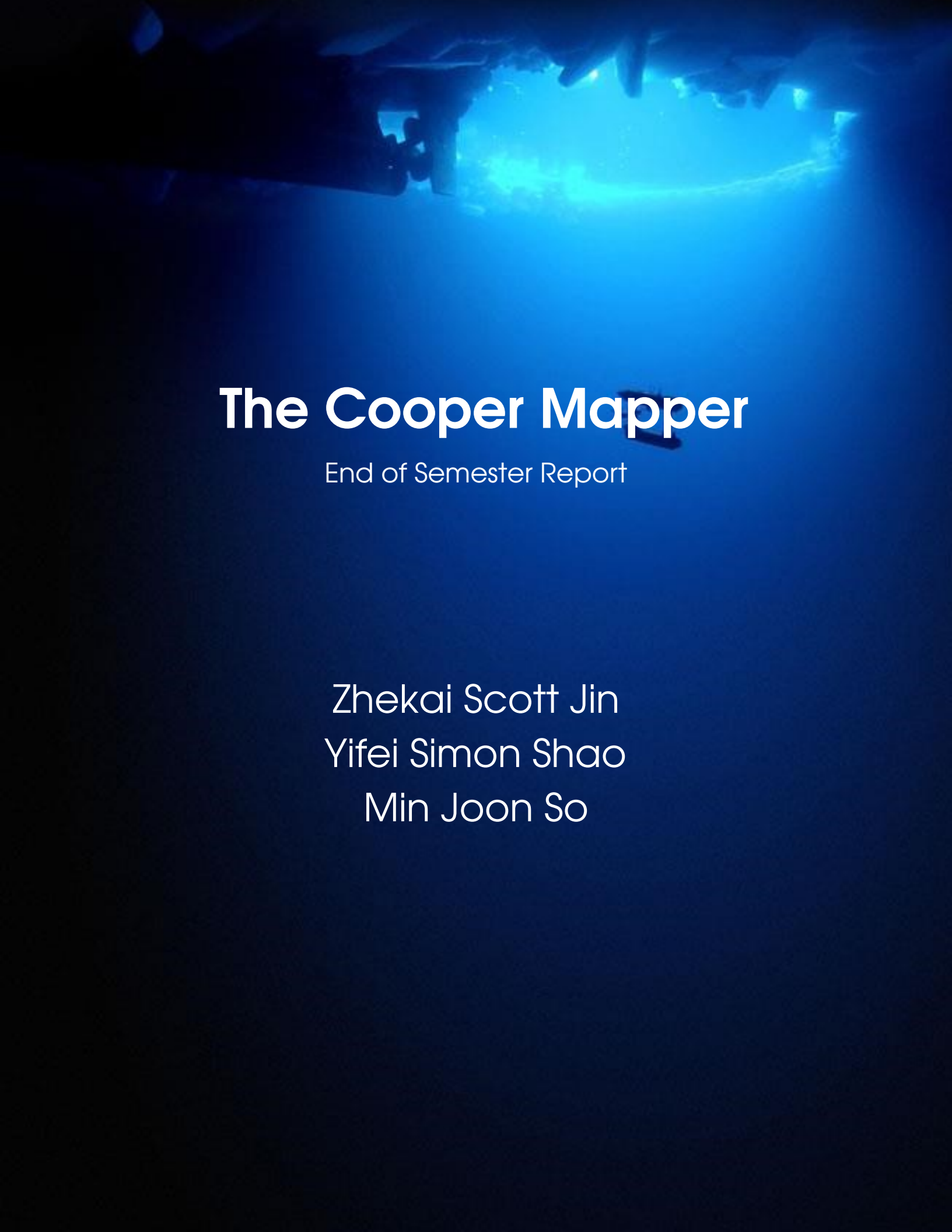
