simple building blocks. The most common activation functions include rectified linear (ReLU) (Agarap, 2018), sigmoid, tanh, and softplus (Ramachandran et al., 2017). These functions are continuous and keep the network differentiable, where *Stochastic Gradient Descent* (SGD) (Bottou, 2012) technique enables iterative optimization of the model parameters. A loss function is defined as the minimization objective, and $L1$ and $L2$ losses are commonly used. Regularization can be implemented to avoid overfitting. A common technique is *weight decay*, where the $L2$ norms of all network weights are added to the loss function multiplied by a scale factor, re-

stricting the network weights to a reasonable value.

A Convolutional Neural Network (CNN) (O’Shea and Nash, 2015) is particularly useful for processing spatial information, such as image processing tasks. It often consists of convolution layers and pooling layers. A convolution layer extracts spatial features from the image. The pooling layer reduces the resulting dimension of the output, compressing the data. Deep networks can model more complex functions, but they are harder to train. Some of the representative network architectures available include VGG (Simonyan and Zisserman, 2014) and ResNet (He et al., 2016). In Chapter 5, a 1D ResNet architecture is used to process segments of IMU data.

2.3. Deep Learning with Uncertainty Estimates

Despite its successes in many areas, deep learning approaches are often considered black boxes and lack performance guarantee. This adds uncertainties when incorporating them into visual-inertial state estimation systems and other safety-critical applications. It is of increasing significance to investigate techniques and algorithms for neural networks to estimate uncertainties to accurately model errors both from data noise and from data unseen in training.

There are two types of uncertainties to take into consideration in neural network modeling: epistemic uncertainty and aleatoric uncertainty (Kiureghian and Ditlevsen, 2009). *Epistemic (model) uncertainty* describes the uncertainty in the model, while *aleatoric (data) uncertainty* describes the inherent randomness in data. Together they inform the *predictive uncertainty* of the model output. Data uncertainty can be further categorized into *homoscedastic* and *heteroscedastic* uncertainties. The former stays constant for different inputs while the latter varies. For a neural network trained on a dataset $\mathcal{D} = \{\mathbf{x}_n, \mathbf{y}_n\}_{n=1, N}$ that maps the input \mathbf{x} to the output \mathbf{y} , the predictive uncertainty is represented by the conditional probability distribution $p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D})$. Two sources of uncertainties contribute to $p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D})$: the noise of the input (aleatoric uncertainty) and the imperfect modeling of the network (epistemic uncertainty). The model uncertainty can be represented as uncertainties on the network weights or architectures, learned as additional parameters. We want a network that has an expressive enough model while at the same time computes the desired $p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D})$ tractably.

The Bayesian probabilistic framework is founded on a theoretical background for reasoning uncertainties with evidence. For example, consider the problem of predicting how fast the Arctic ice cap is melting each year and how confident we are in this prediction with new satellite images as observations. A Bayesian framework defines the variables, functions, and uncertainties for the problem with a mathematical explanation of their relationship. Ideally, we want to combine the advantage of model complexity of neural networks with the mathematical elegance of a Bayesian framework. There are many such attempts with different algorithms. We will provide a brief overview of the existing probabilistic models and their connections to neural networks and focus on easy-to-use practical methods that are scalable to large datasets and complex models.

This section introduces the Bayesian framework and related terminologies, then surveys and discusses the current state-of-the-art uncertainty estimation approaches and benchmarks with deep learning.

2.3.1. Bayesian Modeling

In a Bayesian setting, we describe \mathbf{y} given \mathbf{x} as a parametrized function $\mathbf{y} = f_{\mathbf{w}}(\mathbf{x})$ with model parameters \mathbf{w} . Since \mathbf{y} is not deterministic, it would be natural to train the model parameters to predict the most probable \mathbf{y} from training data

$$p(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \prod_{n=1}^N p(\mathbf{y}_n|\mathbf{x}_n, \mathbf{w}) \quad (2.41)$$

$$\mathbf{w}_{ML} = \arg \max_{\mathbf{w}} \log p(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \arg \max_{\mathbf{w}} \sum_{n=1}^N \log p(\mathbf{y}_n|\mathbf{x}_n, \mathbf{w}) \quad (2.42)$$

where $p(\mathbf{y}|\mathbf{x}, \mathbf{w})$ is called the *likelihood function* and this is the *maximum likelihood estimation* (MLE) approach.

MLE is subject to overfitting and requires us to carefully choose the model complexity given how much data we have. Neural networks are usually overly expressive and subject to this problem. We can resolve this issue by adding a *prior distribution* $p(\mathbf{w})$ on the weights to constrain the model expressiveness under the same architectural design. We subsequently change our objective to finding

model parameters that most likely generate the data using the Bayes rule:

$$p(\mathbf{w}|\mathcal{D}) = p(\mathbf{w}|\mathbf{x}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x}, \mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{x})} \propto \prod_{n=1}^N p(\mathbf{y}_n|\mathbf{x}_n, \mathbf{w})p(\mathbf{w}) \quad (2.43)$$

$$\mathbf{w}_{MAP} = \arg \max_{\mathbf{w}} \log p(\mathbf{w}|\mathcal{D}) = \arg \max_{\mathbf{w}} \sum_{n=1}^N \log p(\mathbf{y}_n|\mathbf{x}_n, \mathbf{w}) + \log p(\mathbf{w}) \quad (2.44)$$

The probability $p(\mathbf{w}|\mathcal{D})$ is called the *posterior distribution*, and maximizing the posterior instead of the likelihood is called *maximum a posteriori* (MAP) approach.

However, both the MLE and MAP approaches can only give a point estimate of the local maximum instead of a distribution. We need the *Bayesian* approach, where we integrate over the posterior distribution to obtain the *predictive distribution*:

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}) = \int p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{w})p(\mathbf{w}|\mathcal{D})d\mathbf{w} \quad (2.45)$$

This requires us to obtain the posterior distribution, which requires the calculation of the term $p(\mathbf{y}|\mathbf{x})$ in (2.43):

$$p(\mathbf{y}|\mathbf{x}) = \int p(\mathbf{y}|\mathbf{x}, \mathbf{w})p(\mathbf{w})d\mathbf{w} \quad (2.46)$$

This is the *marginalization* process, a key characteristic of the Bayesian approach. The term $p(\mathbf{y}|\mathbf{x})$ is called *model evidence* or *marginal likelihood*, and obtaining the predictive distribution in (2.45) is called the *inference* process.

In a neural network, the posterior distribution denotes model uncertainty and can be represented in different forms, such as the distribution of network weights or latent variables. The general guideline in a Bayesian framework is to find the parameters of the network and the posterior by maximizing the marginal likelihood of the training data, and the predictive distribution can be calculated through integration. The calculation of the posterior can be difficult due to the intractable computation of the model evidence. It has a closed-form solution under Gaussian assumptions in Bayesian linear regression, but approximation is needed in more general settings.

2.3.2. Probabilistic Models and Practical Network Approaches

Analytical Solutions and Neural Linear Model

Using neural networks with uncertainty modeling has been studied extensively in the literature. Bayesian Neural Networks (BNN) is one of the most widely studied probabilistic neural models and offers a theoretical statistical framework. These models consider probabilistic network weights, often represented by Gaussian prior distribution and a point estimate for the bias. In a multi-layer BNN, the mapping function is highly nonlinear with respect to all the network weights. This makes the posterior and the predictive distribution intractable to compute. *Laplace approximation* (MacKay, 1992) is a complete Bayesian treatment for probabilistic model weights of a neural network. It makes a Gaussian approximation to the non-Gaussian posterior distribution due to non-linearity and linearizes around the mean of the posterior weights to obtain a Gaussian approximate prediction distribution. This method assumes a small covariance over the posterior so that the network function is approximately linear around the mode. They are easy to train from small datasets and models and provide uncertainty estimates that account for both aleatoric and epistemic uncertainties. However, these models are not scalable to deeper networks and large datasets that modern computer vision tasks require.

One simplification is to only model probabilistic weights in the last layer of the neural network, which effectively transforms the system to a linear one with a nonlinear basis. After which *Bayesian linear regression* can be applied, which has a closed-form solution for the posterior and predictive distribution under Gaussian assumption. This allows us to train a set of expressive nonlinear basis with deep networks and use the Bayesian linear regression model to express the posterior analytically. Moreover, the system is differentiable and can be trained efficiently. This class of approach is called *Neural Linear Models* (NLMs) (Snoek et al., 2015), and they are shown to be a reasonable approximation to full BNNs (Kristiadi et al., 2020). This approach is adopted in various areas such as automated machine learning (Zhou and Precioso, 2019), active learning (Pinsler et al., 2019), and depth completion (Qu et al., 2021), and a benchmark of NLM variants can be found in (Ober and Rasmussen, 2019).

Both Bayesian linear regression and Laplace approximation make Gaussian assumptions on the output distribution, where marginalization over the weights can be analytically solved. To model more general distributions, one effective way is to make use of *latent variables* to model complex distributions with simpler components. More specifically, using a latent variable z , we can express a complex predictive distribution $p(y|x)$ with a simple conditional distribution $p(y|x, z)$ through integration. Finding the posterior latent distribution can be solved by the *expectation-maximization* (EM) algorithm to maximize the likelihood function. A representative example is in *Gaussian Mixture Models* (GMM), where the latent variable is discrete. However, calculating the expectation can be analytically intractable and computationally expensive for high dimensional latent space or complicated posterior distribution.

There are two general classes of approximation schemes to work with such models: *stochastic* or *sampling* approach such as *Markov Chain Monte Carlo* (MCMC) to cover a complex posterior space, and *deterministic* approach which includes *variational inference* (VI) and *expectation propagation* (EP) that provide an approximation in the analytical form. We introduce each of them below.

Sampling and Markov Chain Monte Carlo

One essential tool to approximate the expectation of a complex distribution is sampling. It provides a practical solution to expectation by a finite sum over all samples. The difficulty of sampling approaches lies in generating independent samples that can represent a high dimensional complex distribution, and Markov Chain Monte Carlo (MCMC) is a stochastic technique that works well for such distributions.

Sampling approaches are resource-based approximations: the algorithm can give the exact result if an infinite number of samples are provided. It is computationally demanding to obtain well-distributed samples for accuracy, making these approaches limited to small-scale problems. *Hamiltonian Monte Carlo* (HMC) (Neal, 1995) is a variant of MCMC to generate samples from a complex posterior distribution. However, HMC requires calculating gradient over the whole dataset and does not scale to a large dataset. Stochastic Gradient Langevin Dynamics (SGLD) simplifies the approach by making a stochastic estimate of the gradient of the log-likelihood (Welling and Teh, 2011). This method tends

to generate samples within a single-mode, and similar to other MCMC approaches, the generated samples are correlated. This usually results in the need to generate a large number of samples in order to explore the distribution well.

Variational Inference and Variational Autoencoders

Deterministic approaches make analytical approximations to the complex posterior distribution, for example with a Gaussian. The closed-form approximation of the posterior distribution allows for optimization to maximize the data likelihood. It can be shown that maximizing the marginal likelihood probability is equivalent to minimizing the *Kullback-Leibler* (KL) divergence between the approximate posterior and the true posterior, which effectively maximizes the lower bound of the marginal probability.

Variational inference, or *variational Bayes* is a practical learning algorithm that improves the model log-likelihood by maximizing its lower bound, which involves calculating the expectation over the approximated posterior distribution that can be empirically determined through sampling. The technique of variational inference is first applied to BNNs by Hinton and Van Camp by assuming a factorized Gaussian form of the posterior distribution with independent weights (Hinton and Van Camp, 1993). They demonstrated a VI analytical optimization algorithm with single hidden layer networks. Later Barber and Bishop developed the idea by modeling correlations between weights to improve performance (Barber and Bishop, 1998). With the full covariance, the number of parameters to optimize grows quadratically with the number of model weights, which cannot meet most applications' scalability requirements. To address this limitation, Graves (Graves, 2011) first proposed to approximate the intractable expected log-likelihood using sampling, allowing the network to have more than one layer. Blundell et al. further built on Graves' work, incorporating the reparametrization technique introduced in (Kingma and Welling, 2013) and adding a mixture of Gaussian prior to the weights, increasing the model performance (Blundell et al., 2015). However, due to the Gaussian distribution, the number of parameters in the model is twice the number of network weights. The model is still not scalable enough.

Variational inference on a specified latent variable instead of the weights of the BNN can be tractable,

enabling the usage of complex neural network architecture for expressiveness and a simple latent distribution for practical computation and sampling. *Variational autoencoders* (VAE) (Kingma and Welling, 2013) and *conditional variational autoencoders* (CVAE) (Sohn et al., 2015) are practical methods applying the concept of VI to large neural networks. They have seen great success in image reconstruction and interpolation. Using a reparametrization trick, the sampling process can be made differentiable in the neural network to optimize all model parameters by training end-to-end with stochastic gradient descent.

Expectation Propagation and Lightweight Assumed Density Filtering

Expectation propagation (EP) is an alternative approach to VI but minimizes KL divergence in its reverse form. The approximate posterior is assumed to be a product of factors from the *exponential family*, from which it can be shown that the optimum solution corresponds to matching the moment statistics. Using this property, the algorithm updates the factors one by one by setting the moments equal to the empirical distribution of data. Unlike variational inference, EP is not guaranteed to converge. It tends to fit an average of different modes instead of collapsing into one mode, which works well in single-mode distributions but not so well for mixture models. However, its convergence usually produces more accurate results and is often used in logistic-type models.

Expectation propagation has shown improvement over variational inference in Hernandez-Lobato and Adams’s work using probabilistic backpropagation (PBP) (Hernández-Lobato and Adams, 2015). Noting that EP is optimizing for the VI objective KL-divergence but in its reverse form, a related algorithm uses α -divergence minimization (Hernandez-Lobato et al., 2016) where the divergence parameter α controls an interpolation between VI and EP. The paper also shows the better performance of a choice of α in between. Usually, several passes through the factors are made in the optimization, and a special case of EP is *assumed density filtering* (ADF) which makes only a one-time pass through all the factors (Ghosh et al., 2016). Similar to VI on BNNs, the number of parameters doubles the number of network weights, making the algorithms less scalable to large applications.

Gast and Roth (2018) proposes a lightweight approach to approximate the posterior only on the activation distributions and keep the weights as point estimates, with the only additional requirement

that the noise parameter needs to be specified in the input for the propagation of uncertainties. This approach avoids doubling the number of parameters and has been applied successfully in robotics applications with large networks.

Normalizing Flow

Normalizing flow is another type of probabilistic modeling that can represent the distribution of data and enable sampling of data at the same time. This approach trains a differentiable and invertible function through neural network design that maps high dimensional input data to a continuous latent variable. In normalizing flow, the latent variable has a standard normal distribution. The differentiable function enables estimation of the data likelihood given the probability density function of the latent, and the invertibility enables sampling from the latent space. This general approach can map any smooth data distribution to any smooth latent distribution and has been widely adopted in computer vision for applications such as image morphing.

Through network architecture design, invertibility and recursive dependency of variables can be achieved while enabling parallel computation. A representative architecture is *autoregressive models*, and in flow networks called *autoregressive flows* (Huang et al., 2018). In order for the method to be practical, the Jacobian determinant needs to be easy to compute and differentiate, and such frameworks are explored in NICE (Dinh et al., 2014) and RealNVP (Dinh et al., 2016). Similar to variational inference, flow networks can also be generalized to conditional models (Winkler et al., 2019).

Non-Bayesian Approaches: Predictive, Ensembling, and MC-Dropout

Other than these approaches based on the Bayesian framework that we have been discussing so far, the current literature also moves towards less strictly Bayesian methods for practical usage. We will discuss a few of these approaches that have been widely used in applications for their simplicity and effectiveness.

Predictive approach directly outputs the mean and covariance of a parametric distribution by training with negative log-likelihood (NLL) loss (Nix and Weigend, 1994). The simplest example would be

to output the mean and the variances assuming off-diagonal terms of the covariance matrix are zeros, which results in a simple modification to double the output dimension. This approach only models aleatoric uncertainty since the network parameters are fixed after training. This simplistic approach has been shown to produce statistically consistent uncertainty estimates in real-world applications (Liu et al., 2020).

Ensembling approach trains multiple models independently and uses the collection of results from these models to infer the mean and variance of the estimate. There are two primary ensemble training methods. *Bootstrapping* (Lakshminarayanan et al., 2017) trains multiple models with different initializations: either with random initial weight values or with different initial training datasets. Each model is trained entirely from scratch. *Snapshot* (Huang et al., 2017) only trains once and saves multiple copies of model weights at different stages of training using controlled learning rate scheduling techniques, known as SGDR learning scheme (Loshchilov and Hutter, 2017). These methods are straightforward and achieve high performances in various experimental settings (Gustafsson et al., 2020; Poggi et al., 2020; Ilg et al., 2018), but require saving multiple models and performing inference on all of them at test time. This approach only estimates epistemic uncertainty. One can change the network to predictive models to incorporate data uncertainty and combine the variances from different models and the network outputs.

According to Gal and Ghahramani (2015), dropout Neural Networks are identical to variational inference in GP. The paper proposes training different networks with dropout as Monte Carlo sampling from Bernoulli distributions of node connections. This approach makes a drastic simplification of the variational inference framework on BNNs with a simple dropout procedure (Gal and Ghahramani, 2016). The inference process at test time has to sample from the same Bernoulli distribution (with the same dropout rate), giving a set of sampled models similar to the ensembling approach and models epistemic uncertainty. Similarly, dropout can also be combined with predictive networks to estimate data and model uncertainties jointly, and Gal (2016) shows superior results by combining the two approaches for the monocular depth estimation task on the NYUv2 Depth dataset.

To conclude, we introduced a variety of representative uncertainty estimation approaches in the

literature with deep neural networks. It is worth noting that there are many different variants of the uncertainty estimation approaches, as either a combination of the approaches we have discussed (Loquercio et al., 2020) using different architectures (Ilg et al., 2018) or adding more application-specific modifications. However, the underlying principles are the same, and we can often draw ideas from the probabilistic frameworks and algorithms we have discussed. The following section will discuss some available benchmark comparisons of their performances.

2.3.3. Evaluation Metrics and Benchmarks

Finding a single good metric to evaluate the performance of uncertainty estimation is hard, and multiple metrics have been proposed.

One evaluation technique is the *sparsification plots* (Ilg et al., 2018), where the errors of the estimates are sorted by their uncertainty estimates, and the average error is computed repeatedly while removing fractions of errors from the largest variance to the lowest. The resulting curve is called *sparsification curve*. An *oracle curve* is obtained by computing the error metrics according to the correct sorting order of the errors. The *sparsification error* is defined as the area difference between the sparsification curve and the oracle curve, called *area under the sparsification error curve* (AUSE). This metric only estimates the *relative* accuracy of uncertainty - the metric does not report over-confident uncertainties as long as they are in the correct relative order to the errors.

An *absolute* error metric for uncertainty estimation is proposed in Gustafsson et al. (2020) using the *expected calibration error* (ECE). A well-calibrated model should not be over-confident or over-conservative. Suppose a model outputs the mean μ and variance σ^2 under the Gaussian assumption. In that case, the prediction intervals can be constructed by $\mu \pm \Phi^{-1}(\frac{p+1}{2})\sigma$ where Φ is the CDF of a standard normal distribution. For computer vision tasks, each pixel has an output distribution estimate. The ECE error curve is constructed by moving the confidence p from 0 to 1 and finding the proportion of pixels falling within the prediction interval. The oracle curve for this metric is exactly p (a straight line from 0 to 1). The area in between the two curves is called *area under the calibration error curve* (AUCE). With AUSE and AUCE, together with the usual RMSE and NLL loss used for training, we can obtain a more comprehensive understanding of the performance of

uncertainty estimation.

The comparison metrics evaluate the accuracy of the predictive distribution but cannot show the distinction between aleatoric and epistemic uncertainties. Modeling epistemic uncertainty comes at a higher cost, and Kendall and Gal (2017) argues that data uncertainty plays a more critical role than model uncertainty. However, an accurate estimate of the epistemic uncertainty benefits the application of networks in safety-critical systems since it models the distribution of the training data on the latent space. The toy regression example in Gustafsson et al. (2020) generates a small set of data along a ground truth curve with Gaussian noise. It compares a predictive neural network that captures the aleatoric uncertainty to a Gaussian mixture model using approximate Bayesian inference that captures both aleatoric and epistemic uncertainties. The result shows that the Gaussian mixture model can account for the errors in modeling that the predictive neural network cannot.

Gustafsson et al. (2020) also compares ensembling and MC-dropout to other approximate Bayesian methods. The error metric is the KL-divergence between the predicted distribution and the actual Gaussian distribution used to generate the toy dataset. For small-scale problems, HMC (Neal et al., 2011) is often considered the golden standard for approximating inference. However, due to its scalability issues to larger datasets and complex models, two MCMC variants are considered instead: SGLD and SGHMC. The result shows that the ensembling method provides better approximations than stochastic gradient MCMC variants such as SGLD. The paper also shows that ensembling consistently outperforms MC-dropout for depth completion tasks on the KITTI dataset. A similar result has also been shown in Poggi et al. (2020) for depth estimation on the KITTI dataset. On optical flow estimation, Ilg et al. (2018) presents a comprehensive comparison of the performances of predictive, dropout, and different ensemble types, and the predictive method is shown as the best approach to model aleatoric uncertainty.

We observe that these simple-to-use approaches (predictive, ensembling, and MC-dropout) - with off-the-shelf adaptation from existing deep neural networks with minimum changes - do not strictly follow a Bayesian probabilistic framework yet produce competitive or even better results compared to Bayesian approaches. However, in these approaches, the model uncertainty and data uncertainty

are often disconnected. They lack the explanation of the relationship between uncertainties that a Bayesian framework provides. This can cause failures that cannot be explained for unseen datasets. Most of the available benchmarks compare non-Bayesian approaches to MCMC variants only, and we need more comprehensive comparisons that include more competitive Bayesian approaches such as variational inference. However, these methods are often popular within their specific research domain. In addition, some methods require expertise to implement and use, which adds to the difficulty of obtaining a benchmark result. There is still a gap between deep learning and Bayesian modeling, and combining the advantage of network model expressivity and Bayesian principles is still an open research question.

In Chapter 5, we will adopt the predictive approach to model uncertainty with a Gaussian distribution for the estimated 3D displacement from networks. This method requires very little modification to the network training process and is proven effective with experimental results. However, since epistemic uncertainty is not modeled with this approach, it is shown that this method fails to detect scenarios drastically different from the training set.

CHAPTER 3

SEMI-DENSE VIO ON CONSTRAINED PLATFORMS

The first case of information deficiency that we investigate is operation in low texture environments with limited computation. A practical scenario in robotic tasks is infrastructure inspection. As vision is the dominant source of information in visual-inertial odometry, not having enough visual features can lead to tracking failure. This difficulty can often be alleviated by using a dense or semi-dense visual front-end, which makes use of not only pixels of extracted corners but also pixels with gradients, giving them superior performance in low texture areas. However, these methods are computationally more expensive. As the platform’s scale shrinks down, the available computation of the onboard CPU becomes limited, making autonomous navigation using optimization-based semi-dense tracking a challenging problem.

What we present in this chapter is a direct semi-dense stereo VIO algorithm working on computationally constrained platforms, in particular, for quadrotor platforms under Size, Weight and Power (SWaP) constraints. We address the challenges of operation in low-texture environments and explore a simple semi-dense VIO framework that works for a smartphone-grade processor while leaving enough CPU room for control and planning that enables real-time autonomous flight. We show that this direct semi-dense VIO method performs significantly better than sparse methods in low texture conditions encountered in indoor navigation with a low texture dataset obtained with our quadrotor platform. Our method takes less CPU than the state-of-the-art semi-dense method, VI-Stereo-DSO, due to a simpler framework in the algorithm and a multi-threaded code structure allowing us to run real-time state estimation on an ARM board. Our direct semi-dense VIO performs comparably to other state-of-the-art methods while taking less CPU than other optimization-based approaches, making it suitable for computationally-constrained small platforms. The proposed approach is validated through experiments on a 250 g, 22 cm diameter quadrotor equipped with a stereo camera and an IMU, and we demonstrate autonomous flight with the proposed state estimation on this platform.

3.1. Introduction

State estimation and mapping are essential perceptual building blocks of an autonomous robotic system. For Micro Aerial Vehicle (MAV) platforms under SWaP constraints, real-time operation on a cellphone-grade processing unit such as an ARM board is often required. The low cost and small form factor makes the fusion of cameras and Inertial Measurement Unit (IMU) sensors the preferred option for SWaP-constrained platforms. Approaches for visual-inertial navigation have been well studied in the literature (Gui et al., 2015). With precision tracking from high resolution images combined with high frequency IMU data tracking aggressive movements, this complementary sensor suite provides accurate and robust state estimates sufficient for most autonomous flight operations.

However, there are situations where feature tracking becomes difficult that pose challenges to the state estimation algorithm running on a small board. In this chapter we especially place focus on low texture scenarios with computationally constrained platforms. Nuclear reactor inspection is one of the extreme cases of such application scenarios, and Thakur et al. (2018) tried to tackle the challenge of inspecting Fukushima Daiichi Nuclear Power Plant, where the size of the vehicle is severely constrained by the access passage in the facility and the low texture appearance inside the plant makes feature tracking difficult. This work explores the benefits and trade-offs of Visual-Inertial Odometry (VIO) algorithms and engineering design choices under such constraints.

Existing VIO approaches can be categorized into (a) feature-based or indirect methods (Qin et al., 2018b; Mourikis and Roumeliotis, 2007) and (b) photometric or direct methods (Von Stumberg et al., 2018; Bloesch et al., 2017). Compared to feature-based methods which use geometric feature reprojection errors, direct methods use pixel intensity errors without requiring any feature detector. This enables the use of a large number of pixel observations including edge and gradient information in the same optimization framework. As a result, direct methods are more robust in less textured conditions (Engel et al., 2018) as they are capable of using as much information as possible from the image.

The visual front-end of VIO can also be divided into sparse, dense, and semi-dense categories based

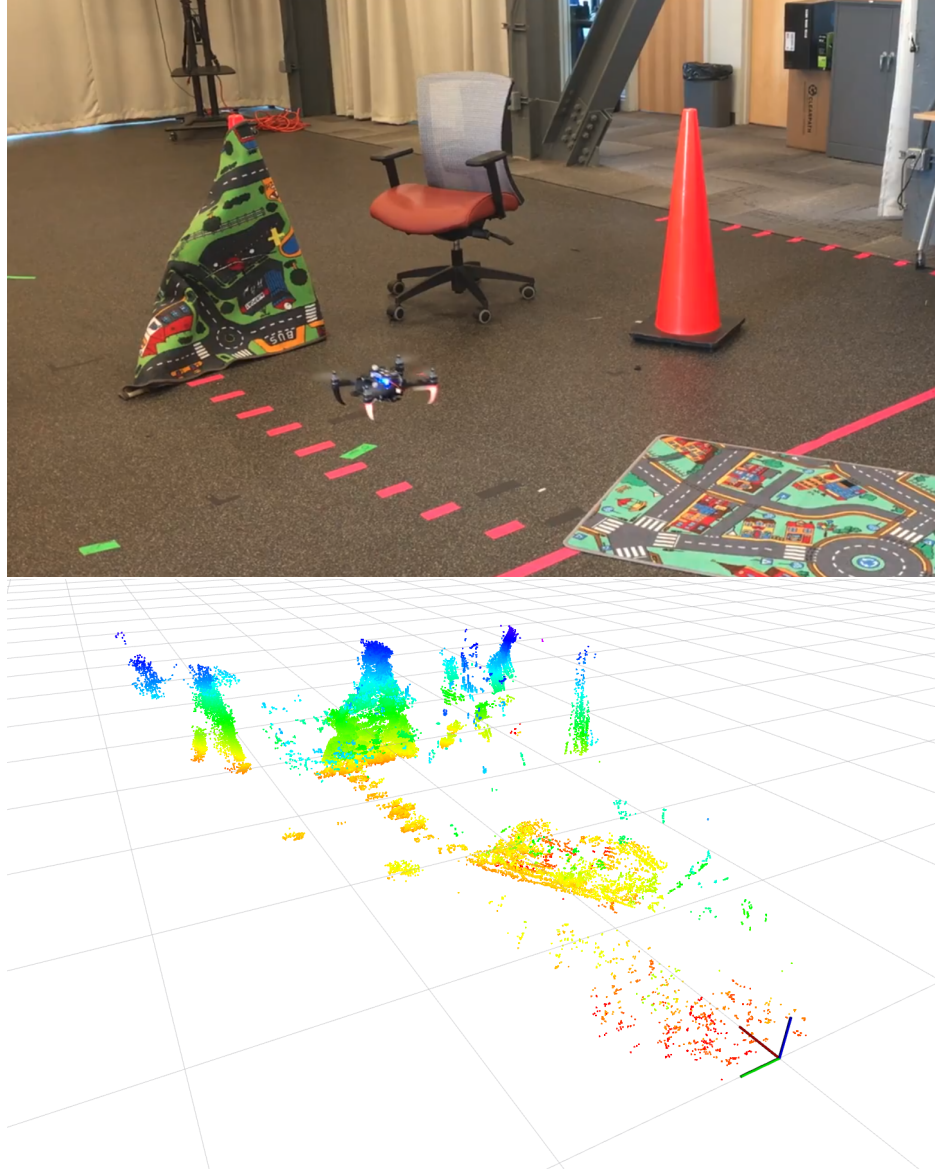


Figure 3.1: The semi-dense map created by the vehicle while following a square trajectory. The picture on top shows a snapshot of the environment. The carpet on the ground, the chair, and the cone to the right can be clearly seen in the semi-dense map.

on the number of visual measurements used for tracking. Sparse methods carefully select a small number of selected feature points to track, while dense methods take into account every pixel in the image yet are computationally expensive. Semi-dense methods use only thresholded high gradient pixels, providing a leverage to control the required computation by the custom design of the threshold. In addition, a semi-dense map can be obtained as a by-product of the state estimation process and used for planning by the robot. Figure 3.1 shows a snapshot of our quadrotor platform flying autonomously in an indoor environment with our semi-dense direct VIO running onboard and the map it creates.

Although direct semi-dense methods have the unique benefits of robust tracking in low texture areas and creating maps, very few works have addressed the problem of flying aerial vehicles based on state estimates from these approaches, particularly for small MAVs under computational constraints. In this work, we investigate the implementation of a tightly-coupled direct semi-dense VIO approach suitable for smartphone-grade platforms to enable fully autonomous flight with onboard processing, and we show the trade-off between computation and accuracy of different modules in this framework on a small quadrotor with closed-loop control. In this chapter, we

- develop a semi-dense, direct visual-inertial odometry and mapping system using a small baseline stereo camera that runs in real time on a smartphone grade processor.
- show that our semi-dense direct VIO implementation has the accuracy comparable to other state-of-the-art methods while being efficient enough to run on an ARM processor. We validate that our method is more robust in low textured areas using our low texture datasets.
- analyze the various parameters affecting the system performance and discuss the trade-off between accuracy and CPU usage. We also explain our multi-threaded implementation which is an important factor that enables real-time performance on the computationally-constrained board.
- demonstrate autonomous flight of a small and lightweight quadrotor system using the developed VIO system for state estimation.

