State Estimation and Vision-based Occupancy Mapping for Off-Road Driving

Wei-Hsin Chou

CMU-RI-TR-17-58

August 2017

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Masters of Science in Robotics.

For my parents.

Abstract

Autonomous navigation in off-road driving scenarios requires accurate and reliable vehicle localization. At the same time, planning for vehicles traversing rough terrain and cluttered areas necessitates the need of efficient and scalable mapping on the surrounding environment. This thesis explores methods of efficient localization and mapping for a self-driving all-terrain vehicle in off-road environment. We present a generalized extended Kalman filtering approach for estimating vehicle state globally and locally with minimal sensor fusion. We then investigate the capability of a state-of-the-art visual SLAM method using single stereo camera in off-road driving cases. Finally, we propose a 3D occupancy mapping framework using stereo vision and integration of visual SLAM. We evaluate our approaches with real-time driving tests and experiments on the publicly available dataset as well as our own off-road dataset from the test field.

Acknowledgements

This thesis would not have been possible without the help and support of many people. Most of all I would like to thank my advisor, Dr. George Kantor for providing me valuable advice, insight and direction throughout my studies at Carnegie Mellon. I am truly thankful for your patience and support along the way and I enjoy working with you.

I also thank my thesis committee members Dr. Michael Kaess and Ming Hsiao for all the help and useful feedback on my research.

I am grateful for my classmates and colleagues. I enjoyed collaborating with Masa Uenoyama, Jay Hiramatsu, Srinivasan Vijayarangan and George Skoptsov on the Yamaha Viking project. I would like to express my sincere gratitude to my classmates and friends Guan-Horng Liu, Po-Wei Chou and Ming Hsiao. I appreciate every valuable discussion and thank you all for your patience and tireless guidance. I would not have been able to make it through for the last two years in CMU without the help from each one of you. It is a privilege to meet you guys and I have learned so much from every one of you. I also want to give special thanks to Wenhua Lin for the encouragement during the tough times. I enjoyed and treasured all the fun conversation and precious time we have together.

Finally, I would like to thank my parents and my sister for their ceaseless love and support, I would not have accomplished any of these without your love.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Recently, great advancement have been made in the development of self-driving technology. Leaders in the industry (Fig. 1.1) are pushing forward to put autonomous driving into our life. Self-driving cars will become a central part of our daily movement. They can improve life quality by preventing accidents, reducing commuting time and bringing unprecedented utility in all kinds of applications.

To drive and navigate by itself, an autonomous vehicle must always be aware of its position and orientation with respect to its surrounding to make the next decision. The essential problem is known as localization. With no prior knowledge of the environment, this seems to be a chicken-and-egg problem. Without a map, one cannot know where one is; without knowing one's location, one cannot build a map. The paradox can be resolved by simultaneous localization and mapping (SLAM). Throughout the history of robotics, there have been many effective SLAM strategies using a variety of different sensors. SLAM methods start to be applied on various mobile robots and autonomous driving systems. However, the key problem in the urban-driving domain becomes more challenging when it comes to off-road driving.

Off-road driving requires additional care on both the ground situation and surrounding environment. Off-road vehicles often need to traverse harsh terrain with high irregularity, rough

Figure 1.1: Well known autonomous driving systems: (a) Tesla Model S (b) Google's Waymo project (c) Uber's self-driving taxi. Image source: (a) *Tesla, Inc.* (b) *Digital Trends* (c) *Uber Technologies, Inc.*

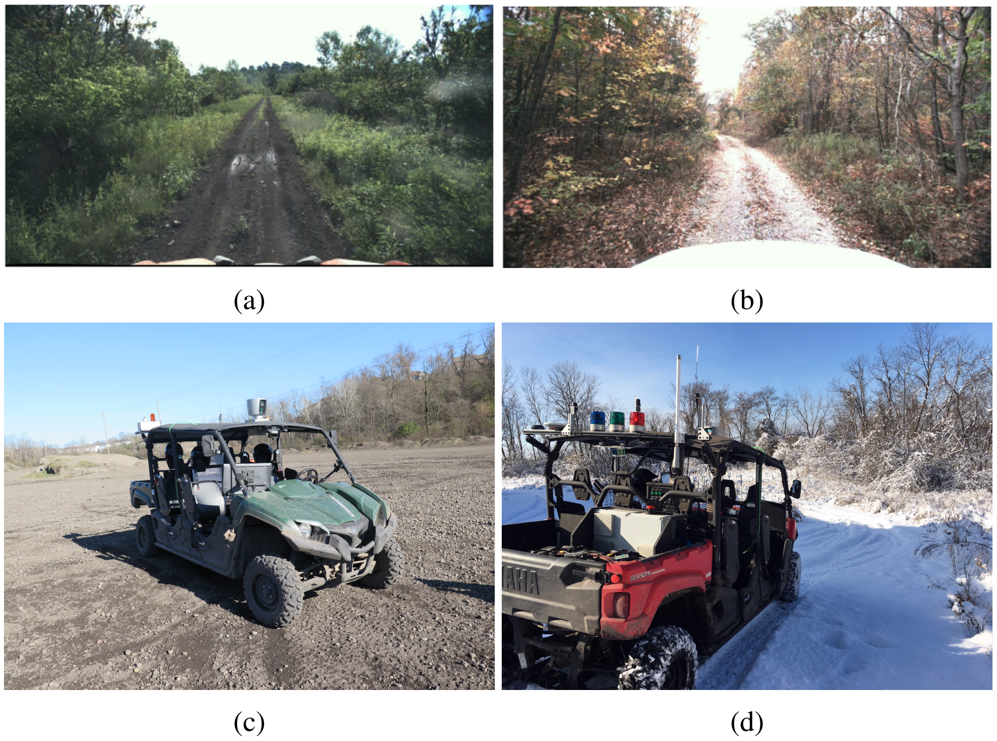


Figure 1.2: Images of off-road environment (a)(b) and our testing vehicles: (c) Leaf (d) Erik

surfaces and varying elevation as shown in Fig. 1.2. Unlike urban places, the lack of distinct structured landmarks in wide open area makes it difficult to extract salient features for localization. Driving in unknown areas without regular lanes to follow, an off-road vehicle needs to navigate itself through the traversable trails while avoiding untraversable and risky terrain. For example, there could be a deep puddle on the road which can potentially sink the vehicle's exhaust pipe, or the vehicle is driving toward an area with slippery ground because of the ice on the road. The vehicle has to be able to sense these details in the environment while driving.

Hence, in order to accomplish fully autonomous navigation in off-road environment, it is crucial to have instant information about the vehicle's location and motion as well as the geometry representation of the surroundings with spatial details, and as a results, reliable localization and mapping capability are required.

This thesis builds upon the work in off-road localization and mapping. We investigate traditional localization method with state estimation using multi-sensor fusion technique. We also explore the application of the state-of-the-art visual SLAM method in off-road driving scenarios. In addition, we propose an efficient mapping approach using vision sensing only. Specifically, we are interested in mapping off-road environment efficiently for navigation using minimal sensory modality.

1.2 Outline

In Chapter 2 we briefly go through the system overview and approaches used in this thesis. In Chapter 3 we present the state estimation approach for off-road driving using filter-based sensor fusion, followed by the formulation of local and global state estimation, finally the experimental results on the field are given. In Chapter 4 we evaluate a state-of-the-art visual SLAM method on the off-road data with experimental results and discussion. In Chapter 5 we propose a 3D occupancy mapping framework using only stereo vision, followed by introduction of each module in the system, and finally the mapping results on two different datasets. Finally, we conclude our work in Chapter 6 and suggests possible future work.

Chapter 2

Background

2.1 System Overview

Our testing platform is an utility side-by-side vehicle (UTV) manufactured by Yamaha Motor Company. The UTV is an all-terrain vehicle that is capable of traversing rough terrain in high speed. The vehicle is equipped with customized drive-by-wire system, velocity controller and multiple sensors for perception and navigation, including GPS, IMU, stereo camera and wheel speed sensor. The vehicle also has a high-precision GPS/INS on-board, which we take as the ground-truth sensor. Fig. 2.1 shows the UTV and sensors that are used in this work.

In this work we use the main on-board sensors shown in Fig. 2.1 for vehicle localization and environment mapping tasks. For validation, we use measurements reported by the GPS/INS with RTK signal as our ground-truth data. The on-board IMU comes with the gyroscopes and accelerometers and it reports inertial data at 200 Hz. The stereo camera has a $80^\circ \times 45^\circ$ FOV, 54 cm baseline, and takes 0.5 megapixel at 30 fps. The WAAS-enabled GPS reports position at 1 Hz with error within 3 m. Lastly, the wheel speed sensor (tachometer) reports vehicle's instantaneous velocity at 100 Hz.

2.2 State Estimation

For navigation tasks, it is crucial to always know where the vehicle is and how it orients in the world while driving. The process of estimating the position and orientation with respect to a certain coordinate system is the key problem of "localization" in robotics or the so-called "pose estimation" in computer vision. In addition to the pose, one would also need information about the speed of movement including linear and angular velocities for better control, navigation and motion planning. Tracking all these information in the same time forms the problem of state estimation.

There are two main approaches to perform state estimation: filtering and smoothing (batch optimization). The filtering approach estimates the state probabilistically in a recursive way using only the latest observations from the sensors, usually in two steps: the prediction and the update. The smoothing approach maintains a history of sensor observations and performs a nonlinear batch optimization over the past states to give the optimal solution given all the observations.

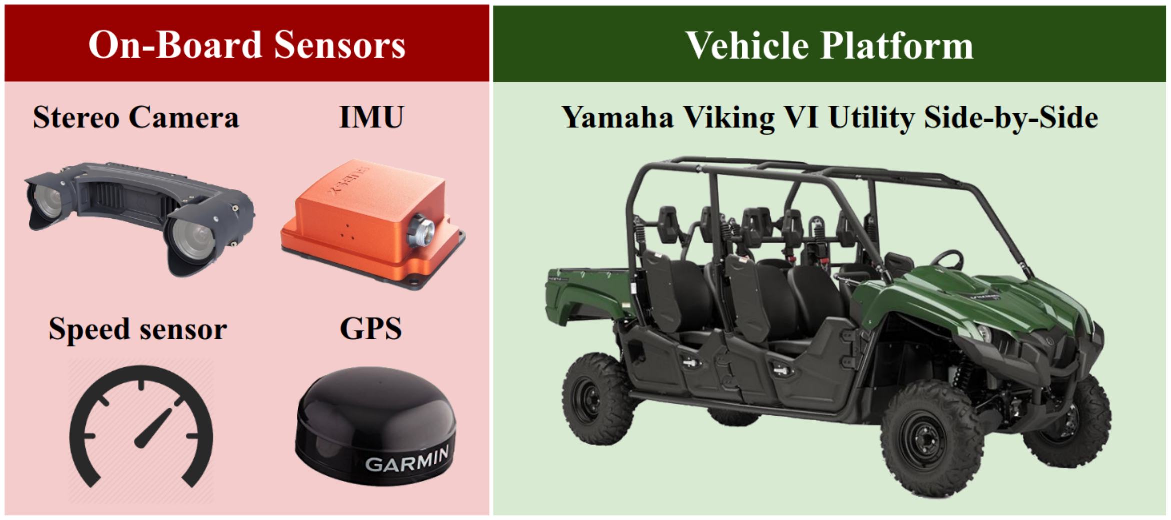


Figure 2.1: The testing vehicle and on-board sensors.

One popular filtering method is the Kalman filter [5], which is one type of recursive Bayesian estimation. By assuming the normal distribution of the state, a Kalman filter recursively estimates the state by incorporating sensor measurements and their uncertainties in the prediction and update steps. Kalman filters have been studied for over half-century with many extensions, generalizations and variants. They have numerous applications in modern technology and are used extensively in navigation and guidance systems.

Moore et al. [6] investigated the problems of state estimation using either a global frame or the body-centric frame. They described the disadvantages of global-frame-based localization and mapping due to discontinuity caused by GPS error. They therefore presented the strategy of using a local frame for simultaneous local and global state estimation. Recently, Moore and Stouch [7] proposed a generalized extended Kalman filter implementation specifically for robot localization, which allows the data fusion of unlimited number of navigation sensors (odometry, IMU and GPS) and the flexibility of choosing the sensor state variables for updating the state estimate.

2.3 Visual SLAM

Visual SLAM refers to the problem of vision-based simultaneous localization and mapping. In this type of SLAM system, localization and mapping are performed using image information only. It has the goal of localizing the camera while reconstructing the environment. By using graph-based optimization or bundle adjustment, visual SLAM systems have the ability to counteract the drift in the estimated trajectory.

A great advantage of SLAM systems is the capability to recognize the visited places and detect loop closures for maintaining a globally consistent map. These so-called loop closures are of great importance for reducing drift. Consequently, visual SLAM is preferred by mobile platforms traversing unknown areas to other odometry sources for the global consistency.

There are two dominant visual SLAM methods: feature-based methods and direct methods. Feature-based methods abstract the images to a sparse set of feature correspondences. Based on the feature correspondences between subsequent frames, the relative motion between these frames is computed by minimizing the geometric reprojection error. In contrast to feature-based methods, direct methods use the whole image for tracking by minimizing the photometric error. As direct methods minimize the photometric error over all pixels in the images, they are computationally intense and thus usually slower than feature-based methods.

Over the past few years, there have been many breakthrough in the development of visual-SLAM systems using keyframe-based design and multi-threading implementation. The recent S-PTAM by Pire et al. [8] is a feature-based stereo system follows the parallel-tracking-and-mapping strategy and performs local bundle adjustment. The LSD-SLAM by Engel et al [9, 10] is a semi-dense direct approach that minimizes photometric error in image regions with high gradient. Both the monocular and stereo version of LSD-SLAM have proved the effectiveness of the direct method. Mur-Artal et al. [11] proposed a feature-based system using ORB features, named ORB-SLAM. They recently published the new version, ORB-SLAM2 [2], which can be used on a monocular, a stereo or a RGB-D camera. The stereo ORB-SLAM2 achieved exceptional accuracy and robustness on various public datasets and benchmarks, arguably the state-of-the-art feature-based visual-SLAM method. In this work, we would like to test its limit in the more challenging off-road environment and utilize its localization capability in the proposed mapping framework.

2.4 Occupancy Mapping

Mapping is a crucial and necessary task for a wide range of robotics applications. The ability to construct a map allows a robot to localize itself and navigate through the environment autonomously. It is one of the key problems in robot perception and is strongly tied to localization, since a robot needs to know its location over time to build a consistent map.