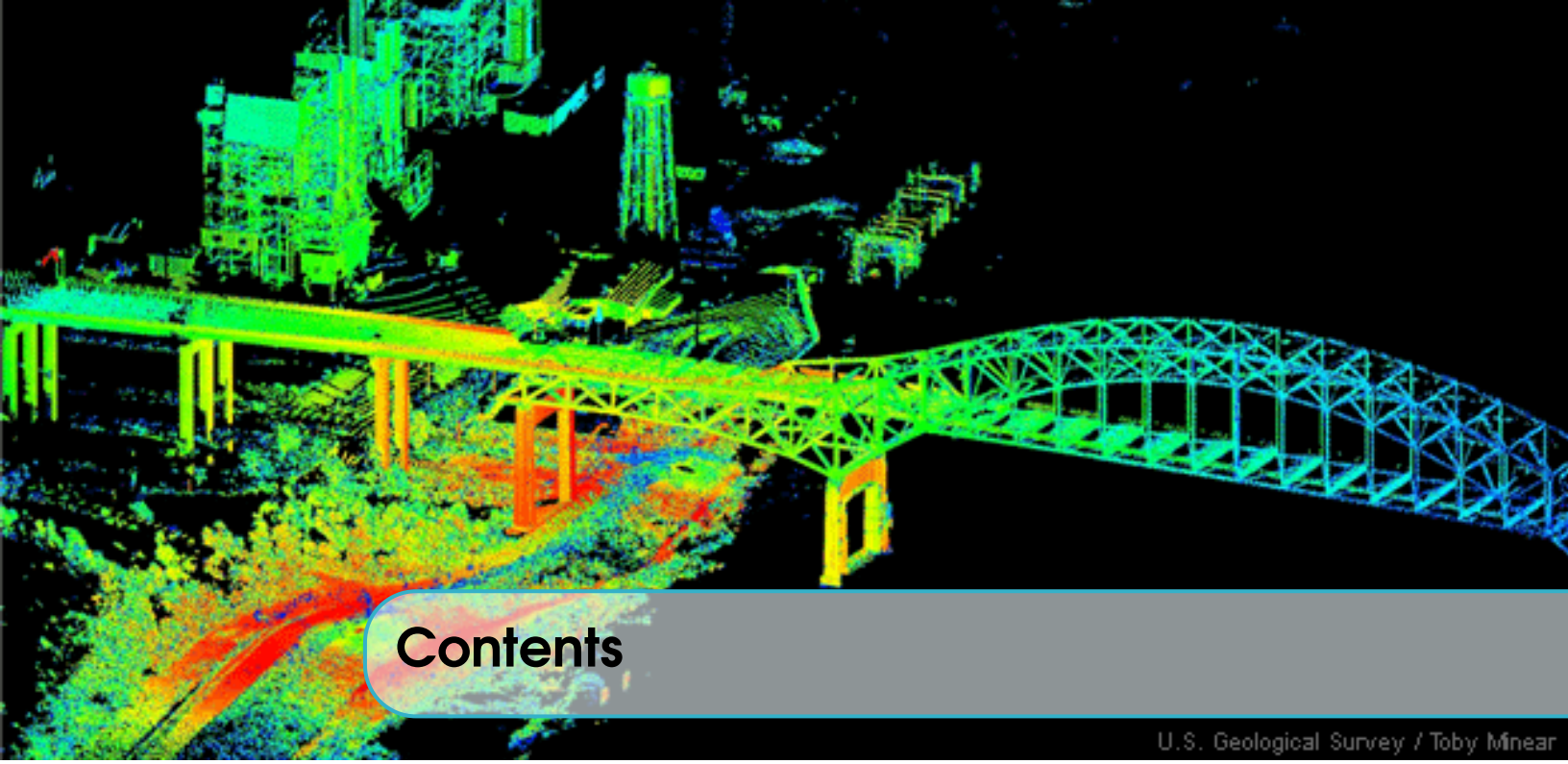