- provide the code and the complete low texture dataset for educational purposes.

The rest of this chapter is organized as follows. Section 3.2 briefly discusses the related literature. Section 3.3 provides a high-level overview of the state estimation and control system. Section 3.4 presents in detail our VIO algorithm. Section 3.5 looks into implementation details and runtime analysis on the SWaP-constrained platform. Section 3.6 evaluates the algorithm accuracy and computation usage on the EuRoC Datasets and demonstrate autonomous flight performance on the small platform. Section 3.7 analyze the performance on a low-texture dataset collected by our platform. Finally in Section 3.8 we conclude with some key observations in this work.

3.2. Related Works

Visual-Inertial Odometry is widely used in robotics as the basis for autonomous navigation and there is a variety of solutions to this problem. Tracking between camera frames requires visual features, which are characterized by spatial changes in pixel intensity. Based on the cost function, visual tracking can be divided into direct and indirect methods. Indirect methods explicitly extract image features using feature detectors such as FAST (Bloesch et al., 2017) or ORB (Mur-Artal and Tardós, 2017) and use geometric reprojection errors as the cost function for image registration (Hartley and Zisserman, 2003). These methods often have more control over the precision of the detected features, especially with recent advances on descriptors more robust to viewing angles and lighting conditions and robust outlier rejection through RANSAC (Zhang et al., 2020). Direct methods on the other hand, use the corresponding pixel intensity difference as the cost for tracking (Irani and Anandan, 1999). These methods allow for semi-dense (Engel et al., 2013) and dense (Newcombe et al., 2011) approaches which can leverage large amount of pixel information based on gradients, making it a compelling choice for low texture environments using RGB cameras. The sparsity of the tracked features can be freely adjusted (Engel et al., 2018), and optimizing for photometric error is also less computationally expensive than frame-to-frame feature matching (Forster et al., 2017b).

Direct methods make use of raw intensity values of the image which are easy to compute, but they come with certain pitfalls. Intensity value for one pixel is sensitive to lighting condition changes and by itself does not describe any properties of a local feature. There are methods to compensate for the

illumination changes by applying local contrast normalization (Irani and Anandan, 1999), modeling global affine lighting conditions in the optimization (Engel et al., 2015), or perform photometric calibration to further maintain brightness consistency across the image space (Engel et al., 2016). We show in this work how the photometric calibration for vignetting effect improves tracking on low texture datasets. Photometric cost function is also highly non-convex since image gradient is defined locally in a small window. Using a pyramid image structure that starts tracking from lower image resolution increases the region of convergence, and integrating IMU measurements instead of using constant velocity model provides a better initial guess (Usenko et al., 2016). This can be solved by tightly coupling vision and inertial measurements into one optimization framework, making the algorithm more robust to intense movement, especially rotations. Comparing to loosely-coupled approaches (Weiss et al., 2012; Meier et al., 2011), a tightly-coupled VIO system is less prone to tracking failure and saves computation by reducing the number of iterations in optimization.

In this work we extend our previous loosely coupled approach (Liu et al., 2018) to a tightly-coupled one which significantly improves the accuracy and robustness of the algorithm. There are a few other works on tightly-coupled direct semi-dense VIO (Usenko et al., 2016; Xu et al., 2018), however our approach is specifically designed for a system under SWaP constraints. We show how a simple direct semi-dense VIO algorithm can provide comparable tracking accuracy to the state-of-the-art, while saving computation to enable state estimation and control to run smoothly on a smart-phone grade processor in real time. An interesting work using semi-dense VO on a smartphone (Schöps et al., 2014) shows the ability of using semi-dense methods for computationally constrained CPU for hand-held devices, and we extend this direction by tightly coupling IMU measurements and demonstrate a semi-dense VIO on-board a small-scale autonomous flying vehicle.

3.3. System Overview

We use a forward-facing stereo camera pair and an IMU as our sensor suite for state estimation. The tightly-coupled VIO algorithm estimates poses at camera rate of between 10 Hz to 30 Hz, and we obtain 500 Hz state estimates of the vehicle with IMU propagation to enable on-board control. The position controller uses the estimated state as feedback to follow trajectories which are the outputs

of a high-level trajectory planner. In earlier works, a backstepping approach was used for quadrotor control because the attitude dynamics are faster than the dynamics governing the position, and linearized controllers were used for both loops (Michael et al., 2010; Weiss et al., 2011; Hérissé et al., 2012). However, we need the system to be capable of large deviations from the hover configuration to take advantage of the agility that is inherent with small platforms. Therefore, we use a nonlinear controller from Lee et al. (2010) which has a much larger basin of attraction.

The tightly-coupled VIO system combines both photometric and IMU residuals into one cost function and jointly optimizes for the full state which includes the 6 DoF pose and the linear velocity of the vehicle. We make use of the concept of keyframes to reduce drift (Klein and Murray, 2009). Instead of constantly tracking between adjacent frames, we select some frames heuristically to be keyframes from which a window of frames are tracked with respect to the keyframe. The criteria on switching to a new keyframe are detailed in Section 3.4.2.

A newly captured image is first processed with Gaussian blur and sub-sampling to form an image pyramid to perform coarse-to-fine optimization for better convergence. Pixels to track are selected only from the keyframes. While the visual residuals are defined between the current frame and the last keyframe, the IMU residuals are defined between adjacent frames using IMU preintegration technique detailed by Forster et al. (2017a). Then we optimize for the state that minimizes the combination of both residuals when each new camera frame is captured. In our system, only the left image is used for tracking, and we use the stereo pair for depth.

In our previous work we used the Inverse Compositional Approach (ICA) (Baker and Matthews, 2004) where the Jacobian depends on the gradient of the keyframe, and therefore the Jacobian only needs to be calculated once for each optimization. In comparison, the Forward Compositional Approach (FCA) calculates the Jacobian based on the gradient in the current frame, and the Jacobian needs to be re-calculated in every iteration as the reprojection errors change. Since we are solving for the pose of the current frame with an IMU residual on its adjacent frame, despite the computational efficiency of ICA method, we use FCA to avoid accumulating the noisy IMU measurements through the keyframe window and to make sure that the information from IMU is not reused.

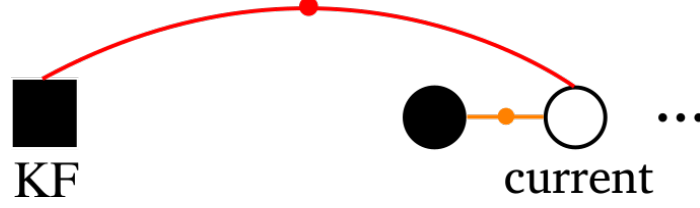


Figure 3.2: Factor graph of the system. The solid square represents the latest image keyframe, and the solid circle represents the frame adjacent to the current frame. The system solves for the state of the current frame, with an image alignment factor (red) to the keyframe and an IMU pre-integration factor (yellow) to the last frame.

This can be better illustrated with a factor graph of the system shown in Figure 3.2. In FCA, the optimization happens on the current frame, while in ICA the optimization is on the pose of the keyframe assuming a fixed current frame. In order to use ICA with the IMU factor, one must accumulate the IMU measurements from the keyframe to the current frame, which is prone to integration noise. The same IMU measurements from past frames will also be used many times whenever a new frame is added. Therefore we adopt FCA in the proposed approach. Also note that because of the simplistic framework without windowed optimization, the IMU biases cannot be estimated accurately (see Section 3.6.2). This is a drawback in this approach as a fixed offline calibration is used, which limits the accuracy of the algorithm.

3.4. Semi-Dense Direct Visual-Inertial Odometry

3.4.1. Optimization framework

We define the frame of the IMU to be the body frame, and we define the world frame to be the same as the initial body frame. Our goal is to optimize for the relative transformation between the current body frame and the world frame at IMU rate. We use an optimization framework to solve the state estimation problem. We formulate the visual-inertial state estimation problem as a maximum a posteriori estimation problem, from which we derive the cost function for optimization. We define the state of our system as:

$$\mathbf{x} \doteq \{ {}^w\mathbf{R}_b, {}^w\mathbf{p}_b, {}^w\mathbf{v}_b \} \quad (3.1)$$

where ${}^w\mathbf{R}_b$, ${}^w\mathbf{p}_b$ and ${}^w\mathbf{v}_b$ are the rotation, position and velocity of the current body frame in the world frame. In comparison to most other optimization and filter frameworks (Forster et al., 2017a; Engel et al., 2018), our method only optimizes the state of the current frame instead of a window of frames to save computation. Because we have a calibrated stereo system to provide good depth estimates, we leave pixel depths out of the state. We show with experiments that even though our state space consists of only one frame, the method is able to perform tracking accurately enough to enable autonomous flight of a quadrotor system.

As the state is not defined on the Euclidean space, we find the minimum on the manifold by optimizing on its tangent space around the current estimate. We optimize for the error state where the rotation is defined on the tangent space in the body frame:

$$\delta\mathbf{x} \doteq \{\delta\phi, \delta\tilde{\mathbf{p}}, \delta\tilde{\mathbf{v}}\}, \quad (3.2)$$

and update the state \mathbf{x} as follows:

$$\begin{aligned} {}^w\mathbf{R}_b &\leftarrow {}^w\mathbf{R}_b \cdot \text{Exp}(\delta\phi) \\ {}^w\mathbf{p}_b &\leftarrow {}^w\mathbf{p}_b + \delta\tilde{\mathbf{p}} \\ {}^w\mathbf{v}_b &\leftarrow {}^w\mathbf{v}_b + \delta\tilde{\mathbf{v}}. \end{aligned} \quad (3.3)$$

We adopt a keyframe-based approach (Klein and Murray, 2009) for visual tracking. We denote the set of measurements in our probabilistic model as $\mathcal{Z} = \{\mathcal{C}, \mathcal{I}\}$, where \mathcal{C} represents the image measurements from the latest keyframe, which consists of the high gradient pixel measurements $z_l \in \mathcal{C}$ for each pixel l in the keyframe that is also observed in the current frame. The IMU measurements are denoted as \mathcal{I} , which consists of the gyroscope and accelerometer measurements between the last frame and the current frame.

The camera and IMU measurements are independent and the measurements for different pixels are assumed to be uncorrelated. Thus, we can express the posterior probability of the state \mathbf{x} given

measurements \mathcal{Z} and prior $p(\mathbf{x}_0)$ as,

$$\begin{aligned} p(\mathbf{x}|\mathcal{Z}) &\propto p(\mathbf{x}_0)p(\mathcal{Z}|\mathbf{x}) = p(\mathbf{x}_0)p(\mathcal{C}, \mathcal{I}|\mathbf{x}) \\ &= p(\mathbf{x}_0)p(\mathcal{I}|\mathbf{x}) \prod_l p(z_l|\mathbf{x}). \end{aligned} \quad (3.4)$$

The vision measurement \mathcal{C} and IMU measurement \mathcal{I} depends on the states of the latest keyframe and last image as well, but because they are not included in our state space and only the current frame state is being optimized, we consider them fixed and exclude them from the probability formulation. This means that after the state of an image frame is estimated, we do not make adjustments to it anymore. We do this out of consideration of computational constraints and show the stability of the algorithm with experiments.

The goal is to find the optimum state \mathbf{x} that has the maximum log likelihood. Under the assumption of white Gaussian noise, it is the same as minimizing the sum of squared residuals:

$$\begin{aligned} \mathbf{x}^* &= \arg \max_{\mathbf{x}} p(\mathbf{x}|\mathcal{Z}) \\ &= \arg \min_{\mathbf{x}} (\|\mathbf{r}_{\mathcal{C}}\|_{\Sigma_{\mathcal{C}}}^2 + \|\mathbf{r}_{\mathcal{I}}\|_{\Sigma_{\mathcal{I}}}^2), \end{aligned} \quad (3.5)$$

where $\mathbf{r}_{\mathcal{C}}$, $\mathbf{r}_{\mathcal{I}}$ are measurement residuals and the total cost is the sum of the Mahalanobis norms of these residuals with $\Sigma_{\mathcal{C}}$, $\Sigma_{\mathcal{I}}$ representing the corresponding covariance matrices of the distributions. We consider the prior distribution a constant and we do not include prior residual in the optimization. We elaborate the vision and IMU residuals in the following subsections.

3.4.2. Visual factor

Upon receiving a new frame from the camera, the image is first rectified, and based on a set of heuristics determined if it is selected to be a new keyframe. In practice, a new keyframe is created if any of the following conditions are met: 1) the percentage of points visible during tracking falls below 70%; 2) the number of frames processed without creating a new keyframe exceeds 50; 3) the angle between the current frame and the last keyframe exceeds 0.2 rad; 4) the translation from last keyframe exceeds 10% of the average scene depth.



Figure 3.3: The left image shows the grayscale image, and the right image shows only the high gradient pixels from feature extraction.

In our framework, feature extraction and registration are performed only on the keyframes. Feature extraction selects the high gradient pixels by computing the gradient and applying an adaptive threshold based on local gradient regions to select the pixels whose gradient is higher relative to their neighbors. An illustration of pixel selection is shown in Figure 3.3. Each pixel i selected contributes to a visual measurement z_i . The depths of the selected pixels are obtained using a block matcher on the calibrated stereo camera images. Here, we abuse the word “feature” to describe a single high gradient pixel without feature descriptors.

We find the photometric error of the feature points between their projections on the keyframe and the current frame. Linear interpolation is used to approximate the intensities of the projected points in sub-pixel coordinates, and we only take into account the feature points that fall into the field of view of the current frame. We implemented our optimizer using Levenberg-Marquardt algorithm to optimize for speed and performance and to have adaptive Jacobian sizes during the iterations of the optimization.

We denote the keyframe image by I^* and the current frame image by I . The photometric residual r_i for a single pixel i with depth d_i is defined as follows:

$$r_i(\mathbf{x}) = I^*(\mathbf{u}_i) - I(\pi(({}^b\mathbf{T}_{\text{kf}})_c \pi^{-1}(\mathbf{u}_i, d_i))). \quad (3.6)$$

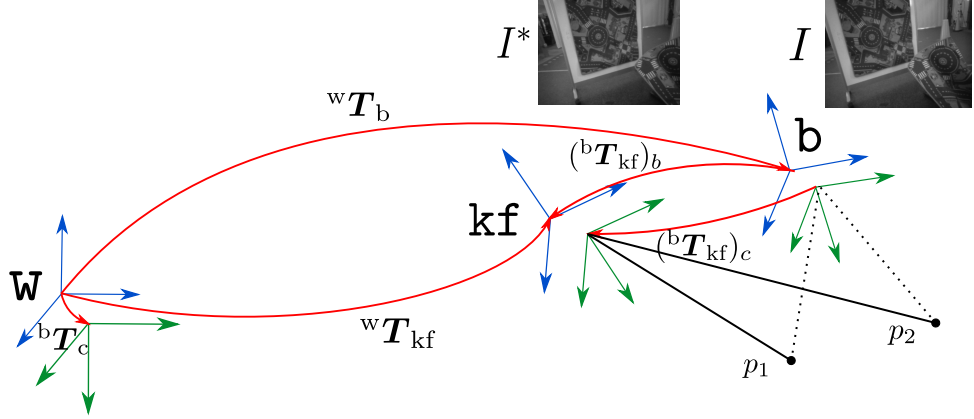


Figure 3.4: Coordinate system illustration. This figure shows the initial body frame as the world frame w , pose of the last keyframe kf and the current body frame b . Blue and green frames represent body and camera frames respectively, and they are rigidly attached with relative transformation bT_c in $SE(3)$.

Here u_i is the pixel coordinate in the keyframe and π^{-1} projects a point on the image plane with its depth information to 3D using pinhole camera model. $({}^bT_{kf})_c$ is the relative camera pose transformation from the keyframe to the current frame in $SE(3)$. Since the camera is rigidly mounted on the body frame with transformation bT_c , we use the adjoint to switch to body frame representation and write $({}^bT_{kf})_c$ as a function of wT_b in the state:

$$({}^bT_{kf})_c = {}^bT_c^{-1} ({}^bT_{kf})_b {}^bT_c = {}^bT_c^{-1} {}^wT_b^{-1} {}^wT_{kf} {}^bT_c. \quad (3.7)$$

We consider ${}^wT_{kf}$ as fixed in the prior. Figure 3.4 visualizes the transformation and coordinate systems.

The photometric cost is defined as the Mahalanobis norm of the error vector r_c , containing the error terms of all the tracked pixels, weighted by the inverse covariance matrix Σ_c^{-1} :

$$\|r_c\|_{\Sigma_c}^2 = r_c^T \Sigma_c^{-1} r_c. \quad (3.8)$$

Note that because the residual is defined on pixel intensities, Σ_c is not directly obtainable. There are two sources of noise that contribute to the photometric error: the image intensity noise coming from

pixel sensors and the disparity noise coming from the stereo block matcher. The image intensity noise is additive to the residual, however the function from disparity to pixel intensity is nonlinear. In order to model the noise as Gaussian, we assume that the noise on disparity is small such that the projected point with noise still falls within the same gradient region, then approximate the uncertainty of the intensity residuals by a gradient-dependent weight ω_u as the diagonal elements of Σ_c^{-1} that down-weights pixels with high gradient. We adopt the formulation in Engel et al. (2018) with a constant coefficient c and $G(u)$ represents the gradient value of a pixel u :

$$\omega_u = \frac{c^2}{c^2 + \|G(u)\|_2^2}. \quad (3.9)$$

However, stereo matcher produces lots of erroneous depth estimates. We use Huber weights to reduce the effect of outliers. The final information matrix Σ_c^{-1} has diagonal elements as the product of the Huber weights and the gradient weights. The off-diagonal entries are set to zeros.

3.4.3. Inertial factor

The discrete IMU measurement includes linear acceleration \mathbf{a} and angular velocity $\boldsymbol{\omega}$ expressed in the IMU frame. The measurement is modeled as the sum of the true values with white noise $\boldsymbol{\eta}$ and sensor bias \mathbf{b} :

$$\boldsymbol{\omega} = \boldsymbol{\omega}_{\text{true}} + \mathbf{b}^g + \boldsymbol{\eta}^g, \quad \mathbf{a} = \mathbf{a}_{\text{true}} + \mathbf{b}^a + \boldsymbol{\eta}^a. \quad (3.10)$$

From the kinematic motion model of rotation, velocity and position

$${}^w\dot{\mathbf{R}}_b = {}^w\mathbf{R}_b [\boldsymbol{\omega}]_{\times}, \quad {}^w\dot{\mathbf{v}}_b = {}^w\mathbf{R}_b \mathbf{a} + \mathbf{g}, \quad {}^w\dot{\mathbf{p}}_b = {}^w\mathbf{v}_b, \quad (3.11)$$

we have the state propagation with IMU measurements:

$$\begin{aligned} \mathbf{R}_{k+1} &= \mathbf{R}_k \text{Exp}((\boldsymbol{\omega}_k - \mathbf{b}_k^g - \boldsymbol{\eta}_k^g)\Delta t) \\ \mathbf{v}_{k+1} &= \mathbf{v}_k + \mathbf{g}\Delta t + \mathbf{R}_k(\mathbf{a}_k - \mathbf{b}_k^a - \boldsymbol{\eta}_k^a)\Delta t \\ \mathbf{p}_{k+1} &= \mathbf{p}_k + \mathbf{v}_k\Delta t + \frac{1}{2}\mathbf{g}\Delta t^2 + \frac{1}{2}\mathbf{R}_k(\mathbf{a}_k - \mathbf{b}_k^a - \boldsymbol{\eta}_k^a)\Delta t^2. \end{aligned} \quad (3.12)$$

We omit the superscripts and subscripts of coordinate frames and Δt denotes the time interval between two consecutive IMU measurements.

To form a constraint between two image frames i and j , we need to integrate the IMU measurements in between. Following (3.12) and iterating through every IMU measurement k between i and j , we have:

$$\begin{aligned} \mathbf{R}_j &= \mathbf{R}_i \prod_{k=i}^{j-1} \text{Exp}((\boldsymbol{\omega}_k - \mathbf{b}_k^g - \boldsymbol{\eta}_k^g)\Delta t) \\ \mathbf{v}_j &= \mathbf{v}_i + \mathbf{g}\Delta t_{ij} + \sum_{k=i}^{j-1} \mathbf{R}_k(\mathbf{a}_k - \mathbf{b}_k^a - \boldsymbol{\eta}_k^a)\Delta t \\ \mathbf{p}_j &= \mathbf{p}_i + \sum_{k=i}^{j-1} \left(\mathbf{v}_k\Delta t + \frac{1}{2}\mathbf{g}\Delta t^2 + \frac{1}{2}\mathbf{R}_k(\mathbf{a}_k - \mathbf{b}_k^a - \boldsymbol{\eta}_k^a)\Delta t^2 \right), \end{aligned} \quad (3.13)$$

where $\Delta t_{ij} = \sum_{k=i}^{j-1} \Delta t$. We adopt the term of IMU preintegration defined by Forster et al. (2017a) to obtain a measurement $\mathbf{x}_{ij}^\Delta \doteq \{\Delta \mathbf{R}_{ij}, \Delta \mathbf{v}_{ij}, \Delta \mathbf{p}_{ij}\}$ independent of the previous states, so that the constraint is always relative and does not bear global information for the optimization to remain consistent:

$$\begin{aligned} \Delta \mathbf{R}_{ij} &= \mathbf{R}_i^\top \mathbf{R}_j \text{Exp}(\delta \phi_{ij}), \\ \Delta \mathbf{v}_{ij} &= \mathbf{R}_i^\top (\mathbf{v}_j - \mathbf{v}_i - \mathbf{g}\Delta t_{ij}) + \delta \tilde{\mathbf{v}}_{ij}, \\ \Delta \mathbf{p}_{ij} &= \mathbf{R}_i^\top \left(\mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i\Delta t_{ij} - \frac{1}{2}\mathbf{g}\Delta t_{ij}^2 \right) + \delta \tilde{\mathbf{p}}_{ij}, \end{aligned} \quad (3.14)$$

with the integrated noise terms expressed as:

$$\begin{aligned} \text{Exp}(\delta \phi_{ij}) &= \prod_{k=i}^{j-1} \text{Exp}(\Delta \mathbf{R}_{k+1}^\top \mathbf{J}_r^k ((\boldsymbol{\omega}_k - \mathbf{b}_k^g)\Delta t) \boldsymbol{\eta}_k^g \Delta t), \\ \delta \tilde{\mathbf{v}}_{ij} &= \sum_{k=i}^{j-1} (\Delta \mathbf{R}_{ik} \boldsymbol{\eta}_k^a \Delta t - \Delta \mathbf{R}_{ik} [\mathbf{a}_k - \mathbf{b}_k^a]_\times \delta \phi_{ik} \Delta t), \\ \delta \tilde{\mathbf{p}}_{ij} &= \sum_{k=i}^{j-1} \left(\delta \tilde{\mathbf{v}}_{ik} \Delta t - \frac{1}{2} \Delta \mathbf{R}_{ik} [\mathbf{a}_k - \mathbf{b}_k^a]_\times \delta \phi_{ik} \Delta t^2 + \frac{1}{2} \Delta \mathbf{R}_{ik} \boldsymbol{\eta}_k^a \Delta t^2 \right). \end{aligned} \quad (3.15)$$

This form allows us to model the noise on the preintegrated measurements in a recursive manner.

With (2.4), it can be shown that up to first order, the measurement noise terms:

$$\delta \tilde{\mathbf{x}}_{ij}^{\Delta} \doteq \{\delta \phi_{ij}, \delta \tilde{\mathbf{v}}_{ij}, \delta \tilde{\mathbf{p}}_{ij}\}$$

are zero-mean and Gaussian, so that noise propagation can be modeled in a linear system where the covariance of the measurements $\Sigma_{\mathcal{I}}$ can be calculated along with the integrated measurements (Forster et al., 2017a). We can therefore write out a closed form representation of the Jacobian for the inertial measurements. We assume the biases to be constant across optimization iterations in this paper to simplify the problem.

We define the IMU residuals between the last two image frames as

$$\mathbf{r}_{\mathcal{I}} = [\mathbf{r}_{\Delta \mathbf{R}_{ij}}, \mathbf{r}_{\Delta \mathbf{p}_{ij}}, \mathbf{r}_{\Delta \mathbf{v}_{ij}}]^{\top} \in \mathbb{R}^9 \quad (3.16)$$

given the preintegrated IMU measurements $\Delta \mathbf{R}_{ij}$, $\Delta \mathbf{v}_{ij}$ and $\Delta \mathbf{p}_{ij}$ and the state estimates denoted with a *hat* as follows:

$$\begin{aligned} \mathbf{r}_{\Delta \mathbf{R}_{ij}}^{\top} &= \text{Log}(\Delta \mathbf{R}_{ij}^{\top} \hat{\mathbf{R}}_i^{\top} \hat{\mathbf{R}}_j) \\ \mathbf{r}_{\Delta \mathbf{v}_{ij}}^{\top} &= \hat{\mathbf{R}}_i^{\top} (\hat{\mathbf{v}}_j - \hat{\mathbf{v}}_i - \mathbf{g} \Delta t_{ij}) - \Delta \mathbf{v}_{ij} \\ \mathbf{r}_{\Delta \mathbf{p}_{ij}}^{\top} &= \hat{\mathbf{R}}_i^{\top} \left(\hat{\mathbf{p}}_j - \hat{\mathbf{p}}_i - \hat{\mathbf{v}}_i \Delta t_{ij} - \frac{1}{2} \sum_{k=i}^{j-1} \mathbf{g} \Delta t^2 \right) - \Delta \mathbf{p}_{ij} \end{aligned} \quad (3.17)$$

To obtain the covariance matrix $\Sigma_{\mathcal{I}}$ we first write the error propagation in state space form with IMU measurement noise $\boldsymbol{\eta} = [\boldsymbol{\eta}^g, \boldsymbol{\eta}^a]^{\top}$:

$$\delta \tilde{\mathbf{x}}_{ij}^{\Delta} = \mathbf{A}_{j-1} \cdot \delta \tilde{\mathbf{x}}_{ij-1}^{\Delta} + \mathbf{B}_{j-1} \cdot \boldsymbol{\eta}_{j-1}. \quad (3.18)$$

Abbreviating $\mathbf{J}_r^{j-1} = \mathbf{J}_r^{j-1}((\boldsymbol{\omega}_{j-1} - \mathbf{b}_i^g)\Delta t)$ and \mathbf{I} as identity matrix, we have:

$$\delta \tilde{\mathbf{x}}_{ij}^\Delta = \begin{bmatrix} \Delta \mathbf{R}_{j-1j}^\top & 0 & 0 \\ -\Delta \mathbf{R}_{ij-1} [\mathbf{a}_{j-1}]_\times \Delta t & \mathbf{I} & 0 \\ -\frac{1}{2} \Delta \mathbf{R}_{ij-1} [\mathbf{a}_{j-1}]_\times \Delta t^2 & \Delta t & \mathbf{I} \end{bmatrix} \delta \tilde{\mathbf{x}}_{ij-1}^\Delta + \begin{bmatrix} \mathbf{J}_r^{j-1} \Delta t & 0 \\ 0 & \Delta \mathbf{R}_{ij-1} \Delta t \\ 0 & \frac{1}{2} \Delta \mathbf{R}_{ij-1} \Delta t^2 \end{bmatrix} \boldsymbol{\eta}_{j-1}. \quad (3.19)$$

The state space representation gives us the propagation on covariance matrix $\boldsymbol{\Sigma}_{\mathcal{I}}$ as:

$$\boldsymbol{\Sigma}_{\mathcal{I}_{ij}} = \mathbf{A}_{j-1} \boldsymbol{\Sigma}_{\mathcal{I}_{ij-1}} \mathbf{A}_{j-1}^\top + \mathbf{B}_{j-1} \boldsymbol{\Sigma}_\eta \mathbf{B}_{j-1}^\top, \quad (3.20)$$

where $\boldsymbol{\Sigma}_\eta$ is the covariance on raw IMU measurement noise and $\boldsymbol{\Sigma}_{\mathcal{I}}$ is initialized to be zero. We can then obtain the inertial cost as the preintegrated measurement residuals weighted by the inverse covariance on the measurement noise, calculated recursively:

$$\|\mathbf{r}_{\mathcal{I}}\|_{\boldsymbol{\Sigma}_{\mathcal{I}}}^2 = \mathbf{r}_{\mathcal{I}}^\top \boldsymbol{\Sigma}_{\mathcal{I}}^{-1} \mathbf{r}_{\mathcal{I}}. \quad (3.21)$$

3.5. Implementation and Runtime Analysis

In this section, we discuss the core parameters and implementation details of the algorithm and analyze their effect on the computational cost on our SWaP-constrained UAV platform.

Figure 3.5 shows our compact experimental platform. It is equipped with a Qualcomm[®] Flight[™] board with a smartphone-grade Snapdragon[™] 801 processor and a 8 cm-baseline VGA stereo camera. For the same algorithm, the CPU usage on board is around 4-5 times the CPU usage on an Intel i7-6600U laptop on average.¹ Our state estimation algorithm takes about 1.5 cores on board, leaving enough space for other system blocks including planning and control to run concurrently. Our method also outputs a semi-dense map which can be used for planning. We achieve this CPU usage on the constrained platform through various parameter selections and algorithm parallelization as discussed below.

¹We are using a different CPU for runtime analysis from the experiments on EuRoC datasets, for the two sets of experiments are conducted at different times.

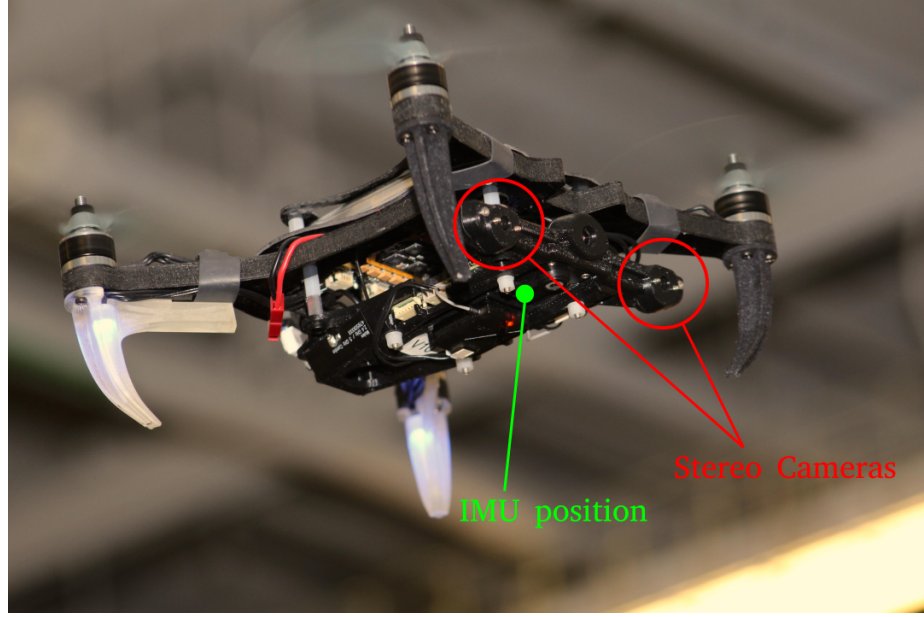


Figure 3.5: A 22 cm, 250 g quadrotor with a forward-facing stereo camera pair (Liu et al., 2021b).