For self-driving vehicles, mapping has become an integral part of enabling fully autonomous driving. This is especially critical for off-road navigation. An off-road vehicle usually drives in unknown areas where no existed maps with scene details available. In addition, off-road vehicles often need to traverse irregular terrain along with overhanging objects and obstacles such as tree branches and tall bushes. Hence, a three-dimensional dense map with spatial details on the field is important for off-road planning and navigation.

Most of the dense 3D reconstruction methods use a point-based representation by storing 3D range measurements directly. The occupied space in the environment is modeled with 3D point clouds returned by range sensors such as laser scanners, stereo sensors or RGB-D cameras. The point cloud approach has been used in several 3D mapping systems such as those presented by Geiger et al.[12] and Alcantarilla et al. [13] using stereo images as well as SLAM approach of Nüchter et al.[14]. The drawbacks of this kind of representation are that neither the free space nor the unknown areas are modeled and the sensor noise and dynamic objects cannot be dealt with directly. As a result, point clouds are only suitable for high precision sensors in static environments without the need to represent unknown areas. Moreover, the memory consumption of this representation increases rapidly with the number of measurements over time.

Another approach to model environments in 3D is to use a grid of equal-size cubic volumes, called voxels, to discretize the space of mapped area. Moravec [15] presented the early work of using such a representation. However, rigid grids have large memory requirement and need to be initialized with knowing the extent of mapped area beforehand, which is not tractable for large-scale out-door mapping. Recently, an octree-based 3D mapping approach is proposed by Hornung et al [4]. The octree data structure avoids one of the main shortcomings of fixed grid: they delay the initialization of map volumes until measurements need to be integrated. As a result, the size of mapped area does not need to be known beforehand and the map only contains volumes that have been mapped, which significantly reduces the memory consumption.

Chapter 3

State Estimation for Off-Road Driving

In this chapter, we present a filtering method to estimate the 3D state of our autonomous UTV driving in off-road environment. We use a generalized extended Kalman filter to fuse measurement data from the on-board sensors and estimate the position, orientation and velocities of the vehicle locally and globally in real time. We present the estimation framework and provide experimental results from the field tests.

3.1 Generalized Extended Kalman Filter

The Kalman filter is by far the most popular and useful estimation algorithm to date [16]. Kalman filtering is a recursive method of estimating the state of a dynamical system in the presence of uncertainties. With the assumption of Gaussian distributed errors, Kalman filter gives the optimal solution for state estimate in linear models. However, for nonlinear systems, an extended version of Kalman filter is required to deal with the uncertainty propagation through linearization.

In this work, we employ an open-source application of a generalized framework[1] to conduct the state estimation for vehicle localization [7]. The software package is specifically designed for robot localization via state estimation and is fully implemented for the Robot Operating System (ROS) [17]. The unique feature of this framework is that it allows for an unlimited number of sensor inputs and also allows per-sensor control of which variables of a sensor data are fused with the state estimate.

The formulation and algorithm of extended Kalman filter (EKF) are well-known [5, 16]. Here we detail the EKF algorithm for state estimation in 3D space. The goal is to fuse the associate sensor data and estimate the full 3D (6DoF) pose and velocity of the vehicle over time.

The state transition process of a robot can be described as a nonlinear dynamical system,

$$\boldsymbol{x}_k = f(\boldsymbol{x}_{k-1}) + \boldsymbol{\epsilon}_k \tag{3.1}$$

$$\boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q}_k) \tag{3.2}$$

where $\boldsymbol{x}_k$ is the robot's state (i.e., 3D pose and velocity) at time k, f is a nonlinear state transition function which encodes the system motion model, and $\boldsymbol{\epsilon}_k$ is the process noise assumed to be

[1]http://docs.ros.org/indigo/api/robot_localization/html

normally distributed with zero mean. The 12-dimensional state vector $\boldsymbol{x}$ comprises the robot's 3D 6DoF pose and their respective velocities as follows.

$$\boldsymbol{x} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}]^\top \tag{3.3}$$

The 3D rotation is expressed as Euler angles. In addition, given the current state the observation process can be described as

$$\boldsymbol{z}_k = h(\boldsymbol{x}_k) + \boldsymbol{\delta}_k \tag{3.4}$$

$$\boldsymbol{\delta}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_k) \tag{3.5}$$

where $\boldsymbol{z}_k$ is the measurement received at time k, h is a nonlinear sensor model which maps the state into the measurement space, and $\boldsymbol{\delta}_k$ is the normally distributed measurement noise with zero mean.

For the filter, a state estimate at time k can be expressed as a normal distribution.

$$\hat{\boldsymbol{X}}_k \sim \mathcal{N}(\hat{\boldsymbol{x}}_k, \boldsymbol{\Sigma}_k) \tag{3.6}$$

where $\hat{\boldsymbol{x}}_k$ is the mean and $\boldsymbol{\Sigma}_k$ is the covariance.

In the prediction step, the current state estimate and error covariance are propagated forward in time through the motion model:

$$\bar{\boldsymbol{x}}_k = f(\hat{\boldsymbol{x}}_{k-1}) \tag{3.7}$$

$$\bar{\boldsymbol{\Sigma}}_k = \boldsymbol{F}_k \boldsymbol{\Sigma}_{k-1} \boldsymbol{F}_k^\top + \boldsymbol{Q}_k \tag{3.8}$$

In our case, without control inputs, f is a standard 3D kinematic model derived from Newtonian mechanics which only depends on the current state estimate and the time, see A.1. This formulation therefore has all the state variables to be updated in the correction step by the corresponding sensor data. The error covariance estimate, Σ, is projected via $\boldsymbol{F}$, the Jacobian of f, and then perturbed by $\boldsymbol{Q}$, the process noise covariance. The bar notation here denotes the predicted estimates.

The correction step is carried out for updating the estimates with sensor measurements:

$$\boldsymbol{K}_k = \bar{\boldsymbol{\Sigma}}_k \boldsymbol{H}_k^\top (\boldsymbol{H}_k \bar{\boldsymbol{\Sigma}}_k \boldsymbol{H}_k^\top + \boldsymbol{R}_k)^{-1} \tag{3.9}$$

$$\hat{\boldsymbol{x}}_k = \bar{\boldsymbol{x}}_k + \boldsymbol{K}_k(\boldsymbol{z}_k - \boldsymbol{H}_k \bar{\boldsymbol{x}}_k) \tag{3.10}$$

$$\boldsymbol{\Sigma}_k = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k) \bar{\boldsymbol{\Sigma}}_k (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k)^\top + \boldsymbol{K}_k \boldsymbol{R}_k \boldsymbol{K}_k^\top \tag{3.11}$$

Note that the Joseph form of covariance update equation [18] is used in the correction step to maintain filter stability by ensuring that $\boldsymbol{\Sigma}_k$ remains positive semi-definite.

In the standard EKF formulation, $\boldsymbol{H}$ is specified as the Jacobian matrix of the sensor model function h. In order to support a broad array of sensors, this framework assumes that each sensor produces measurements of the state variable directly, i.e. $h(\boldsymbol{x}) = \boldsymbol{H}\boldsymbol{x}$. In practice, this allows for partial updates of the state vector, which accommodates the common situation that a sensor data does not measure every variable in the state vector. Specifically, when measuring only m variables, $\boldsymbol{H}$ becomes an m by 12 matrix of rank m, with its only nonzero values (in this case, ones) existing in the columns of the measured variables, see A.2.

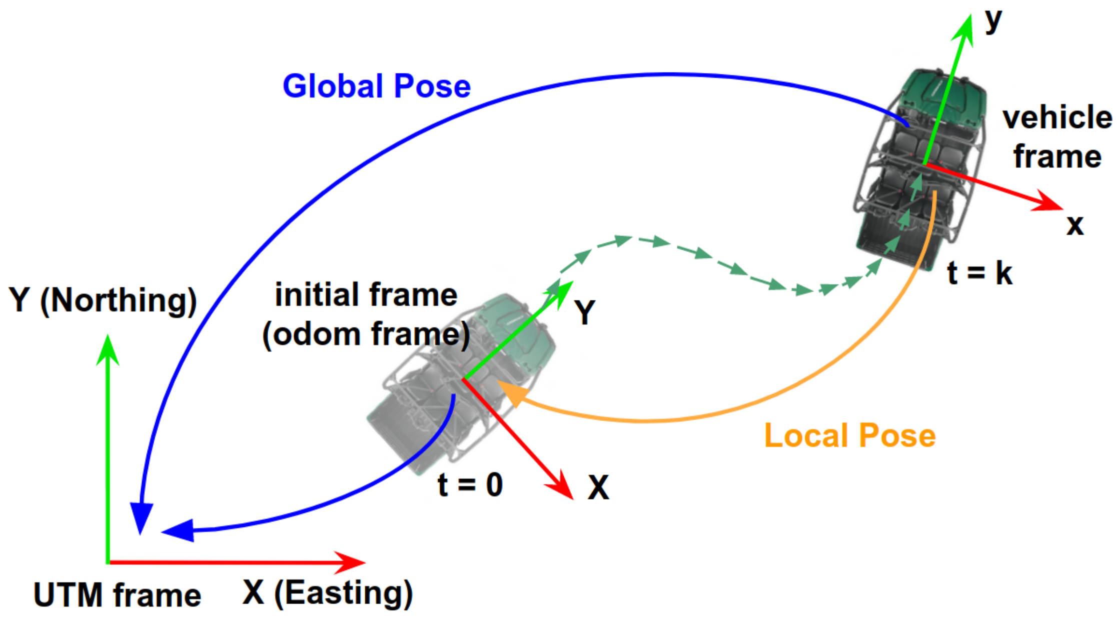


Figure 3.1: Relation between global pose and local pose

Theoretically speaking, the assumption on the sensor model does not hold for large vehicles or mobile platforms with on-board sensors placed far apart each other, in which case the "lever-arm" effect must be taken into consideration. However, for standard size vehicle or small mobile robot with considerate position error in the sensor, e.g. 3m to 5m for a GPS, the lever-arm effect is not significant and the assumption can hold for general operations.

3.2 Local and Global State Estimation

Autonomous navigation and planning tasks often require instant report of the robot state which contains its position and orientation as well as the inertial information including linear velocities, angular velocities and accelerations.

Global planning algorithms usually perform in an absolute earth-referenced coordinate system for navigational task such as waypoint following. On the other hand, local planning policies usually focus on short-term robot movement and relative motion with respect to a local frame, for example, path planning for obstacle avoidance. Consequently, the robot's pose with respect to a global and a local frame are both necessary for autonomous driving. Fig. 3.1 shows a 2D illustration of the relation between the global and local poses. A global pose refers to the absolute position and rotation in the Universal Transverse Mercator (UTM) coordinate system usually reported by a GPS and a magnetometer, and therefore it gives the geographic location and earth-referenced orientation. A local pose represents the vehicle position and orientation with respect to a local coordinate system, often called odometry frame or "odom" frame. The odom frame is a world-fixed frame that is usually represented by the initial pose (initial body frame) where the motion starts [19]. Combined with the inertial data measured by an IMU, one can track the overall motion locally and globally, which forms the problems of local and global

Table 3.1: Sensor measurements and state vectors

Sensor	Measurement	Description
GPS	$\boldsymbol{z}^{GPS} = [X, Y, Z]$	UTM coordinates in X, Y altitude in Z
IMU	$\boldsymbol{\Phi}^{IMU} = [\alpha, \beta, \gamma]$ $\boldsymbol{\omega}^{IMU} = [\omega_\alpha, \omega_\beta, \omega_\gamma]$	Roll, pitch and yaw[2] Roll rate, pitch rate and yaw rate
Wheel sensor	$\boldsymbol{z}^{Wheel} = v_y$	Vehicle velocity in y-direction
Estimator	**State vector**	**Description**
Global EKF Local EKF	$\hat{\boldsymbol{x}}^G = [x^G, y^G, z^G, \phi^G, \theta^G, \psi^G, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}]$ $\hat{\boldsymbol{x}}^L = [x^L, y^L, z^L, \phi^L, \theta^L, \psi^L, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}]$	Global state, pose wrt UTM frame Local state, pose wrt odom frame

state estimation.

To achieve both local and global estimation, we utilize two instances of the generalized EKF introduced in the last section to fuse the data from three main sensors on the vehicle. The sensor measurements and state vectors of both EKFs are shown in Table 3.1 (all in row vector form). The layout of the sensor fusion is shown in Fig. 3.2. A superscript either denotes the type of sensor (GPS, IMU or wheel sensor) that acquires the measurement or the reference frame (local or global) a state variable is with respect to.

For the local EKF, only the IMU and the wheel sensor are fused to estimate the local state. The sensor inputs include the forward speed v_y measured by the wheel sensor and the orientation angles and rates, $\boldsymbol{\Phi}^{IMU}$ and $\boldsymbol{\omega}^{IMU}$, from the IMU. Note that since there is no side-way velocity measurements (v_x, v_z) available in the system, for practical consideration, the zero velocities, $v_x = 0, v_z = 0$ should be fed into the filter to indicate that the vehicle cannot move instantaneously sideways. Since at the update step, only the orientation and velocities are corrected by the measurements, the uncertainties in positions will increase, which will be revealed in the next section.

For the global EKF, all three sensors are fused together to estimate the global state. GPS coordinates are used to corrected the positions in the global frame at the update step. For IMU inputs, we only feed the roll and pitch data into the filter since there is no absolute heading measurement available. Despite the lack of direct yaw update, the horizontal GPS position and yaw rate measurements together can help correct the absolute heading estimate since the vehicle's motion is constrained from moving sideways.

Regarding the covariances, since it is not trivial to tune the covariance values, we use a set of nominal covariance values which are given in the appendix, A.3.