The Cooper Mapper

End of Semester Report

Zhekai Scott Jin
Yifei Simon Shao
Min Joon So

SENIOR PROJECT RESEARCH, THE COOPER UNION FOR THE ADVANCEMENT OF SCIENCE AND ART
This research was done under the supervision of Dr. Carl Sable, Dr. Dirk Luchtenburg, Dr. Neveen Shlayan, Dr. Sam Keene in the 2018- 2019 acedemic year at Cooper Union.
First release, December 2018



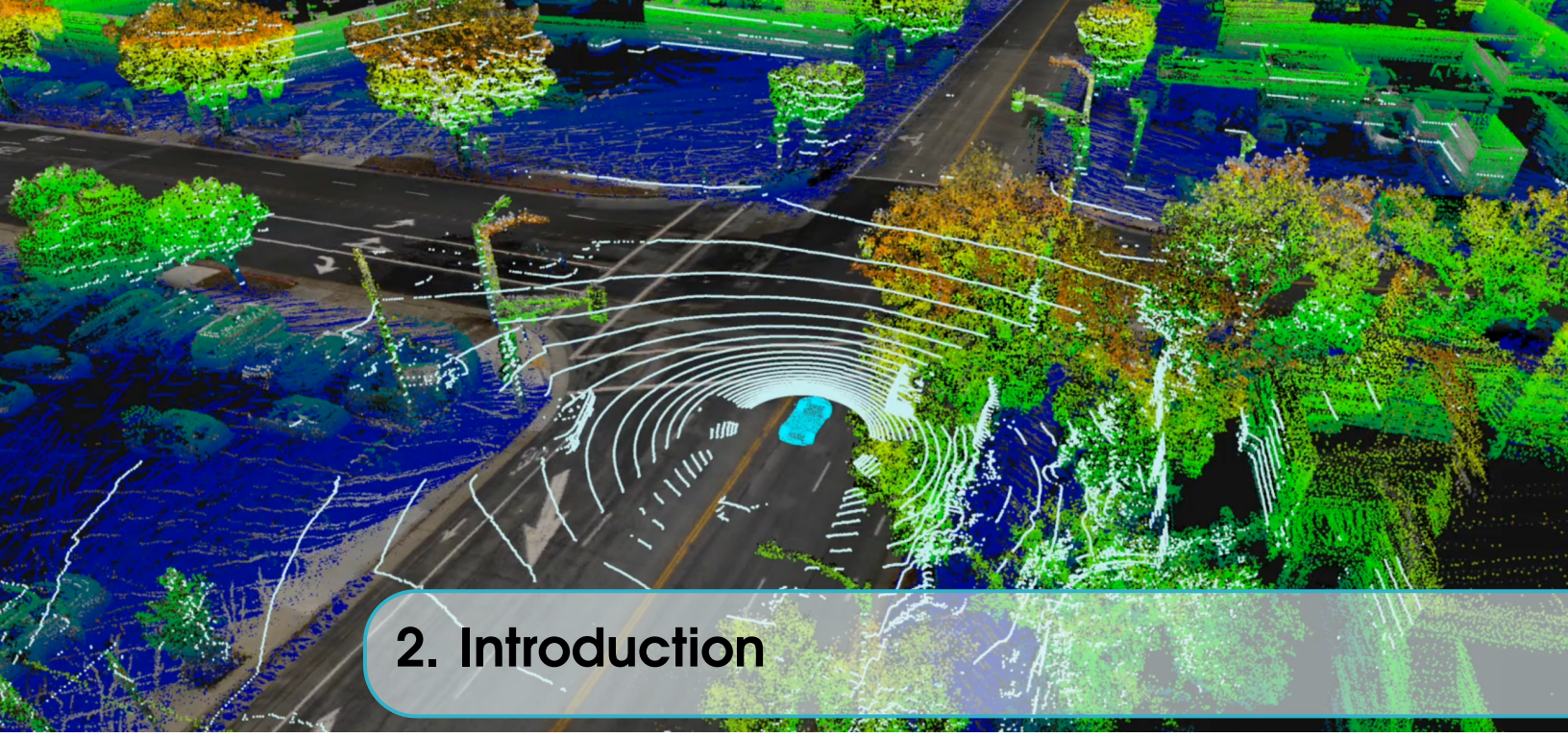
U.S. Geological Survey / Toby Minear

Contents

1. Abstract

Nowadays, autonomous driving technologies are advancing at a fast speed. To operate and to perform tasks autonomously in unstructured environments like highway and crowded city blocks, mobile robots have to maintain localization and mapping parameters to make sense of their environment. One of the challenges in such tasks is the inaccuracies and uncertainties in sensor measurements, and various techniques have been prompted to handle such noises. Multisensor data fusion has become a dominant paradigm due to its potential advantages like uncertainty reduction, accuracy improvement, cost reduction. One of the main challenges robot facing is to create the perception of the robot without any prior information about its initial position or its surrounding, known as simultaneous localization and mapping (SLAM) problem. With the GPS reception denied, indoor SLAM problem becomes significantly difficult. Our work aims to come up with a general realtime multisensor data fusion algorithm for combining visual data and Lidar data to improve indoor SLAM. We have built a custom differential-based vehicle and implemented Lidar SLAM and Visual SLAM independently. We are at the stage of combining each module and applying various multisensor data fusion algorithms. We will evaluate our results with KITTI benchmark for motion estimation accuracy and test the robustness of the algorithm by moving the sensor suite at high speed under significant ambient lighting changes.

Keywords: sensor fusion, information fusion, inference, imputation, fusion model, Kalman Filter, inference, autonomous driving, robotics.



2. Introduction

Today, robotics applications have moved from the manufacturing industry to more unpredictable and more complex environments. Due to the high demand for service robots, autonomous intelligent mobile robots are replacing traditional industrial robots to some extent. Such autonomous robots can adapt to the dynamics of the environment and perform decision makings to perform their assigned tasks. Applications such as space exploration, floor cleaning, mowing lawns, and material transportation, etc. are suitable for such robots. To achieve autonomy in such tasks, a robot has to perform functions including sensor-based exploration, motion planning, localization, and mapping. [6, 78, 2, 31] A literature study shows that intelligent autonomous robots are able to deal with uncertainties in such a complex environment in an independent fashion [32, 12, 32, 3, 70] and a fully autonomous robot can work for an extended period without any human intervention based on the gained information about its surrounding and constantly adapt to it. [45, 39, 51, 48] For example, an autonomous robot 'URMAD' assists the patients in hospitals and an autonomous mobile robot 'MOVAID' is in service to assist the disabled and elderly people [20]. Robots like 'ABIO' can perform self-docking to charge their batteries [14]. Autonomous robots in the service industry are in high demand for laborious jobs like domestic chores, laundry handling, cleaning and attending elderly persons, etc. [5, 75, 58] And such tasks are often performed in an indoor environment. Indoor operation rules out the use of GPS to bound the localization error, thus making the localization

particularly difficult. [13] Multisensor data fusion is a common technique in recent years to solve such a problem and is primarily used in tasks like Simultaneous Localization and Mapping, which can perform mapping and localization in the absence of GPS information.