The most important parameters to balance the trade-off between accuracy and computational cost are frame rate, image resolution and grid size.

1. **Frame rate:** There are three frame rate modes available on the cameras mounted on the robot: 10 Hz, 15 Hz and 30 Hz. Higher frame rate makes it easier for the tracked pixels to fall into the gradient region of convergence for direct methods, increasing tracking accuracy. High frame rate also benefits the fusion of an inexpensive IMU as integrating the noisy IMU measurements leads to large drift. This can result in jumps in the high frequency state estimates when images are observed, while a smoother estimate is preferred by the controller. However, a frame rates of 30 Hz overloads the CPU and causes the control to be unstable. We find that a frame rate of 10 Hz to 15 Hz is a reasonable compromise.
2. **Image resolution:** The highest image resolution from the on-board cameras is 640×480 . However as shown in Figure 3.6, keyframe processing is much more computationally expensive for the original resolution comparing to a down-sampled resolution of 320×240 . This is due to the disparity generation and high-gradient point extraction operations conducted on the whole image as observed from the computation breakdown in Figure 3.7. The runtime

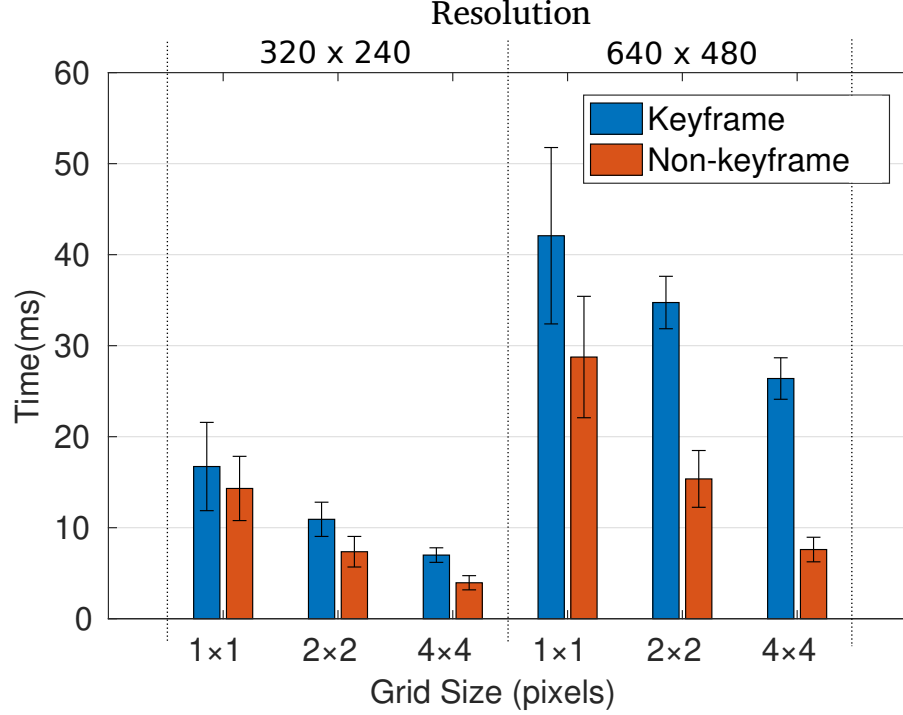


Figure 3.6: Average and standard deviation of the processing time of keyframes and non-keyframes with different resolutions and grid sizes tested on a 2.60 GHz Intel i7-6600U laptop. The time required by processing a non-keyframe is approximately the same if we compare the two resolutions with similar number of high-gradient points (e.g. 320×240 with no grid vs. 640×480 with 2×2 grid size).

for the keyframe determines the resolution we use for real-time state estimates, which we require to be at least 10 Hz for control purposes. Although higher resolution gives finer details of the gradient information, we resort to using the down-sampled resolution of 320×240 as the largest layer in the image pyramid on board. We also found that using two pyramid levels works the best for overall system performance.

3. **Number of tracked pixels:** Tracking a larger number of pixels in direct method increases tracking accuracy (Forster et al., 2017b), but it also results in proportionally longer computational time. To control the number of tracked pixels and spacing them uniformly throughout the image space, we divide the image into pixel grids and only select the highest gradient point in each grid cell from all the pixels selected from an adaptive threshold with predetermined parameters. The smaller the grid size, the more points we use for optimization, thus the higher

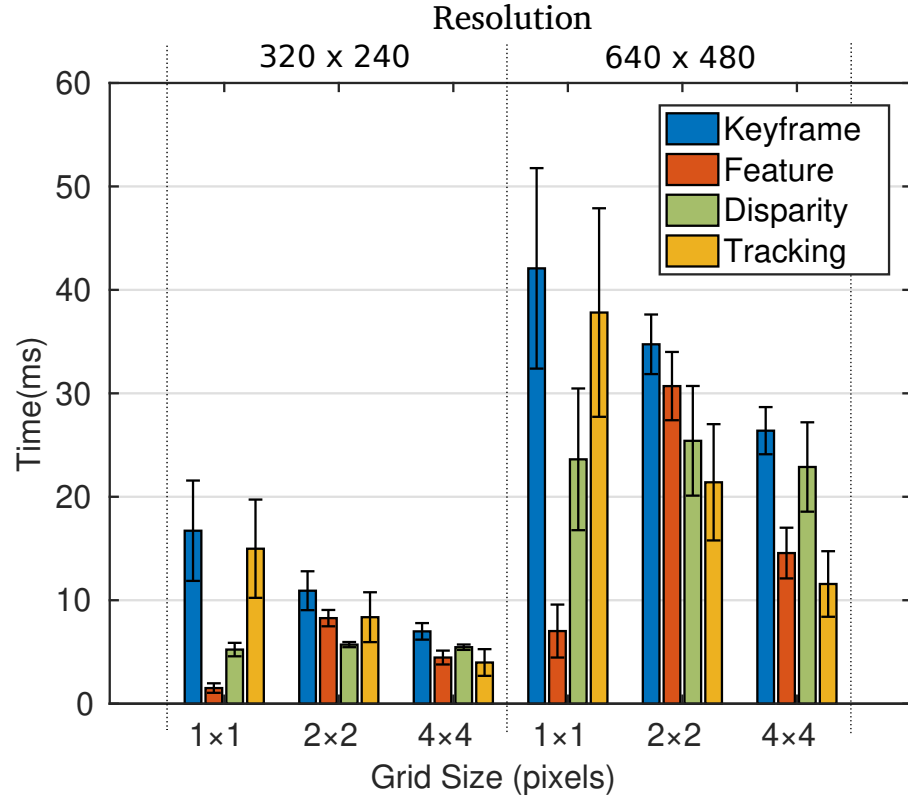


Figure 3.7: Average and standard deviation of the total processing time of a keyframe and its three parallel threads: feature extraction, disparity generation and tracking. The processing time for keyframe disparity generation and high-gradient point selection is proportional to the total number of pixels in an image, where the original resolution results in 4 times the computation than the downsampled resolution.

Time [ms]	Mean	Std Dev	Min	Max
Keyframe (total)	59.1	8.2	43.7	80.2
– extract features	45.5	6.8	35.1	66.4
– feature tracking	41.7	13.1	6.3	71.2
– disparity	18.7	4.9	14.7	35
Non Keyframe	32	11.3	5	57.9

Table 3.1: Time statistics on our platform with the selected settings for keyframes and non-keyframes, as well as the major threads in keyframe processing. Having the processing time below 100 ms allows us to run the system at 10 Hz in real time for flight. Note that the times for keyframe processing don’t add up since they are running on separate threads.

the computational cost. In Figure 3.7, the “tracking” bars characterize the optimization time, and in lower resolution the optimization time dominates. We choose 2×2 grid size for our application. When there is no grid (1×1 in Figure 3.7), the points selected are only dependent on the threshold and since there is no grid pruning step, the feature extraction time is minimal.

Our final system uses two image pyramid levels with the base layer set to the sub-sampled 320×240 resolution, a grid size of 2×2 , and running the image stream and vision pipeline at 10 Hz. In addition to these parameters, we utilize parallelization to cut down on the time required for each new frame. Two threads are used for left and right images to do pyramid construction. The feature extraction and disparity map generation for each pyramid resolution are also put into different threads. The tracking is done with respect to the last keyframe and can run in parallel with the feature extraction and disparity computation for new keyframes. This parallelization scheme greatly reduces the time for keyframe processing, and is one of the main factors allowing us to run the pipeline on the constrained platform. Figure 3.7 and Table 3.1 shows the running time statistics of the threads composing the keyframe pipeline.

3.6. System Performance and Benchmark Evaluation

In this section we evaluate our performance in accuracy and CPU usage using the standard EuRoC Dataset as benchmark. We compare our semi-dense direct VIO approach against other state estimation algorithms on a 3.60 GHz Intel i9-9900K desktop CPU. We then validate the suitability of this approach on SWaP-constrained platforms through autonomous flight.

	Error	Solver	Features
Ours	photometric	optim	semi-dense
VI-Stereo-DSO	photometric	optim	semi-dense
VINS-Fusion	geometric	optim	sparse
MSCKF	geometric	filter	sparse
ROVIO	photometric	filter	sparse
Basalt	geometric	optim	sparse
OpenVINS	geometric	filter	sparse

Table 3.2: List of state-of-the-art stereo VIO algorithms we compare with on EuRoC Dataset and their categories.

3.6.1. Evaluation on EuRoC Vicon Room Datasets

The EuRoC dataset is a UAV flight dataset with aggressive maneuvers in a room with varying lighting conditions, which presents a good challenge for state estimation algorithms. The stereo images are collected by global shutter cameras at 20 Hz and the IMU data is collected at 200 Hz. A snapshot of the proposed semi-dense VIO algorithm running on EuRoC dataset is illustrated in Figure 3.8.

We evaluate our algorithm on the six Vicon Room EuRoC dataset in comparison with the approaches listed in Table 3.2. We select the algorithms that compliment each other in their cost function and solver type to obtain insight on their differences. We select VI-Stereo-DSO (Von Stumberg et al., 2018) as the most similar semi-dense baseline. Among other sparse approaches, VINS-Fusion (Geneva et al., 2020) and Basalt (Usenko et al., 2019) are optimization approaches, where MSCKF (Sun et al., 2018b), OpenVINS (Geneva et al., 2020) and ROVIO (Bloesch et al., 2017) are filter approaches, and ROVIO uses photometric cost. We compare in both accuracy and CPU usage to show that our method is a good alternative for state estimation on low computational capability platforms.

For metric evaluation, each dataset is run three times on each algorithm to take the average. We use the evo open-source tool² to align the estimated and ground truth trajectories by rotation and translation. We use Root Mean Square Error (RMSE) as our error metric and we take the average of the three trials as the value shown in Figure 3.9. The average CPU usage is shown in Figure 3.10.

²<https://github.com/MichaelGrupp/evo>

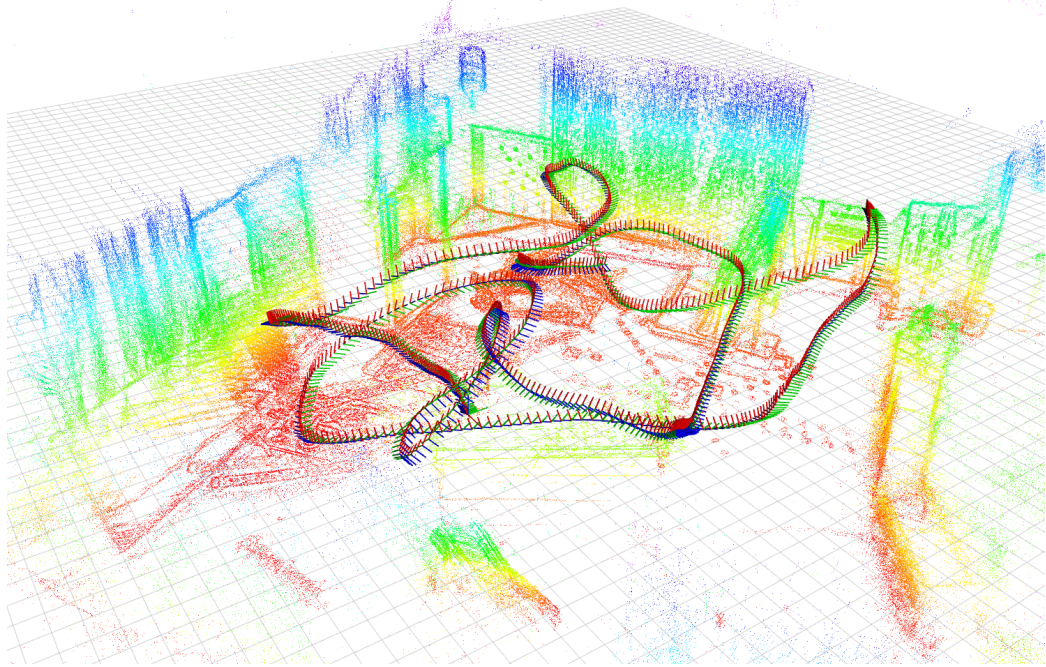


Figure 3.8: State estimates from direct VIO and the semi-dense map created concurrently on EuRoC dataset V1-02-medium. The 6DoF state estimates are represented by the axes and the map is represented by the point cloud. The computationally efficient block matching algorithm gives noisy disparity map but most of the noisy points are avoided as the algorithm selects only the high gradient pixels.

We remove the invalid data when the algorithm cannot keep track of the full sequence: V1-03 and V2-02 for fast setting of VI-Stereo-DSO due to not enough points, and V2-03 for MSCKF due to brightness inconsistency between the stereo cameras.

Our method strikes a balance between computation and accuracy for semi-dense photometric optimization to achieve on-board real-time performance while creating a semi-dense map. VI-Stereo-DSO (Von Stumberg et al., 2018) is the closest to our approach except that we only do current frame to keyframe alignment without point features in the state. The VI-Stereo-DSO includes feature depths in a windowed optimization framework, thus is more accurate in its default settings (5-7 frames, 2000 points) despite heavier computation. Its performance becomes unstable when limited to fast settings (4-6 frames, 800 points). Our approach uses grid to control the maximum number of feature points, and uses adaptive threshold to control the selection quality. The number of points tracked in our approach on EuRoC dataset ranges between 2000 to 6000 and changes with every keyframe

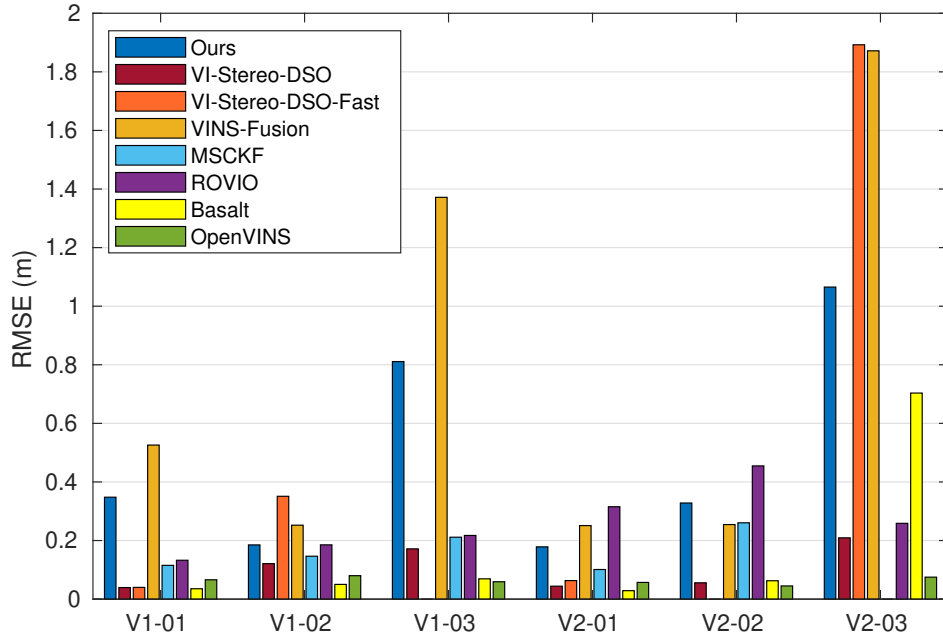


Figure 3.9: Average RMSE comparison on EuRoC Vicin room Dataset showing a reasonable performance on accuracy given the simplistic framework adopted in our approach.

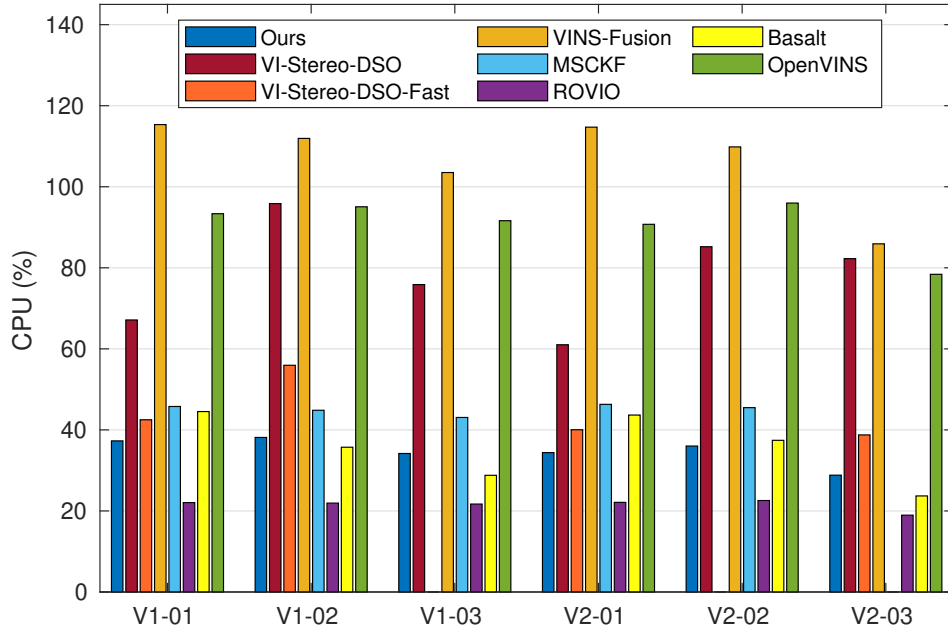


Figure 3.10: Average CPU usage on EuRoC Vicin room Dataset for real time execution excluding the initialization stage. 100% indicates one full core on a 3.60 GHz Intel i9-9900K desktop CPU. Our method shows competitive efficiency in CPU usage which is suitable for on-board flight.

based on the scene. On the low texture dataset, the number of points our algorithm takes varies between 500 to 3000. We can maintain an upper limit of the number of points by resizing the grid size dynamically. We show that without windowed optimization, our algorithm can still track decently to allow autonomous flight. It inevitably increases drift but an additional drift of less than 20 cm over the whole trajectory for non-aggressive flight is acceptable for our application.

In Figure 3.10 we report the average CPU usage at stable execution stage over three runs of each algorithm. For Basalt and VI-Stereo-DSO, the algorithms run at maximum processing speed and the CPU usage is calculated from the run time with minimum number of threads allowed. It is also worth noting that the CPU usage not only represents the algorithm design, but also implementation. A well-engineered system can achieve superior computational efficiency bounded by the algorithm requirements. To observe in more detail of the computation cost in a semi-dense approach, we fix the number of frames to a constant for VI-Stereo-DSO on V1-01 dataset and control the window and feature sizes to record the CPU usage shown in Figure 3.11. We are unable to use other datasets because the algorithm gets unstable and does not provide enough meaningful data. The minimum number of frames for VI-Stereo-DSO to work is 3, while our algorithm only has the current frame in the state.

3.6.2. Autonomous Flight

We demonstrate two different trajectories on our experimental platform, a square trajectory that is repeated 3 times and a 10×2 meters rectangular tour in our flying space. Figure 3.12 shows a 3D plot of the square trajectory, and Figure 3.13 shows the position estimates for both the trajectories. All computations during these flights are done on board and the ground station is only used for storage and visualization. The quadrotor follows the designed trajectories smoothly given the high frequency state estimates. The RMSE for the two trajectories are 0.055 m and 0.273 m respectively. The second trajectory has some drift starting at time 60 s due to turning towards a low textured area, and it drifted in the vertical direction because the features are mainly vertical on the curtains. With only vertical features, the movement on the vertical direction is observable only from IMU data, but the problem with IMU data is that there are time varying biases present in the measurements. Since our framework

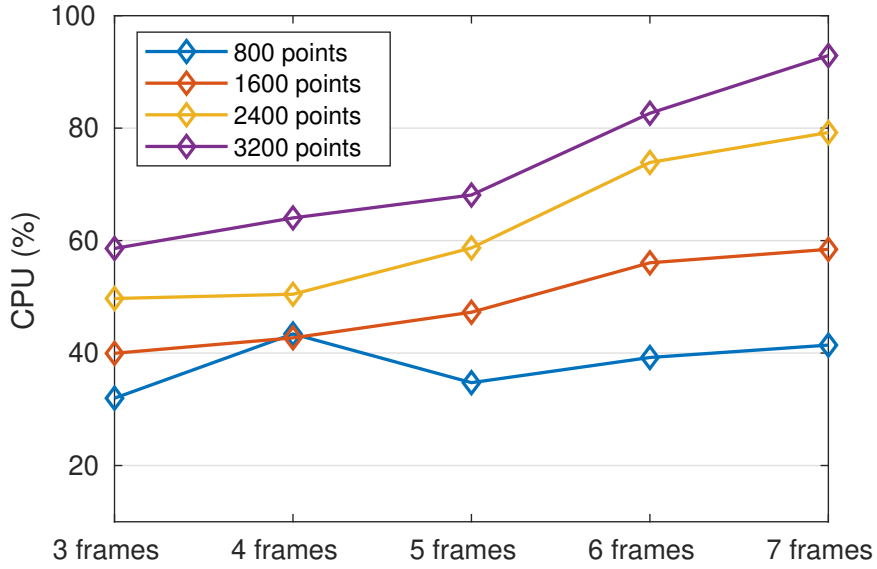


Figure 3.11: CPU usage visibly reduces by decreasing the number of feature points and window size in VI-Stereo-DSO optimization framework.

uses a fixed prior with only the latest pose in the state, we cannot have a reliable bias estimate as the random walk modeling of biases requires multiple states and their uncertainties to be effective. We tried including the accelerometer and gyroscope biases in the state with only the current pose in the state using the Jacobians based on Forster et al. (2017a), but observed no significant improvement in the performance on our dataset due to the limitation of the framework design. The lack of accurate bias estimate causes our approach to be prone to drift when image gradient does not provide enough motion information and is one of the main drawbacks in our simplified framework. Nevertheless, these experiments demonstrate that our algorithm is suitable for on-board autonomous flight on such small, computationally constrained platforms.

3.7. Low-Texture Scenarios

To demonstrate the reason we adopt semi-dense direct approach over sparse approaches, we collected a *low texture dataset* with our computationally constrained quadrotor platform flying/hand-carried with the camera facing towards the large curtains in our VICON space – a common case scenario of indoor flight where sparse feature-based tracking often fails. The curtains do not have features on them and only the vertical lines formed by the natural curvature of the curtain can be seen from the

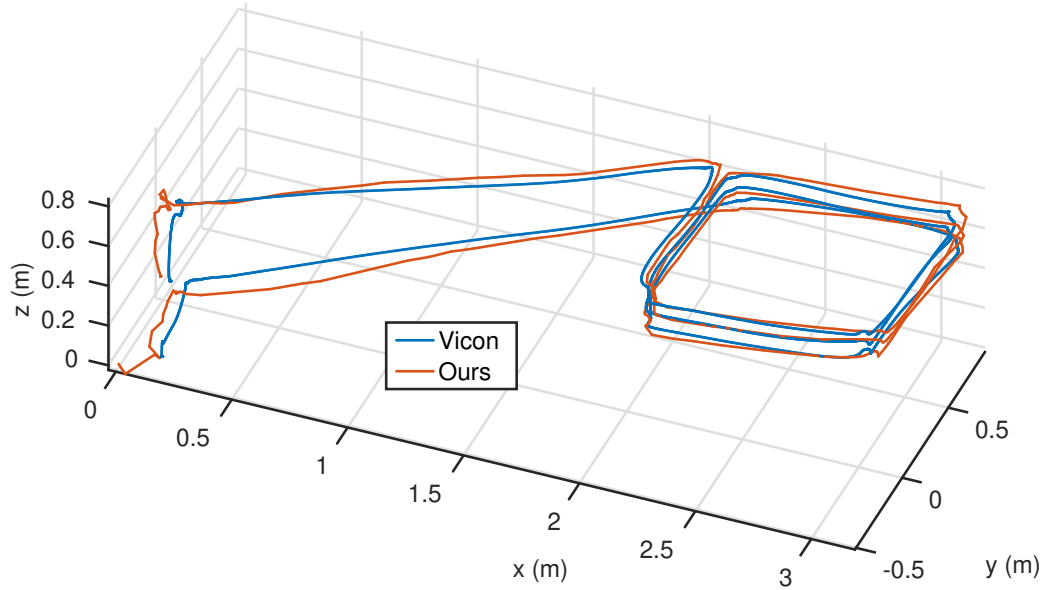


Figure 3.12: Square trajectory state estimates against VICON ground truth.

camera, which provides a near-perfect scenario with only gradient information. Figure 3.14 shows example camera images during flight.

The dataset consists of 5 hand-carried sequences at image framerate of 10 Hz, 4 sequences with a bright blob gradient feature projected by a handheld torchlight onto the curtain, and 2 flying sequences at 30 Hz. 500 Hz of IMU data and 100 Hz of Vicon Mocap groundtruth data is included. All sequences are collected next to the curtain area with cameras facing the curtain at some time. The dataset is collected in the format of ROS bagfiles, with camera calibration information included as rostopics. A separate file is provided with the complete IMU and camera calibration.

We compare only to MSCKF and ROVIO, two sparse filtering approaches, for this dataset with affordable CPU usage to run on our small platform. MSCKF tracks FAST features with KLT algorithm using geometric cost, and ROVIO tracks image patches selected from high gradient pixels using photometric cost. Figure 3.15 shows the estimated position of these approaches on an example sequence comparing to the ground truth provided by the VICON motion capture system.

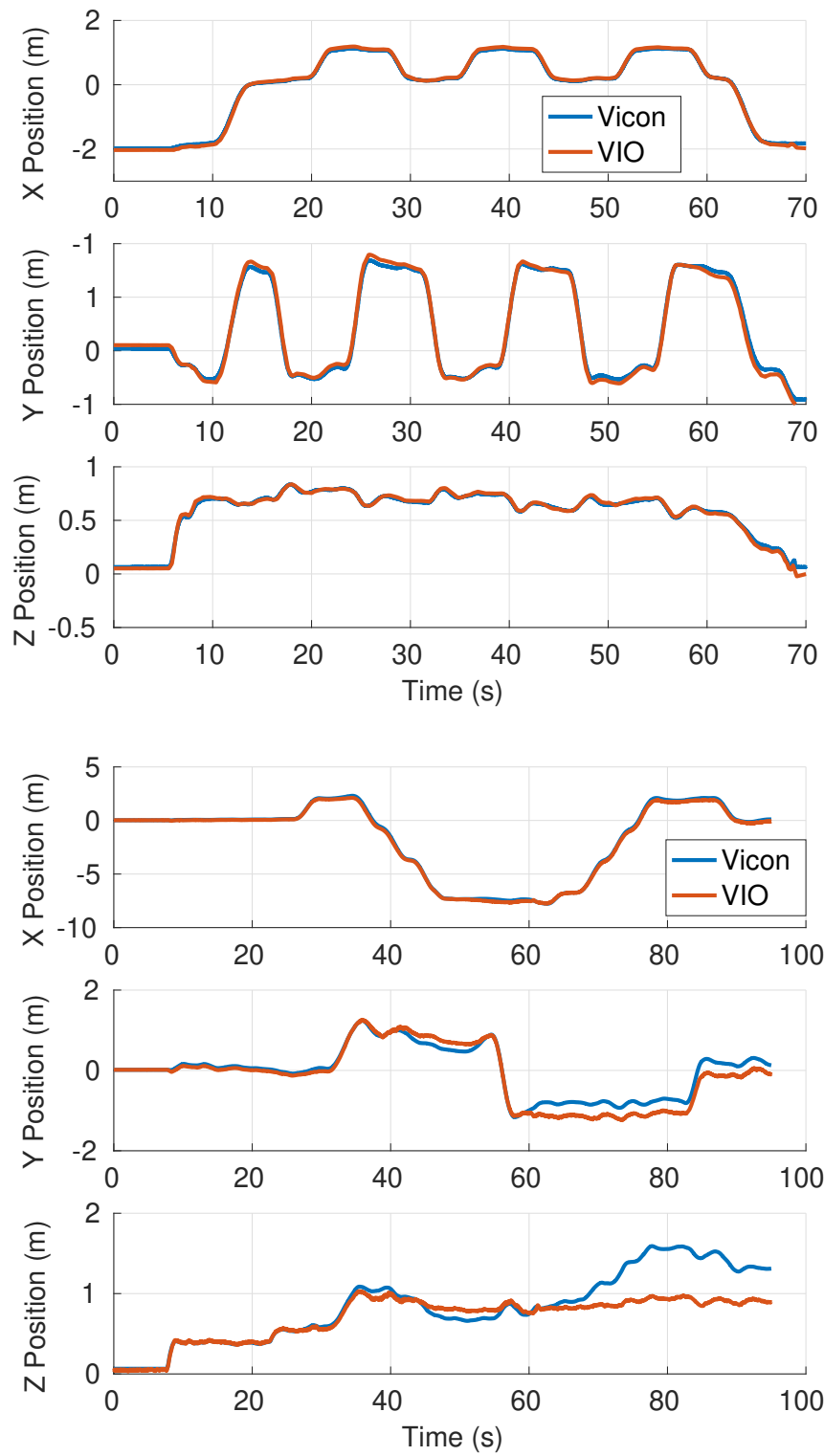


Figure 3.13: Position estimates during the square and tour trajectories plotted against time.

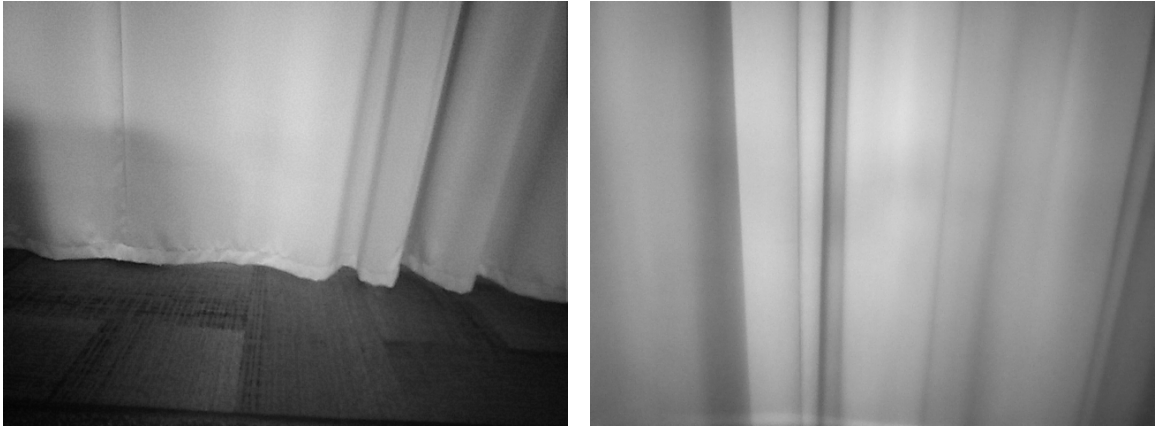


Figure 3.14: Forward facing camera view of the curtain area.