[2]Note that our IMU does not report an absolute earth-referenced heading, the yaw measurement ψ here is the relative angle with respect to the initial heading when the sensor is turned on

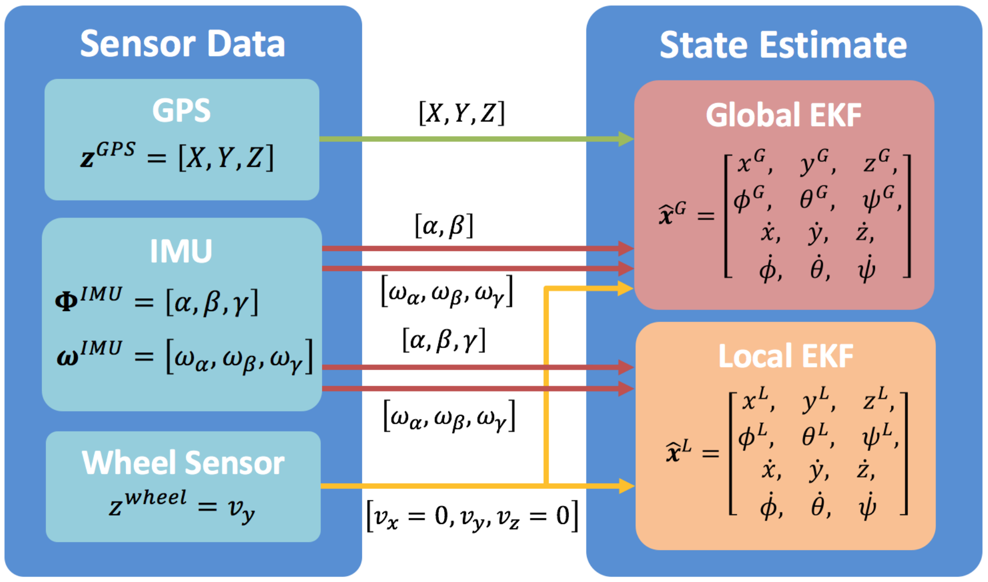

Sensor Data
GPS
$z^{GPS} = [X, Y, Z]$
IMU
$\Phi^{IMU} = [\alpha, \beta, \gamma]$
$\omega^{IMU} = [\omega_\alpha, \omega_\beta, \omega_\gamma]$
Wheel Sensor
$z^{wheel} = v_y$
$[X, Y, Z]$
$[\alpha, \beta]$
$[\omega_\alpha, \omega_\beta, \omega_\gamma]$
$[\alpha, \beta, \gamma]$
$[\omega_\alpha, \omega_\beta, \omega_\gamma]$
$[v_x = 0, v_y, v_z = 0]$
State Estimate
Global EKF
$\hat{x}^G = \begin{bmatrix} x^G, & y^G, & z^G, \\ \phi^G, & \theta^G, & \psi^G, \\ \dot{x}, & \dot{y}, & \dot{z}, \\ \dot{\phi}, & \dot{\theta}, & \dot{\psi} \end{bmatrix}$
Local EKF
$\hat{x}^L = \begin{bmatrix} x^L, & y^L, & z^L, \\ \phi^L, & \theta^L, & \psi^L, \\ \dot{x}, & \dot{y}, & \dot{z}, \\ \dot{\phi}, & \dot{\theta}, & \dot{\psi} \end{bmatrix}$

Figure 3.2: EKF sensor fusion layout

3.3 Experimental Results

To evaluate the performance of the EKFs for off-road driving cases, we collected sensor data from the test field for offline testing as well as conducted real-time experiments during autonomous driving.

As shown in Fig. 3.3, the selected off-road sample trails are used to test the EKFs. The loop-shape routes shown in Fig. 3.3(a) and (b) are good for testing position drift after the loop closure. In addition, all the locations of picked trails have significant elevation in terrain around 5 to 9 meters difference in height between the lowest and the highest points in a route, which is critical for testing the estimation of vehicle movement in the vertical direction. We also performed long-term testing on a long distance trail as shown in Fig. 3.7.

Both the local and global EKFs are tested on the field and compared with the ground-truth GPS/INS sensor. The ground-truth and estimated trajectories at four different locations are shown in Fig. 3.4. Note that for consistent comparison the trajectories from ground-truth and global EKF are both transformed onto the local frame so that all poses in a trajectory are with respect to the initial pose (start from origin). The full 6DoF pose estimation results at the two loop-shape trails are shown in Fig. 3.5 and 3.6. In addition to the trajectory, the time series of position and orientation are given for visualizing the change of vehicle pose over time. Also, to show the performance of global EKF, the trajectories in the global coordinate frame (UTM) are shown in Fig. 3.7 with the satellite image.

We use two different evaluation metrics, the absolute translation error proposed in [20] and the relative pose error proposed in [21]. For estimation with local EKF, we would like to inspect the drift in pose over the trajectory, for which the relative error is suitable to represent. On the other hand, for the global EKF, the absolute error metric is preferred to evaluate the accuracy in global positioning.

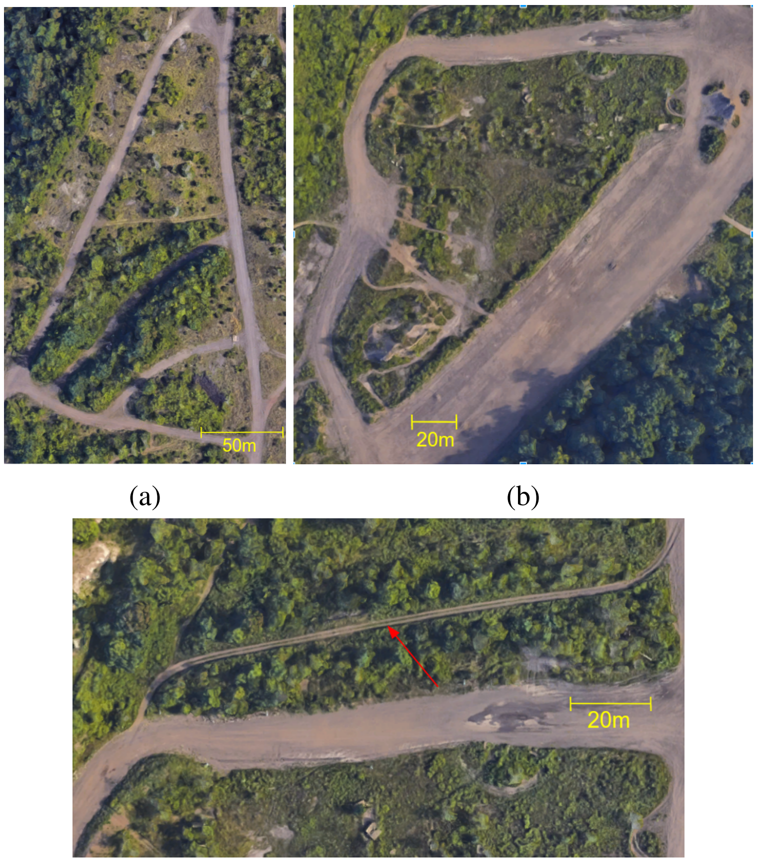
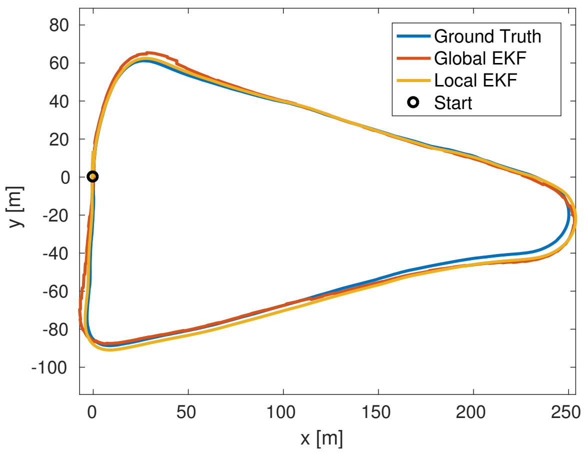


Figure 3.3: Google Maps images of sample trails [1] (a) triangle loop (b) slope loop (c) narrow trail

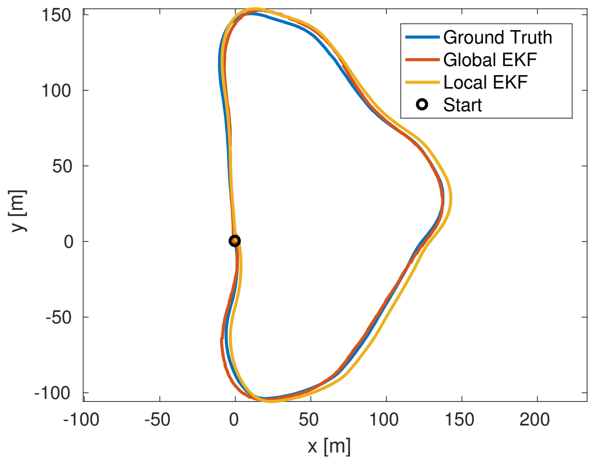
The evaluation results of local EKF at four selected trails are shown in Table 3.2. The translation and rotation errors are both mean RMSE from averaging over intervals of different travelled distances (100m, 200m to ... 1600m) [20]. The evaluation results of global EKF are shown in Table 3.3.

3.4 Discussion

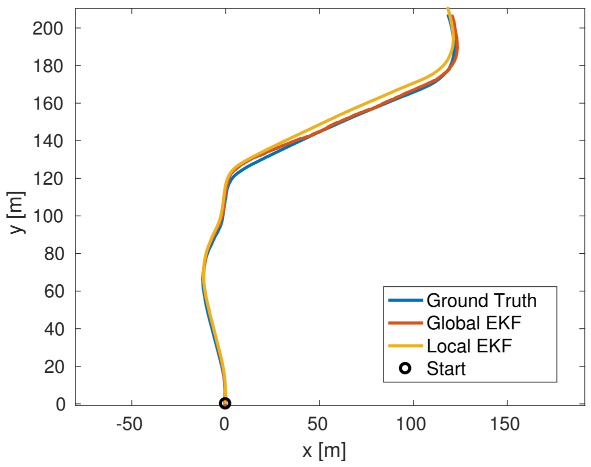
The evaluation results shows that the local EKF is effective in short-term localization with an average 1.7% of position drift and 1.29 degree of rotation error for routes under 700 m distance. However, without correction from the absolute position update, the local EKF is prone to drifting for long-distance movement due to the error accumulation and wheel slip issue as shown in Fig. 3.4(d). Additionally, the drifting in the vertical direction is significant due to the measurement bias from IMU as shown in Fig. 3.5 and 3.6, with an average RMSE of 5.5 m in the Z-direction. Despite the incapability in long-term navigation, the local EKF provides continuous pose estimates and accurate relative information in the local frame with high update rate, which



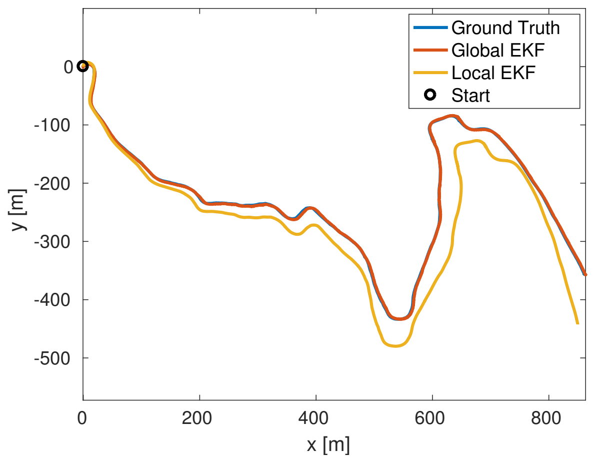
(a)



(b)



(c)



(d)

Figure 3.4: Estimated trajectory and ground-truth (a) triangle loop (b) slope loop (c) narrow trail (d) long trail

is good for tasks involving local planning, navigation and mapping in small area. For the rest of this work, the local EKF is taken as a baseline pose source for evaluating the localization performance in the local frame.

For global estimation, the EKF with GPS integration is able to track the vehicle motion with respect to the absolute earth-referenced coordinates. The evaluation result shows an average RMSE of 2.4 m in absolute translation error. As shown in Fig. 3.7, the global EKF is proved to be reliable in long-term driving despite the lack of absolute heading measurement. However, due to the integration of GPS measurements, the trajectory contains discontinuities in the position, which is not suitable for mapping or data registration in the local region. As a results, we take the global EKF as the global pose source which provides the absolute pose information for long-term and large-scale planning and navigation tasks.

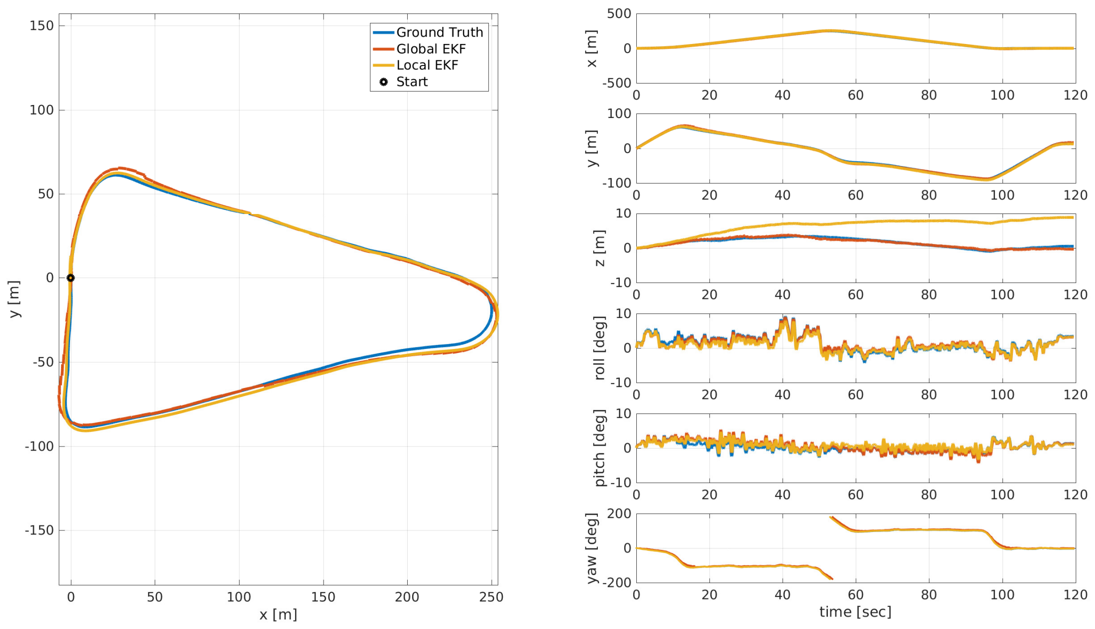


Figure 3.5: EKF estimation results at triangle loop

Table 3.2: Relative pose error of local EKF

Location	Distance (m)	Relative Pose Error Translation (%)	Rotation (deg)
Narrow trail	287	1.82	0.84
Slope loop	657	1.87	1.67
Triangle loop	684	1.55	1.35
Long trail	1605	5.02	6.55

· mean translation RMSE (%), avg. over 100m, 200m to ... 1600m intervals.
· mean rotation RMSE (deg/100m), avg. over 100m, 200m to ... 1600m intervals.