While SLAM with Multisensor data fusion has enormous implication for indoor mobile robots, multisensor data fusion has become a paradigm to solve problems concerning how to combine or fuse data from multiple sources to support various advanced tasks in the autonomous vehicle such as navigation and decision-making. [6, 11, 44] Autonomous vehicles are receiving considerable attention due to their great potentials for enhancing safety and improving the throughput of transportation systems. [52, 42, 16] However, an autonomous driving system is a complex system. It leverages multiple inputs from both in-car sensors and external sensors to make informed decisions. These sensors often generate their measurements in their own units and data representations as they operate. This fact leads to a challenging topic regarding multiple sensors fusion. As raw input, Lidar would generate its measurement as a point cloud, typically represented as an array of [(x,y,z), ...], while a camera would generate an array of tuples of RGB measurements with different albedo values: [(R, G, B), ...], preprocessing has to be performed to allow the system to make sense of the data coming in different modalities. As seen in figure 2.1, an autonomous car must incorporate different kinds of sensors to complement each other sensor's weaknesses and thus create a combined effect. Such sensors include Lidar, camera, ultrasonic sensor, radar and odometry sensor. More detailed descriptions of each sensor will be discussed in the Background section, but it is important to note the difficulties lying in utilizing each sensor's strength and prevent its weaknesses from adversely affecting the system. And it is extremely hard to for a intelligent system to know when and how to update the system accurately, and which source of information should it use to do so.

Another aspect of the intelligent mobile system is embedded hardware. Embedded hardware in each sensor has strong real-time processing capability. For a camera operating at 60 fps, for example, there is a 16 ms interval between each frame. Even though the sensors can collect data fast enough, the computer used to analyze the data from the camera may not be able to process all the data in the 16 ms interval before the next frame of data coming in. When there are multiple sensors (Lidar, camera, radar, etc.), the realtime data processing becomes extremely hard, and the

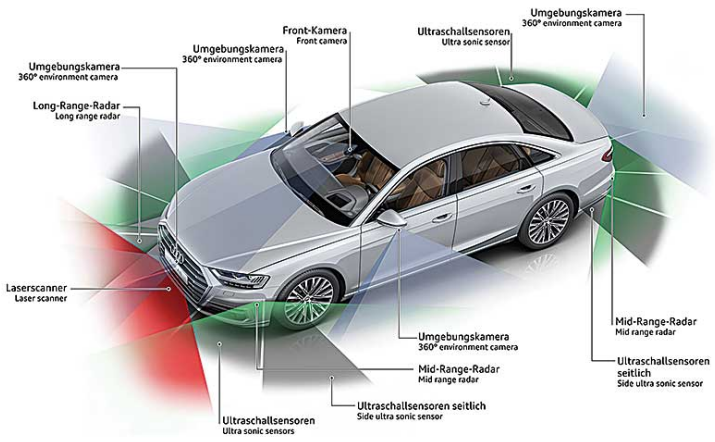


Figure 2.1: An Autonomous Automobile With Different Sensors

system faces challenges like how often to update the information contained in the system (whether this information is necessary to feed into the system), and how to decide when this information becomes outdated and can be discarded. An ill-designed system is subject to dangers like missing traffic signals, and yielding severe consequences. Therefore, a suitable computing platform and a robust algorithm that enable large-scale real-time processing, allow in-time notification to the drivers, and perform intelligent decision for them is of crucial importance to the reliability and continuous operation of an autonomous car. At the same time, to ensure reliability and comfort-ability, the computing platform has to have reasonably low power consumption, efficient heat dissipation, and proper physical size.

A couple of years ago, a Tesla car was on Autopilot, and it crashed into a white truck in the middle of the road. At the time, Tesla vehicles mainly used visual input along with Radar to perceive the environment. The bright reflection from the truck caused a specularity in the perception system and blinded the driver. This accident took the driver's life. The robustness of Tesla's intelligent mobile system has been challenged in such a scenario where the vehicle is moving at high speed, and the environment has a substantial light change. If Lidar system is installed and its information properly handled, such tragedy may be avoided.