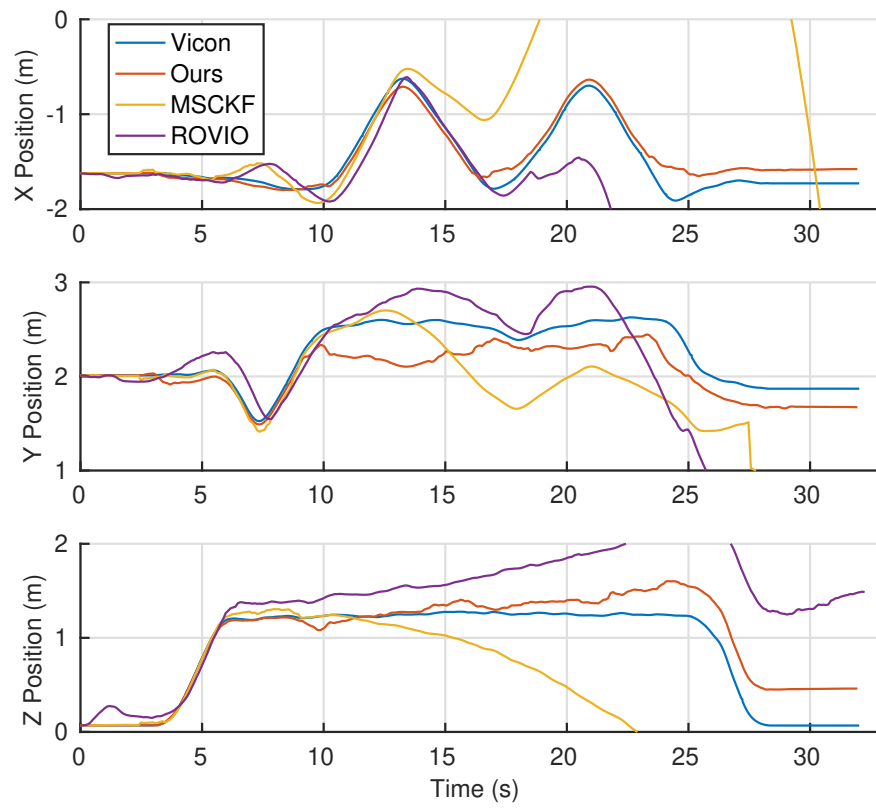


Figure 3.15: Position estimates in the low texture environment. The figure shows that feature-based methods lose track a short while after the camera sees the curtain, while semi-dense method keeps track with a reasonable small drift.

We also explored the use of photometric calibration on the low texture datasets using the approaches in Engel et al. (2016). Figure 3.16 shows that running the photometric calibration by compensating for the vignetting effect on the camera image greatly improves the tracking accuracy.

We ran MSCKF, ROVIO, and our approach with (Ours-calib) and without photometric calibration on the entire low texture dataset. We report the average RMSE over three runs for each algorithm in Figure 3.17. Feature-based MSCKF quickly lose track with the view of the curtain. ROVIO takes gradient patches as features so it works well when enough gradient features are still present. Yet the number of features is limited in its state and the algorithm is not able to keep track for sequences with a full view of the curtain.

3.8. Summary

In this work, we present a semi-dense direct visual-inertial odometry system that can be deployed onboard a small MAV platform with computational constraints. We show a detailed analysis of our algorithm’s accuracy and computational cost on the standard EuRoC dataset compared to other state-of-the-art methods. Our algorithm design made various simplifications compared to the standard windowed optimization approaches. We show that our design presents a good trade-off solution for semi-dense methods to work efficiently for onboard real-time flights. We discuss the parameters that affect the balance between accuracy and CPU usage and the parallelization approaches to enable the algorithm to run on a smartphone-grade processor.

We also demonstrated on a low texture dataset that direct semi-dense approaches have the advantage of tracking in environments where very few corners can be detected. We show cases where feature-based tracking fails while our semi-dense approach performs well, making it the preferred choice when operating in low texture environments. However, with the computational efficiency coming from a simplified framework, our approach has inherent limitations. Most notably, the ineffective IMU bias estimation causes more drift when the gradient information does not render all degrees of freedom in the state observable.

We identify that this research is on the boundary of computation constraints. If this information

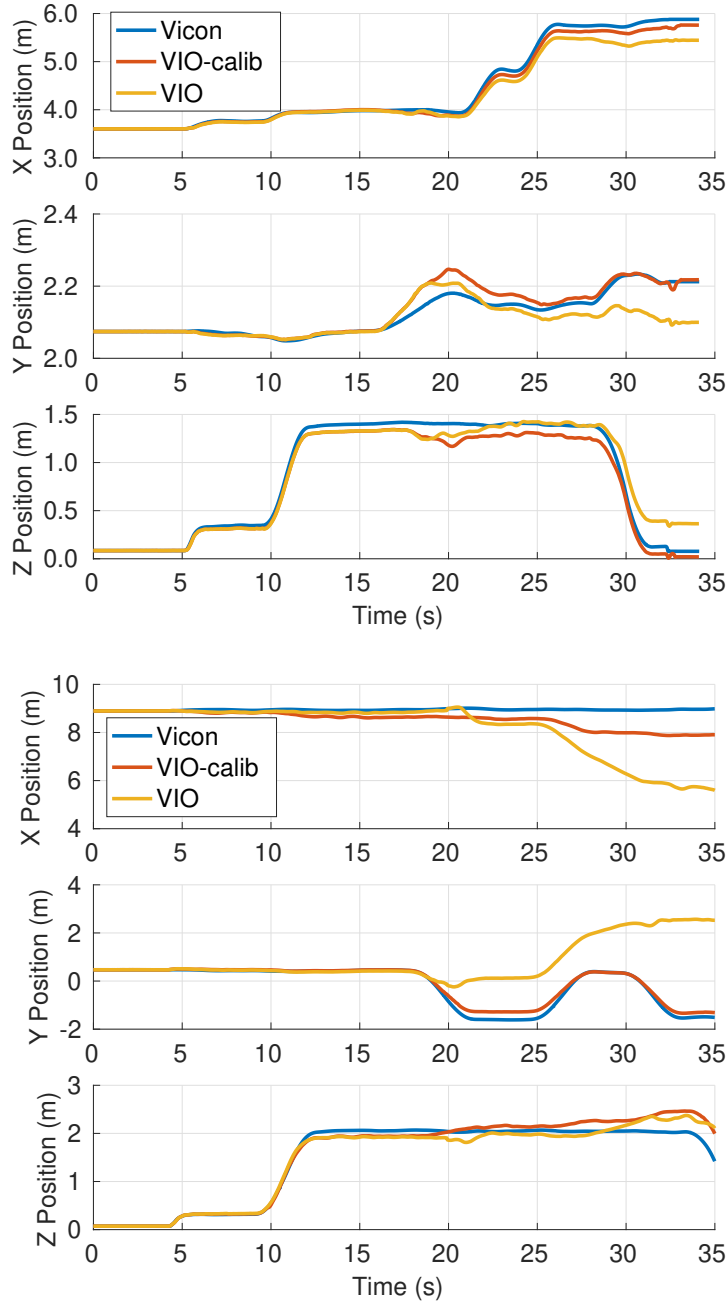


Figure 3.16: Improvement by photometric calibration on two other low texture datasets. *VIO-calib* shows the VIO output after compensating for the vignetting effect. Comparing to the original *VIO*, adding photometric calibration reduces error in tracking and can in some cases prevent the system from losing track, as is the case in the second illustration.

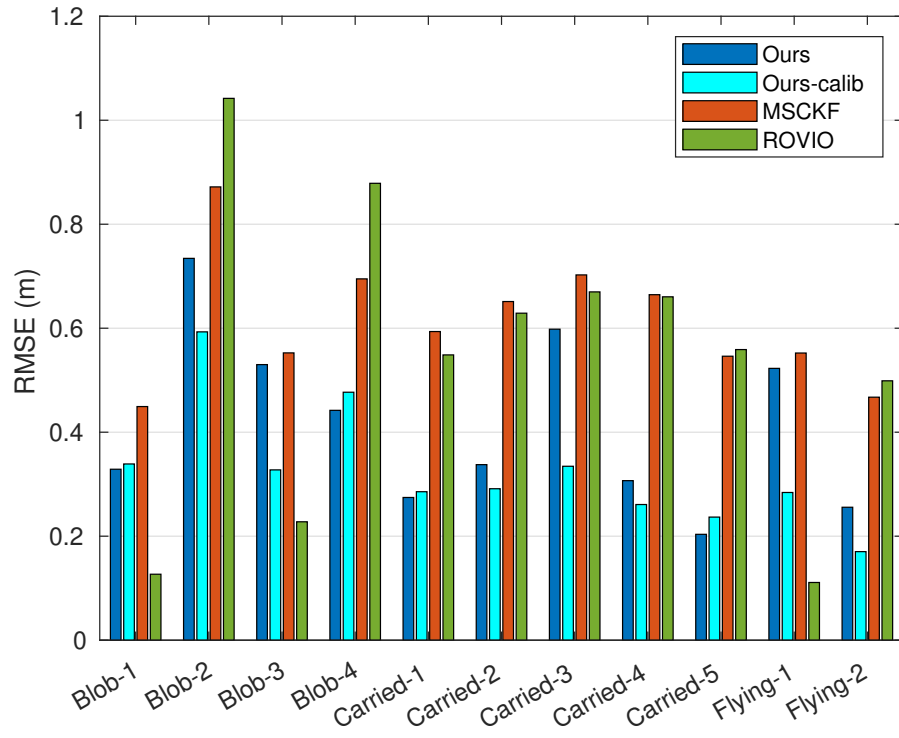


Figure 3.17: Average RMSE of our proposed semi-dense approach comparing to MSCKF and ROVIO. Photometric calibration visibly helps the direct semi-dense algorithm in such extreme low texture cases.

deficiency case is unavoidable, we summarize the following challenges for future work:

1. With the limitation of CPU resources on constrained platforms, the efficiency of hardware usage can be improved by utilizing GPU for image processing tasks
2. Depth estimation and stereo matching is a bottleneck in computation in this method. Designing small footprint algorithms to lower CPU usage is vital for small platforms
3. Optimization is another bottleneck in computation. Tracking quality is unstable, causing slow convergence. Implementing more robust outlier rejection functionality can help improve the efficiency of direct image alignment and reduce the recursive optimization steps
4. After improvements on the above aspects, with additional CPU resources available, windowed batch optimization is a strongly recommended extension to incorporate effective IMU bias estimation and further improve tracking accuracy

CHAPTER 4

LEARNING EGOMOTION IN A HYBRID STRUCTURE

This chapter considers the case of monocular visual odometry without the presence of inertial data. Inertial data provides scale information to visual-inertial state estimation systems. For a calibrated set of stereo cameras with a known baseline, the scale is observable. However, this is not the case for monocular applications.

We aim to use the recent rapid development in computer vision with deep learning to compensate for this information loss. Convolutional Neural Networks (CNNs) are suitable for processing spatially structured data like images, and we have seen successful examples in depth (Godard et al., 2017) and optical flow (Sun et al., 2018a) prediction, where network-based methods are now state of the art. With a learned prior, the network can retrieve scale information from a single monocular image and maintain constant-time computation over the full image.

We propose a method that combines unsupervised deep learning predictions for optical flow and monocular depth with a model-based optimization procedure for camera pose. Given the flow and disparity predictions from the network, we apply a RANSAC outlier rejection scheme to find an inlier set of flows and disparities, which we use to solve for the camera pose in a least-squares fashion. We show that this pipeline is fully differentiable, allowing us to combine the pose with the network outputs as an additional unsupervised training loss to further refine the predicted flows and disparities. This method not only allows us to directly regress pose from the network outputs, but also automatically segments away pixels that do not fit the rigid scene assumptions that many unsupervised structure from motion methods apply, such as on independently moving objects. We evaluate our method on the KITTI driving dataset and demonstrate state-of-the-art results, even in the presence of challenging, independently moving objects.

4.1. Introduction

The advent of unsupervised methods for neural networks has resulted in the rapid growth of works of unsupervised deep learning for visual odometry (Zhou et al., 2017). By utilizing the ability of

neural networks to learn a prior for the scale of a scene, these networks can predict camera poses and depths without the scale ambiguity problem for monocular inputs. However, these methods abstract away the function between the image and the pose within the network weights, therefore cannot provide guarantees on robustness or safety. Furthermore, outliers such as independently moving objects remain a challenging problem, as incorporating outlier rejection schemes into the pose when it is directly regressed from the network has proven difficult.

Most state-of-the-art learning frameworks for visual odometry have a network structure which estimates 6 DoF pose from 2D images using convolutions. However, these learning-based pose estimation methods still cannot match the performance of geometric optimization approaches using feature correspondences and outlier rejection with epipolar constraints (Delmerico and Scaramuzza, 2018). In this work, we decompose the classical direct visual odometry pipeline and introduce a learning-based front end. We combine the best of both worlds by leveraging the learning ability of neural networks to predict dense optical flow correspondences and disparities from images, and applying a robust optimization backend using RANSAC to estimate pose from the network outputs.

By applying RANSAC (Fischler and Bolles, 1981) for pose estimation, we are able to apply outlier rejection at a pixel level to extract only the set of predicted flow and disparity values that best fit the camera pose estimates. There have been a number of works that try to filter out these objects, for example by directly predicting a mask for the valid pixels in the scene from the network (Zhou et al., 2017; Vijayanarasimhan et al., 2017), or by detecting mismatches between the predicted optical flow and the disparity and pose (Ranjan et al., 2018; Yang et al., 2018b). Our method does this through a principled geometric matching algorithm without the need to separately learn these objects.

Given the pose estimated from RANSAC, we propose a set of additional refinement losses to further improve the performance of the proposed pipeline. Similar to previous works, we use a rigid motion model to estimate optical flow from the generated disparities and pose (Zhou et al., 2017). However, we also perform the inverse, and estimate disparity from optical flow and pose. These new estimated values are used to apply additional photometric losses on the original network predictions, and we show that applying these losses at training time produces meaningful improvements in

network performance.

This work has three contributions:

- An end-to-end unsupervised structure from motion pipeline which uses two fully convolutional networks to predict optical flow and disparity from a pair of images, and a RANSAC outlier rejection scheme to robustly estimate camera pose.
- A set of photometric losses that uses the camera pose to estimate disparity from flow and vice versa, allowing for additional supervision of each modality from another image.
- Evaluation on the KITTI datasets, where we show robustness to independently moving objects, with extensive ablations demonstrating improvements with RANSAC.

4.2. Related Works

There have been a number of recent methods that leverage the principles of photoconsistency and image warping (Jaderberg et al., 2015) to perform unsupervised learning of image motion. Garg et al. (2016) and Godard et al. (2017) showed that metric depth can be learned from a monocular camera by warping stereo images onto one another. Similarly, Jason et al. (2016) and Meister et al. (2017) use a similar transformation to learn optical flow from a pair of images. Zhou et al. (2017) also showed that 3D camera motion can also be learned from this regime, by jointly predicting the egomotion of the camera, the depth of the scene, and combining the two to warp each pixel in the image. Since then, a number of methods have improved upon this scheme. Zhan et al. (2018) add a feature reconstruction loss to resolve ambiguities seen with a photometric loss, such as in textureless regions. Li et al. (2018) use the egomotion to align the point clouds associated with the predicted depths, while Wang et al. (2018b) introduce a recurrent neural network to replace the feed-forward CNN.

However, these methods rely on a rigid scene assumption, and so are corrupted by independently moving objects in the scene. A number of works have tried to remove these objects from the loss function, including the works by Zhou et al. (2017) and Vijayanarasimhan et al. (2017), which predict

masks to remove invalid pixels. Moreover, the works by Ranjan et al. (2018), Luo et al. (2018) and Yang et al. (2018b) use the mismatches between egomotion and depth and optical flow predictions to generate these masks.

In a similar vein to our work, Wang et al. (2018a) train a depth network by applying direct visual odometry optimization using the predicted depths to minimize the image reprojection error, and Yang et al. (2018a) incorporate a separately trained depth network into the Direct Sparse Odometry (Engel et al., 2018) framework, providing a monocular odometry pipeline that is able to estimate metric trajectories comparable to a stereo setup.

Our work, on the other hand, uses an interpretable geometric, model-based backend to jointly detect outliers and regress pose, while leveraging the strength of the network in 2D to predict both correspondences and depths.

4.3. Method

Our pipeline consists of two neural networks: one predicts optical flow from a pair of images, and one predicts disparity from a single image. Both networks are based on the FlowNet-S (Fischer et al., 2015) architecture, which consists of an encoder-decoder pipeline with skip connections and a multi-scale loss. We have significantly reduced the size of the network, which we discuss in Sec. 4.3.3.

Each network can be trained separately, with a combination of photometric, smoothness and geometric losses. At training time, we sample four images randomly from the dataset, consisting of two stereo pairs from times t_0 and t_1 . We then pass these images into both networks to predict a forward, $F_{t_0 \rightarrow t_1}(\vec{x})$, and backward, $F_{t_1 \rightarrow t_0}(\vec{x})$, flow for each image in the stereo pairs, and a disparity for each image, $D(\vec{x})$, resulting in four flow and disparity predictions respectively for every two stereo image pairs. As both flow and disparity can both be modeled by a general image warping function that warps a pixel \vec{x} to another location: $W(\vec{x}_i) \rightarrow \vec{x}_j$, we will describe our loss functions in terms of this general warping, which can then be substituted by either optical flow or disparity.

We use RANSAC for relative pose estimation and outlier rejection, taking the flow and disparity from the networks as inputs. The estimated pose is then used to apply a second set of *refinement*

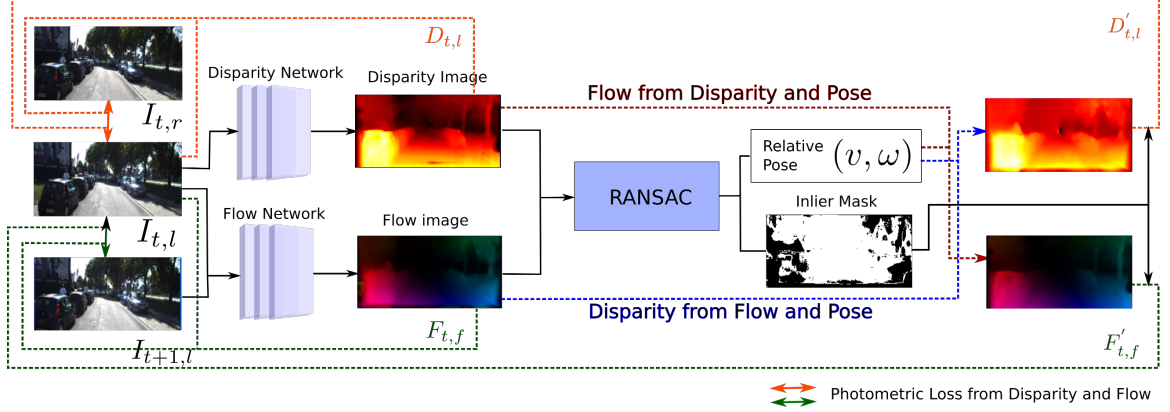


Figure 4.1: The pipeline given three images as inputs. Only forward and backward images are used as inputs, but a third image from the stereo is used for the disparity network and end-to-end training with RANSAC.

losses. First, a rigid motion model is applied to estimate disparity from pose and flow, and flow from disparity and pose. Photometric losses are then applied to the spatial and temporal images warped by this second set of flow and disparity estimates. This pipeline is depicted in Fig. 4.1. As the inlier mask from RANSAC removes points that do not fit the rigid motion model, any moving objects are also automatically segmented. As a result, we only back-propagate through the set of inliers, and the whole pipeline is trainable end-to-end.

4.3.1. Loss Function Design

Appearance Loss

Given a pair of images, I_i , I_j , and a warping between them, $W_{i \rightarrow j}$, we apply a photoconsistency assumption, which assumes that the correct warp should warp pixels from I_i to pixels at I_j with the same intensity. We therefore apply the following photometric loss to enforce the constraint:

$$\mathcal{L}_{\text{photo}}^{W_{i \rightarrow j}}(\vec{x}) = \rho(I_i(\vec{x}) - I_j(W_{i \rightarrow j}(\vec{x}))) \quad (4.1)$$

$$\rho(x) = \sqrt{x^2 + \epsilon^2} \quad (4.2)$$

where $\rho(x)$ is the robust Charbonnier loss function used in Jason et al. (2016).

We combine this photometric loss with a structural similarity (SSIM) loss (Wang et al., 2004) to

form our appearance loss:

$$\mathcal{L}_{\text{appearance}}^{W_{i \rightarrow j}}(\vec{x}) = (1 - \alpha) \text{SSIM}(\vec{x}) + \alpha \mathcal{L}_{\text{photo}}(\vec{x}) \quad (4.3)$$

Geometric Consistency Loss

Several works, such as Godard et al. (2017) and Meister et al. (2017), have proposed additional losses to constrain geometric consistency between the forward and backward (or left and right) estimates of a warp. That is, the backward warp, warped to the previous image, should be equivalent to the negative of the forward warp. We apply this constraint to both the disparity and flow:

$$\mathcal{L}_{\text{consistency}}^{W_{i \rightarrow j}}(\vec{x}) = \rho(W_{i \rightarrow j}(\vec{x}) + W_{j \rightarrow i}(W_{i \rightarrow j}(\vec{x}))) \quad (4.4)$$

Smoothness Regularization Loss

We further apply a constraint for the warp to be locally smooth. This is applied with an edge aware smoothness loss, which weighs the loss lower at pixels with high image gradient:

$$\mathcal{L}_{\text{smooth}}^{W_{i \rightarrow j}}(\vec{x}) = \rho\left(\delta_x W_{i \rightarrow j}(\vec{x}) e^{-|\delta_x I(\vec{x})|}\right) + \rho\left(\delta_y W_{i \rightarrow j}(\vec{x}) e^{-|\delta_y I(\vec{x})|}\right) \quad (4.5)$$

Motion Occlusion Mask

For a given warp function, there may be pixels that are warped out of the image. Applying a photometric or geometric consistency loss at these pixels would introduce errors into the model, as we cannot sample from points from outside the image. To resolve this issue, we generate a mask, $M_{\text{occ}}^{W_{i \rightarrow j}}(\vec{x})$, which is 0 for pixels that are warped out of the image, and 1 otherwise, and is used for the photometric and geometric consistency losses. Note that this mask is computed directly from the predicted flows and disparities, and does not contain any learnable components.

Total Loss

For a single warp, $W_{i \rightarrow j}(\vec{x})$, the total loss is:

$$\mathcal{L}_{\text{total}}^{W_{i \rightarrow j}} = \sum_{\vec{x}} M_{\text{occ}}^{W_{i \rightarrow j}}(\vec{x}) \left(\mathcal{L}_{\text{photo}}^{W_{i \rightarrow j}}(\vec{x}) + \lambda_1 \mathcal{L}_{\text{consistency}}^{W_{i \rightarrow j}}(\vec{x}) \right) + \lambda_2 \mathcal{L}_{\text{smoothness}}^{W_{i \rightarrow j}}(\vec{x}) \quad (4.6)$$

The final loss, then, is the sum of $\mathcal{L}_{\text{total}}$ for each of the flow and disparity predictions:

$$\mathcal{L}_{\text{total}} = \sum_{c \in \{l, r\}} \mathcal{L}_{\text{total}}^{F_{0 \rightarrow 1, c}} + \mathcal{L}_{\text{total}}^{F_{1 \rightarrow 0, c}} + \sum_{t \in \{0, 1\}} \mathcal{L}_{\text{total}}^{D_{l \rightarrow r, t}} + \mathcal{L}_{\text{total}}^{D_{r \rightarrow l, t}} \quad (4.7)$$

4.3.2. RANSAC Outlier Rejection and Refinement Loss

Given the flow and disparity predictions, F , D , we can use the RANSAC algorithm (Fischler and Bolles, 1981) to estimate the best set of inliers to be used to estimate the pose of the cameras. RANSAC is a robust inlier selection scheme that allows us to filter out outliers such as independently moving objects from the optimization to estimate pose. The motion field equation that constrains pose, depth and optical flow is:

$$\begin{aligned} F(\vec{x}) &= \frac{1}{Z(\vec{x})} Av + B\omega \\ &= \frac{1}{Z(\vec{x})} \begin{bmatrix} -1 & 0 & x \\ 0 & -1 & y \end{bmatrix} v + \begin{bmatrix} xy & -(1+x^2) & y \\ 1+y^2 & -xy & -x \end{bmatrix} \omega \end{aligned} \quad (4.8)$$

$$Z(\vec{x}) = \frac{fb}{D(\vec{x})} \quad (4.9)$$

where x and y represent the coordinates in the normalized camera frame of the corresponding pixel, v and ω are the linear and angular velocity of the camera in the camera frame, $Z(\vec{x})$ is the depth in the camera frame, and Av and $B\omega$ are the 2-dimensional vectors corresponding to linear and angular velocity terms. f is the focal length of the camera and b is the baseline between the cameras. Note that, in this work, we estimate displacements rather than velocity, although we will keep the same notation for brevity. i.e. $v \times t$ and $\omega \times t$ instead of v and ω .

Using $F(\vec{x})$ and $D(\vec{x})$ from the network, this is a least squares problem for v and ω . We perform

3-point RANSAC by randomly sampling 3 points from $F(\vec{x})$ and $D(\vec{x})$, and solving for v and ω . The solution is then used to find the set of inlier points that best fit the estimated model. The threshold for inliers is set to be the absolute difference between the network and RANSAC flow estimates, in terms of pixels. Because the motion field equation above assumes a static scene, the computed inlier mask will automatically filter out independently moving (non-static) objects.

As noted in prior work by Brachmann et al. (2017), the hard thresholding to determine inliers and the argmax function used to compute the best set of inliers is non differentiable. However, given the set of inliers, the function to estimate camera velocity from the inlier set of flows and disparities is a simple matrix inversion, which is differentiable. In this work, we treat the inlier set as a fixed, non-learned mask. As a result, gradient flows from the estimated pose through the inlier flows and disparities. This makes the refinement pipeline differentiable with respect to the set of inlier points, but not the selection of inliers themselves.

Refinement Loss

Using the pose estimate from RANSAC and the motion field equation in (4.8), we can estimate the optical flow from the disparities and pose using the forwards equation, as in Zhou et al. (2017). However, as our flow predictions also directly contribute to the final pose estimate, we also estimate disparities using the flow and pose through the backwards equation derived from (4.8). To estimate disparity from flow and pose, we would like to find the disparities, $\hat{D}(\vec{x})$, that minimize $\|L(\vec{x})\|_2^2$, where:

$$L(\vec{x}) = F(\vec{x}) - \frac{\hat{D}(\vec{x})}{fb} Av - B\omega \quad (4.10)$$

This is equivalent to finding the scale between the vector v_1 and the projection of \vec{v}_2 onto \vec{v}_1 :

$$\hat{D}(\vec{x}) = \frac{\vec{v}_2^T \vec{v}_1}{\vec{v}_1^T \vec{v}_1} \quad (4.11)$$

$$\vec{v}_1 = Av/fb, \vec{v}_2 = F(\vec{x}) - B\omega \quad (4.12)$$

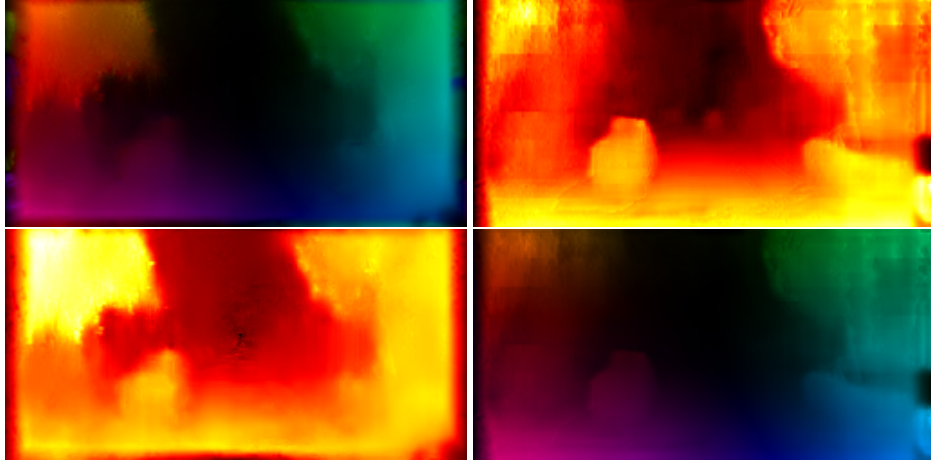


Figure 4.2: Using the RANSAC pose, we can use the predicted flow (top left) to estimate disparity (bottom left), and disparity (top right) to estimate flow (bottom right). Best viewed in color.

Using these estimated flows and disparities, we can then compute the appearance loss, (4.1), for the other image sequence. That is, we can use the disparities estimated from flow to compute the appearance loss on the stereo pair, and the flow estimates from disparity on the temporal pair. These losses allow each network to learn from an additional pair of images. An example of these estimates can be found in Fig. 4.2

4.3.3. Network Architecture

One disadvantage of our method is that it requires dense per-pixel optical flow and disparity predictions from two networks in order to estimate pose, as compared to other networks that directly regress pose using a, potentially smaller, CNN. In order to compensate for this, we use a much smaller pair of networks in our pipeline than prior works, such as Godard et al. (2017) and Meister et al. (2017). Our network is based on the FlowNet-S architecture (Fischer et al., 2015), but with a significantly smaller model. In short, we remove the final three convolution layers from the encoder and the first two convolution layers from the decoder, and halve the number of output channels at each layer. In our experiments, our network reduces an input image with resolution 128x448 to activations with resolution 8x28 and 256 channels at the bottleneck.

In Tab. 4.1, we compare the number of parameters in this proposed architecture with several methods, including Monodepth (Godard et al., 2017), SFM-Learner (Zhou et al., 2017), FlowNet (Fischler and

Method	# Parameters
Ours	2,943,496
Monodepth	31,596,936
SFM-Learner	33,187,568
FlowNet-S	35,885,068
Monodepth Resnet50	58,445,736

Table 4.1: Comparison of model size.

Bolles, 1981), as well as the Resnet50 (He et al., 2016) variant from Monodepth. Our mini network is less than 10% of the size of these networks.

4.4. Performance Evaluation

We evaluate our proposed pipeline on the KITTI Dataset (Geiger et al., 2013) and make comparisons to other state-of-the-art networks for depth, flow and pose estimation. We provide ablation studies to evaluate the contribution of the proposed refinement losses and we show qualitative examples where the outlier mask is able to correctly segment independently moving objects as outliers in Fig. 4.5.