Table 3.3: Absolute translation error of global EKF

Location	Distance (m)	Absolute Translation Error RMSE (m)
Narrow trail	287	1.50
Slope loop	657	2.69
Triangle loop	684	2.88
Long trail	1605	2.53

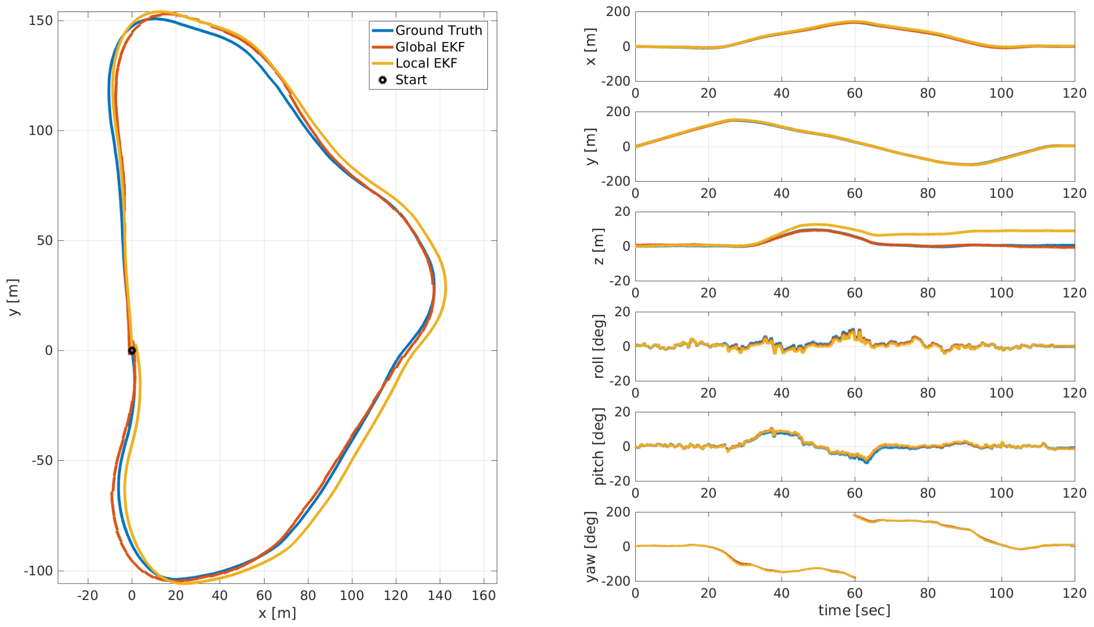


Figure 3.6: EKF estimation results at slope loop

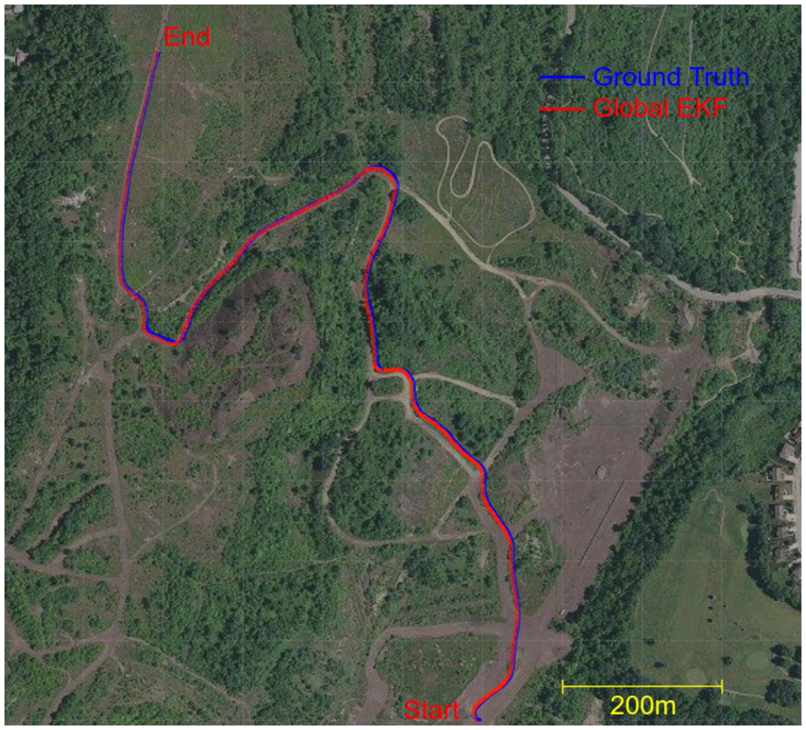


Figure 3.7: Satellite image of global EKF and ground-truth trajectories at long trail

Chapter 4

Visual SLAM for Off-Road Driving

In this chapter, we introduce the state-of-the-art feature-based visual-SLAM algorithm, ORB-SLAM2 [2]. We focus on the stereo version of ORB-SLAM2 and conduct experiments using our off-road image data collected by the autonomous UTV. We then present the results of stereo ORB-SLAM2 to demonstrate the localization and mapping performance in the off-road environment. We also compare the evaluation results of ORB-SLAM2 with those of EKF sensor fusion presented in Sec. 3.2. Finally, we discuss the issues when performing ORB-SLAM2 in off-road cases.

4.1 ORB-SLAM2

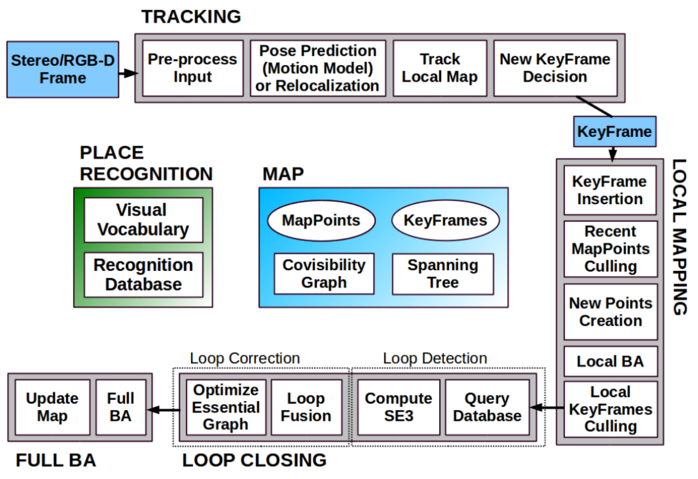
ORB-SLAM2 is an open-source visual-SLAM system for monocular, stereo and RGB-D cameras[1]. It is one of the state-of-the-art visual SLAM algorithms and arguably the most robust feature-based method currently. The keyframe-based algorithm is based on the use of ORB features [22]. ORB features are robust to rotation and scale change in viewpoint and they are extremely efficient to compute.

The system overview of ORB-SLAM2 is shown in Fig. 4.1. The multi-threading implementation of ORB-SLAM2 enabling the real-time processing on a CPU. We give brief description on the main system threads as follows.

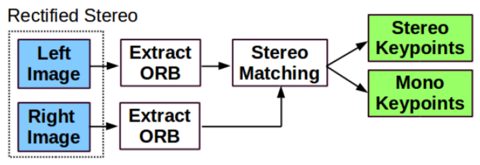
The system contains three main threads:

- Tracking thread: Tracks camera pose at every frame by finding ORB feature matches in the local map and minimizes the reprojection error by applying motion-only bundle adjustment (BA). Tracking thread also decides the insertion of a new keyframe.
- Local mapping thread: Creates new map points and performs local BA to optimize the local map.
- Loop closing thread: Detects loops and correct the accumulated drifts by performing a pose-graph optimization, which launches another thread to perform the full BA to compute the optimal structure and poses.

[1]https://github.com/raulmur/ORB_SLAM2



(a)



(b)

Figure 4.1: ORB-SLAM2 system overview [2] (a) system threads and modules (b)Stereo input pre-processing

At the input pre-processing step, ORB features are extracted at salient keypoint locations. For stereo cameras, the keypoints are classified and defined as follows:

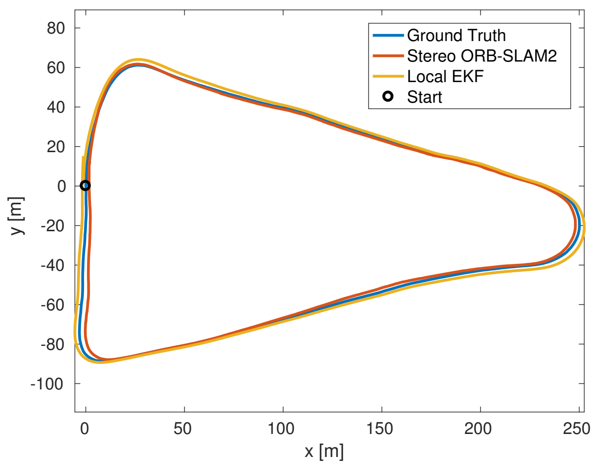
- Stereo keypoint: The ORB matched point defined by three coordinates $\boldsymbol{x}_s = (u_L, v_L, u_R)$ with (u_L, v_L) being the pixel coordinates in the left image and u_R the horizontal coordinate of the corresponding point in the right image. A stereo keypoint is further classified to close and far points.
    - Close keypoint: a stereo keypoint with depth less than 40 times the stereo baseline.
    - Far keypoint: otherwise.
- Monocular keypoint: Defined by $\boldsymbol{x}_m = (u_L, v_L)$ on the left image and correspond to all those ORB features for which a stereo match could not be found.

Close keypoints can be triangulated from one frame as depth is accurately estimated. They provide scale, translation and rotation information. On the other hand, far points provide accurate rotation information but weaker scale and translation information. The far points are triangulated when they are supported by multiple views. The monocular keypoints are only triangulated from multiple views and do not provide scale information, but they contribute to the rotation and translation estimation. The stereo matched keypoints are crucial for tracking the camera motion and the distribution of close and far points in a frame has a impact on the pose estimation, which will be revealed in the next section.

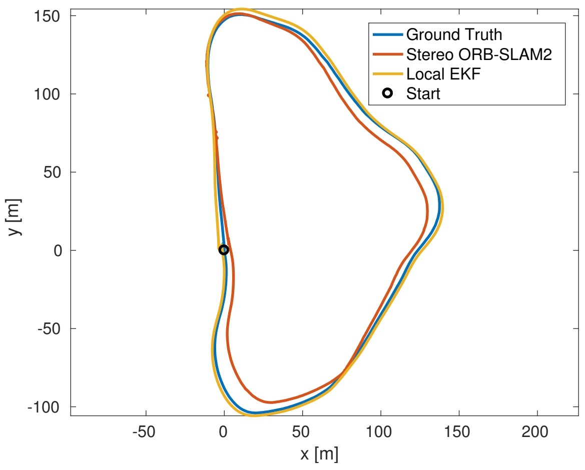
4.2 Experimental Results

To evaluate the performance of stereo visual-SLAM in off-road driving, we use the same dataset collected from the off-road trails shown in Fig. 3.3.

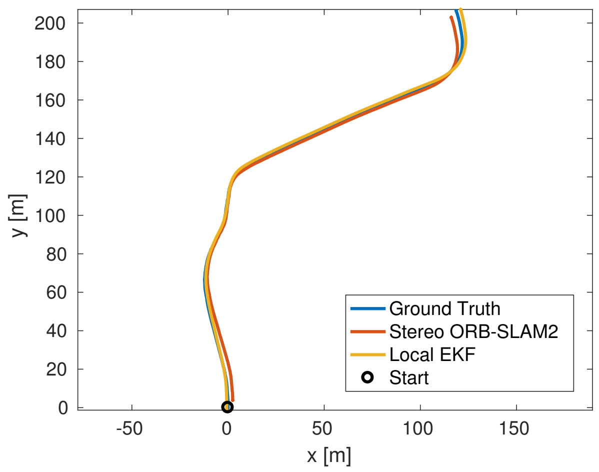
We ran the ORB-SLAM2 on each sequence 5 times and show the median results to account



(a)



(b)



(c)

Figure 4.2: Stereo ORB-SLAM2, local EKF and ground-truth at (a) triangle loop (b) slope loop and (c) narrow trail

for the non-deterministic nature of the multi-threading system. Our stereo camera has a ∼21 cm baseline with 30 fps and image resolution of 1024 × 544 pixels. Due to the processing time limit of ORB-SLAM2, we have to downsample our image inputs so that the frame rate is trackable for ORB-SLAM2, which is about half of the original frame rate (∼15 fps).

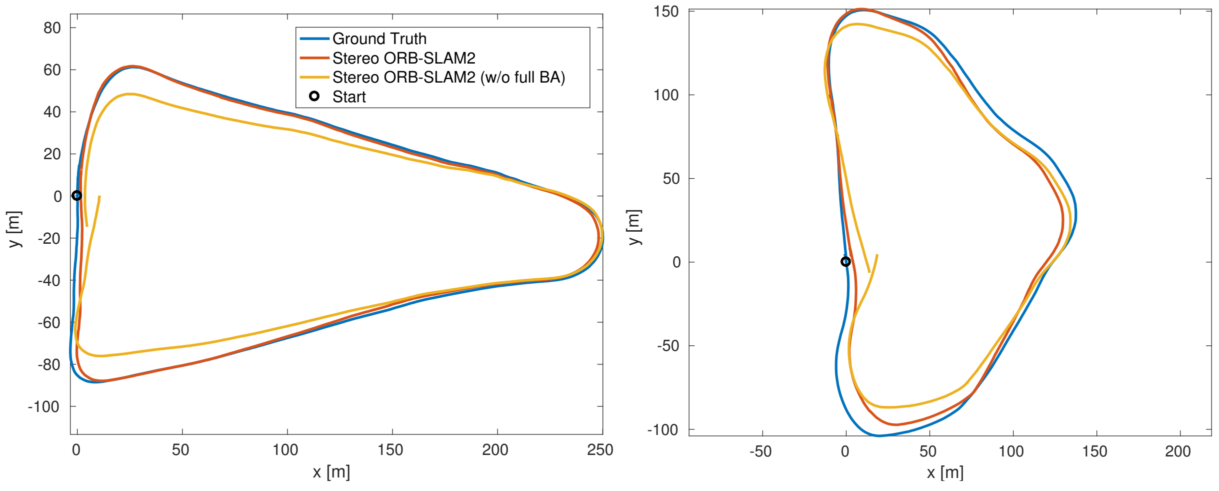
The resulting trajectories are shown in Fig. 4.2. For comparison, the trajectories of local EKF are also shown in the figure. We use the same two evaluation metrics used in Sec. 3.3, relative pose error and absolute translation error. The evaluation results are shown in Table 4.1. In addition, the sparse reconstructions of the trails by ORB-SLAM2 are shown in Fig. 4.4

Since ORB-SLAM2 performs the global bundle adjustment on the full graph after detecting a loop closure, we also like to know the effect of loop closure correction on the estimated trajectory. For comparison, we save the trajectory right before the loop closure happens in each sequence, as shown in Fig. 4.3. Table 4.2 shows the comparison of evaluation results on trajectories with and without loop closure correction.