To better understand and formulate the problem regarding to handling data in multiple modalities, and to come up with better sensor fusion algorithm, we are building a research vehicle for sensor

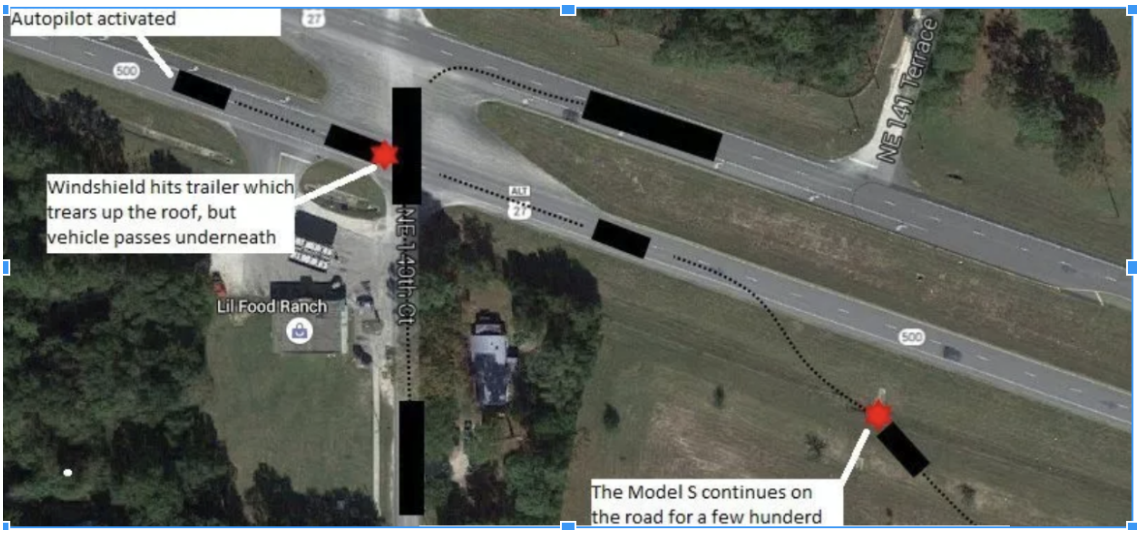


Figure 2.2: Tesla Accident

fusion, the Cooper Mapper. Mapping can be a great benchmark mechanism for understanding how well a robot understands its surrounding. And we believe a better fusion algorithm will improve the accuracy of the map and lead to better effect in other tasks like navigation and planning for the autonomous vehicles. The goal is to compare an accurate map created by the sensor fusion algorithm to the actual floor plan to evaluate the performance of new algorithms. Further, the accuracy of motion estimation will be evaluated by the mapping result. The robustness of this algorithm will then be tested by moving the sensor suite at high speed under significant ambient lighting changes.

The rest of this report is organized as follows: in Section 3 we give a thorough background ranging from SLAM, to common sensors utilized in an autonomous vehicle, to multisensor data fusion, along with the major related contributions in those fields that have been made so far. The project status and description are described in Section 4. Section 5 provides a Gantt chart report of where we are and the future work. And in Section 6, we present the concluding summary for this report.

3. Background

In this section, we provide the reader with the background information from the SLAM problem, to the common sensors used in autonomous driving vehicles along with their characteristics, to the multisensor data fusion problem. In the end, we present the multi-robot SLAM problem, where sensor fusion plays a considerable role.

3.1 SLAM

SLAM is the process by which a mobile robot can build a map of an environment and at the same time use this map to compute its own location. [26] In other words, it comprises the simultaneous estimation of the state of a robot equipped with on-board sensors, and the construction of a model (the map) of the environment that the sensors are perceiving. In simple instances, the robot state is described by its pose (position and orientation), although other quantities may be included in the state, such as robot velocity, sensor biases, and calibration parameters. The map, on the other hand, is a representation of aspects of interest (e.g., the position of landmarks, obstacles) describing the environment in which the robot operates. [13]

There are many reasons why a map is needed. First, a map may be in need to support other tasks like informed path planning, intuitive visualization for operators. Second, the map allows limiting the error committed in estimating the state of the robot. [13] In the absence of a map, dead-

reckoning would quickly drift over time; on the other hand, using a map, e.g., a set of distinguishable landmarks, the robot can "reset" its localization error by re-visiting known areas (so-called loop closure). Therefore, SLAM finds applications in all scenarios in which a prior map is not available and needs to be built. [13]