4.4.1. Implementation Details

Our model was trained by first training each network for 8 epochs without the refinement losses, and then training with refinement losses for a total of 50 epochs. Input images were resized using bilinear interpolation to 128x448 pixels for both models. 100 RANSAC steps were run at each step, with an estimate considered an inlier if the norm between the RANSAC flow and network flow was less than 1 pixel.

Models used for pose evaluation were trained on the KITTI Odometry training sequences 00-08. Models used for optical flow and depth evaluation were trained on a subset of the KITTI Raw dataset, consisting of the city, residential and road sequences, with all images overlapping with the Eigen depth test split (Eigen et al., 2014), KITTI Stereo 2015 dataset and the KITTI Flow 2012 and 2015 datasets removed. Note that this is strictly less training data than training with each individual train/test split.

4.4.2. Experimental Results

We perform evaluations of each component in our pipeline on the KITTI Odometry 2012 and Flow and Stereo 2012 and 2015 datasets.

Pose Estimates

For quantitative pose evaluation, we compute average RMSE translation (%) and rotation ($^{\circ}/100m$) drift for all methods, as prescribed by Geiger et al. (2012). We compare our method against the ablation studies, as well as competing unsupervised deep SFM methods SFMLearner (Zhou et al., 2017), UnDeepVO (Li et al., 2018), Zhan et al. (2018), Luo et al. (2018) and DVSO (Yang et al., 2018a). In addition, we provide comparisons against the monocular optimization based visual odometry pipeline ORBSLAM (Mur-Artal et al., 2015b). As SFMLearner and Luo et al. are monocular methods that generate poses up to a scale, the trajectories are scale aligned before computing each error. These results can be found in Tab. 4.2.

From these results, our method outperforms UnDeepVO and Zhan et al., the instantaneous methods that predict up to scale depths across almost all sequences. In addition, these results strongly outperform monocular optimization frameworks which are unable to resolve the scale ambiguity such as ORBSLAM (Mur-Artal et al., 2015b), as noted in the results in UnDeepVO. We also outperform SFMLearner, and perform comparably to Luo et al. on the test set. However, as Luo et al. only regress depths up to a scale factor, the pose results have been compensated for this scale. Overall, DVSO strongly outperforms all instantaneous methods. However, this is to be expected as DVSO incorporates a windowed optimization method to reduce drift, which is the main contributor of error in instantaneous methods. We present our method as a step towards closing the gap between instantaneous and more global methods, with the additional benefit of providing interpretable pose results compared to learning only methods.

For qualitative results, Fig. 4.3 shows trajectories generated by the two models in our ablation against ground truth on 6 of the 11 sequences. Note that sequence 01 has a much higher translation error than the other datasets. This is because this is the only sequence on a highway in a very open space

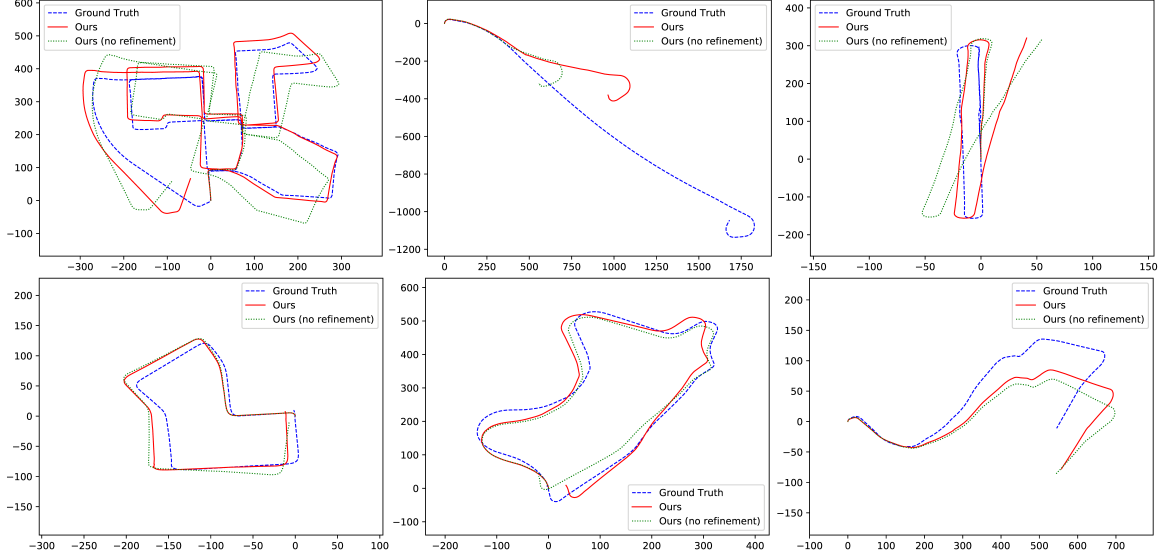


Figure 4.3: Selected trajectories of the models trained on KITTI raw before and after refinement, as well as our final model, against ground truth. Sequence numbers from top left to bottom right are 00, 01, 06, 07, 09, 10. Best viewed in color.

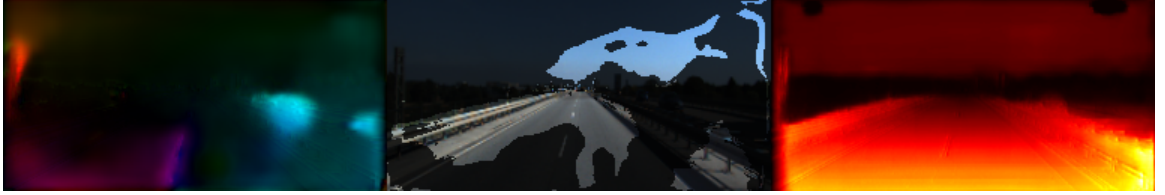


Figure 4.4: An example of a failure case from sequence 01. From left to right are flow, inlier mask and disparity respectively. The network consistently underestimates the flow over the textureless road section, causing RANSAC to select a large part of the sky as inliers. Best viewed in color.

without many other structures, as can be seen in Fig. 4.4. The network consistently underestimates the flow over the textureless road section, causing RANSAC to select a large part of the sky in the inlier set, and produce an underestimate of the translation. Note also that, despite the high translation error, the pipeline performs very well in terms of rotation error for sequence 01.

Depth Estimates

We evaluate our depth network prediction based on the test split and metrics used in Eigen et al. (2014). We compare our results according to the 50 m cap. The Eigen test split has around 697 images in total selected from KITTI Raw sequences. In addition, we test our model on the KITTI Stereo 2015 dataset (Menze et al., 2018) using the same metrics. Both sets of experiments use the

crop from Garg et al. (2016). We report both results and compare to other methods, and report the values as in Table 4.4.

From these results, our model outperforms Garg et al. (2016), SFMLearner (Zhou et al., 2017) and UnDeepVO (Li et al., 2018), while performing comparably to Zhan et al. (2018), which uses a larger image of 160x680 pixels, and performs worse than Godard et al. (2017) and DVSO (Yang et al., 2018a), which use images of 256x512 pixels, and are semi-supervised in the case of DVSO, by depths initialized by DSO (Engel et al., 2018).

Flow Estimates

Our optical flow evaluation is performed on KITTI Flow 2015 training dataset (Menze et al., 2015). We have the ground truth from 200 images in the training set and we calculate the average Endpoint Error (EPE) as an error metric. We then calculate the percentage of outliers (flows with $EPE > 3$ and $> 5\%$ of the true magnitude) on both the non-occluded (noc) pixels and the whole image (all) using the metrics defined by the KITTI Dataset (Geiger et al., 2013).

We compare to other state-of-the-art unsupervised methods including the work by Luo et al. (2018), UnFlow (Meister et al., 2017), and GeoNet (Yin and Shi, 2018). From these results, we perform significantly worse than the competing methods, likely due to the lower capacity and smaller input resolution in our network. However, we show in Sec. 4.4.4 that our RANSAC method is able to reject the high error flow values, resulting in statistics that outperform the competitors in these metrics.

4.4.3. Ablation on Refinement Loss

We perform an ablation study on the effect of the proposed refinement losses, by training an additional network without these losses (i.e. with (4.3.1) as the loss). Results for pose, depth and flow can be found in Tab. 4.3, 4.4, 4.5, respectively.

From these results, we can see that the refinement losses result in improvements in pose and flow, with a neutral effect on depth. This corresponds from our observations during training that disparity estimation is an easier problem than optical flow for a dataset with a consistent environment. Disparity estimation is a 1D search problem, and is constrained to be positive, while flow is 2D and

Sequence	00		01		02		03		04		05		06	
	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}
Ours	4.95	1.39	45.5	1.78	6.40	1.92	4.83	2.11	2.43	1.16	3.97	1.20	3.49	1.02
SFMLearner (Zhong et al., 2017)	66.4	6.13	35.2	2.74	58.8	3.58	10.8	3.92	4.49	5.24	18.7	4.10	25.9	4.80
UnDeepVO (Li et al., 2018)	4.14	1.92	69.1	1.60	5.58	2.44	5.00	6.17	4.49	2.13	3.40	1.50	6.20	1.98
Zhan et al. (2018)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Luo et al. (2018)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DVSO (Yang et al., 2018a)	0.71	0.24	1.18	0.11	0.84	0.22	0.77	0.18	0.35	0.06	0.58	0.22	0.71	0.20
	07		08		09*		10*		Mean*		Mean (all)			
	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}		
Ours	4.50	1.78	4.08	1.17	4.66	1.69	6.30	1.59	5.48	1.64	8.28	1.59		
SFMLearner (Zhong et al., 2017)	21.3	6.65	21.9	2.91	18.8	3.21	14.3	3.30	16.6	3.26	27.0	4.23		
UnDeepVO (Li et al., 2018)	3.15	2.48	4.08	1.79	7.01	3.61	10.6	4.65	8.82	4.13	11.2	2.75		
Zhan et al. (2018)	-	-	-	-	11.9	3.60	12.6	3.43	12.3	3.52	-	-		
Luo et al. (2018)	-	-	-	-	3.72	1.60	6.06	2.22	4.89	1.91	-	-		
DVSO (Yang et al., 2018a)	0.73	0.35	1.03	0.25	0.83	0.21	0.74	0.21	0.79	0.21	0.77	0.20		

Table 4.2: Evaluation against competing instantaneous unsupervised pose learning methods, as well as DVSO (Yang et al., 2018a), which uses a windowed optimization. Translation t_{rel} (%) and rotation r_{rel} (°/100 m) RMSE on the 11 KITTI Odometry training sequences is presented. Bold indicates best instantaneous result for each column. Final means are computed over the training set (sequences 09 and 10) and all sequences, respectively. As SFMLearner (Zhou et al., 2017) and Zhan et al. (2018) are monocular methods, their results have been corrected for scale. Only results for 09 and 10 are provided in Zhan et al. (2018) and Luo et al. (2018).

can also be negative. By estimating disparity from flow, our refinement losses allow our network to reduce a component of the optical flow problem into a 1D search in the positive disparity space. On the other hand, estimating flow from disparity converts a 1D problem to 2D, and provides negligible gains. This improvement in flow error also helps to validate the accuracy of the estimated pose, as the refinement losses are only valid provided an accurate pose estimate.

4.4.4. Inlier Quality

In this section, we investigate the quality of the flows and depths selected by RANSAC as inliers. In Tab. 4.6 and Tab. 4.7, we present the flow and depth errors, computed only over inlier points. For the depth results, we also provide results without the crop from Garg et al. (2016), as this already removed many difficult points that would be marked as outliers. From these results, we can also see that RANSAC selects points that have significantly lower error than the average. From the flow results, we can see that the “noc” and “all” results are very similar, suggesting that almost all of the occluded points have been rejected correctly by RANSAC. Essentially, by using RANSAC for inlier

Sequence	00		01		02		03		04		05	
	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}
Ours	4.95	1.39	45.5	1.78	6.40	1.92	4.83	2.11	2.43	1.16	3.97	1.20
Ours (no refinement)	5.45	1.78	63.9	1.53	6.21	1.76	4.27	1.79	2.05	1.04	3.82	1.38
	06		07		08		09		10		Mean	
	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}
Ours	3.49	1.02	4.50	1.78	4.08	1.17	4.66	1.69	6.30	1.59	8.28	1.59
Ours (no refinement)	4.02	1.63	5.47	2.56	3.73	1.09	5.11	1.95	6.71	2.52	10.06	1.73

Table 4.3: Pose ablation study on the effect of the proposed refinement losses with models trained with and without the proposed refinement losses. Models were trained on sequences 00-08.

	Split	RMSE	RMSE(log)	Abs Rel	Sq Rel	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Ours (no refinement)	stereo2015	6.753	0.274	0.182	2.570	0.807	0.924	0.963
Ours	stereo2015	6.394	0.257	0.168	2.114	0.820	0.931	0.968
Ours (no refinement)	eigen	4.240	0.218	0.147	0.897	0.816	0.938	0.975
Ours	eigen	4.366	0.230	0.154	1.021	0.808	0.933	0.972
DVSO (Yang et al., 2018a)	eigen	3.390	0.177	0.092	0.547	0.898	0.962	0.982
Godard et al. (2017) resnet pp	eigen	3.729	0.194	0.108	0.657	0.873	0.954	0.979
Garg et al. (2016)	eigen	5.104	0.273	0.169	1.080	0.740	0.904	0.962
SFMLearner (Zhong et al., 2017)	eigen	5.181	0.264	0.201	1.391	0.696	0.900	0.966
UnDeepVO (Li et al., 2018)	eigen	6.570	0.268	0.183	1.730	-	-	-
Zhan et al. (2018)	eigen	4.204	0.216	0.128	0.815	0.835	0.941	0.975

Table 4.4: Depth evaluation results on the Eigen (50 m cap) test split and KITTI stereo 2015. Left: lower is better, right: higher is better.

selection, we are able to estimate camera velocity from the equivalent of a much better model.

This is likely possible due to the fact that the network tended to learn easier parts of the image first, such as high gradient corners, before slowly learning progressively harder regions such as textureless areas. Thus, the network generates reasonable flow and disparity values for at least some pixels, even early on in training or for very small networks. RANSAC is then able to select these correct values, even in the presence of challenging outliers.

We show some examples of RANSAC segmenting independently moving objects in Fig. 4.5. As the outlier rejection is purely geometric, it does not rely on the network to learn that these objects have independent motion. The outlier masks also show that RANSAC segments regions where the networks tend to struggle, such as strong shadows (second row), sky (fifth row), thin structures (sixth

	EPE		Error Perc	
	noc	all	noc	all
Ours	10.66	19.74	45.84%	54.19%
Ours (no refinement)	11.80	21.16	46.77%	54.99%
Luo et al. (2018)	3.86	5.66	-	-
UnFlow (Meister et al., 2017)	-	8.10	-	23.27%
GeoNet (Yin and Shi, 2018)	8.05	10.81	-	-

Table 4.5: Comparison and ablation of flow prediction on the KITTI Flow 2015 dataset. noc = occluded pixels removed. A pixel is considered an outlier if its EPE is higher than 3 pixels and 5% of its true magnitude.

	EPE on Inliers		Error Perc. on Inliers	
	noc	all	noc	all
Ours	3.21	3.56	20.81	21.12
Ours (no refinement)	3.25	3.39	21.07	21.24

Table 4.6: Evaluation of inlier quality with flow errors on the KITTI Flow 2015 dataset. Errors are computed only over inlier pixels, $\sim 21\%$ of pixels.

row) and vegetation (seventh row).

4.5. Discussion and Insights

We demonstrate a novel structure from motion pipeline that combines unsupervised learning with geometric optimization. Our method is able to compete with state-of-the-art methods which directly predict pose from a network, while providing extra guarantees about robustness and automatic detection of independently moving objects. We show that this pipeline can achieve state-of-the-art accuracy with a significantly reduced deep learning model. We identify the following key observations from this work:

1. With depth estimation from the network, our method has significant improvement over monocular visual odometry optimization frameworks as the learned predictions resolve scale ambiguities.
2. The proposed hybrid pipeline outperforms other learning-based approaches that regress pose directly from the network on instantaneous relative pose estimates.

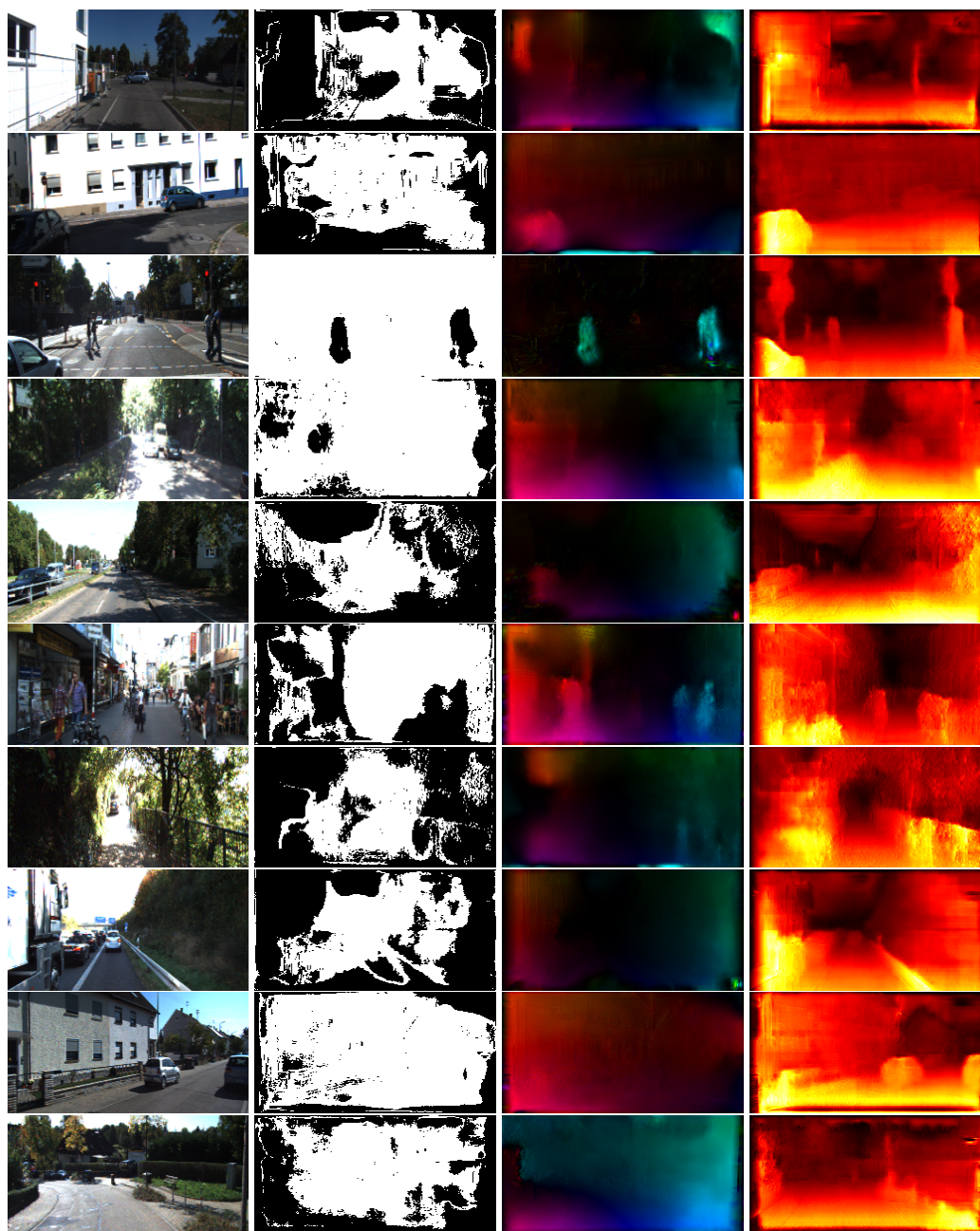


Figure 4.5: Interesting qualitative results from our pipeline. Left to right: Grayscale image, inlier mask, predicted flow, predicted disparity. The inlier mask demonstrates RANSAC’s ability to remove areas over which the network has high uncertainty, such as vegetation and textureless regions such as the sky. In addition, it allows us to automatically remove independently moving objects from the scene. Note that in the fourth image, there is a person slight left of the center of the image. Best viewed in color.

	RMSE	RMSE (log)	Abs Rel	Sq Rel
Ours	12.63	0.30	0.24	22.63
Ours (inliers)	8.00	0.24	0.16	3.63
Ours (inliers, no ref.)	9.00	0.24	0.17	7.92

Table 4.7: Evaluation of inlier quality with depth errors from the KITTI Stereo 2015 dataset, without the crop from Garg et al. (2016). The first row is computed over all pixels, while the last two are computed only over inlier pixels, $\sim 35\%$ of pixels.

3. The refinement loss connects the training process of flow and disparity and implicitly uses more data, therefore improves the performance within our unsupervised pipeline.
4. The RANSAC procedure successfully filters out inconsistent estimates from the networks, including moving objects, vegetation, and textureless areas.
5. Training data selection is important for system performance, and this approach is application-specific to driving scenarios consistent with training data.

A future direction is to extend this work to a batch optimization back-end using multiple images to compete with frameworks with a complete VO pipeline. To add additional constraints between multiple camera poses, correspondence information is needed. However, in our current framework, only pairwise flow information is available, which is a barrier to direct system extension. How to design a system that preserves the benefits in the current hybrid framework is an open research question.

CHAPTER 5

TIGHT-LEARNED INERTIAL ODOMETRY

In this chapter, we remove the most critical information from the equation, which is vision. Assuming camera failure or a power-saving option in tracking, we investigate the problem of inertial-only state estimation in a pedestrian setting with applications in AR/VR. The MEMS IMU sensor data is corrupted by noise and bias. The double integration from linear acceleration to position in the traditional kinematics motion model means the bias gets accumulated very quickly, causing a large amount of drift in a short period of time. We introduce a new approach featuring the advantage of incorporating deep learning in a tightly-coupled Extended Kalman Filter framework for IMU-only state estimation, and show how this approach effectively improves the state-of-the-art inertial tracking performances. We demonstrate a network that regresses 3D displacement estimates and its uncertainty, allowing us to tightly fuse the relative state measurement into a stochastic cloning EKF to solve pose, velocity, and sensor biases. We show that our network, trained with pedestrian data from a headset, can produce statistically consistent measurement and uncertainty to be used as the update step in the filter, and the tightly-coupled system outperforms both velocity integration approaches in position estimates and the AHRS attitude filter in orientation estimates.

Video materials and code can be found on our project page: cathias.github.io/TLIO

5.1. Introduction

Visual-Inertial Navigation Systems (VINS) in recent years have seen tremendous success, enabling a wide range of applications from mobile devices to autonomous systems. Using only light-weight cameras and Inertial Measurement Units (IMUs), VINS achieve high accuracy in tracking at low cost and provide one of the best solutions for localization and navigation on constrained platforms. With the unique combination of these advantages, VINS have been the de facto standard for demanding applications such as Virtual/Augmented Reality (VR/AR) on mobile phones or headsets.

Despite the impressive performance of state-of-the-art VINS algorithms, demands from consumer AR/VR products are posing new challenges on state estimation, pushing the research frontier. Visual-

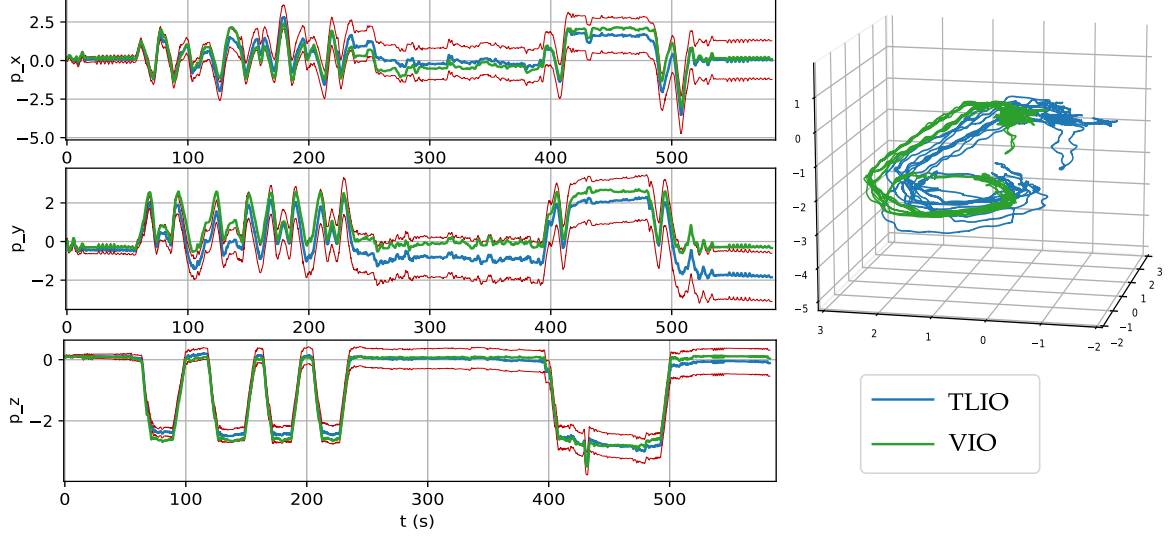


Figure 5.1: An example 3D trajectory estimated by TLIO from a single MEMS IMU, showing the user repeatedly walking up and down a staircase. Our method estimates full orientation and translation in 3D: On the left, we show the estimated x, y, z position (blue) as well as associated uncertainties ($\pm 3\sigma$ in red). On the right, we show the resulting 3D trajectory. For both plots, we show the output of state-of-the-art visual-inertial odometry (green) for comparison.

Inertial Odometry (VIO) largely relies on consistent image tracking, leading to failure cases under extreme lighting or camera blockage conditions, such as in a dark room or inside a pocket. High frequency image processing makes power consumption a bottleneck for sustainable long-term operations. In addition, widespread camera usage carries privacy implications. Targeting an alternative to the state-of-the-art VINS algorithms for pedestrian applications, this work focuses on consumer grade IMU-only state estimation problem known as strap-down Inertial Navigation System (INS) or Dead Reckoning (Groves, 2015).

Inertial Pedestrian Dead Reckoning (PDR) has gained increasing interest with the price decline and widespread usage of MEMS sensors in the past two decades (Harle, 2013). Strap-down pedestrian IMU navigation faces the challenge of the accumulation of sensor errors since the IMU kinematic model provides only relative state estimates. To compensate for the errors and reduce drift without the aid of other sensors or floor maps, the existing approaches rely on the prior knowledge of human walking motion, in particular, steps. One way to make use of steps is Zero-velocity UPdaTe (ZUPT) (Foxlin, 2005). It detects when the foot touches the ground to generate pseudo-measurement updates

in an Extended Kalman Filter (EKF) framework to calibrate IMU biases. However, it only works well when the sensor is attached to the foot where step detection is obvious. Another category is step counting (Brajdic and Harle, 2013) which does not require the sensor to be attached to the foot. Such systems consist of multiple submodules: the identification and classification of steps, data segmentation and step length prediction, all of which require heavy tuning of hand-designed heuristics or machine learning.

In parallel, recent research advances (IONet (Chen et al., 2018a) and RoNIN (Yan et al., 2020)) have shown that integrating average velocity estimates from a trained neural network results in highly accurate 2D trajectory reconstruction using only IMU data from pedestrian hand-carried devices. These results showed the ability of networks to learn a translation motion model from pedestrian data. In this work, we draw inspiration from these new findings to build an IMU-only dead reckoning system trained with data collected from a head-mounted device, similar to a VR headset. This work has two major contributions:

1. We develop a network design to regress both the 3D displacement and the corresponding covariance, and show the network’s capability of providing both accurate and statistically consistent outputs trained on our pedestrian dataset.
2. We demonstrate a complete state estimation system combining the neural network with an EKF in a tightly-coupled formulation that jointly estimates position, orientation, velocity, and IMU biases with only pedestrian IMU data.

Our approach presents significant advantages over other IMU-only state estimation approaches. Comparing to traditional PDR methods, it avoids the limitations and complexities of step-counting or stride and gait detection by learning a displacement motion model valid for any data segments, simplifying the process and improving accuracy and robustness. Comparing to common deep learning approaches like RoNIN, it elevates the problem onto 3D domain and does not require an external ZUPT-based orientation estimator. This tight fusion approach reduces average yaw and position drift by 27% and 33% respectively on our test dataset comparing to the best performing RoNIN velocity

concatenation baseline approach. Fig. 5.1 shows an example of the estimated trajectory. We name our method TIGHT LEARNED INERTIAL ODOMETRY (TLIO).

5.2. Related Works

VIO. Visual-Inertial Odometry has been well studied in the literature (Chen et al., 2018b). Without power and privacy issues and when static visual features are clearly trackable, VIO is the standard method for state estimation.

PDR. Solutions to Pedestrian Dead Reckoning problems can be divided into two categories: Bayesian filtering (Jiménez et al., 2010) and step counting (Brajdic and Harle, 2013). With IMU as the only source of information for both propagation and update steps of an EKF, the measurements need to be carefully constructed. Different types of pseudo measurement approaches include Zero-velocity UPdaTe (ZUPT) (Foxlin, 2005) and Zero Angular Rate Update (ZARU) (Rajagopal, 2008) which detects when the system is static, and heuristic heading reduction (HDR) (Borenstein et al., 2009) which identifies walking in a straight line to calibrate gyroscope bias. Foot-mounted IMU sensor enables reliable detection of contact from which accurate velocity constraints can be derived (Ju et al., 2016), however such information is not present for hand carried devices, and various signal processing and machine learning algorithms such as peak detection in time/frequency domain and activity classification have been explored (Brajdic and Harle, 2013). These algorithms are complex and involve lots of tuning to accurately estimate a trajectory. It is also an active research direction as deep learning approaches solving for subproblems such as gait classification (Dehzangi et al., 2017) and stride detection (Beaufils et al., 2019) are investigated.

Deep Learning. The emergence of deep learning provides new possibilities to extract information from IMU data. Estimating average velocity from IMU data segments has seen great success in position estimation. IONet (Chen et al., 2018a) first proposed an LSTM structure to output relative displacement in 2D polar coordinates and concatenate to obtain position. RIDI (Yan et al., 2018) and RoNIN (Yan et al., 2020) both assume orientation is known from an Android device to rotate IMU data into a gravity-aligned frame. While RIDI regresses velocity to optimize bias but still uses double integration from the corrected IMU data for position, RoNIN directly integrates the regressed

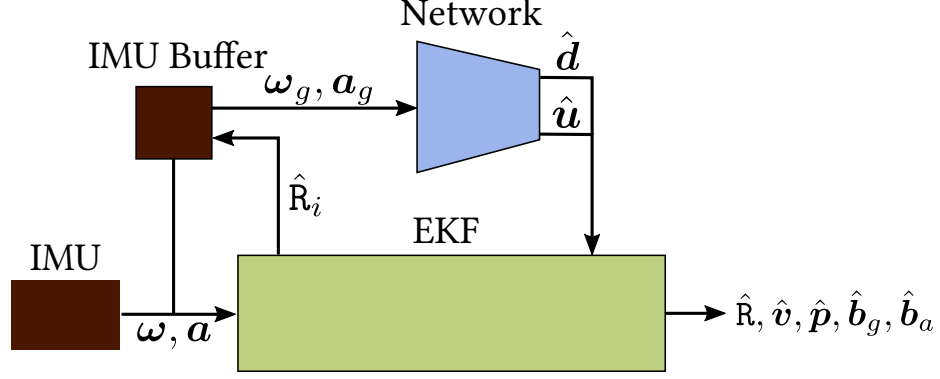


Figure 5.2: System block diagram. The IMU buffer provides segments of gravity-aligned IMU measurements to the network, using rotation from the filter state. The network outputs displacement \hat{d} and uncertainty \hat{u} used as measurement update to the filter. The filter estimates rotation, velocity, position and IMU biases at IMU rate.

velocity and showed better results. This shows that a statistical IMU motion model can outperform the traditional kinematic IMU model in scenarios that can be captured with training data.