Table 4.1: Results of stereo ORB-SLAM2 in the off-road dataset

Location	Distance (m)	Stereo ORB-SLAM2 t_{rel}	Stereo ORB-SLAM2 r_{rel}	Stereo ORB-SLAM2 t_{abs}	Local EKF t_{rel}	Local EKF r_{rel}	Global EKF t_{abs}
Narrow trail	287	3.28	2.29	2.83	1.82	0.84	1.50
Slope loop	657	6.51	4.60	7.15	1.87	1.67	2.69
Triangle loop	684	2.50	1.66	1.88	1.55	1.35	2.88

$\cdot t_{rel}$: mean relative translation RMSE (%), avg. over 100m to 600m intervals.
$\cdot r_{rel}$: mean relative rotation RMSE (deg/100m), avg. over 100m to 600m intervals.
$\cdot t_{abs}$: absolute translation RMSE (m), after 6DoF alignment.



(a) (b)

Figure 4.3: Comparison of stereo ORB-SLAM2 trajectories before and after the loop-closure

4.3 Discussion

The evaluation result shows that stereo ORB-SLAM2 can give good pose estimate in general off-road driving cases. However it also shows that ORB-SLAM2 is not robust in the off-road environment in terms of localization performance. One of the possible causes is the lighting condition. We have seen large variation in image intensity over short period of time in one sequence (slope loop), which caused the number of matched points to drop significantly and leads to the concentration of feature points in the far region of the image as shown in Fig 4.5. Since far points only contribute to the estimation in orientation but provide weak information for translation, the loss of track on the close points cause the estimation in translation to be off.

Despite the sensitivity to environment, ORB-SLAM2 shows impressive capability in localization using only image input in challenging off-road driving tests. For every test on the loop-shape trails, ORB-SLAM2 successfully detected loop-closure in a short amount of time. The ability to detect loop-closure and trajectory correction is critical to conduct consistent mapping, which will be demonstrated in the next chapter.

Table 4.2: Comparison of stereo ORB-SLAM2, before and after loop-closure

Location	Distance (m)	After loop-closure (w/ Global BA)			Before loop-closure (w/o Global BA)		
		t_{rel}	r_{rel}	t_{abs}	t_{rel}	r_{rel}	t_{abs}
Slope loop	657	6.51	4.60	7.15	9.61	11.3	12.1
Triangle loop	684	2.50	1.66	1.88	7.10	8.24	9.29

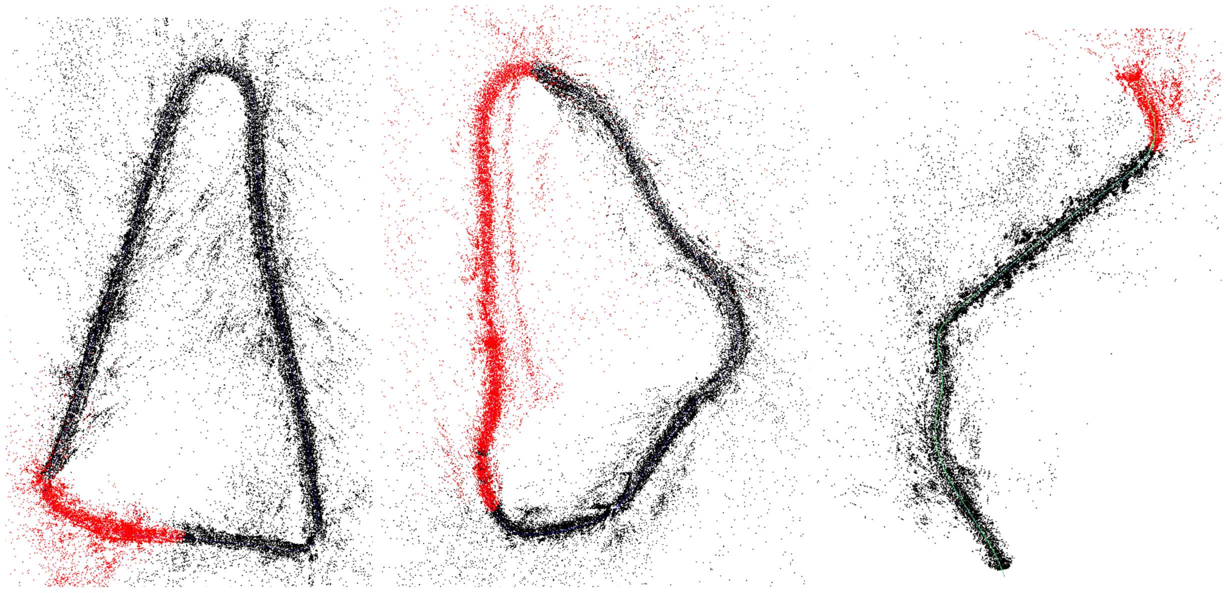
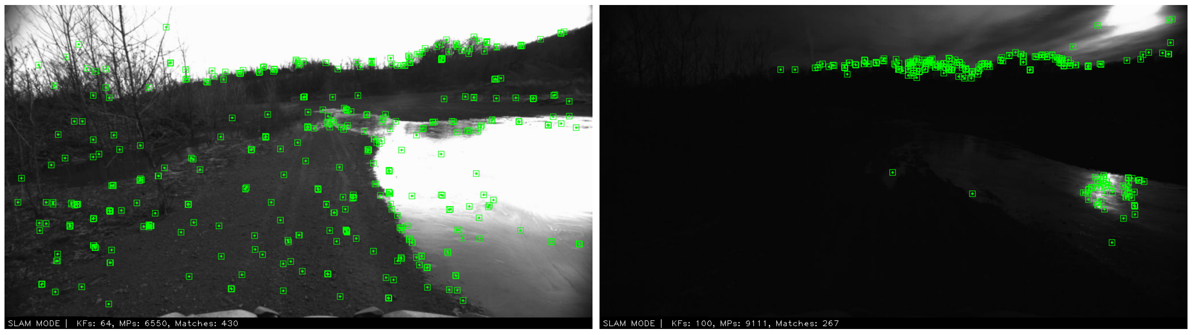


Figure 4.4: ORB-SLAM2 sparse reconstruction of the off-road trails, map points (black and read), current local map points (red).



SLAM MODE | KFs: 64, MPs: 6550, Matches: 430

SLAM MODE | KFs: 100, MPs: 9111, Matches: 267

(a) (b)

Figure 4.5: Effect of lighting condition. (a) tracked feature points under normal illumination. (b) feature tracking affected by unpredictable lighting.

Chapter 5

Occupancy Mapping using Stereo Vision

In this chapter, we present an occupancy mapping framework using stereo vision. We explore the capability of using single stereo camera for large-scale dense mapping in the off-road environment. We first introduce the stereo-based mapping framework, followed by the method of dense stereo matching and point cloud construction. Then, we detail the core 3D occupancy mapping approach used in this framework. Finally, we present experimental results on the KITTI benchmark dataset [21] and our own off-road dataset.

5.1 Stereo-based Occupancy Mapping Framework

Stereo cameras can provide depth information at each frame through two-view geometry, which enables their use for 3D reconstruction. There have been many related work of using stereo images for large-scale dense 3D reconstruction [12, 13]. However, most of the previous work requires the storage and registration of large amount of 3D point clouds, which limits the real-time application in large-scale environment due to the problem of point cloud growth over time.

Instead of using point representation, occupancy grid map [15, 23, 24] provides an memory-efficient geometric representation of the environment by discretizing the space into cell blocks, the so-called voxels. Using the 3D range measurements, a robot can sense the environment and create a 3D grid map with occupancy probability at each voxel.

The pipeline of the proposed stereo-based 3D mapping framework is shown in Fig 5.1. The main procedures are listed as follows:

1. Acquire stereo images of the current scene and perform stereo image processing.
2. Use the rectified stereo pair and camera parameters to generate the disparity map through dense stereo matching.
3. Construct the 3D point cloud from the disparity map at the current frame.
4. Estimate the camera pose using stereo visual-SLAM algorithm.
5. Continuously perform 3D occupancy mapping over time with camera pose and point cloud at each frame.

Note that in a more general framework, the sensor does not have to be a camera, it could be a lidar, laser scanner, radar, or any type of sensor that is capable of range sensing. Also, the sensor

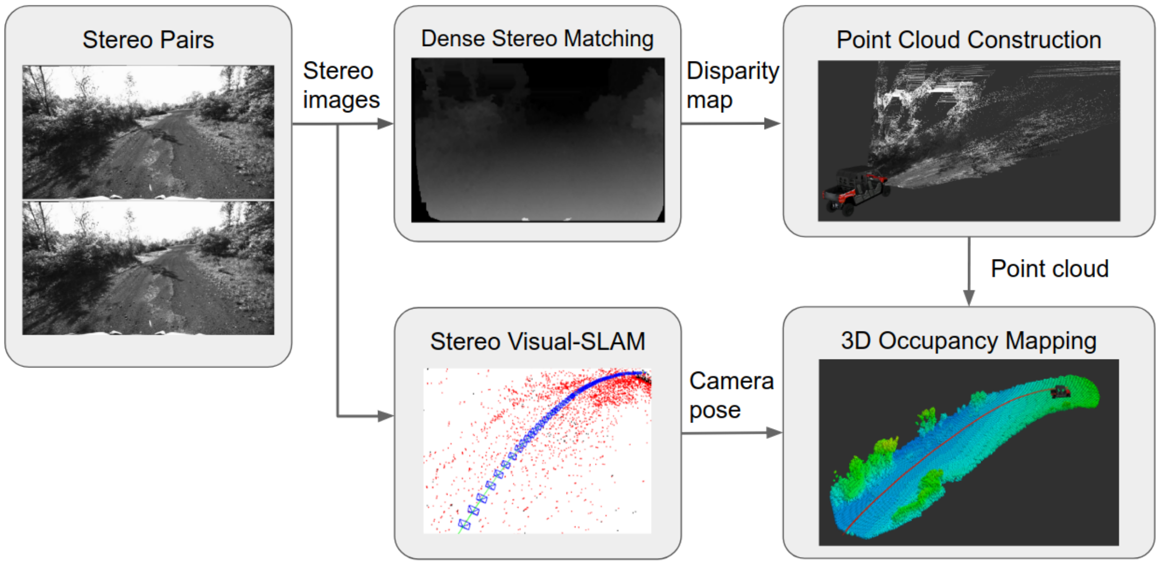


Figure 5.1: Stereo-based occupancy mapping pipeline

pose needs not come from the same sensor itself, it could be the pose from estimation and fusion on other sensor measurements such as odometry, IMU or GPS, provided proper registration.

In our case, we intend to minimize the use of available sensors and exam the ability of localization and mapping using pure vision with single stereo camera due to the fact that stereo vision provides a low-cost and non-intrusive alternative for inferring geometric information of the surrounding world compared to expensive devices such as lidars and laser range finders.

The other thing worth mentioning is that the proposed framework takes a known-pose mapping approach rather than a SLAM approach despite the use of a visual-SLAM module inside. The visual-SLAM subsystem does not correct the resulting occupancy map over time, nor does the mapping module updates the pose estimation from visual-SLAM. The occupancy mapping module uses the estimated camera pose directly and builds the map accordingly.

5.2 Stereo Matching and 3D Reconstruction

To perform three-dimensional mapping, first we need to infer the 3D information from 2D images. This step can be achieved by matching the left and right stereo images using multi-view geometry.

Stereo matching denotes the classic computer vision problem of finding dense correspondences in a pair of images in order to perform 3D reconstruction. Fig. 5.2 outlines the 3D reconstruction pipeline using the stereo matching approach. Note that throughout this chapter we assume that the stereo pair has been rectified such that corresponding points lie on the same horizontal epipolar line. The image rectification process can be carried out using standard methods given a calibrated stereo camera [25]. As shown in Fig. 5.2, corresponding points, p and p', have different pixel locations in the left and right images when the cameras see the same scene from different viewpoints. The amount of horizontal displacement (in pixels) is called the dis-

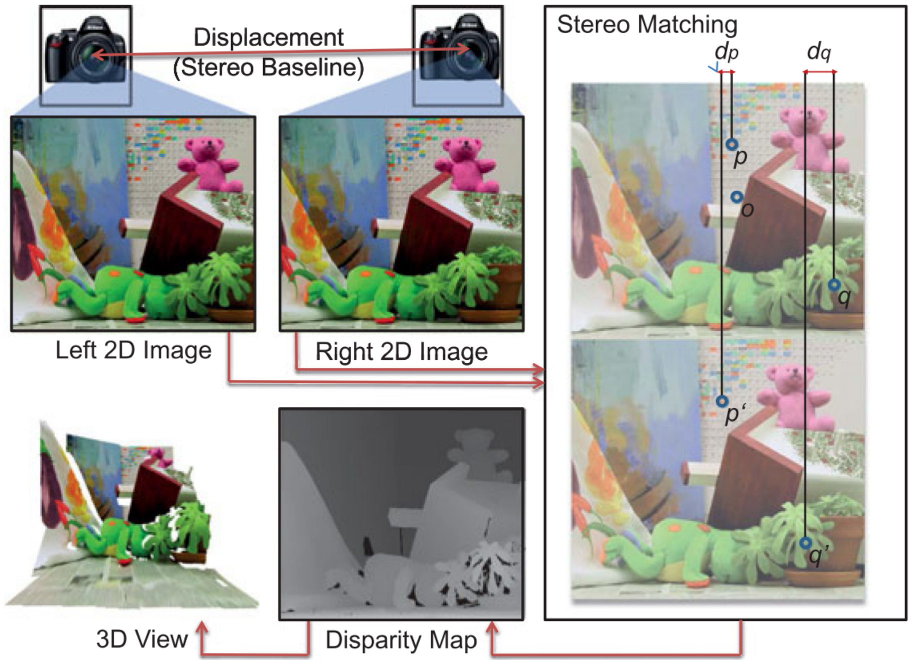


Figure 5.2: 3D reconstruction via stereo matching [3]

parity, denoted by d_p in the figure. The important observation is that disparity of the foreground pixels (close points, q and q') is larger than that of the background pixels (far points, p and p'), i.e. $d_q > d_p$. The disparity of a pixel is inversely proportional to the real distance (depth) of the point to the camera. Storing the disparity value at each pixel gives the so-called disparity map which is usually represented by an intensity image where dark pixels encode low disparity (high depth) and bright ones encode large disparity (small depth).

Once a disparity map is obtained, one can easily infer the depth information of the scene and perform 3D reconstruction by projecting the disparity image back to the 3D space.