SLAM has been formulated and solved as a theoretical problem in a number of different forms. [26] And it has also been implemented in a number of different domains from indoor robots, to outdoor, underwater and airborne systems. At a theoretical and conceptual level, SLAM can now be considered a solved problem. However, substantial issues remain in practically realizing more general SLAM solutions and notably in building and using perceptually rich maps as part of a SLAM algorithm. [26] In 1986 IEEE Robotics and Automation Conference, probabilistic SLAM problem was introduced as the consistent probabilistic mapping problem. The result of this conversation was a recognition that consistent probabilistic mapping was a fundamental problem in robotics with major conceptual and computational issues that needed to be addressed.
Later, work by Smith and Cheesman [72] and Durrant-Whyte [27] showed that there must be a high degree of correlation between estimates of the location of different landmarks in a map and that indeed these correlations would grow with successive observations. At the same time Ayache and Faugeras [7] were undertaking early work in visual navigation, Crowley [19] and Chatila and Laumond [15] in sonar-based navigation of mobile robots using Kalman filter-type algorithms. These two directions of research had a lot in common and showed that as a mobile robot moves through an unknown environment taking relative observations of landmarks, the estimates of these landmarks are all necessarily correlated with each other because of the common error in estimated vehicle location. This implied that a solution to SLAM problem needs a joint internal state keeping the information of the vehicle pose and every landmark position, which will be updated following each landmark observation. The memory and computation requirement to hold such a state vector put SLAM research in halt, until the realization that the combined mapping and localization problem, once formulated as a single estimation problem, was convergent. [26] From then on, rapid and exciting progress in solving the SLAM problem together with many compelling implementations of SLAM methods has been seen, such as Extended Kalman Filters, Rao-Blackwellised Particle Filters,

and maximum likelihood estimation. Moreover, researchers have delineated the basic challenges connected to efficiency and robust data association, which has become a hot research topic in the past decade. A detailed formulation of probabilistic SLAM problem could be found in [26] where the structures and implementation of probabilistic SLAM are also described. This question of "is SLAM solved?" is often asked within the robotics community, c.f. [30]. This question is difficult to answer because SLAM has become such a broad topic that the question is well posed only for a given robot/environment/performance combination. [13] Now, researchers are working hard in these fields to help improving SLAM:

- robust performance: A robust SLAM system operates with a low failure rate for an extended period in a broad set of environments.
- high-level understanding: Beyond basic geometry reconstruction, SLAM system aims to obtain a high-level understanding of the environment (semantic, physics, etc.).
- resource awareness: SLAM system should be suitable for a range of available sensing and computational sources with means to adjust the computational load for the corresponding resources.

3.2 Sensors

For mobile robot applications, fusion refers to any stage in the integration process where an actual combination of different sources of information takes place. [59] And the most obvious part where fusion processes take place in is the part where the system takes in sensors in different modalities. Mainstream autonomous car platforms use Radar, Lidar and camera as the three major components of perception. And in this section we give a quick overview of all the sensors related in a typical self-driving platform.

3.2.1 Camera

The camera is a typical type of passive sensors, which are the true observers of the environment. [68] In another word, a typical camera does not actively perceive its surrounding but passively gain information from the environment. It captures signals that are generated by other sources in the

environment.[1]

Figure 3.1: Picture of different sensors on a Domino's self-driving research vehicle

The camera transcribes the driver's vision. It is very often used to understand the environment with artificial intelligence by classifying roads, pedestrians, signs, etc.

3.2.2 Lidar

Lidar is a typical range measurement device. Since it is very precise, efficient and the output does not require much computation to process, it is nowadays extensively used in self-driving vehicles. However, they are costly though there is a sign of decreasing cost in the Lidar market. Problems with laser scanners are looking at certain surfaces including glass, where they can give terrible readings (data output). Also, laser scanners cannot be used underwater since the water disrupts the light and the range is drastically reduced.

3.2.3 Sonar

Sonar was used intensively some years ago. They are very cheap compared to laser scanners. Their measurements are not very good compared to laser scanners, and they often give bad readings. Where laser scanners have a single straight line of measurement emitted from the scanner with a width of as little as 0.25 degrees, a sonar can easily have beams up to 30 degrees in width. Underwater, though,

[1] Note that time of flight camera is an exception, being an active sensor.

they are the best choice and resemble the way dolphins navigate. The type used is often a Polaroid sonar. It was initially developed to measure the distance when taking pictures in Polaroid cameras. Sonar has been successfully used in [43].

3.2.4 Vision

Vision-based range finding has been found to be a computationally intensive process and error-prone due to sudden changes in light conditions [13]. In a dark room, a vision-only autonomous system would not work. In recent years, though, there have been some exciting advances within this field [68]. Using vision resembles the way humans look at the world and thus may be more intuitively appealing than laser or sonar has been prompted, and stereo camera are now extensively used in various applications like augmented reality, virtual reality, and autonomous robotics. Also, there is a lot more information in a picture compared to laser and sonar scans. This feature used to be the bottleneck, since all this data needed to be processed, but with advances in algorithms and computation power this is becoming less of a problem and can be considered as an advantage of vision system [66]. A successful autonomous robotic application used vision-based range finding has been done in [71].

3.2.5 Odometric sensors

IMU, GPS, and wheel encoders are sensors commonly used to estimate the odometry of autonomous mobile robots. However, IMU, though has a high frame rate, is subject to the drift of error due to a lack of ground truth value. A differential GPS can provide a high accuracy to indicate the robot's location based on triangulation of known satellites location. However, GPS is often sensitive to occlusion and signal reception problems. And its low frequency in refresh rate fails to provide a high temporal resolution for autonomous robots to perform real-time decision making in high speed. Odometry in a wheeled vehicle like a car can be improved by measuring the rotation of the wheels instead of mapping motor power into velocity [8]. However, wheel-encoder based odometry estimation also lacks a ground truth for drift correction and are often susceptible to dirt, oil, and dust contaminants.