In addition to using networks alone for pose estimates, Backprop KF (Haarnoja et al., 2016) proposes an end-to-end differentiable Kalman filter framework. Because the loss function is on the accuracy of the filter outputs, the noise parameters are trained to produce the best state estimate, and do not necessarily best capture the measurement error model. AI-IMU (Brossard et al., 2019) uses this approach to estimate IMU noise parameters and measurement uncertainties for car applications. In this work, we also combine deep learning with a Kalman filter. However, instead of training an end-to-end system, we leverage a state-of-the-art visual-inertial fusion algorithm as ground-truth for supervised learning of the measurement function itself. We use a likelihood loss function to jointly train for 3D displacement and uncertainty, which are directly used as input to the EKF in the measurement update.

5.3. System Design

Our estimation system takes IMU data as input and has two major components: the network and the filter. Figure 5.2 shows a high-level diagram of the system.

The first component is a convolutional neural network trained to regress 3D relative displacement and uncertainty between two time instants given the segment of IMU data in between. The network is

required to infer positional displacement over a short timespan from acceleration and angular velocity measurements alone, without access to the initial velocity. This is intentional: It forces the network to learn only a prior expressing "typical human motion" and leaves model-based state propagation, i.e. acceleration integration to propagate velocity, and respective uncertainty propagation, to the second system component, the EKF. In order for this to work well, we found that it is important to include the inferred prior's uncertainty to the network output, allowing the network to encode how much motion model prior it obtained from the input measurements.

The EKF estimates the current state: 3D position, velocity, orientation and IMU biases, and a sparse set of past poses. The EKF propagates with raw IMU samples and uses network outputs for measurement updates. We define the measurement in a local *gravity-aligned frame* to decouple global yaw information from the relative state measurement (see Sec. 5.5.4). The propagation from raw IMU data provides a model-based kinematic motion model, and the neural network provides a statistical motion model. The filter tightly couples these two sources of information.

At runtime, the raw IMU samples are interpolated to the network input frequency and rotated to a local gravity-aligned frame using rotations estimated from the filter state and gyroscope data. A gravity-aligned frame always has gravity pointing downward. Placing data in this frame implicitly gives the gravity direction information to the network.

Note that in the proposed approach, IMU data is being used twice, both as direct input for state propagation and indirectly through the network as the measurement. In fact, with overlapping windows of IMU data in the network inference, IMU data is used more than twice. This violates the independence assumptions the EKF is based on: the errors in the IMU network input (initial orientation, sensor bias, and sensor noise) would propagate to its output which is used to correct the propagation results polluted with the same noise, which could lead to estimation inconsistency. We address this issue two-fold: We deliberately design the system to separate the purpose of the network model and the propagation model. We exclude initial velocity from the network input, preventing the network from learning the IMU kinematic model, and learn a statistical model of human walking patterns instead. We further leverage training techniques (see Sec. 5.4) to reduce error propagation by adding

random perturbations to IMU bias and gravity direction during training. We show in Section 5.7 that these techniques successfully improved the robustness of the network to sensor bias and rotation inaccuracies.

5.4. Neural Statistical Motion Model

5.4.1. Architecture and Loss Function Design

Our network uses a 1D version of ResNet18 architecture proposed in He et al. (2016). The input dimension to the network is $N \times 6$, consisting of N IMU samples in the gravity-aligned frame. The output of the network contains two 3D vectors: the displacement estimates $\hat{\mathbf{d}}$ and their uncertainties $\hat{\mathbf{u}}$ which parametrize the diagonal entries of the covariance. The two vectors have independent fully-connected blocks extending the main ResNet architecture.

We make use of two different loss functions during training: the Mean Square Error (MSE) and the Gaussian Maximum Likelihood loss. The MSE loss on the trained dataset is defined as:

$$\mathcal{L}_{\text{MSE}}(\mathbf{d}, \hat{\mathbf{d}}) = \frac{1}{n} \sum_{i=1}^n \|\mathbf{d}_i - \hat{\mathbf{d}}_i\|^2 \quad (5.1)$$

where $\hat{\mathbf{d}} = \{\hat{\mathbf{d}}_i\}_{i \leq n}$ are the 3D displacement output of the network and $\mathbf{d} = \{\mathbf{d}_i\}_{i \leq n}$ are the ground truth displacement. n is the number of data in the training set.

We define the Maximum Likelihood loss as the negative log-likelihood of the displacement according to the regressed Gaussian distribution:

$$\begin{aligned} \mathcal{L}_{\text{ML}}(\mathbf{d}, \hat{\Sigma}, \hat{\mathbf{d}}) &= \frac{1}{n} \sum_{i=1}^n -\log \left(\frac{1}{\sqrt{8\pi^3 \det(\hat{\Sigma}_i)}} e^{-\frac{1}{2} \|\mathbf{d}_i - \hat{\mathbf{d}}_i\|_{\hat{\Sigma}_i}^2} \right) \\ &= \frac{1}{n} \sum_{i=1}^n \left(\frac{1}{2} \log \det(\hat{\Sigma}_i) + \frac{1}{2} \|\mathbf{d}_i - \hat{\mathbf{d}}_i\|_{\hat{\Sigma}_i}^2 \right) + \text{Cst} \end{aligned} \quad (5.2)$$

where $\hat{\Sigma} = \{\hat{\Sigma}_i\}_{i \leq n}$ are the 3×3 covariance matrices for i^{th} data as a function of the network uncertainty output vector $\hat{\mathbf{u}}_i$. $\hat{\Sigma}_i$ has 6 degrees of freedom, and there are various covariance parametrizations for neural network uncertainty estimation (Russell and Reale, 2019). In this work, we simply

assume a diagonal covariance output, parametrized by 3 coefficients written as:

$$\hat{\Sigma}_i(\hat{\mathbf{u}}_i) = \text{diag}(e^{2\hat{u}_{xi}}, e^{2\hat{u}_{yi}}, e^{2\hat{u}_{zi}}) \quad (5.3)$$

The diagonal assumption decouples each axis, while regressing the logarithm of the standard deviations removes the singularity around zero in the loss function, adding numerical stabilization and helping the convergence in the optimization process. This choice constrains the principal axis of the uncertainty ellipses to be along the gravity-aligned frame axis.

5.4.2. Data Collection and Implementation Details

We use a dataset collected by a custom rig where an IMU (Bosch BMI055) is mounted on a headset rigidly attached to the cameras. The full dataset contains more than 400 sequences totaling 60 hours of pedestrian data that pictures a variety of activities including walking, standing still, organizing the kitchen, playing pool, going up and down the stairs etc. It was captured with multiple different physical devices by more than 5 people to depict a wide range of individual motion patterns and IMU systematic errors. A state-of-the-art visual-inertial filter based on Mourikis and Roumeliotis (2007) provides position estimates at 1000 Hz on the entire dataset. We use these results both as supervision data in the training set and as ground truth in the test set. The dataset is split into 80% training, 10% validation and 10% test subsets randomly.

For network training, we use an overlapping sliding window on each sequence to collect input samples. Each window contains N IMU samples of total size $N \times 6$. In our final system we choose $N = 200$ for 200 Hz IMU data. We want the network to capture a motion model with respect to the gravity-aligned IMU frame, therefore the IMU samples in each window are rotated from the IMU frame to a gravity-aligned frame built from the orientation at the beginning of the window. We use visual-inertial ground-truth rotation for that purpose. The supervision data for the network output is computed as the difference of the ground-truth position between two instants expressed in the same headset-centered, gravity-aligned frame.

During training, because we assume the headset can be worn at an arbitrary heading angle with re-

spect to the walking direction, we augment the input data for the network to be yaw invariant by giving a random horizontal rotation to each sample following RoNIN (Yan et al., 2020). In our final estimator, the network is fed with potentially inaccurate input from the filter, especially at the initialization stage. We simulate this at training time by random perturbations on the sensor bias and the gravity direction to reduce network sensitivity to these input errors. To simulate bias, we generate additive bias vectors with each component independently sampled from uniform distribution in $[-0.2, 0.2] m/s^2$ or $[-0.05, 0.05] rad/s$ for each input sample. Gravity direction is perturbed by rotating those samples along a random horizontal rotation axis with magnitude sampled from $[0, 5^\circ]$.

Optimization is done through the Adam optimizer. We used an initial learning rate of 0.0001, zero weight decay, and dropouts with a probability of 0.5 for the fully connected layers. We observe that training for \mathcal{L}_{ML} directly does not converge. Therefore we first train with \mathcal{L}_{MSE} for 10 epochs until the network stabilizes, then switch to \mathcal{L}_{ML} until the network fully converges. It takes around another 10 epochs to converge and a total of 4 hours of training time is needed on an NVIDIA DGX computer.

5.5. Stochastic Cloning Extended Kalman Filter

The EKF in our system tightly integrates the displacements predicted by the network with a statistical IMU model as used in other inertial navigation systems. As the displacement estimates from the network express constraints on pairs of past states, we adopt a stochastic cloning framework (Roumeliotis and Burdick, 2002). Similar to Mourikis and Roumeliotis (2007), we maintain a sliding window of m poses in the filter state. In contrast to Mourikis and Roumeliotis (2007), however, we only apply constraints between *pairs* of poses, and these constraints are derived from the network described in the preceding section, rather than camera data.

5.5.1. State Definition

At each instant, the full state of the EKF is defined as:

$$\mathbf{X} = (\xi_1, \dots, \xi_m, \mathbf{s})$$

where $\xi_i, i = 1, \dots, m$ are past (cloned) states, and s is the current state. More specifically,

$$\xi_i = ({}^w_iR_i, {}^w_i p_i), \quad s = ({}^w_iR, {}^w v, {}^w p, b_g, b_a)$$

We express w_iR as the rotation matrix that transforms a point from the IMU frame to the world frame, and ${}^w v$ and ${}^w p$ are respectively the velocity and position expressed in the world frame. In the following, we will drop the explicit superscript for conciseness. b_g, b_a are the IMU gyroscope and accelerometer biases.

As commonly done in such setups, we apply the error-based filtering methodology to linearize locally on the manifold of the minimal parametrization of the rotation. More specifically, the filter covariance is defined as the covariance of the following error-state:

$$\tilde{\xi}_i = (\tilde{\theta}_i, \delta \tilde{p}_i), \quad \tilde{s} = (\tilde{\theta}, \delta \tilde{v}, \delta \tilde{p}, \delta \tilde{b}_g, \delta \tilde{b}_a)$$

where *tilde* indicates errors for every state as the difference between the estimate and the real value, except for rotation error which is defined as $\tilde{\theta} = \log_{SO3}(\hat{R}\tilde{R}^{-1}) \in \mathfrak{so}(3)$ where \log_{SO3} denotes the logarithm map of rotation. The full error-state is of dimension $6m + 15$, where m is the total number of past states, and \tilde{s} has dimension of 15.

5.5.2. IMU model

We assume that the IMU sensor provides direct measurements of the non-gravitational acceleration a and angular velocity ω , expressed in the IMU frame. We assume as it is common for this class of sensor that the measurements are polluted with noise $n_{g;a}$ and bias $b_{g;a}$.

$$\omega = \omega_{\text{true}} + b_g + n_g, \quad a = a_{\text{true}} + b_a + n_a.$$

n_g and n_a are random noise variables following a zero-centered Gaussian distribution. Evolution of biases is modeled as a random walk process with discrete parameters η_{gd} and η_{ad} over the IMU sampling period Δt .

5.5.3. State Propagation and Augmentation

The filter propagates the current state \mathbf{s} with IMU data using a kinematic motion model. If the current timestamp is associated to a measurement update, stochastic cloning is performed together with propagation in one step. During cloning, a new state ξ is appended to the past state.

Propagation Model

We use the strapdown inertial kinematic equation assuming a uniform gravity field ${}^w\mathbf{g}$ and ignoring Coriolis forces and the earth's curvature:

$${}^w\hat{\mathbf{R}}_{k+1} = {}^w\hat{\mathbf{R}}_k \exp_{SO3}((\boldsymbol{\omega}_k - \hat{\mathbf{b}}_{gk})\Delta t) \quad (5.4)$$

$${}^w\hat{\mathbf{v}}_{k+1} = {}^w\hat{\mathbf{v}}_k + {}^w\mathbf{g}\Delta t + {}^w\hat{\mathbf{R}}_k(\mathbf{a}_k - \hat{\mathbf{b}}_{ak})\Delta t \quad (5.5)$$

$${}^w\hat{\mathbf{p}}_{k+1} = {}^w\hat{\mathbf{p}}_k + {}^w\hat{\mathbf{v}}_k\Delta t + \frac{1}{2}\Delta t^2({}^w\mathbf{g} + {}^w\hat{\mathbf{R}}_k(\mathbf{a}_k - \hat{\mathbf{b}}_{ak})) \quad (5.6)$$

$$\hat{\mathbf{b}}_{g(k+1)} = \hat{\mathbf{b}}_{gk} + \boldsymbol{\eta}_{gdk} \quad (5.7)$$

$$\hat{\mathbf{b}}_{a(k+1)} = \hat{\mathbf{b}}_{ak} + \boldsymbol{\eta}_{adk}. \quad (5.8)$$

Here, \exp_{SO3} denotes the SO(3) exponential map, the inverse function of \log_{SO3} . The discrete-time noise $\boldsymbol{\eta}_{gdk}$, $\boldsymbol{\eta}_{adk}$ have been defined above and follow a Gaussian distribution.

The linearized error propagation can be written as:

$$\tilde{\mathbf{s}}_{k+1} = \mathbf{A}_{k(15,15)}^s \tilde{\mathbf{s}}_k + \mathbf{B}_{k(15,12)}^s \mathbf{n}_k \quad (5.9)$$

where $\mathbf{n}_k = [\mathbf{n}_{\omega k}, \mathbf{n}_{ak}, \boldsymbol{\eta}_{gdk}, \boldsymbol{\eta}_{adk}]^T$ is a vector containing all random input. The superscript s indicates that these matrices correspond to the current state \mathbf{s} , and the subscript brackets indicate matrix dimensions. The expanded mathematical expression can be found in Appendix A.1. The

corresponding propagation of the state covariance \mathbf{P} is:

$$\mathbf{P}_{k+1} = \mathbf{A}_k \mathbf{P}_k \mathbf{A}_k^T + \mathbf{B}_k \mathbf{W} \mathbf{B}_k^T, \quad (5.10)$$

$$\mathbf{A}_k = \begin{bmatrix} \mathbf{I}_{6m} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_k^s \end{bmatrix}, \mathbf{B}_k = \begin{bmatrix} \mathbf{0} \\ \mathbf{B}_k^s \end{bmatrix} \quad (5.11)$$

with $\mathbf{W}_{(12,12)}$ the covariance matrices of sensor noise and bias random walk noise. The state covariance \mathbf{P} is of the dimension $(6m + 15)$ of the full state \mathbf{X} . In the implementation, multiple propagation steps can be combined together to reduce the computational cost (Roumeliotis and Burdick, 2002).

State Augmentation

In our system, state augmentations are performed at the measurement update frequency. During an augmentation step, the state dimension is incremented through propagation with cloning:

$$\mathbf{P}_{k+1} = \bar{\mathbf{A}}_k \mathbf{P}_k \bar{\mathbf{A}}_k^T + \bar{\mathbf{B}}_k \mathbf{W} \bar{\mathbf{B}}_k^T, \quad (5.12)$$

$$\bar{\mathbf{A}}_k = \begin{bmatrix} \mathbf{I}_{6m} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_k^\xi \\ \mathbf{0} & \mathbf{A}_k^s \end{bmatrix}, \bar{\mathbf{B}}_k = \begin{bmatrix} \mathbf{0} \\ \mathbf{B}_k^\xi \\ \mathbf{B}_k^s \end{bmatrix} \quad (5.13)$$

$\bar{\mathbf{A}}_k$ is now a copy operation plus the augmented and current state propagation operations. \mathbf{A}_k^ξ and \mathbf{B}_k^ξ are partial propagation matrices for rotation and position only. After the increment, the dimension of the state vector increases by 6. Old past states will be pruned in the marginalization step, see Sec.5.5.5.

5.5.4. Measurement Update

It would be natural to define the measurement function as the displacement \mathbf{d} expressed in the world frame, however such measurement function would imply heading observability at the filter level. Heading is theoretically unobservable as the IMU propagation model and the learned prior are invariant to the change of yaw angle. In order to prevent injecting spurious information into the filter,

we carefully define the measurement function h as the 3D displacement in a *local gravity-aligned* frame. This frame is anchored to a clone state i . Vectors in this frame can be obtained by rotating the corresponding world frame vectors by the yaw rotation matrix R_γ of the state rotation matrix R_i . We decompose R_i using extrinsic "XYZ" Euler angle convention: $R_i = R_\gamma R_\beta R_\alpha$, where α, β, γ correspond to roll, pitch and yaw respectively. h then writes:

$$h(\mathbf{X}) = R_\gamma^T ({}^w\mathbf{p}_j - {}^w\mathbf{p}_i) = \hat{\mathbf{d}}_{ij} + \boldsymbol{\eta}_{d_{ij}}. \quad (5.14)$$

$\hat{\mathbf{d}}_{ij}$ is the network output displacement between state i and j , using IMU samples rotated to the local gravity-aligned frame anchored at pose i as input. $\boldsymbol{\eta}_{d_{ij}}$ is a random variable that, we assume, follows the normal distribution $\mathcal{N}(0, \hat{\Sigma}_{ij})$ given by the network.

The linearization of the measurement function h with respect to the error state yields the linearized measurement matrix $\mathbf{H}_{(3, 6m+15)}$. It has only non zero values in the 3×3 blocks corresponding to $\tilde{\boldsymbol{\theta}}_i, \delta\tilde{\mathbf{p}}_i$ and $\delta\tilde{\mathbf{p}}_j$.

$$\mathbf{H}_{\tilde{\boldsymbol{\theta}}_i} = \frac{\partial h(\mathbf{X})}{\partial \tilde{\boldsymbol{\theta}}_i} = \hat{R}_\gamma^T [{}^w\hat{\mathbf{p}}_j - {}^w\hat{\mathbf{p}}_i]_\times \mathbf{H}_z \quad (5.15)$$

$$\mathbf{H}_{\delta\tilde{\mathbf{p}}_i} = \frac{\partial h(\mathbf{X})}{\partial \delta\tilde{\mathbf{p}}_i} = -\hat{R}_\gamma^T \quad (5.16)$$

$$\mathbf{H}_{\delta\tilde{\mathbf{p}}_j} = \frac{\partial h(\mathbf{X})}{\partial \delta\tilde{\mathbf{p}}_j} = \hat{R}_\gamma^T \quad (5.17)$$

$$\text{where } \mathbf{H}_z = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \cos \gamma \tan \beta & \sin \gamma \tan \beta & 1 \end{bmatrix} \quad (5.18)$$

In the above expression, $[\mathbf{x}]_\times$ is the skew-symmetric matrix built from vector \mathbf{x} . The singularity $\cos \beta = 0$ occurs when the person wearing the headset is looking straight down or straight up: we simply discard the update for these cases. Note that these situations are unlikely and never occur in our dataset. The mathematical proof for $\mathbf{H}_{\tilde{\boldsymbol{\theta}}_i}$ can be found in Appendix A.2.

Finally, \mathbf{H} and $\hat{\Sigma}_{ij}$ are used to compute the Kalman gain to update the state \mathbf{X} and the covariance

P as follows:

$$K = PH^T(HPH^T + \hat{\Sigma}_{ij})^{-1} \quad (5.19)$$

$$X \leftarrow X \oplus \left(K(h(X) - \hat{d}_{ij}) \right) \quad (5.20)$$

$$P \leftarrow (I - KH)P(I - KH)^T + K\hat{\Sigma}_{ij}K^T \quad (5.21)$$

Operator \oplus denotes the regular addition operation except for rotation where the update operation writes $R \leftarrow \text{Exp}(\tilde{\theta})R$.

We use a χ_2 test to protect the filter estimate against occasional wrong network output: we discard the update when the normalized innovation error $\|d_{ij} - \hat{d}_{ij}\|_{HPH^T + \hat{\Sigma}_{ij}}$ is greater than a threshold. We choose here the threshold value of 11.345 which corresponds to the 99th percentile of the χ_2 distribution with 3 degrees of freedom.

In practice, when using the measurement covariance $\hat{\Sigma}_{ij}$ from the network, we scale the covariance by a factor of 10 to compensate for the temporal correlation of measurements as noted in Sec. 5.7.1.

5.5.5. State size, Marginalization and Initialization

As soon as an update is processed, all states prior to this update window are marginalized. This is performed by removing the corresponding ξ from the state and the corresponding covariance entries. The number of states kept in the filter depends on two parameters: (i) the displacement duration window for which the network is trained and (ii) the update frequency. For instance, for an update frequency of 20 Hz and a window of 1 s, there will be at most 21 past states in the filter. The two parameters can be set independently: if the period of update is smaller than the displacement window, the consecutive measurements will be generated from the overlapping windows of IMU data, while if the update period is too long, some IMU measurements would not be given to the network. We observe that our system has better performance with a higher update frequency (see Sec. 5.7.2), and we use 20 Hz in our final system.

Our EKF needs a good initialization to converge. In this work we assumed the initial state is given

by an external procedure. In our experiments we initialize the speed, roll, and pitch with the ground-truth estimate assigning a large covariance, while the yaw and the position are initialized with a strong prior in order to fix the gauge. Biases are initialized as with the values of an initial factory calibration, and we refine the biases online to account for its evolution due to turn-on change, temperature effect, or aging.

In practice, we use the following values to initialize the state covariance: $\sigma_{w_v} = 0.1$ m/s, $\sigma_{b_a} = 0.2$ m/s², $\sigma_{b_g} = 1.10^{-4}$ rad/s, $\sigma_\theta = \text{diag}(10, 10, 0.1)$ deg. We also compensate all the input IMU samples for non-orthogonality scale factor and g-sensitivity from the factory calibration.

5.6. Experimental Results

We evaluate our system on the test split of our dataset. Each of these 37 trajectories contains between 3 to 7 minutes of human activities. We compare the performance of two pose estimators against a state-of-the-art VIO implementation based on Mourikis and Roumeliotis (2007) and considered as ground-truth (GT):

- TLIO: the tightly-coupled EKF with displacement update from the trained network presented in this work.
- 3D-ROBIN: an estimator where the displacements from the same trained network are concatenated in the direction given by an engineered AHRS attitude filter, resembling what smartphones have. We borrow the name from Yan et al. (2020), but we extended the method to 3D and we trained the network on our dataset to get a fair comparison. Note that our dataset contains trajectories involving climbing staircases, sitting motions, and walking outdoors on uneven terrain, where the 2D PDR method of Yan et al. (2020) is not directly applicable.

5.6.1. Metrics definitions

In order to assess our system performance, for each dataset of length n , we define the following metrics:

- ATE (m): $\sqrt{\frac{1}{n} \sum_i^n \| {}^w\mathbf{p}_i - {}^w\hat{\mathbf{p}}_i \|^2}$

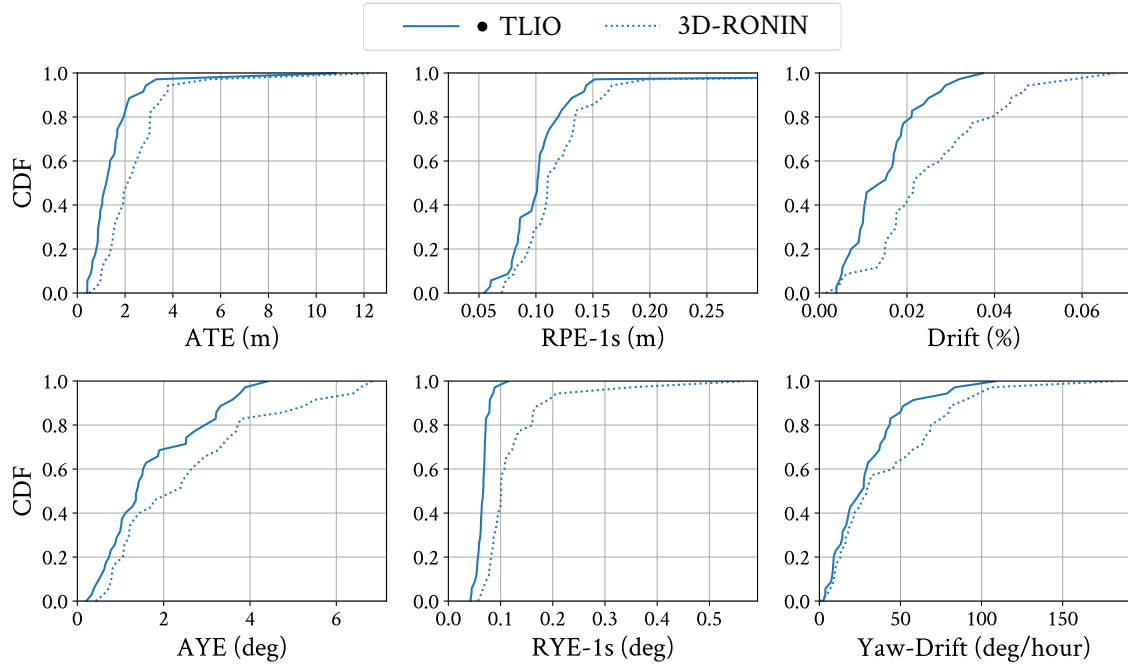


Figure 5.3: Comparing TLIO to 3D-RONIN on the error metrics with respect to the ground-truth. Each plot shows the cumulative density function of the chosen metric on the entire test set. The steeper the plots are the better the performance. TLIO shows significantly lower value than 3D-RONIN for all presented metrics.

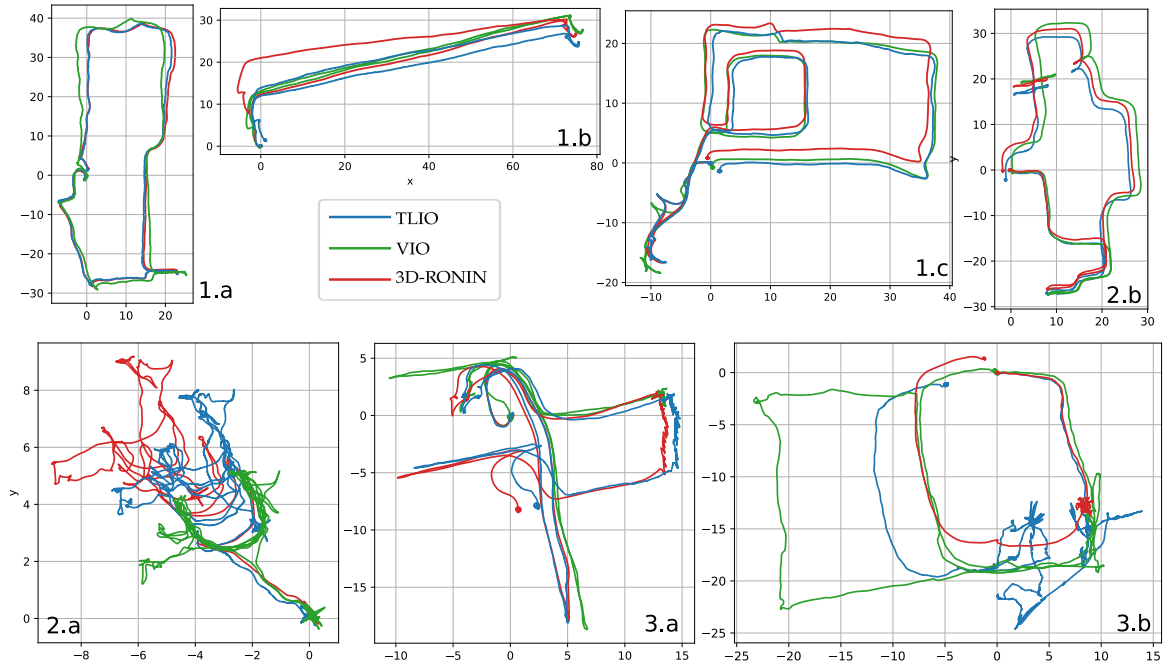


Figure 5.4: Selection of trajectories. Each graph has x and y axes with unit in meters. 1.x and 2.x are the good and hard cases and 3.x show examples of failures. 2.a. shows a case with very wrong network outputs. 2.b shows a case of bad initialization in the filter leading to yaw drift. In 3.a and 3.b, unusual motions not present in the training set are performed: side-stepping repeatedly for 3 minutes and rolling on a chair.

The Absolute Translation Error indicates the spatial closeness of position estimate to the GT over the whole trajectory, computed as root-mean square error (RMSE) between these sets of values.

- **RTE- Δt (m):**

$$\sqrt{\frac{1}{n} \sum_i^n \| {}^w\mathbf{p}_{i+\Delta t} - {}^w\mathbf{p}_i - \mathbf{R}_\gamma \hat{\mathbf{R}}_\gamma^T ({}^w\hat{\mathbf{p}}_{i+\Delta t} - {}^w\hat{\mathbf{p}}_i) \|^2}$$

The Relative Translation Error indicates the local spatial closeness of position estimate to the GT over a window of duration Δt . We use 1 s in this analysis. We remove the yaw drift at the beginning of the window so that the relative measure is not affected by accumulated errors.

- **DR (%) :** $(\| {}^w\mathbf{p}_n - {}^w\hat{\mathbf{p}}_n \|) / (\text{trajectory-length})$

The final translation drift over the distance traveled.

We compute similar metrics for yaw angle γ , measuring the quality of the unobservable orientation around gravity:

- **AYE (°):** $\sqrt{(\frac{1}{n} \sum_i \|\gamma_i - \hat{\gamma}_i\|^2)}$ is the Absolute Yaw Error.
- **RYE- Δt (°):** $\sqrt{\frac{1}{n} \sum_i^n \|\gamma_{i+\Delta t} - \gamma_i - (\hat{\gamma}_{i+\Delta t} - \hat{\gamma}_i)\|^2}$ is the Relative Yaw Error
- **Yaw-DR (°/hour):** $(\gamma_n - \hat{\gamma}_n) / (\text{sequence-duration})$ is the Yaw Drift over time.

5.6.2. System Performance

Figure 5.3 shows the distribution of the metrics across the entire test set. It demonstrates that our system consistently performs better than the decoupled orientation and position estimator 3D-RONIN on all metrics.