5.2.1 Efficient Large-Scale Stereo Matching

In this work we use an efficient large-scale stereo matching algorithm called ELAS (Efficient LArge-scale Stereo)[1] [26] to perform dense stereo matching and produce the disparity map. ELAS is a local matching method which uses a generative probabilistic model for stereo matching. The method builds a prior over the disparity space by forming triangulation on a set of robustly matched correspondences, called "support points". Having the piecewise linear prior, ELAS does not suffer in the presence of low-textured and slanted surfaces and it allows dense matching with small aggregation windows. The efficient performance of ELAS enables stereo matching at the frequency close to frame rate which is ideal for our mapping framework. Fig. 5.3 shows a few examples of stereo matching results on the KITTI dataset and our off-road dataset.

[1]http://www.cvlibs.net/software/libelas/

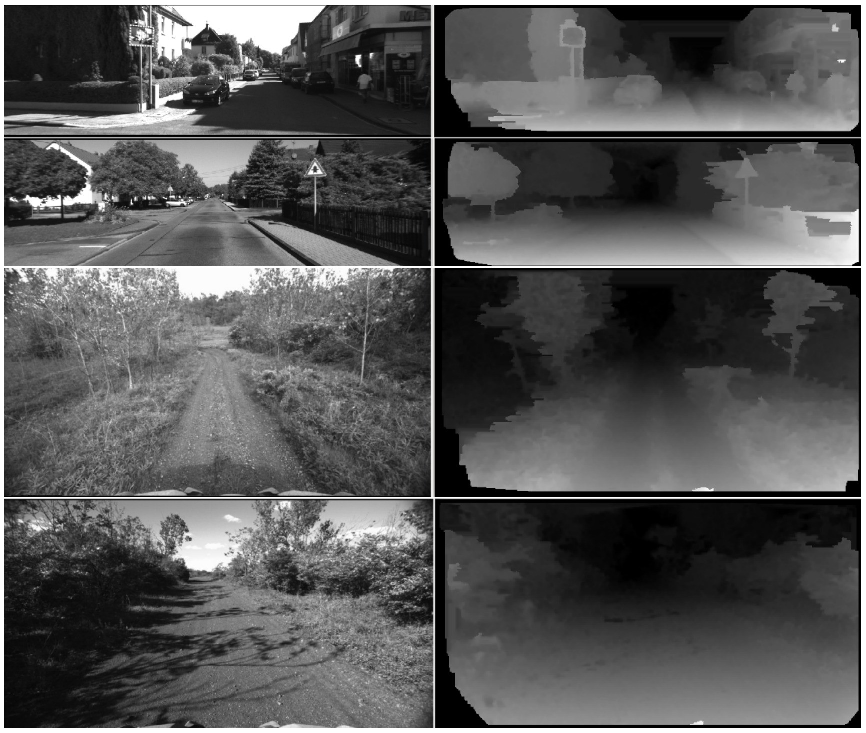


Figure 5.3: Examples of stereo matching results on KITTI dataset (upper two rows) and our off-road dataset (3rd and 4th rows). Left column shows the rectified left image of the stereo pair, right column shows the registered disparity image from ELAS.

5.3 Octomap

In this section, we introduce the core module of occupancy mapping in our framework, the Octomap [4]. Octomap is an open-source framework that performs 3D occupancy mapping based on octrees[2].

An octree is a hierarchical data structure used to represent the 3D space as shown in Fig 5.4. Each node in an octree represents a cubic volume of space usually called a voxel. This volume is recursively subdivided into eight sub-volumes until a given minimum voxel size is reached. Hence, the minimum voxel size determines the resolution of the octree.

Essentially, Octomap performs the probabilistic occupancy estimation at each observed voxel to maintain updatability and to deal with sensor noise. To represent not only the occupied space, but also the free and unknown areas, Octomap explicitly models the free volumes in the tree using ray-casting in the sensor model for range measurements. Fig. 5.5 illustrates how Octomap models the occupied and free voxels through ray-casting. An example of 3D grid map built by Octomap is shown in Fig. 5.6 where blue volumes denote the occupied voxels while green ones represent the free voxels.

[2]https://Octomap.github.io/

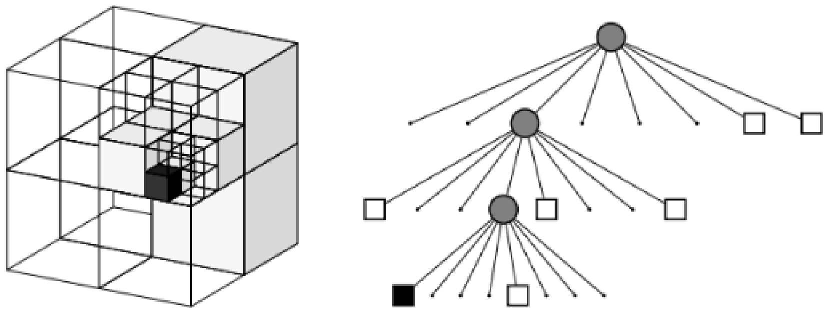


Figure 5.4: Example of an octree [4]. The volumetric model is shown on the *left* and the corresponding octree representation is on the *right*. The white squares on the right denotes the free voxels (shaded) and the black square denotes the occupied cell on the left.

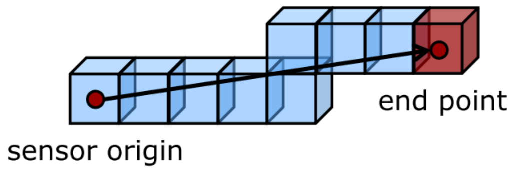


Figure 5.5: Ray-casting from sensor origin to end point, the last voxel is marked as occupied, all the other voxels along the ray are marked as free.

5.3.1 Occupancy Probability Estimation

Octomap integrates the sensor measurements using occupancy grid mapping method which takes a Bayes filtering approach. The probability of a leaf node n to be occupied given the sensor data $z_{1:t}$ is estimated according to

$$P(n|z_{1:t}) = \left[1 + \frac{1 - P(n|z_t)}{P(n|z_t)} \frac{1 - P(n|z_{1:t-1})}{P(n|z_{1:t-1})} \frac{P(n)}{1 - P(n)}\right]^{-1} \tag{5.1}$$

The update of occupancy probability depends on the previous estimate $P(n|z_{1:t-1})$, the current measurement z_t and a prior probability $P(n)$. $P(n|z_t)$ denotes the probability of voxel n to be occupied given the measurement z_t. It represents the inverse sensor model and its value depends on the sensor that generates z_t. With the assumption of an uniform prior probability $P(n) = 0.5$ (unknown) and using the log-odds notation, Eq. 5.1 becomes

$$L(n|z_{1:t}) = L(n|z_{1:t-1}) + L(n|z_t) \tag{5.2}$$

with

$$L(n) = log\left[\frac{P(n)}{1 - P(n)}\right] \tag{5.3}$$

Instead of using Eq. 5.2 directly, Octomap updates the occupancy estimate using a clamping update policy [27]:

$$L(n|z_{1:t}) = max(min(L(n|z_{1:t-1}) + L(n|z_t), l_{max}), l_{min}) \tag{5.4}$$

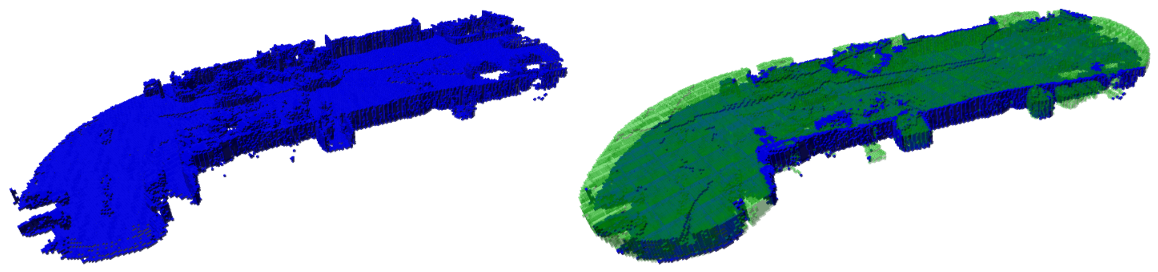


Figure 5.6: 3D map example of an urban street scene, blue: occupied volume, green: free volume

where l_{min} and l_{max} denote the lower and upper bound on the log-odds value. The modified update rule limits the number of updates needed to change the state of a voxel, which leads to two advantages: 1. The confidence in the map remains bounded and 2. The model can adapt to dynamic change in the environment quickly.

5.3.2 Sensor Model

For a distance sensor, the sensor model relates distance measurements to the probability an obstacle is present. It is used to update the occupancy grid map as explained in Sec. 5.3.1, where the current probability is combined with the previous belief about the occupancy of a cell.

Due to the measurement noise, the reliability of an occupancy grid map depends on the inverse sensor model. This model is responsible for representing the uncertainty in measurements, and for ensuring that the probabilities describing the occupancy grid map are correctly updated when new measurements are incorporated.

Octomap employs a beam-based inverse sensor model which assumes the endpoint of a measurement correspond to the obstacle surface and the line of sight between the sensor origin and the endpoint does not contain any obstacles. To determine which of the map cells need to be updated using ray-casting, Octomap implements a 3D variant Bresenham algorithm to approximate the beam in 3D [28]. Volumes along the beam are updated using the following inverse sensor model:

$$L(n|z_t) = \begin{cases} l_{occ}, & \text{if beam hits the volume.} \\ l_{free}, & \text{if beam passes through volume.} \end{cases} \tag{5.5}$$

We used log-odds values of $l_{occ} = 0.85$ and $l_{free} = -0.4$, corresponding to probability of 0.7 and 0.4 for occupied and free volumes, respectively. The clamping thresholds are set to $l_{min} = -2$ and $l_{max} = 3.5$, corresponding to probabilities of 0.12 and 0.97.

5.4 Experimental Results

We tested the proposed mapping framework on two different datasets, the KITTI dataset [21] and Yamaha dataset. The KITTI vision dataset contains 22 sequences of stereo images collected from an urban driving car. The stereo camera has a ~54 cm baseline and works at 10 fps frame rate with image resolution of 1392 $\times$ 512 pixels. We selected 2 sequences (00 and 07) from the training set, both of which contain loops.

The Yamaha dataset is collected by the Yamaha autonomous UTV from the off-road test field at Gascola, Penn Hills, PA. The on-board stereo camera has a $\sim$21 cm baseline recording at 30 fps with resolution of 1024 $\times$ 544 pixels. We tested the mapping at three off-road trails as shown in Fig. 3.3.

Since ORB-SLAM2 performs the global bundle adjustment (BA) after detecting a loop, it is expected to see difference between a map built with the trajectory before the loop closure and the other one created with the optimized trajectory after loop closure and global BA. As a result, in the following sections, at each sequence that contains a loop, we show the results of two 3D maps, one is built with the current tracked camera pose till loop closure, the other one is built with the BA-optimized trajectory after the loop closure.

The following sections show the results of both datasets. For consistency, we use the same set of parameters in the Octomap for both datasets which are shown in Table 5.1. Note that all the 3D occupancy maps are shown with color coded height.

5.4.1 KITTI Dataset

The 3D occupancy maps of KITTI sequences are shown in Fig. 5.7 and 5.8. The zoom-in views are added for details of the 3D map. Additionally, we put two close-up views from different viewpoints around the loop closure to show the effect of the global bundle adjustment.

Fig. 5.7 shows the partial map of the KITTI sequence 00 from the beginning to the first loop closure. We successfully mapped out all the area passed by the vehicle using single stereo camera. Due to the fact of using "front-facing" cameras on a forward moving vehicle, there are several unmapped areas (white holes) in the resulting map. Most of these areas are the regions which are visually blocked by the cars parked on the street and are not covered in the camera's field of view.

It can be seen from both Fig. 5.7(a) and 5.8(a) that there is a color mismatch at the end of the sequence before loop closure. The mismatch denotes the different heights at the same place, which indicates that the trajectory drifts in the vertical direction. Fig. 5.7(b) and 5.8(b) show the capability of this mapping framework in correcting the mapping errors from odometry drift with the help of visual-SLAM.

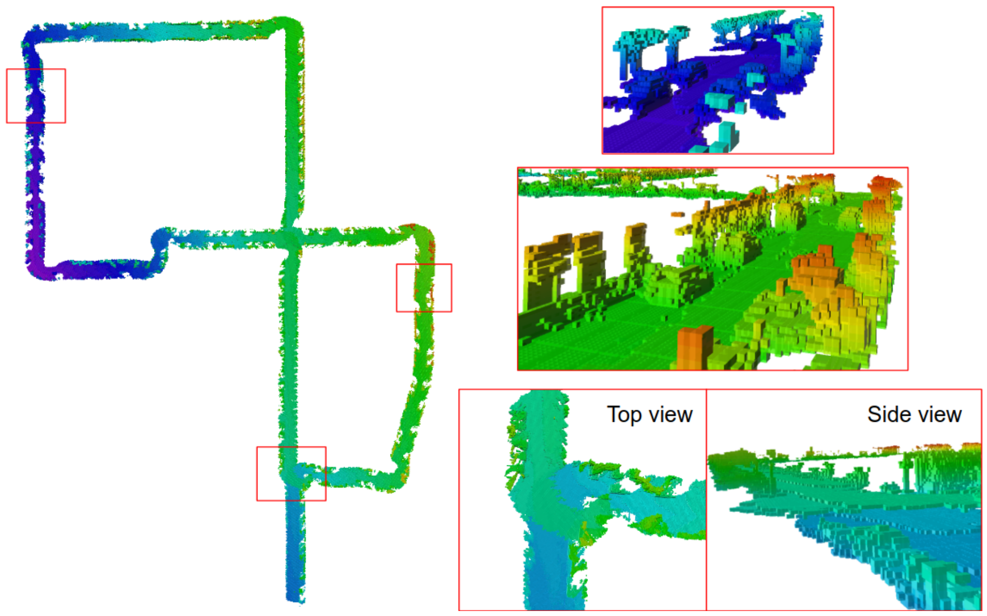
Table 5.1: Main parameters for Octomap

Parameter	Value	Description
Resolution	0.20 (m)	Leaf node voxel size for octree.
Max range	10.0 (m)	Maximum range for integrating point cloud measurements
Hit/Miss prob.	0.7/0.4	Probabilites for hit and miss in the sensor model
Min/Max clamping prob.	0.12/0.97	Minimum and maximum probability for clamping

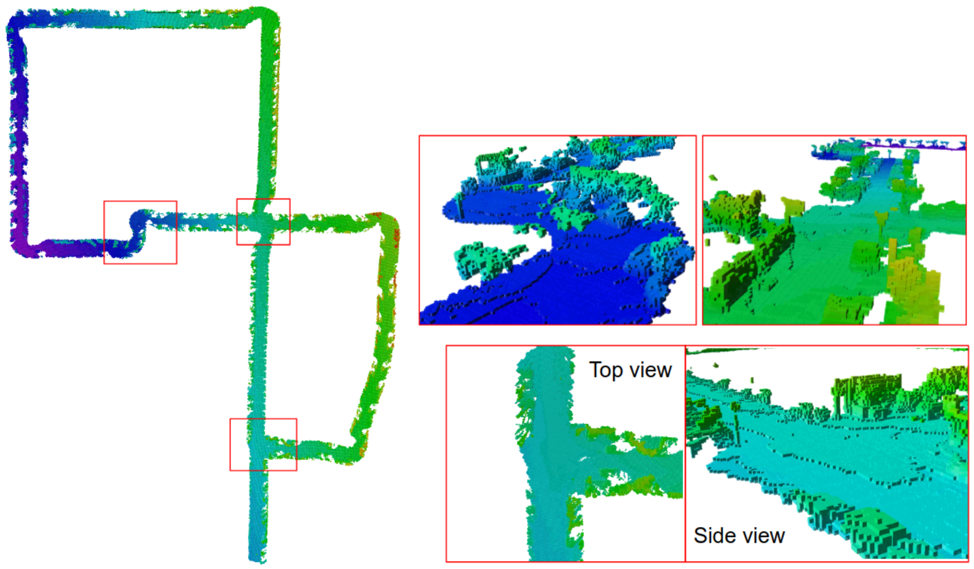
5.4.2 Yamaha Dataset

The mapping results of Yamaha off-road datasets are shown in Fig. 5.9 through 5.11. The mapping performs well in the off-road data both in the wide open areas as shown in Fig. 5.9 and 5.10 or the cluttered narrow trail with tall bush on the sides shown in Fig. 5.11.