3.2.6 GPS

The Global Positioning System (GPS) provides the position and altitude data from receiving satellite signals. The system includes 32 GPS satellites, 1 ground control station, 3 data injection station, etc. The satellites continuously broadcast signal with time stamp given from atomic clocks. After receiving the signal, the user terminal will calculate the signal's propagation time, which then will yield the distance between the satellite and the user terminal. If there are at least three satellites detected, then the position of the user terminal can be uniquely determined. With an addition of a fourth satellite, the altitude can also be calculated. By integrating the data from multiple satellites and optimizing the algorithm, the precision of civil GPS can be controlled within 10 meters. Because the positioning process only depends on the precision of the current signal, the position error will not accumulate by time. However, the satellite can be easily interfered by weather condition and building, making its less adaptive to different environments. Plus, the low precision and slow sampling frequency of the GPS make it unpractical to support autonomous driving by itself. Therefore, GPS data needs to be combined with input from other sensors.

3.2.7 IMU

The Inertia Measuring Unit (IMU) is the sensor that measures the acceleration and the angular speed through an accelerometer and a gyroscope. These two types of sensors are typically made on a single chip as a MicroElectroMechanical System (MEMS). The principle of MEMS is to transfer the inertial force to the change by electrical signals using mechanical structure engraved on the chip. The price of IMUs vary, from low-end type used for commercial electronics to the military type for satellite and missile navigation. For the autonomous driving, we usually choose low to middle-level ones that can cost tens to thousands of dollars. The advantages of IMUs mainly lies in two parts. First, they are isolated from the external world so they can adapt to a wide range of application environment. Second, their high sampling rate—usually 1k Hz—ensures a real-time acquisition of the position data. This gives the system enough time to react. The disadvantage of IMU is that the error will accumulate. This is because the position and posture are derived from integrating the acceleration twice and the angular speed once. This means that the previous error will also affect the later result. Within a short period, as time goes on, the calculated position will drift further, far

from the actual position. Combining the data from both GPS and IMU will achieve the best result in both short-term accuracy and long-term stability. Nevertheless, since we are testing in the indoor environment, there is no GPS signal, so we will only use the IMU data for our project.

3.2.8 A final note

Each of these sensors has advantages and disadvantages. Multisensor data fusion aims to use the strengths of each to precisely understand its environment. A camera serves an excellent tool for detecting roads, reading signs and recognizing a vehicle. The Lidar, on the other hand, is better at accurately estimating the position of this vehicle while the Radar is better at accurately assessing the speed. Table 3.1 compares the capabilities of different sensors in long-distance performance, resolution, robustness against extreme temperature and other metrics. We can see that each sensor has its pros and cons. Not one sensor can ensure robustness and reliability in all situations.

Table 3.1: Comparison of pros and cons of each sensor

	Lidar	Camera	Millimeter Radar	GPS+IMU
Long Distance Performance	Good	Good	Good	Good
Resolution	Fair	Good	Good	Good
False Reading	Fair	Fair	Good	Good
Extreme Temperature Performance	Good	Good	Good	Good
Bad Weather Performance	Poor	Poor	Good	Good
Dust/Moist Performance	Poor	Poor	Good	Poor
Low Cost	Poor	Good	Good	Fair
Low Computation Cost	Poor	Poor	Good	Fair

3.3 Multisensor Data fusion

There has been a confusion in the terminology for fusion systems. The terms "sensor fusion", "data fusion", "information fusion", "multi-sensor data fusion", and "multi-sensor integration" have been widely used in the technical literature to refer to a variety of techniques, technologies, systems, and applications that use data derived from multiple information sources. A detailed review of the definitions on sensor fusions and sensor integration can be found in [36]. Here, we adhere to the definitions of sensor fusion and sensor integration definitions given in [28].