On 3D position, TLIO performs better than 3D-RONIN which integrates average velocities. Integration approach has the advantage of being robust to outliers since the measurements at each instant are decoupled. The result shows not only the benefit of integrating IMU kinematic model, but also the overall robustness of the filter, which comes from the quality of the covariance output from the network and the effectiveness of outlier rejection with χ_2 test.

Our system also has a smaller yaw drift than 3D-RONIN, indicating that even without any hand engineered heuristics or step detection, using displacement estimates from a trained statistical model outperforms a smartphone AHRS attitude filter. This shows that the EKF can accurately estimate not only position and velocity, but also gyroscope biases.

Figure 5.4 shows a hand-picked collection of 7 trajectories containing the best, typical and worst cases of TLIO and 3D-RONIN, while Figure 5.1 shows a 3D visualization of one additional sequence. See the corresponding captions for more details about the trajectories and failure cases.

5.7. Components and Variation Studies

We investigate different variations of the network settings in three aspects: input IMU frequency, time interval Δt_{ij} over which displacement is computed, and the total time interval included in the network input. We do not consider future data because we aim for a causal filter. The name convention we use is built from these aspects. For example, 200hz-05s-3s is a model that takes 3 s of 200 Hz input data and regresses for the displacement over the last 0.5 s. To easily keep track of the different models, in all following figures, we prefix with “●” the version of our algorithm whose results were presented in Sec. 5.6: 200hz-1s-1s.

5.7.1. Network Statistical Model

In this section, we evaluate the deep learning component of our system in isolation.

Does using more data help?

Figure 5.5 shows a comparison of several variations of the window duration, keeping the IMU frequency fixed to 200 Hz. It is surprising that lower MSE loss over the same displacement window does not imply lower ATE. This is because MSE does not say anything about the sum of the errors: the network can make more accurate predictions per sample by seeing more data, while at the same time produces more correlated results as the inputs overlap over a longer period of time. This introduces a trade-off when integrating displacement, and we observe similar trajectory ATEs in different network time window variants. For further experiments we will use 200hz-1s-1s except noted otherwise.

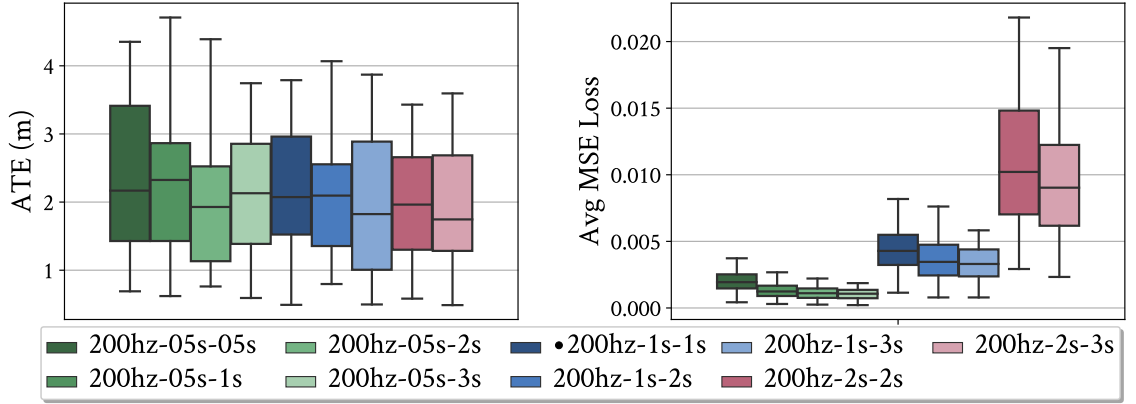


Figure 5.5: Network variations with different past and total window sizes evaluated on the test set. Upper left: ATE of 3D-RONIN concatenation result comparing to GT trajectories. Upper right: average MSE loss on the test set. Adding past data to the network input reduces average MSE loss over samples, but it does not imply lower ATE over trajectories.

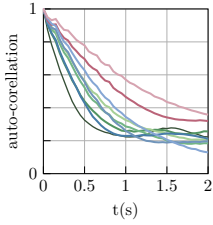


Figure 5.6: Left: Auto-correlation function of the norm of the network displacement error over one sequence. The norms are computed at 20 Hz using overlapping windows of IMU data. Models with more input data have lower MSE but greater temporal correlation of errors.

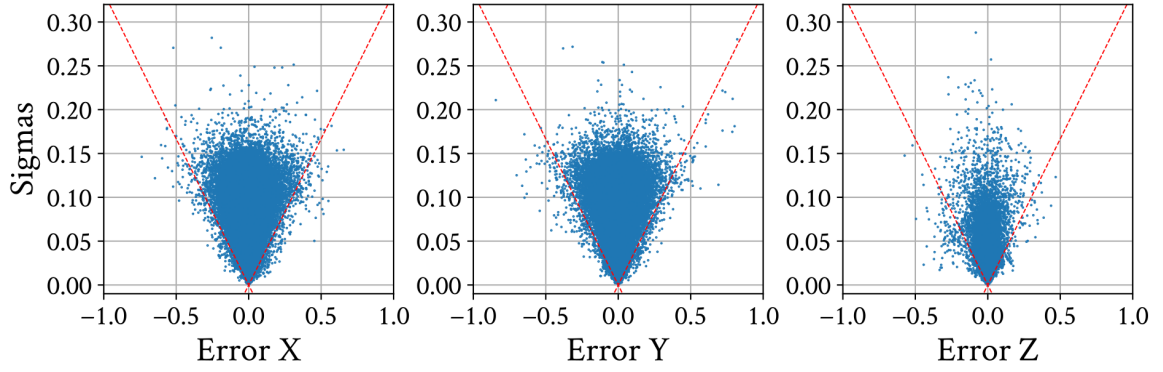


Figure 5.7: Uncertainty returned by the network (standard-deviation σ) plotted against errors (m) on three axes x, y, z respectively. Points with larger errors have larger variance outputs, and over 99% of the points fall inside the 3σ cone region indicated by the dashed red line. We have 0.70% for x, y and 0.47% for z axis of error points outside 3σ bounds respectively.

Consistency of learned covariance

We collected around 60 000 samples at 5 Hz on the entire test set for this analysis and the sensitivity analysis section below. Fig. 5.7 plots the sigma outputs of the network against the errors on the displacement. We observe over 99% of the error points fall inside the 3σ region, showing the network outputs are consistent statistically. In addition, we compute the Mahalanobis distance of the output displacement errors scaled by the output covariances in 3D. We found that only 0.30% of the samples were beyond the 99 percentile critical value of the χ^2 distribution, all of which are from the failure case shown in Fig. 5.4.3.b). From this analysis, we conclude that the uncertainty output of the network grows as expected with its own error albeit being often slightly conservative. We observe an asymmetry between horizontal and vertical errors distribution; along vertical axis errors and sigmas concentrate on lower values. This is not surprising: people usually either walk on a flat ground or on stairs which may make expressing a prior easier on the z axis.

Sensitivity Analysis

Figure 5.8 shows the network output robustness to input perturbation on bias and gravity direction. We observe that the models trained with bias perturbation are more robust to IMU bias, in particular accelerometer bias. The model trained with gravity perturbation is significantly more robust to gravity direction errors. These network training techniques effectively help reduce the propagation of errors from input to the output, protecting the independence assumption required in a Kalman

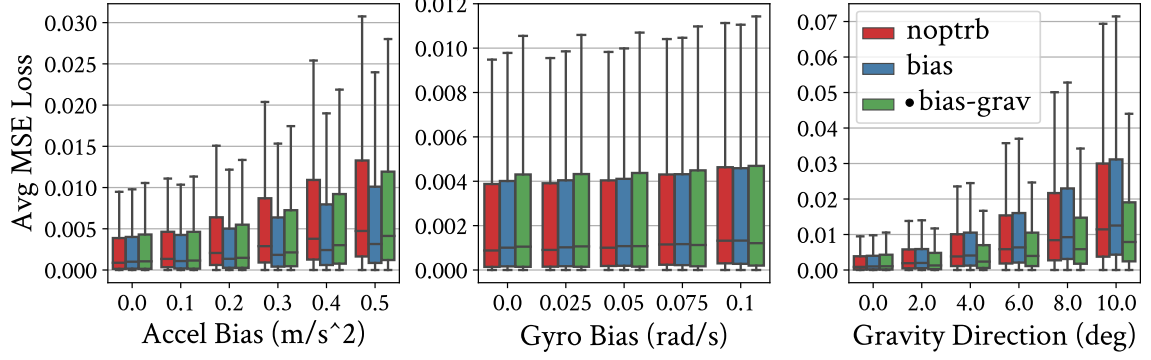


Figure 5.8: Sensitivity of the network prediction on bias and gravity direction noise. At test time the input IMU samples are perturbed for accelerometer bias, gyroscope bias, and gravity direction as described in Sec. 5.4.2 but with various ranges. The three models under comparison are trained with different data augmentation procedures: no perturbation (noptrb), perturbing only the bias (bias), and perturbing both the bias and the gravity direction (bias-grav). We observe that training with perturbation significantly improves robustness on accelerometer bias and gravity direction.

Filter framework. Therefore we choose 200hz-1s-1s trained with bias and gravity perturbation as our system model.

5.7.2. EKF System

Ablation Study

We compare system variants using a hand-tuned constant covariance and networks trained without likelihood loss to show the benefit of using the regressed uncertainty for TLIO. To find the best constant covariance parameters, we use the standard deviation of the measurement errors on the test set - $\text{diag}(0.051, 0.051, 0.013)\text{m}$ - multiplied by a factor yielding the best median ATE tuned by grid-search. Fig. 5.9 shows the Cumulative Density Function of ATE, Drift and AYE over the test set of various system configurations. We observe that using 3D-RONIN, training the network with only MSE loss gives a more accurate estimate. However, using a fixed covariance, such network variants in a filter system achieve similar performance. What makes a difference is the regressed covariance from the network, which significantly improves the filter in ATE and Drift. We also notice a dataset where using a fixed covariance loses track due to initialization failure, showing the adaptive property of the regressed uncertainty for different inputs, improving system robustness. Comparing to 3D-RONIN-mse, TLIO has an improvement of 27% and 31% on the average yaw and position drift respectively.

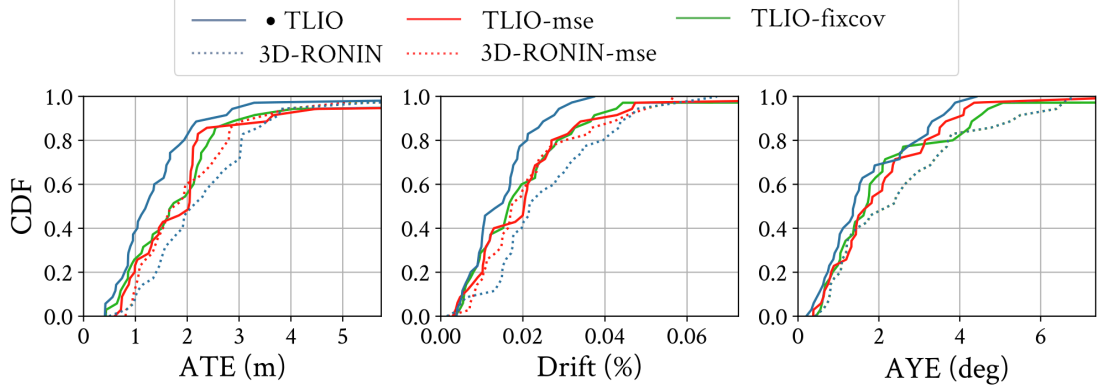


Figure 5.9: Performance comparison showing the effectiveness of using a learned covariance. TLIO-mse and 3D-RONIN-mse use a network trained with only MSE loss. TLIO-fixcov and TLIO-mse use a hand-tuned constant covariance for all the measurements. 3D-RONIN and 3D-RONIN-mse concatenate displacement estimates directly without considering uncertainty. TLIO achieves the best performance with the covariance regressed from the network. Constant covariance approaches do not reach 100% for ATE and drift in this illustration due to a failure case.

Timing parameters

Fig. 5.10 shows the performance comparison between variations on network IMU frequency, Δt_{ij} , and filter measurement-update frequency. ATE and drift are reduced when using high frequency measurements. Once again, we observe minor differences between network parameter variations on the monitored metrics. This shows that our entire pipeline is not sensitive to these choices of parameters, and TLIO outperforms 3D-RONIN in all cases.

5.8. Conclusion and Insights

This chapter presented a tightly-coupled inertial odometry algorithm introducing a learned component in an EKF framework. We present a network regressing 3D displacement and uncertainty, and an EKF framework fusing the displacement measurement to estimate the full state. We train a network that learns a prior on the displacement distributions given IMU data from statistical motion patterns. We show through experimental results and variation studies that the network outputs are statistically consistent. The filter outperforms the state-of-the-art trajectory estimator using velocity integration on position estimates and a model-based AHRS attitude filter on orientation. This demonstrates that an IMU sensor alone can provide enough information for low drift pose estimation

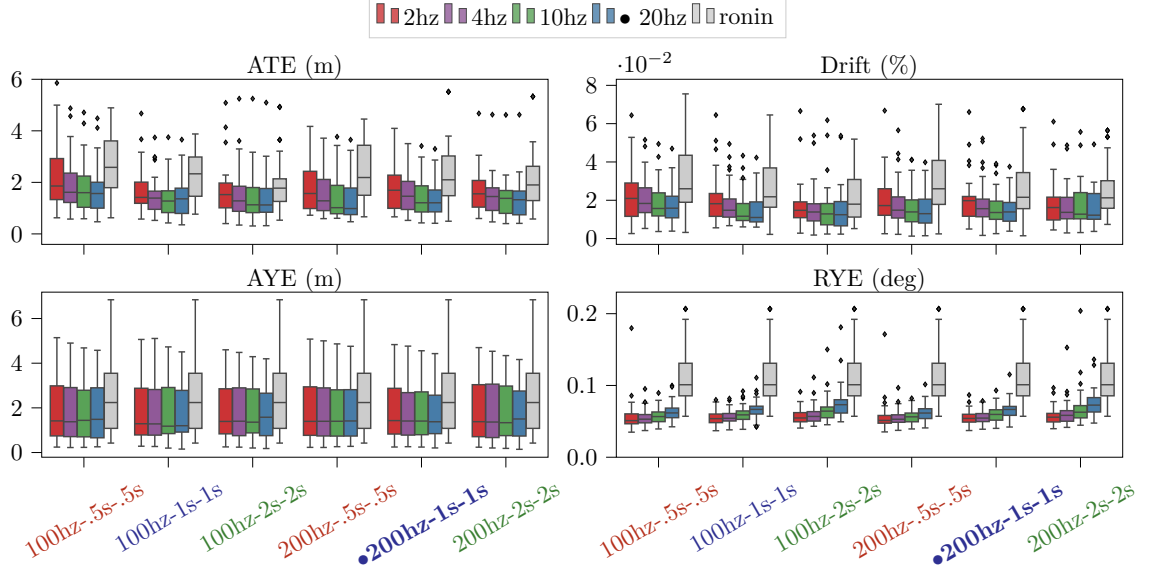


Figure 5.10: Performance of different system configurations. Each group corresponds to a network model, as indicated by the x axis labels. Within each group, the colored boxplots differ by update frequency, and the grey ones show 3D-ronin as baseline. The filtering approach constantly outperforms 3D-ronin on all the experimented models. High frequency update yields the best ATE and drift in spite of temporal correlation of the measurements. RYE-1s increases with update frequency, indicating more jitter on the yaw estimates.

and calibration for pedestrian dead-reckoning with a learned prior.

This result shows us an example of fusing deep learning into a standard EKF state estimation framework. Unlike the 3D-ronin method which only considers instantaneous estimates and concatenates the results, the TLIO system design considers prior information and the state change in a probability distribution. This allows the system to reject outliers from the networks based on predictions with a χ^2 test and produce state estimates with uncertainties, which is essential for safety-critical applications such as self-driving cars and micro aerial vehicles.

As common with learning approaches, the system is limited by the scope of training data. Unusual motions cause system failure, as discussed with examples. In the Kalman filter framework, the key to an optimal and consistent system lies in the accurate measurement uncertainty in the update step. It is of research interest to train a network model that produces correct mean and covariance estimates, and find solutions that address the time correlation to model measurement noise accurately.

CHAPTER 6

CONCLUSION

6.1. Research Summary

This thesis presented systematic research on information-deficient state estimation. The works focus on visual-inertial state estimation systems, addressing three information deficiency cases in three application scenarios. We summarize these three works below.

Semi-dense direct VIO on resource-constrained platforms

This work experiments with a system solution of a tightly-coupled VIO framework in a low texture environment and under computational constraints on an autonomous flying platform. A semi-dense direct approach is chosen for the state estimation system to gain an edge in low texture scenarios presenting limited visual information. Consequently, more computation is needed than sparse approaches. A minimal optimization framework and camera-specific engineering choices all contribute to a small footprint VIO system running in real-time onboard. The algorithm is tested through benchmark comparisons against the state of the arts on both the standard EuRoC Dataset and our Low Texture Dataset. Finally, the application of semi-dense VIO on resource-constrained platforms is validated through the autonomous flight of a 250g quadrotor equipped with a smartphone-grade processor.

Unsupervised learning of egomotion with a hybrid structure for monocular camera

This work introduces a new approach to combining recent deep learning advancements in vision with an optimization framework with RANSAC. Using a monocular camera is the minimum setup for visual odometry, but the metric scale information is not observable. Deep neural networks provide this information through accurate depth and flow estimates in the image plane. The relative pose is then derived through least-squares optimization with RANSAC, applying motion-field equation constraints. This process selectively uses the set of consistent estimates from the network outputs, and the pose estimate, in turn, provides additional supervision to training. This hybrid model can

estimate accurate visual odometry with compact neural network models on the driving datasets.

Tight learned inertial odometry

This work shows an inertial state estimation framework design that significantly reduces drift for pedestrian AR/VR headset applications. Without vision, the method fuses displacement and uncertainty estimates from deep learning into a tightly-coupled extended Kalman filter framework. The network can produce statistically consistent uncertainty outputs specific to the input data, improving the filter performance. The system tracks short-term movements with the IMU motion model in the prediction step and long-term movements through accurate measurement updates from the neural network. The system achieves an overall 1.3% median translational drift and better yaw drift than the AHRS filter, indicating that the system has better calibration estimates.

6.2. Contribution and Insights

The projects study the state estimation problem with three critical information deficiency cases in visual-inertial navigation and contribute to a systematic study of the topic in the following ways:

1. The proposed VIO system for low texture environment on resource-constrained platforms investigates the application boundaries and algorithmic limits when little visual feature information is present. The research presents a feasible real-time semi-dense system solution on a smartphone-grade processor for autonomous flight. Result analysis also points out the computation bottlenecks that need to be addressed to improve system accuracy under such constraints further.
2. The proposed hybrid VO framework addresses the problem of scale ambiguity in monocular systems with deep learning, but with a twist of model-based optimization with RANSAC to enforce motion constraints. This combination effectively improves system performance and enables high accuracy pose estimation despite having smaller models and erroneous network outputs. This work proves the effectiveness of using unsupervised learning in a hybrid model for monocular visual odometry with a robust system design.

3. The tightly-coupled IO system is a successful attempt at finding a solution for single IMU navigation problems for pedestrian applications. The power of deep neural networks in learning pedestrian motion patterns is backed by the experimental results of low drift in both translation and yaw. Full state estimation without any visual information, which has always been considered difficult due to the inherent drawback of amplified error accumulation in kinematic IMU motion models from integration, is shown feasible for the first time with tracking accuracy comparable to a visual-inertial state estimation system.

We gained some high-level insights for working with information-deficient state estimation systems by researching the three problems.

The first work looks into a low texture environment as one type of visual information deficiency. It focuses on the system engineering aspects of the challenges of a semi-dense VIO optimization framework under application constraints. The adopted minimum keyframe-based optimization framework analysis reveals two areas for improvement corresponding to the bottlenecks in computation: optimization for tracking and keyframe image processing. Further improving outlier rejection and robust estimator heuristics would contribute to a clean gradual convergence down the image pyramid for optimization. The image processing cost can be alleviated by computing gradient and depth by individual points instead of the whole image or mitigating specific image processing tasks from the CPU. For example, one can use FPGAs for feature-related operations or GPUs for dense depth estimation with neural network solutions. After these bottlenecks in the computation are alleviated, a faster frame rate or windowed optimization can be considered for further accuracy and robustness for the direct semi-dense approach. As of the current direct implementation using standard open-source libraries, the state estimation consumes 1.5 cores on the smartphone-grade processor running at a maximum of 15Hz of image frame rate. Due to its processing time and CPU usage, the small quadrotor platform cannot afford bundle adjustment with multiple keyframes for fully autonomous flight.

The hybrid egomotion estimation framework looks into the monocular visual tracking problem, a major research problem for the simplicity of usage and widespread applications. However, the prob-

lem is inherently ill-posed due to scale ambiguities. Using neural networks trained unsupervised on stereo data allows for recovering depth information from monocular sequences, compensating for the lack of scale information. It also leverages state-of-the-art data-driven methods for driving datasets, providing dense and accurate depth and flow estimation in similar application scenarios. However, the learned depth and flow images cannot do magic - they are most accurate on recognizable structures and objects but have high variance on vegetation and textureless areas. The model-based optimization with RANSAC successfully compensates for network errors and low accuracy areas. It maintains relative pose estimation accuracy, which allows for more flexibility on the network size and performance and provides a performance guarantee based on geometric constraints. Comparison with other state-of-the-art model-based approaches (ORB-SLAM) shows the effectiveness of using deep learning to provide scale information, and the comparison with other state-of-the-art network approaches highlights the benefits of a hybrid model. To further improve this approach to compete with methods such as DVSO that has a complete VO pipeline, the instantaneous relative pose estimation needs to be extended to include optimization over multiple frames instead of only the adjacent two. Since there is no tracking of features across images but only a flow estimate for the current frame, correspondence information is not kept in this framework, which needs to be addressed in the extension.

TLIO considers an extreme case of incomplete information: when vision information is not available. This situation can occur in dark rooms, for power saving strategies, or out of privacy considerations. The successful use of neural networks on IMU data indicates that pedestrian movement patterns can be recognized. Its displacement over a short (0.5s-2s) time window can be accurately estimated through learning. Using a learned deep neural network is a much more accurate modeling approach for translation than the kinematic motion model using integration, which is prone to drift due to sensor noise and bias. This advantage is the key contributing factor to TLIO performance. Another key contributor to the filter performance is the network estimated adaptive uncertainty, which has been shown to increase robustness in filter convergence and reduce drift. The network estimated uncertainty trained from the NLL loss from Gaussian distribution is statistically consistent to serve as measurement noise in an extended Kalman filter, a standard framework for state estimation. Using

neural network outputs with uncertainty estimates in a probabilistic state estimation framework is a compelling upgrade from state-of-the-art. One limitation of this approach is the time correlation between adjacent displacement estimates. When the displacement time windows overlap to increase filter update frequency, the network input overlaps, which causes time correlation in the output, breaking the filter assumption of uncorrelated measurement noise. We correspondingly up-scaled the network estimated uncertainty to compensate for this factor; however, there is not yet a principled translation from time correlation to a statistical increase in uncertainty for filter performance. Another critical factor to consider when using this approach in practice is the sensitivity of timestamp alignment in the training data. Since the filter depends on network estimated displacements over time windows, a large percentage of work goes into obtaining accurately aligned ground truth displacement for training, which plays a crucial part in the system performance.

The three works address information deficiencies in state estimation from the following aspects: the choice of algorithms; unsupervised learning on data where the missing information is enforced through training; and supervised learning with uncertainty estimate to incorporate into a probabilistic state estimation framework. We have summarized the effectiveness and limitations of each approach in addressing each specific problem. We have evidence to believe that machine learning is an effective tool in addressing information deficiencies in state estimation systems and beyond, where the learning module compensates for the lack of information with a learned prior from data not seen at processing time. We also have the confidence to believe from experimental system comparisons that utilizing a data-driven approach as a module within a model-based framework, instead of replacing such existing frameworks, provides the most accurate and robust solution to state estimation. The learning-optimization/learning-filter approaches can not only incorporate prior information from trained models but also provide tools and statistical gating mechanisms based on principled geometric constraints and kinematics to control errors and high variance outputs from the learning module. The fusion between the data-driven and model-based methods involves identifying what each part contributes and how they influence each other. Developing practical tools and frameworks that best utilize the advantages of both aspects and enable cross supervision for performance guarantees will increase overall system performance.

One crucial factor that influences the inner working of the hybrid frameworks is uncertainty estimation from the learning module. In this thesis, we introduced an application of the predictive method in filtering. At the same time, it is worth noting that various other uncertainty estimation approaches use neural networks that can include both aleatoric and epistemic uncertainty. Accurate uncertainty modeling can significantly benefit the state estimation system in scenarios outside the manifold in which the learned prior resides. However, the research field of neural network uncertainty estimation that marries Bayesian principles is not yet mature. Based on the current benchmarks in the literature, Bayesian approaches have not yet surpassed the non-Bayesian approaches, such as predictive and ensembling, in estimation accuracy. There is also a lack of standard benchmark comparison across the full scope of different approaches. Because there is a lack of explanation of the relationship between uncertainties in the non-Bayesian approaches, using these approaches in practice, albeit having a better performance in benchmarks, may not reveal the true benefit of having a principled uncertainty estimation module in state estimation systems immediately. Eventually, we want the network model to have an accurate data uncertainty estimation and the capability to argue about out-of-distribution data. The Bayesian approach provides one way to associate predicted uncertainty with data distribution.

6.3. Future Directions

Future research can explore two major directions to further solidify the research field of state estimation with information deficiency. One direction is to investigate further down the line within the scope of visual-inertial state estimation to refine and develop algorithms and tools to improve the state-of-the-art. Another direction is to expand to a broader range of sensors and application scenarios to calibrate the current approaches and enrich the methodologies of this research field.

6.3.1. Information deficiency in visual-inertial state estimation

There are various information components in a visual-inertial state estimation system that are of different importance and can become unavailable at different stages of computation. This thesis discussed the missing sensory input of inertial or visual data as an extreme case of critical information deficiency. A low texture environment is another type of critical information deficiency since the

existence of visual features is one of the basic assumptions for visual tracking. Assuming gradient information is not entirely unavailable, we adopt a denser algorithm to collect as much gradient information as possible, which transformed into a system engineering problem considering the application resource constraints.

Many other types of information deficiency have not been discussed. For example, in image processing, there is the lack of knowledge of moving objects; the lack of knowledge of depth discontinuities which causes additional/disappearing visual features at the edge from different viewpoints; the lack of knowledge of reflections, to name a few. These can all cause errors in visual tracking.

Some information is not available immediately but can be derived. For example, depth discontinuities can be inferred from the depth map, which needs to be estimated from the original image; the moving objects can be inferred after consistency checks from the flow and depth images. If known earlier in the pipeline, this information can help the system make more informed decisions about feature selection and tracking. Engineering techniques such as parallel computing and workflow design can help, or one can use learning techniques to skip intermediate steps.

Some information cannot be recovered through subsequent processing, such as reflections. For example, if the reflection surface is perfectly smooth, there is no way to tell whether the textures observed are in the reflection or at a distance until colliding 3D structures are detected. The recovery of this information is more complex and may involve more preliminary information to be collected before this information can be inferred. In these cases, deep learning may be the most convenient and accurate way of obtaining such information.

Identifying, obtaining, and utilizing unavailable information in the state estimation system is a promising research direction.

6.3.2. Extensions to other sensors and applications

The definition of information deficiency does not have to be restricted to visual-inertial state estimation systems, and each application scenario has its specific constraints and characters. This thesis looks at three application scenarios: resource-constrained autonomous flight platforms, driv-

ing with a frontward-facing camera, and AR/VR headsets for pedestrians. Some of the approaches we have developed can be directly migrated to other scenarios, with additional considerations on the application-specific details. For example, semi-dense visual-inertial odometry would encounter similar computational constraints on all other platforms equipped with a smartphone-grade processor. Instead of planning and control for aerial robots, some other processes may compete with state estimation for computation resources. In some other cases, the application scenario provides additional information and unique constraints that need to be addressed. An example of an application-specific system design would be using TLIO for quadrotors for flying in fog or smoke.

The high performance of TLIO on pedestrian headsets comes down to the existence and interpretability of human movement patterns from segments of IMU data. Unlike pedestrians, quadrotor motion is much less confined and often contains degeneracies: the robot freely moves in 3D space based on the user-defined goal trajectory, and moving at a constant velocity following a straight line is expected. At the same time, however, we also have the advantage of having information on the control inputs, and we can take advantage of the aerodynamic modeling. The available information to us now includes not only IMU data but also the motor speeds from the ESCs and power draw from the board. The momentum theory and blade element momentum theory connect rotor speeds with quadrotor dynamics and velocity, providing an alternative information source for state propagation. The state will not only include the poses, velocity, and IMU biases but also parameters in the quadrotor dynamic description. IMU data still provides learned priors with uncertainty and kinematic information, and these can be combined into measurement updates in the filter.

We have done some preliminary tests on the performance of the learned displacement. The preliminary result shows that the learning approach is still effective on IMU data from the quadrotor platform, despite the differences from the pedestrian setting: the motion is coupled with quadrotor tilt, and short-term patterns may not be present when flying a smooth trajectory. However, the training data includes only limited trajectories, and collecting a dataset covering the full range of motions with a quadrotor is challenging. Utilizing simulations and experimental design that shows the effectiveness of aerodynamic constraints are promising directions to explore. In addition to extending

TLIO to quadrotor navigation scenarios, underwater navigation is also a closely relevant application.

Information deficiency can take various forms with different sensors and different application scenarios. In this thesis, we primarily focus on visual-inertial state estimation systems. Investigating more error sources in visual-inertial state estimation systems and extensions to other sensors and applications all present ample opportunities to gradually improve and complete the theoretical and technological aspects of this research topic.

APPENDIX

DETAILS IN THE TLIO EXTENDED KALMAN FILTER

A.1. State Propagation

The state propagation Eq. 5.9 is the following:

$$\begin{aligned}
 \tilde{s}_{k+1} &= A_k^s \tilde{s}_k + B_k^s n_{IMU} + C_k^s \eta = \begin{bmatrix} \tilde{\theta}_{k+1} \\ \delta \tilde{v}_{k+1} \\ \delta \tilde{p}_{k+1} \\ \delta \tilde{b}_{g(k+1)} \\ \delta \tilde{b}_{a(k+1)} \end{bmatrix} \\
 &= \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & -\hat{R}_{k+1} J_r((\omega_k - \hat{b}_{gk})\Delta t)\Delta t & \mathbf{0} \\ -[\hat{R}_k(a_k - \hat{b}_{ak})]_{\times} \Delta t & \mathbf{I} & \mathbf{0} & \mathbf{0} & -\hat{R}_k \Delta t \\ -\frac{1}{2}[\hat{R}_k(a_k - \hat{b}_{ak})]_{\times} \Delta t^2 & \mathbf{I} \Delta t & \mathbf{I} & \mathbf{0} & -\frac{1}{2} \hat{R}_k \Delta t^2 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \tilde{\theta}_k \\ \delta \tilde{v}_k \\ \delta \tilde{p}_k \\ \delta \tilde{b}_{gk} \\ \delta \tilde{b}_{ak} \end{bmatrix} \\
 &+ \begin{bmatrix} \hat{R}_{k+1} J_r((\omega_k - \hat{b}_{gk})\Delta t)\Delta t & \mathbf{0} \\ \mathbf{0} & \hat{R}_k \Delta t \\ \mathbf{0} & \frac{1}{2} \hat{R}_k \Delta t^2 \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} n_{\omega k} \\ n_{ak} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \eta_{gdk} \\ \eta_{adk} \end{bmatrix}
 \end{aligned}$$

J_r is the right Jacobian of $SO(3)$.