Like the case in KITTI dataset, the drifting in pose results in discontinuity and misalignment in the map which can be corrected by the loop closure. There is a significant drift at the triangle loop as shown in Fig. 5.10. The corrected occupancy map provides a continuous and smooth reconstruction of the loop trail.

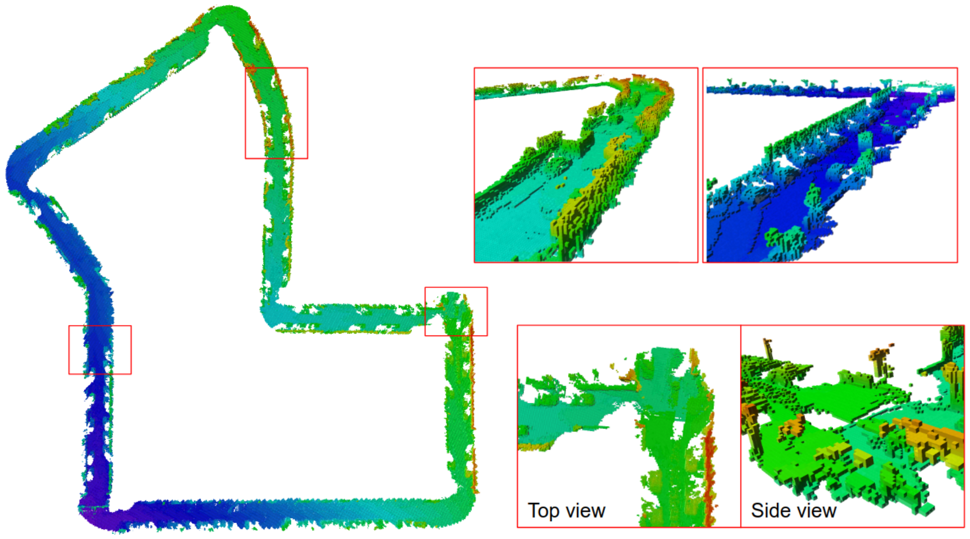


(a)

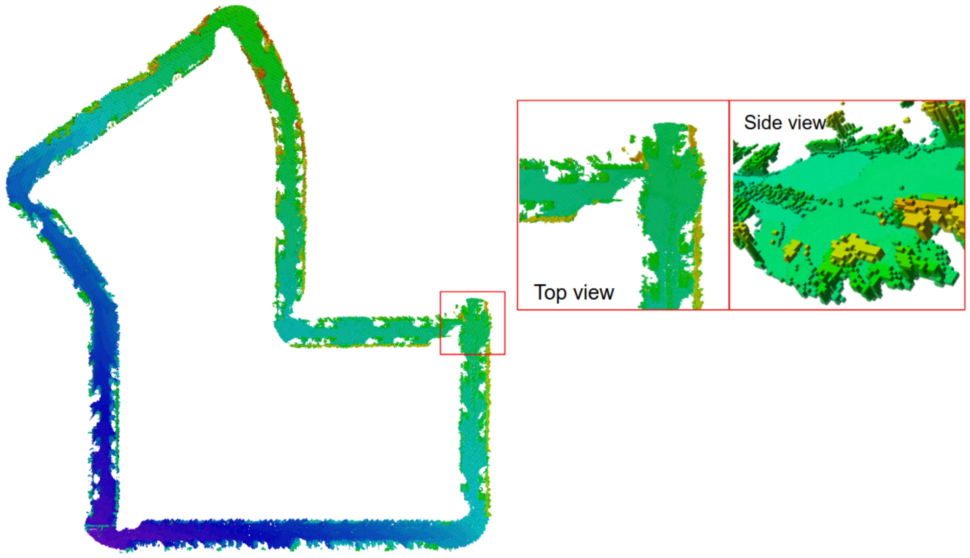


(b)

Figure 5.7: Results of KITTI sequence 00 (a) before loop closure (b) after loop closure.

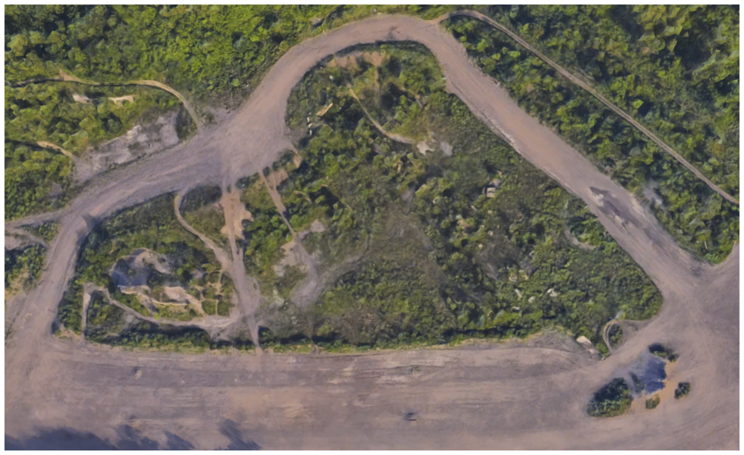


(a)

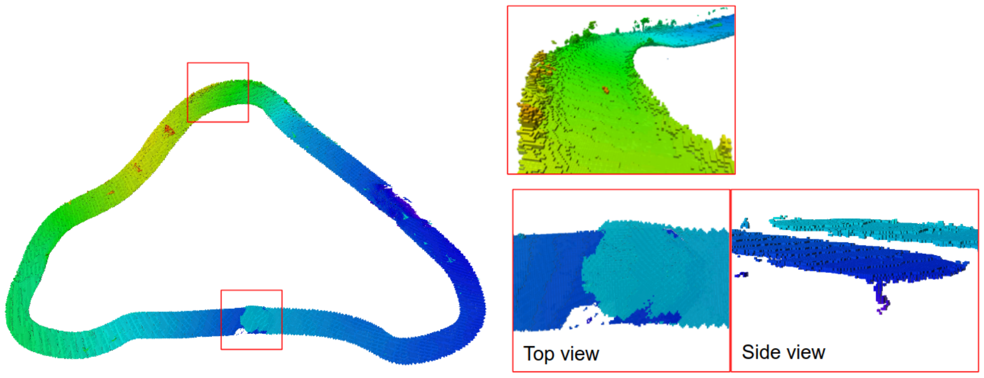


(b)

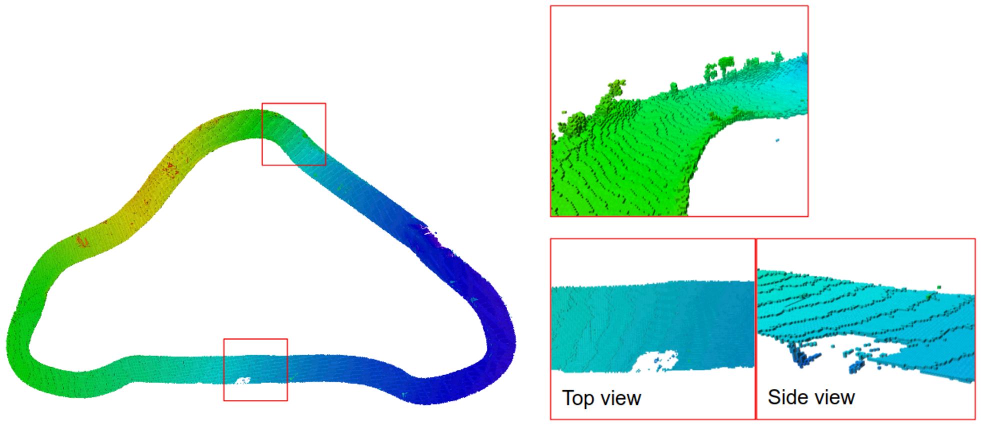
Figure 5.8: Results of KITTI sequence 07 (a) before loop closure (b) after loop closure.



(a)

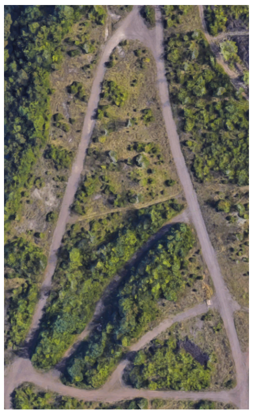


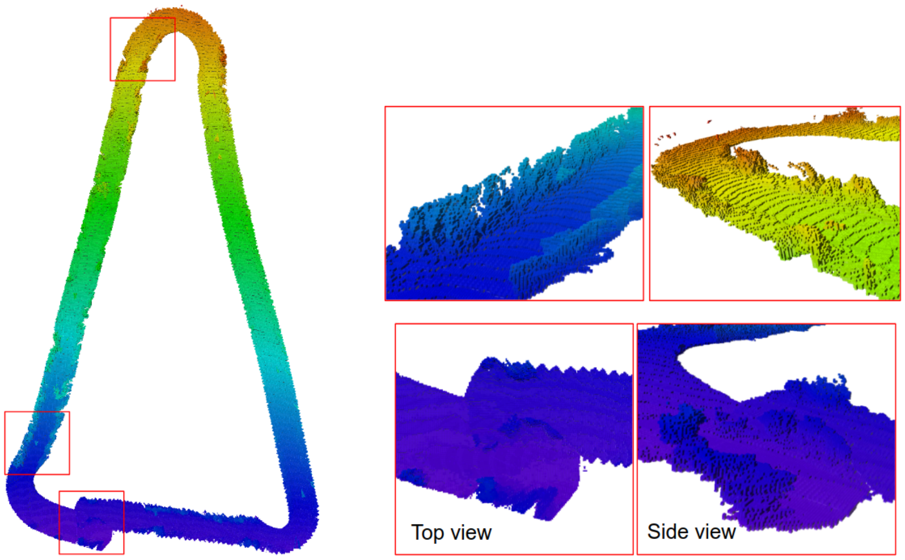
(b)



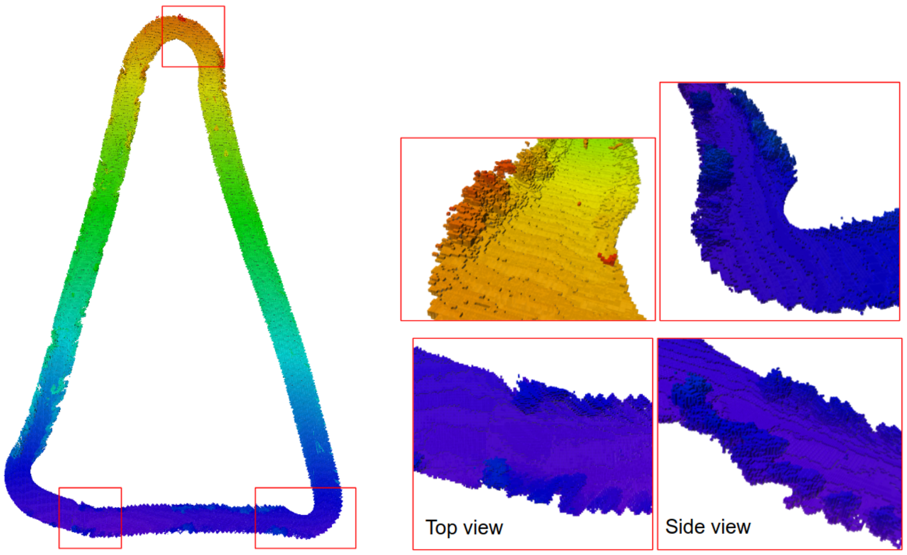
(c)

Figure 5.9: Results of Yamaha dataset at slope loop (a) Google Map image (b) before loop closure (c) after loop closure.


(a)

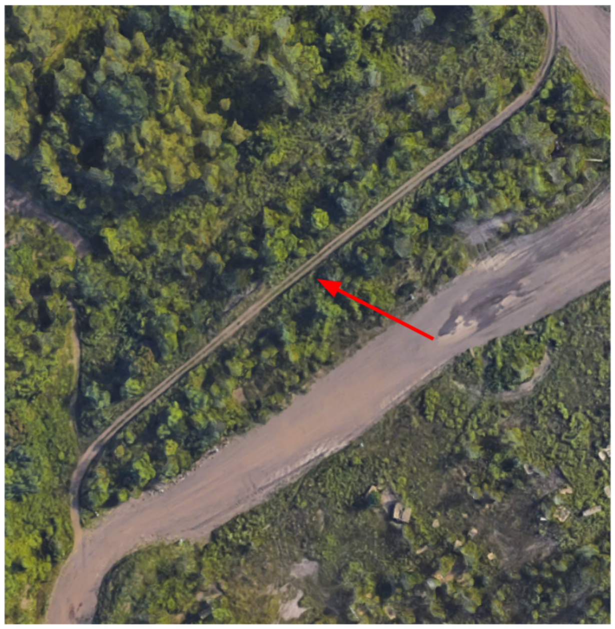


(b)



(c)

Figure 5.10: Results of Yamaha dataset at triangle loop (a) Google Map image (b) before loop closure (c) after loop closure.