A.2. Measurement Update

The proof of $H_{\tilde{\theta}_i}$ in Eq. 5.14 is as follows:

Proof. According to chain rule,

$$H_{\tilde{\theta}_i} = \frac{\partial h(X)}{\partial \theta_i} = \frac{\partial h(X)}{\partial \theta_{iz}} \cdot \frac{\partial \theta_{iz}}{\partial r} \cdot \frac{\partial r}{\partial \theta_i} \quad (\text{A.1})$$

where θ_{iz} is the vector form of $\mathfrak{so}(3)$ of the rotation matrix R_{iz} , defined as $\text{Log}(R_{iz})$, and $\theta_i = \text{Log}(R_i)$. r is defined as $(\alpha, \beta, \gamma)^T$ where α , β and γ corresponds to yaw, pitch and roll angles of the R_i respectively:

$$R_i(\alpha, \beta, \gamma) = R_{\alpha \mathbf{i}} R_{\beta \mathbf{j}} R_{\gamma \mathbf{k}} = \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \gamma & -\sin \gamma \\ 0 & \sin \gamma & \cos \gamma \end{bmatrix},$$

where $\mathbf{i} = (0, 0, 1)^T$, $\mathbf{j} = (0, 1, 0)^T$, $\mathbf{k} = (1, 0, 0)^T$, and $R_{\alpha \mathbf{i}} = R_{iz}$

According to Trawny and Roumeliotis (2005), we have

$$H = \frac{\partial \theta_i}{\partial r} = \begin{bmatrix} \mathbf{i} & R_{\alpha \mathbf{i}} \mathbf{j} & R_{\alpha \mathbf{i}} R_{\beta \mathbf{j}} \mathbf{k} \end{bmatrix} = \begin{bmatrix} 0 & -\sin \alpha & \cos \alpha \sin \beta \\ 0 & \cos \alpha & \sin \alpha \cos \beta \\ 1 & 0 & -\sin \beta \end{bmatrix}$$

and we have $\det(H) = -\cos \beta$. Singularity occurs when $\beta = \pm \frac{\pi}{2}$ and is known as gimbal lock.

From the inverse function theorem, when $\det(H) \neq 0$, we have

$$\frac{\partial r}{\partial \theta_i} = H^{-1} = \begin{bmatrix} \cos \alpha \tan \beta & \sin \alpha \tan \beta & 1 \\ -\sin \alpha & \cos \alpha & 0 \\ \cos \alpha \sec \beta & \sec \beta \sin \alpha & 0 \end{bmatrix}$$

where H^{-1} has a closed form solution. This is the last term of the chain rule in Eq. A.1.

The second term can be easily obtained as $\theta_{iz} = \text{Log}(R_i z) = (0, 0, \alpha)^T$:

$$\frac{\partial \theta_{iz}}{\partial r} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

To obtain the first term we use the same technique for calculating the state propagation matrix, where we expand $h(X)$ by $\tilde{\theta}_{iz}$:

$$\begin{aligned} \delta h(X) &= (e^{\tilde{\theta}_{iz}} R_{iz})^T (p_j - p_i) \\ &= R_{iz}^T e^{-\tilde{\theta}_{iz}} (p_j - p_i) \\ &= R_{iz}^T (\mathbf{I} - [\tilde{\theta}_{iz}]_{\times}) (p_j - p_i) \\ &= R_{iz}^T (p_j - p_i) - R_{iz}^T [\tilde{\theta}_{iz}]_{\times} (p_j - p_i) \\ &= R_{iz}^T (p_j - p_i) - R_{iz}^T [(p_j - p_i)]_{\times} \tilde{\theta}_{iz} \end{aligned}$$

So we have

$$\frac{\partial h(X)}{\partial \theta_{iz}} = -R_{iz}^T [(p_j - p_i)]_{\times}$$

Therefore,

$$\begin{aligned} H_{\tilde{\theta}_i} &= \frac{\partial h(X)}{\partial \theta_i} = \frac{\partial h(X)}{\partial \theta_{iz}} \cdot \frac{\partial \theta_{iz}}{\partial r} \cdot \frac{\partial r}{\partial \theta_i} \\ &= \hat{R}_{iz}^T [\hat{p}_j - \hat{p}_i]_{\times} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos \alpha \tan \beta & \sin \alpha \tan \beta & 1 \\ -\sin \alpha & \cos \alpha & 0 \\ \cos \alpha \sec \beta & \sec \beta \sin \alpha & 0 \end{bmatrix} \\ &= \hat{R}_{iz}^T [\hat{p}_j - \hat{p}_i]_{\times} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \cos \alpha \tan \beta & \sin \alpha \tan \beta & 1 \end{bmatrix} \end{aligned}$$

□

BIBLIOGRAPHY

- Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.
- Arun, K. S., Huang, T. S., and Blostein, S. D. (1987). Least-squares fitting of two 3-d point sets. *IEEE Transactions on pattern analysis and machine intelligence*, (5):698–700.
- Baker, S. and Matthews, I. (2004). Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255.
- Barber, D. and Bishop, C. M. (1998). Ensemble learning in bayesian neural networks. *Nato ASI Series F Computer and Systems Sciences*, 168:215–238.
- Beaufils, B., Chazal, F., Grelet, M., and Michel, B. (2019). Robust stride detector from ankle-mounted inertial sensors for pedestrian navigation and activity recognition with machine learning approaches. *Sensors*, 19(30):4491.
- Bergmann, P., Wang, R., and Cremers, D. (2017). Online photometric calibration of auto exposure video for realtime visual odometry and slam. *IEEE Robotics and Automation Letters*, 3(2):627–634.
- Blösch, M., Burri, M., Omari, S., Hutter, M., and Siegwart, R. (2017). Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback. *The International Journal of Robotics Research*, 36(10):1053–1072.
- Blösch, M., Omari, S., Hutter, M., and Siegwart, R. (2015). Robust visual inertial odometry using a direct ekf-based approach. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 298–304. IEEE.
- Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*.
- Borenstein, J., Ojeda, L., and Kwanmuang, S. (2009). Heuristic reduction of gyro drift in IMU-based personnel tracking systems. In *Optics and Photonics in Global Homeland Security V and Biometric Technology for Human Identification VI*, volume 7306, page 73061H.
- Bottou, L. (2012). Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer.
- Brachmann, E., Krull, A., Nowozin, S., Shotton, J., Michel, F., Gumhold, S., and Rother, C. (2017). Dsac-differentiable ransac for camera localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6684–6692.
- Bradski, G. and Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library*.

" O'Reilly Media, Inc."

- Brajdic, A. and Harle, R. (2013). Walk detection and step counting on unconstrained smartphones. In *ACM international joint conference on Pervasive and ubiquitous computing*, pages 225–234.
- Brossard, M., Barrau, A., and Bonnabel, S. (2019). AI-IMU dead-reckoning. *IEEE Transactions on Intelligent Vehicles*.
- Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., Achtelik, M. W., and Siegwart, R. (2016). The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163.
- Campos, C., Elvira, R., Rodríguez, J. J. G., Montiel, J. M., and Tardós, J. D. (2021). Orb-slam3: An accurate open-source library for visual, visual-inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890.
- Chen, C., Lu, X., Markham, A., and Trigoni, N. (2018a). IONet: Learning to cure the curse of drift in inertial odometry. In *Thirty-Second AAAI Conference on Artificial Intelligence*, pages 6468–6476.
- Chen, C., Zhu, H., Li, M., and You, S. (2018b). A review of visual-inertial simultaneous localization and mapping from filtering-based and optimization-based perspectives. *Robotics*, 7(3):45.
- Clark, R., Wang, S., Wen, H., Markham, A., and Trigoni, N. (2017). Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem. In *Thirty-First AAAI Conference on Artificial Intelligence*.
- Corke, P., Lobo, J., and Dias, J. (2007). An introduction to inertial and visual sensing.
- Davison, A. J. (2003). Real-time simultaneous localisation and mapping with a single camera. In *Computer Vision, IEEE International Conference on*, volume 3, pages 1403–1403. IEEE Computer Society.
- Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. (2007). Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067.
- Dehzangi, O., Taherisadr, M., and ChagalVala, R. (2017). IMU-based gait recognition using convolutional neural networks and multi-sensor fusion. *Sensors*, 17(12):2735.
- Delmerico, J. and Scaramuzza, D. (2018). A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots. *Memory*, 10:20.
- Dinh, L., Krueger, D., and Bengio, Y. (2014). Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*.
- Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2016). Density estimation using real nvp. *arXiv preprint*

arXiv:1605.08803.

- Eigen, D., Puhrsch, C., and Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374.
- Engel, J., Koltun, V., and Cremers, D. (2018). Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625.
- Engel, J., Schöps, T., and Cremers, D. (2014). Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer.
- Engel, J., Stücker, J., and Cremers, D. (2015). Large-scale direct slam with stereo cameras. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 1935–1942. IEEE.
- Engel, J., Sturm, J., and Cremers, D. (2013). Semi-dense visual odometry for a monocular camera. In *Proceedings of the IEEE international conference on computer vision*, pages 1449–1456.
- Engel, J., Usenko, V., and Cremers, D. (2016). A photometrically calibrated benchmark for monocular visual odometry. *arXiv preprint arXiv:1607.02555*.
- Fischer, P., Dosovitskiy, A., Ilg, E., Häusser, P., Hazırbaş, C., Golkov, V., Van der Smagt, P., Cremers, D., and Brox, T. (2015). FlowNet: Learning optical flow with convolutional networks. *arXiv preprint arXiv:1504.06852*.
- Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.
- Forster, C., Carlone, L., Dellaert, F., and Scaramuzza, D. (2017a). On-manifold preintegration for real-time visual-inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21.
- Forster, C., Zhang, Z., Gassner, M., Werlberger, M., and Scaramuzza, D. (2017b). Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*, 33(2):249–265.
- Foxlin, E. (2005). Pedestrian tracking with shoe-mounted inertial sensors. *IEEE Computer graphics and applications*, 25(6):38–46.
- Fraundorfer, F. and Scaramuzza, D. (2012). Visual odometry: Part ii: Matching, robustness, optimization, and applications. *IEEE Robotics & Automation Magazine*, 19(2):78–90.
- Gal, Y. (2016). *Uncertainty in Deep Learning*. PhD thesis.

- Gal, Y. and Ghahramani, Z. (2015). Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*.
- Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*.
- Garg, R., BG, V. K., Carneiro, G., and Reid, I. (2016). Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, pages 740–756. Springer.
- Gast, J. and Roth, S. (2018). Lightweight probabilistic deep networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3369–3378.
- Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237.
- Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Geneva, P., Eckenhoff, K., Lee, W., Yang, Y., and Huang, G. (2020). Openvins: A research platform for visual-inertial estimation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4666–4672. IEEE.
- Ghosh, S., Delle Fave, F. M., and Yedidia, J. S. (2016). Assumed density filtering methods for learning bayesian neural networks. In *AAAI*, pages 1589–1595.
- Godard, C., Mac Aodha, O., and Brostow, G. J. (2017). Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, volume 2, page 7.
- Goldman, D. B. (2010). Vignette and exposure calibration and compensation. *IEEE transactions on pattern analysis and machine intelligence*, 32(12):2276–2288.
- Graves, A. (2011). Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356.
- Groves, P. D. (2015). Principles of GNSS, inertial, and multisensor integrated navigation systems, [book review]. *IEEE Aerospace and Electronic Systems Magazine*, 30(2):26–27.
- Gui, J., Gu, D., Wang, S., and Hu, H. (2015). A review of visual inertial odometry from filtering and optimisation perspectives. *Advanced Robotics*, 29(20):1289–1301.
- Gustafsson, F. K., Danelljan, M., and Schön, T. B. (2020). Evaluating scalable bayesian deep learning methods for robust computer vision. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1289–1298.

- Haarnoja, T., Ajay, A., Levine, S., and Abbeel, P. (2016). Backprop KF" learning discriminative deterministic state estimators. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4376–4384.
- Harle, R. (2013). A survey of indoor inertial positioning systems for pedestrians. *IEEE Communications Surveys & Tutorials*, 15(3):1281–1293.
- Hartley, R. and Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge university press.
- He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- Hérissé, B., Hamel, T., Mahony, R., and Russotto, F. X. (2012). Landing a VTOL Unmanned Aerial Vehicle on a moving platform using optical flow. *IEEE Transactions on Robotics*, 28(1):77–89.
- Hernandez-Lobato, J., Li, Y., Rowland, M., Bui, T., Hernández-Lobato, D., and Turner, R. (2016). Black-box alpha divergence minimization. In *International Conference on Machine Learning*, pages 1511–1520. PMLR.
- Hernández-Lobato, J. M. and Adams, R. (2015). Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869.
- Hesch, J. A., Kottas, D. G., Bowman, S. L., and Roumeliotis, S. I. (2013). Towards consistent vision-aided inertial navigation. In *Algorithmic Foundations of Robotics X*, pages 559–574. Springer.
- Hinton, G. E. and Van Camp, D. (1993). Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13.
- Huai, Z. and Huang, G. (2018). Robocentric visual-inertial odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6319–6326. IEEE.
- Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. (2018). Neural autoregressive flows. In *International Conference on Machine Learning*, pages 2078–2087. PMLR.
- Huang, G. (2019). Visual-inertial navigation: A concise review. In *2019 international conference on robotics and automation (ICRA)*, pages 9572–9582. IEEE.
- Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J., and Weinberger, K. (2017). Snapshot ensembles: Train 1, get m for free. *ArXiv*, abs/1704.00109.
- Huang, W., Liu, H., and Wan, W. (2020). An online initialization and self-calibration method for stereo visual-inertial odometry. *IEEE Transactions on Robotics*, 36(4):1153–1170.

- Huber, P. J. (2011). Robust statistics. In *International encyclopedia of statistical science*, pages 1248–1251. Springer.
- Ilg, E., Çiçek, Ö., Galesso, S., Klein, A., Makansi, O., Hutter, F., and Brox, T. (2018). Uncertainty estimates and multi-hypotheses networks for optical flow. In *ECCV*.
- Irani, M. and Anandan, P. (1999). About direct methods. In *International Workshop on Vision Algorithms*, pages 267–277. Springer.
- Jaderberg, M., Simonyan, K., Zisserman, A., et al. (2015). Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025.
- Jason, J. Y., Harley, A. W., and Derpanis, K. G. (2016). Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In *European Conference on Computer Vision*, pages 3–10. Springer.
- Jiménez, A. R., Seco, F., Prieto, J. C., and Guevara, J. (2010). Indoor pedestrian navigation using an INS/EKF framework for yaw drift reduction and a foot-mounted IMU. In *7th Workshop on Positioning, Navigation and Communication*. IEEE.
- Ju, H., Park, S. Y., and Park, C. G. (2016). Foot mounted inertial navigation system using estimated velocity during the contact phase. In *IPIN*.
- Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J., and Dellaert, F. (2012). isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2):216–235.
- Kaess, M., Ranganathan, A., and Dellaert, F. (2008). isam: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378.
- Kendall, A. and Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? In *NIPS*.
- Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.
- Kiureghian, A. D. and Ditlevsen, O. D. (2009). Aleatory or epistemic? does it matter?
- Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, pages 225–234. IEEE.
- Klein, G. and Murray, D. (2009). Parallel Tracking and Mapping on a Camera Phone. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 83–86, Orlando, USA.

- Kristiadi, A., Hein, M., and Hennig, P. (2020). Being bayesian, even just a bit, fixes overconfidence in relu networks. *ArXiv*, abs/2002.10118.
- Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *NIPS*.
- Lee, T., Leok, M., and McClamroch, N. H. (2010). Geometric tracking control of a quadrotor uav on se(3). In *49th IEEE Conference on Decision and Control (CDC)*, pages 5420–5425.
- Lentaris, G., Stamoulias, I., Soudris, D., and Lourakis, M. (2015). Hw/sw codesign and fpga acceleration of visual odometry algorithms for rover navigation on mars. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(8):1563–1577.
- Leutenegger, S., Furgale, P., Rabaud, V., Chli, M., Konolige, K., and Siegwart, R. (2013). Keyframe-based visual-inertial slam using nonlinear optimization. *RSS*.
- Li, M. and Mourikis, A. I. (2012). Improving the accuracy of ekf-based visual-inertial odometry. In *2012 IEEE International Conference on Robotics and Automation*, pages 828–835. IEEE.
- Li, R., Wang, S., Long, Z., and Gu, D. (2018). Undeepvo: Monocular visual odometry through unsupervised deep learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7286–7291. IEEE.
- Liu, P., Zuo, X., Larsson, V., and Pollefeys, M. (2021a). Mba-vo: Motion blur aware visual odometry. *arXiv preprint arXiv:2103.13684*.
- Liu, W., Caruso, D., Ilg, E., Dong, J., Mourikis, A. I., Daniilidis, K., Kumar, V., and Engel, J. (2020). Tlio: Tight learned inertial odometry. *IEEE Robotics and Automation Letters*, 5:5653–5660.
- Liu, W., Loiano, G., Mohta, K., Daniilidis, K., and Kumar, V. (2018). Semi-dense visual-inertial odometry and mapping for quadrotors with swap constraints. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–6. IEEE.
- Liu, W., Mohta, K., Loiano, G., Daniilidis, K., and Kumar, V. (2021b). Semi-dense visual-inertial odometry and mapping for computationally constrained platforms. *Autonomous Robots*, 45(6):773–787.
- Longuet-Higgins, H. C. (1981). A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135.
- Loop, C. and Zhang, Z. (1999). Computing rectifying homographies for stereo vision. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 1, pages 125–131. IEEE.
- Loquercio, A., Segù, M., and Scaramuzza, D. (2020). A general framework for uncertainty estima-

- tion in deep learning. *IEEE Robotics and Automation Letters*, 5:3153–3160.
- Loshchilov, I. and Hutter, F. (2017). Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*.
- Luo, C., Yang, Z., Wang, P., Wang, Y., Xu, W., Nevatia, R., and Yuille, A. (2018). Every pixel counts++: Joint learning of geometry and motion with 3d holistic understanding. *arXiv preprint arXiv:1810.06125*.
- Lupton, T. and Sukkariéh, S. (2011). Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *IEEE Transactions on Robotics*, 28(1):61–76.
- Ma, Y., Soatto, S., Kosecka, J., and Sastry, S. S. (2012). *An invitation to 3-d vision: from images to geometric models*, volume 26. Springer Science & Business Media.
- MacKay, D. J. C. (1992). A practical bayesian framework for backpropagation networks. *Neural Computation*, 4:448–472.
- Mandal, D. K., Jandhyala, S., Omer, O. J., Kalsi, G. S., George, B., Neela, G., Rethinagiri, S. K., Subramoney, S., Hacking, L., Radford, J., et al. (2019). Visual inertial odometry at the edge: A hardware-software co-design approach for ultra-low latency and power. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 960–963. IEEE.
- Matthies, L., Kanade, T., and Szeliski, R. (1989). Kalman filter-based algorithms for estimating depth from image sequences. *International Journal of Computer Vision*, 3(3):209–238.
- Matthies, L. H. (1989). *Dynamic stereo vision*. Carnegie Mellon University.
- Meier, L., Tanskanen, P., Fraundorfer, F., and Pollefeys, M. (2011). Pixhawk: A system for autonomous flight using onboard computer vision. In *2011 IEEE International Conference on Robotics and Automation*, pages 2992–2997. IEEE.
- Meister, S., Hur, J., and Roth, S. (2017). Unflow: Unsupervised learning of optical flow with a bidirectional census loss. *arXiv preprint arXiv:1711.07837*.
- Menze, M., Heipke, C., and Geiger, A. (2015). Joint 3d estimation of vehicles and scene flow. In *ISPRS Workshop on Image Sequence Analysis (ISA)*.
- Menze, M., Heipke, C., and Geiger, A. (2018). Object scene flow. *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)*.
- Michael, N., Mellinger, D., Lindsey, Q., and Kumar, V. (2010). The Grasp Multiple Micro-UAV Test Bed. *IEEE Robotics and Automation Magazine*, 17(3):56–65.
- Moravec, H. P. (1980). *Obstacle avoidance and navigation in the real world by a seeing robot rover*. PhD thesis, Stanford University.

- Moreno-Noguer, F., Lepetit, V., and Fua, P. (2007). Accurate non-iterative $\mathcal{O}(n)$ solution to the pnp problem. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE.
- Mourikis, A. I. and Roumeliotis, S. I. (2007). A multi-state constraint kalman filter for vision-aided inertial navigation. In *Robotics and automation, 2007 IEEE international conference on*, pages 3565–3572. IEEE.
- Mueggler, E., Gallego, G., Rebecq, H., and Scaramuzza, D. (2018). Continuous-time visual-inertial odometry for event cameras. *IEEE Transactions on Robotics*, 34(6):1425–1440.
- Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D. (2015a). Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163.
- Mur-Artal, R., Montiel, J. M. M., and Tardós, J. D. (2015b). Orb-slam: A versatile and accurate monocular slam system. *IEEE Trans. Robotics*, 31(5):1147–1163.
- Mur-Artal, R. and Tardós, J. D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262.
- Neal, R. M. (1995). *Bayesian Learning for Neural Networks*. PhD thesis, CAN. AAINN02676.
- Neal, R. M. et al. (2011). Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2.
- Newcombe, R. A., Lovegrove, S. J., and Davison, A. J. (2011). Dtam: Dense tracking and mapping in real-time. In *2011 international conference on computer vision*, pages 2320–2327. IEEE.
- Nistér, D. (2004). An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):756–770.
- Nistér, D., Naroditsky, O., and Bergen, J. (2004). Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I. Ieee.
- Nistér, D., Naroditsky, O., and Bergen, J. (2006). Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3–20.
- Nix, D. and Weigend, A. (1994). Estimating the mean and variance of the target probability distribution. *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, 1:55–60 vol.1.
- Ober, S. and Rasmussen, C. (2019). Benchmarking the neural linear model for regression. *ArXiv*, abs/1912.08416.
- O’Shea, K. and Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint*

arXiv:1511.08458.

- Piao, J.-C. and Kim, S.-D. (2019). Real-time visual–inertial slam based on adaptive keyframe selection for mobile ar applications. *IEEE Transactions on Multimedia*, 21(11):2827–2836.
- Pinsler, R., Gordon, J., Nalisnick, E., and Hernández-Lobato, J. M. (2019). Bayesian batch active learning as sparse subset approximation. In *Advances in Neural Information Processing Systems*, pages 6359–6370.
- Poggi, M., Aleotti, F., Tosi, F., and Mattoccia, S. (2020). On the uncertainty of self-supervised monocular depth estimation. *ArXiv*, abs/2005.06209.
- Qin, T., Li, P., and Shen, S. (2018a). Relocalization, global optimization and map merging for monocular visual-inertial slam. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1197–1204. IEEE.
- Qin, T., Li, P., and Shen, S. (2018b). Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020.
- Qu, C., Liu, W., and Taylor, C. J. (2021). Bayesian deep basis fitting for depth completion with uncertainty. *arXiv preprint arXiv:2103.15254*.
- Rajagopal, S. (2008). Personal dead reckoning system with shoe mounted inertial sensors. *Master’s Degree Project, Stockholm, Sweden*.
- Ramachandran, P., Zoph, B., and Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.
- Ranjan, A., Jampani, V., Kim, K., Sun, D., Wulff, J., and Black, M. J. (2018). Adversarial collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. *arXiv preprint arXiv:1805.09806*.
- Roumeliotis, S. I. and Burdick, J. W. (2002). Stochastic cloning: A generalized framework for processing relative state measurements. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1788–1795.
- Russell, R. L. and Reale, C. (2019). Multivariate uncertainty in deep learning. *arXiv preprint arXiv:1910.14215*.
- Scaramuzza, D. and Fraundorfer, F. (2011). Visual odometry [tutorial]. *IEEE robotics & automation magazine*, 18(4):80–92.
- Scaramuzza, D., Martinelli, A., and Siegwart, R. (2006). A toolbox for easily calibrating omnidirectional cameras. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5695–5701. IEEE.

- Scaramuzza, D. and Siegwart, R. (2008). Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles. *IEEE transactions on robotics*, 24(5):1015–1026.
- Scaramuzza, D. and Zhang, Z. (2019). Visual-inertial odometry of aerial robots. *arXiv preprint arXiv:1906.03289*.
- Schöps, T., Engel, J., and Cremers, D. (2014). Semi-dense visual odometry for ar on a smartphone. In *Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on*, pages 145–150. IEEE.
- Shan, M., Feng, Q., and Atanasov, N. (2020). Orcvio: Object residual constrained visual-inertial odometry. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5104–5111. IEEE.
- Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M. M. A., Prabhat, and Adams, R. (2015). Scalable bayesian optimization using deep neural networks. In *ICML*.
- Sohn, K., Lee, H., and Yan, X. (2015). Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pages 3483–3491.
- Srinivasan, A. and Kaiser, K. M. (1981). Information deficiency: implications for information systems design.
- Sun, D., Yang, X., Liu, M.-Y., and Kautz, J. (2018a). Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8934–8943.
- Sun, K., Mohta, K., Pfrommer, B., Watterson, M., Liu, S., Mulgaonkar, Y., Taylor, C. J., and Kumar, V. (2018b). Robust stereo visual inertial odometry for fast autonomous flight. *IEEE Robotics and Automation Letters*, 3(2):965–972.
- Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media.
- Szeliski, R. and Kang, S. B. (1994). Recovering 3d shape and motion from image streams using nonlinear least squares. *Journal of Visual Communication and Image Representation*, 5(1):10–28.
- Thakur, D., Loianno, G., Liu, W., and Kumar, V. (2018). Nuclear environments inspection with micro aerial vehicles: Algorithms and experiments. In *International Symposium on Experimental Robotics*, pages 191–200. Springer.

- Trawny, N. and Roumeliotis, S. (2005). Jacobian for conversion from euler angles to quaternions. *Dept. Comput. Sci. Eng., Univ. Minnesota, Tech. Rep*, 4.
- Triggs, B., McLauchlan, P. F., Hartley, R. I., and Fitzgibbon, A. W. (1999). Bundle adjustment—a modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer.
- Usenko, V., Demmel, N., Schubert, D., Stücker, J., and Cremers, D. (2019). Visual-inertial mapping with non-linear factor recovery. *IEEE Robotics and Automation Letters*, 5(2):422–429.
- Usenko, V., Engel, J., Stücker, J., and Cremers, D. (2016). Direct visual-inertial odometry with stereo cameras. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1885–1892. IEEE.
- Vidal, A. R., Rebecq, H., Horstschaefer, T., and Scaramuzza, D. (2018). Ultimate slam? combining events, images, and imu for robust visual slam in hdr and high-speed scenarios. *IEEE Robotics and Automation Letters*, 3(2):994–1001.
- Vijayanarasimhan, S., Ricco, S., Schmid, C., Sukthankar, R., and Fragkiadaki, K. (2017). Sfm-net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*.
- Von Stumberg, L., Usenko, V., and Cremers, D. (2018). Direct sparse visual-inertial odometry using dynamic marginalization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2510–2517. IEEE.
- Wang, C., Miguel Buenaposada, J., Zhu, R., and Lucey, S. (2018a). Learning depth from monocular videos using direct methods. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Wang, R., Frahm, J.-M., and Pizer, S. M. (2018b). Recurrent neural network for learning densedepth and ego-motion from video. *arXiv preprint arXiv:1805.06558*.
- Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612.
- Weiss, S., Achtelik, M. W., Lynen, S., Chli, M., and Siegwart, R. (2012). Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments. In *2012 IEEE international conference on robotics and automation*, pages 957–964. IEEE.
- Weiss, S., Scaramuzza, D., and Siegwart, R. (2011). Monocular-SLAM-based navigation for autonomous micro helicopters in GPS denied environments. *Journal of Field Robotics*, 28(6):854–874.
- Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688.

- Williams, B., Klein, G., and Reid, I. (2011). Automatic relocalization and loop closing for real-time monocular slam. *IEEE transactions on pattern analysis and machine intelligence*, 33(9):1699–1712.
- Winkler, C., Worrall, D., Hoogeboom, E., and Welling, M. (2019). Learning likelihoods with conditional normalizing flows. *arXiv preprint arXiv:1912.00042*.
- Wu, K., Ahmed, A., Georgiou, G. A., and Roumeliotis, S. I. (2015). A square root inverse filter for efficient vision-aided inertial navigation on mobile devices. In *Robotics: Science and Systems*, volume 2, page 2. Rome, Italy.
- Xu, W., Choi, D., and Wang, G. (2018). Direct visual-inertial odometry with semi-dense mapping. *Computers & Electrical Engineering*, 67:761 – 775.
- Yan, H., Herath, S., and Furukawa, Y. (2020). RoNIN: Robust neural inertial navigation in the wild: Benchmark, evaluations, and new methods. *Proceedings of the IEEE international conference on robotics and automation (ICRA)*, pages 3146–3152.
- Yan, H., Shan, Q., and Furukawa, Y. (2018). RIDI: Robust IMU double integration. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 621–636.
- Yang, N., Stumberg, L. v., Wang, R., and Cremers, D. (2020). D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1281–1292.
- Yang, N., Wang, R., Stückler, J., and Cremers, D. (2018a). Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *European Conference on Computer Vision*, pages 835–852. Springer.
- Yang, Z., Wang, P., Wang, Y., Xu, W., and Nevatia, R. (2018b). Every pixel counts: Unsupervised geometry learning with holistic 3d motion understanding. *arXiv preprint arXiv:1806.10556*.
- Yin, Z. and Shi, J. (2018). Geonet: Unsupervised learning of dense depth, optical flow and camera pose. *arXiv preprint arXiv:1803.02276*.
- Zhan, H., Garg, R., Weerasekera, C. S., Li, K., Agarwal, H., and Reid, I. (2018). Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. *arXiv preprint arXiv:1803.03893*.
- Zhang, Z., Sattler, T., and Scaramuzza, D. (2020). Reference pose generation for visual localization via learned features and view synthesis. *arXiv preprint arXiv:2005.05179*.
- Zhong, Y., Dai, Y., and Li, H. (2017). Self-supervised learning for stereo matching with self-improving ability. *arXiv preprint arXiv:1709.00930*.

- Zhou, T., Brown, M., Snavely, N., and Lowe, D. G. (2017). Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1851–1858.
- Zhou, W. and Precioso, F. (2019). Adaptive bayesian linear regression for automated machine learning. *arXiv preprint arXiv:1904.00577*.
- Zhu, A. Z., Liu, W., Wang, Z., Kumar, V., and Daniilidis, K. (2018). Robustness meets deep learning: An end-to-end hybrid pipeline for unsupervised learning of egomotion. *arXiv preprint arXiv:1812.08351*.