(a)

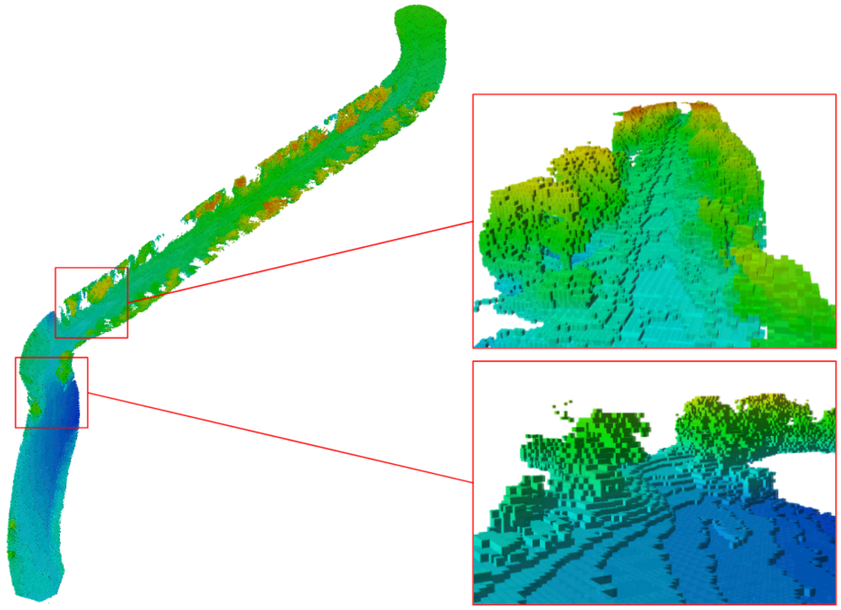

(b)

Figure 5.11: Results of Yamaha dataset at narrow trail (a) Google Map image (b) 3D occupancy map

Chapter 6

Conclusion and Future Work

This work investigates the localization and mapping for autonomous off-road driving. To achieve real-time localization of an autonomous off-road vehicle, we employ a generalized EKF framework to perform multi-sensor fusion and estimate the vehicle state locally and globally. Two instances of EKF are used to fuse different set of on-board sensors for local and global state estimation. We show the experimental results of local and global EKF running on the vehicle which drives in the off-road trails. The evaluation results show that the EKF can provide accurate and reliable pose estimate, both in the local and the global reference frame with high frequency update and low drift. The state estimation framework provides an compact and off-the-shelf solution for reliable localization and navigation in off-road driving.

We also explore the SLAM approach in off-road driving by investigating the performance of a state-of-the-art feature-based visual-SLAM system, ORB-SLAM2. We test the stereo ORB-SLAM2 on our own off-road dataset and show the evaluation results in both relative and absolute error metrics along with the EKF results. The results show the weakness and sensitivity of visual-SLAM system applied in the off-road environment due to the factors such as unpredictable lighting condition and the lack of salient features in unstructured environment. Nevertheless, the ability to perform real-time SLAM using single stereo camera is still impressive. The visual SLAM gives a continuous and globally consistent pose in general off-road driving cases and provides the alternative pose source for local mapping application.

Lastly, we propose a 3D occupancy mapping framework using stereo camera. We integrate stereo matching algorithm, visual-SLAM system and the occupancy mapping module together to achieve efficient dense 3D mapping with accurate camera trajectory estimated by the visual-SLAM. We show both mapping results on the famous KITTI benchmark dataset and also on our off-road dataset. The results show that with the help of loop closure correction, the resulting map do not contain drift and discontinuity, and the scene details are well mapped by the 3D voxels of occupancy grid. In addition, the output files of the octree maps are with extreme compact size which is ideal for various applications such as large-scale motion planning and autonomous navigation in real time.

For future work, we are aiming for the following research directions:

1. **Full integration of odometry, IMU, GPS and visual SLAM**
   The results from the vision-based occupancy mapping demonstrates a proof-of-concept framework using single stereo camera. However, the lack of absolute positioning limits

the use of this kind of mapping framework for global registration and navigation. The full integration with other motion tracking sensors is expected to bring more accuracy, robustness and functionality to both the localization and mapping capability. The full integration through graph-based optimization is able to provide better and more consistent pose estimation with smooth optimized trajectory over time [29], which essentially resolves the predicament of separating local and global localization due to the drifting and discrete jumps respectively.

2. **Real-time mapping system through parallelization with GPU implementation**
   The current implementation of the integrated mapping system demands significant processing time and memory consumption and it is not optimized for real-time operation yet. The present bottleneck is the ray-casting process in the mapping module. To achieve the operation with real frame rate (>30 fps), parallelization in computation is one way to boost the system processing. GPU implementation will be necessary for such architecture.
3. **Octomap-based planning**
   The ultimate goal of all the work presented in this thesis is to help achieve the fully autonomous driving in off-road. To this end, we are looking forward to the real world integration of this perception capability. The real use case of Octomap-based planning can show the power and also the limits of this framework. On the other hand, with planning in the loop, there is also potential to accomplish autonomous exploration for a driving vehicle based on the unmapped area.

Appendix A

State Estimation Details

A.1 Motion Model

The state transition function f is a 3D kinematic model with constant velocity assumption which can be expressed as follows:

$$\begin{aligned} &\boldsymbol{x}_k = f(\boldsymbol{x}_{k-1}) \\ \rightarrow &\begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}_k = \begin{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + R(\phi,\theta,\psi) \cdot \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \cdot \Delta t \\ \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} + R(\phi,\theta,\psi) \cdot \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \cdot \Delta t \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}_{k-1} \end{aligned} \tag{A.1}$$

where Δt is the elapsed time since last update and R denotes the 3D rotation matrix of the corresponding Euler angels:

$$R(\phi,\theta,\psi) = \begin{bmatrix} c_\theta c_\psi & c_\psi s_\theta s_\phi - c_\phi s_\psi & s_\phi s_\psi + c_\phi c_\theta s_\psi \\ c_\theta s_\psi & c_\phi c_\psi + s_\phi s_\theta s_\psi & c_\phi s_\theta s_\psi - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \tag{A.2}$$

with s and c representing sine and cosine functions.

A.2 Sensor Model

As mentioned in Sec. 3.1, the generalized EKF framework assumes that each sensor gives direct measurements of the state variable:

$$\boldsymbol{z} = h(\boldsymbol{x}) = \boldsymbol{H}\boldsymbol{x} \tag{A.3}$$

where $\boldsymbol{H}$ is an m by 12 matrix of rank m with only nonzeros entries (ones) existing in the columns of the measured variables. Here list the $\boldsymbol{H}^i$ for sensor i used in the local and global EKF with $\mathbf{0}_{m\times n}$ and $\boldsymbol{I}_{m\times m}$ denoting the $m \times n$ zero matrix and $m \times m$ identity matrix respectively.

Local EKF:

$$\boldsymbol{H}_L^{wheel} = \begin{bmatrix} \mathbf{0}_{3\times6} & \boldsymbol{I}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \tag{A.4}$$

$$\boldsymbol{H}_L^{imu} = \begin{bmatrix} \mathbf{0}_{3\times3} & \boldsymbol{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \boldsymbol{I}_{3\times3} \end{bmatrix} \tag{A.5}$$

Global EKF:

$$\boldsymbol{H}_G^{wheel} = \begin{bmatrix} \mathbf{0}_{3\times6} & \boldsymbol{I}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \tag{A.6}$$

$$\boldsymbol{H}_G^{imu} = \begin{bmatrix} \mathbf{0}_{2\times3} & \boldsymbol{I}_{2\times2} & \mathbf{0}_{2\times4} & \mathbf{0}_{2\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times2} & \mathbf{0}_{3\times4} & \boldsymbol{I}_{3\times3} \end{bmatrix} \tag{A.7}$$

$$\boldsymbol{H}_G^{gps} = \begin{bmatrix} \boldsymbol{I}_{3\times3} & \mathbf{0}_{3\times9} \end{bmatrix} \tag{A.8}$$

A.3 Covariances

Here list the nominal covariance values used in the local and global EKF. The $diag$ denotes the diagonal matrix of entries with the values in the parentheses.

A.3.1 Local EKF

Initial estimates covariance:

$$\boldsymbol{\Sigma}_0 = diag(1.0e^{-9}, ..., 1.0e^{-9}) \quad (\boldsymbol{\Sigma}_0 \text{ is 12 by 12}) \tag{A.9}$$

Process noise covariance:

$$\boldsymbol{Q} = diag(0.05, 0.05, 0.06, 0.03, 0.03, 0.06, 0.025, 0.025, 0.04, 0.01, 0.01, 0.02) \tag{A.10}$$

Measurement noise covariance:

$$\boldsymbol{R}^{wheel} = diag(0, 0.05, 0) \tag{A.11}$$

$$\boldsymbol{R}^{imu} = diag(0.0175, 0.0175, 0.1571, 4.36e^{-4}, 4.36e^{-4}, 4.36e^{-4}) \tag{A.12}$$

A.3.2 Global EKF

Initial estimates covariance:

$$\mathbf{\Sigma}_0 = diag(1.0e^{-9}, ..., 1.0e^{-9}) \quad (\mathbf{\Sigma}_0 \text{ is 12 by 12}) \tag{A.13}$$

Process noise covariance:

$$\boldsymbol{Q} = diag(10.0, 10.0, 20.0, 3.0, 3.0, 0.6, 0.025, 0.025, 0.04, 0.01, 0.01, 0.02) \tag{A.14}$$

Measurement noise covariance:

$$\boldsymbol{R}^{wheel} = diag(0, 0.05, 0) \tag{A.15}$$

$$\boldsymbol{R}^{imu} = diag(0.0175, 0.0175, 4.36e^{-4}, 4.36e^{-4}, 4.36e^{-4}) \tag{A.16}$$

$\boldsymbol{R}^{gps}$: time-varying with respect to the status of satellites

Bibliography

[1] Google. Google Maps, 2017. URL `https://maps.google.com/`. (document), 3.3

[2] Raúl Mur-Artal and Juan D Tardós. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics*, 2017. (document), 2.3, 4, 4.1

[3] Michael Bleyer and Christian Breiteneder. Stereo Matching: State-of-the-Art and Research Challenges. In *Advanced Topics in Computer Vision*, pages 143–179. Springer, 2013. (document), 5.2

[4] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An Efficient Probabilistic 3D Mapping Framework based on Octrees. *Autonomous Robots*, 34(3):189–206, 2013. (document), 5.3, 5.4

[5] Rudolph Emil Kalman et al. A New Approach to Linear Filtering and Prediction Problems. *Journal of basic Engineering*, 82(1):35–45, 1960. 2.2, 3.1

[6] David C Moore, Albert S Huang, Matthew Walter, Edwin Olson, Luke Fletcher, John Leonard, and Seth Teller. Simultaneous Local and Global State Estimation for Robotic Navigation. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3794–3799. IEEE, 2009. 2.2

[7] Thomas Moore and Daniel Stouch. A Generalized Extended Kalman Filter Implementation for the Robot Operating System. In *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer, July 2014. 2.2, 3.1

[8] Taihú Pire, Thomas Fischer, Javier Civera, Pablo De Cristóforis, and Julio Jacobo Berlles. Stereo Parallel Tracking and Mapping for Robot Localization. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 1373–1378. IEEE, 2015. 2.3

[9] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-Scale Direct Monocular SLAM. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014. 2.3

[10] Jakob Engel, Jörg Stückler, and Daniel Cremers. Large-Scale Direct SLAM with Stereo Cameras. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 1935–1942. IEEE, 2015. 2.3

[11] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, 31(5):

1147–1163, 2015. 2.3

[12] Andreas Geiger, Julius Ziegler, and Christoph Stiller. Stereoscan: Dense 3D Reconstruction in Real-Time. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 963–968. IEEE, 2011. 2.4, 5.1

[13] Pablo F Alcantarilla, Chris Beall, and Frank Dellaert. Large-Scale Dense 3D Reconstruction from Stereo Imagery. Georgia Institute of Technology, 2013. 2.4, 5.1

[14] Andreas Nüchter, Kai Lingemann, Joachim Hertzberg, and Hartmut Surmann. 6D SLAM-3D Mapping Outdoor Environments. *Journal of Field Robotics*, 24(8-9):699–722, 2007. 2.4

[15] H Moravec. Robot Spatial Perceptionby Stereoscopic Vision and 3D Evidence Grids. *Perception*, 1996. 2.4, 5.1

[16] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT press, 2005. 3.1

[17] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. ROS: an Open-Source Robot Operating System. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, 2009. 3.1

[18] Gerald J Bierman and Catherine L Thornton. Numerical Comparison of Kalman Filter Algorithms: Orbit Determination Case Study. *Automatica*, 13(1):23–35, 1977. 3.1

[19] ROS.org. ROS REP 105, 2010. URL `http://www.ros.org/reps/rep-0105.html`. 3.2

[20] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A Benchmark for the Evaluation of RGB-D SLAM Systems. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 573–580. IEEE, 2012. 3.3

[21] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are We Ready for Autonomous Driving? the KITTI Vision Benchmark Suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE, 2012. 3.3, 5, 5.4

[22] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An Efficient Alternative to SIFT or SURF. In *Computer Vision (ICCV), 2011 IEEE international conference on*, pages 2564–2571. IEEE, 2011. 4.1

[23] Yuval Roth-Tabak and Ramesh Jain. Building an Environment Model using Depth Information. *Computer*, 22(6):85–90, 1989. 5.1

[24] Sebastian Thrun. Learning Occupancy Grid Maps with Forward Sensor Models. *Autonomous robots*, 15(2):111–127, 2003. 5.1

[25] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge university press, 2003. 5.2

[26] Andreas Geiger, Martin Roser, and Raquel Urtasun. Efficient Large-Scale Stereo Matching. In *Asian conference on computer vision*, pages 25–38. Springer, 2010. 5.2.1

[27] Manuel Yguel, Olivier Aycard, and Christian Laugier. Update Policy of Dense Maps:

Efficient Algorithms and Sparse Representation. In *Field and Service Robotics*, pages 23–33. Springer, 2008. 5.3.1

[28] John Amanatides, Andrew Woo, et al. A Fast Voxel Traversal Algorithm for Ray Tracing. In *Eurographics*, volume 87, pages 3–10, 1987. 5.3.2

[29] Hauke Strasdat, José MM Montiel, and Andrew J Davison. Visual SLAM: Why Filter? *Image and Vision Computing*, 30(2):65–77, 2012. 1

[30] Konstantin Schauwecker and Andreas Zell. Robust and Efficient Volumetric Occupancy Mapping with an Application to Stereo Vision. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 6102–6107. IEEE, 2014.

[31] Franz Andert. Drawing Stereo Disparity Images into Occupancy Grids: Measurement Model and Fast Implementation. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 5191–5197. IEEE, 2009.