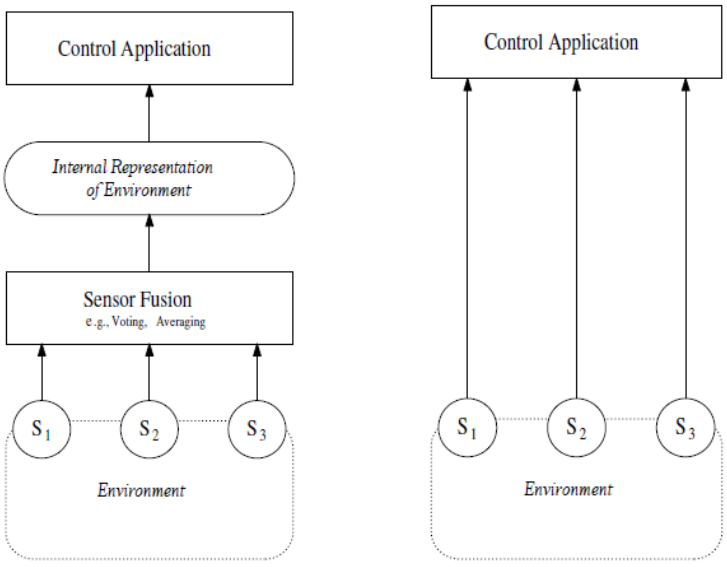
Information Fusion: encompasses theory, techniques and tools conceived and employed for ex-

ploiting the synergy in the information acquired from multiple sources (sensor, databases, information gathered by human, etc.) such that the resulting decision or action is in some sense better (qualitatively or quantitatively, in terms of accuracy, robustness, etc.) than would be possible if any of these sources were used individually without such synergy exploitation [28].

Sensor Fusion: is the combining of sensory data or data derived from sensory data such that the resulting information is in some sense better than would be possible when these sources were used individually [28].

Sensor Integration: Multisensor integration means the synergistic use of sensor data for the accomplishment of a task by a system. Sensor fusion is different to multisensor integration in the sense that it includes the actual combination of sensory information into one representational format [47, 76].

The relationship between Sensor Fusion and Sensor Integration could be referenced in Figure 3.2, Where S1, S2, S3 refer to the actual physical sensors.



(a) Sensor fusion (b) Multisensor integration

Figure 3.2: Block diagram of sensor fusion and multisensor integration

3.3.1 Multisensor fusion

In the last decade, significant progress has been made in the field of multisensor data fusion to solve the problems regarding how to combine multimodel data efficiently and support intelligent robots in decision making [6, 11, 44]. The diversity offered by multiple sensors can positively contribute to the perception task of the intelligent robot. Overcoming heterogeneity of different sensors through robust fusion algorithms lead to effective utilization of the redundancy across the sensors [23]. The following section further discusses the benefit in doing so. However, data coming from different sources are in different formats and has different sensing uncertainties associated with them. Research endeavor is focusing on the effective alignment(either partially, geometrically or temporally) of different sensor streams and the problem are often referred to as multisensor data fusion [46].

Multisensor data fusion could then be subdivided into multiple categories. A detailed classification could be found here [28]. Here, I present a common and general data-oriented classification.

Signal-Level Fusion: signal-level fusion often involves signal enhancement technique such as beamforming using microphone arrays. The resulting signal from multiple sensors is usually of the same form as the original signal but with a greater quality [59].

Pixel-Level Fusion: Pixel-Level Fusion refers to the fusion process in the form of pixels. For example, Lidar output, 3D point cloud, could be represented in a voxel-level resolution and fused with image pixels in the same region of interest. Common fusion approaches often involve pixel-by-pixel fusion or associating neighbor pixels to extract useful information.

Feature-Level Fusion: Nowadays, raw sensor data are often pre-processed into feature vector representations to give a dense and allow better semantic understanding of the data. The feature representation can then be feed into models like neural networks and compared with matrix operations. However, it is still a difficulty in making sense of different data sources and unifying them into one unified feature representation as the raw data often lacks a specific semantic meaning or represent rather different information.

Symbol-Level Fusion: The statistical inference can be used for symbol level fusion where the fusion of symbols is represented in the form of conditional probability [59].

A general description of the sensor fusion pipeline is drawn in Figure 3.3.

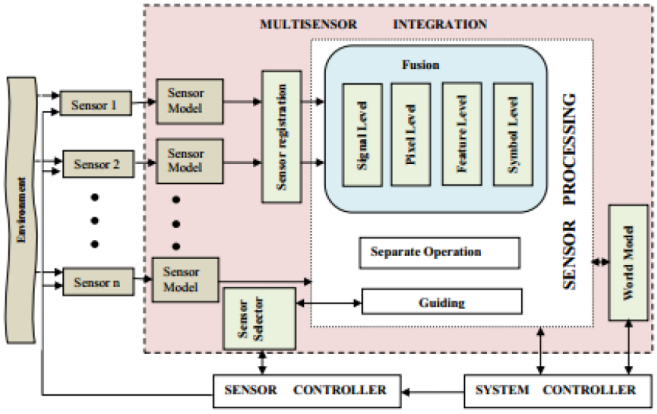


Figure 3.3: Functional Diagram of Multisensor Integration and Fusion

3.3.2 Multisensor integration

Multisensor integration, on the other hand, refers to the synergistic use of sensor data for the accomplishment of a task by a system [47, 28]. Figure 3.3 depicts sensor integration as a pre-processing task before multisensor data fusion. It can be interpreted as a software architecture to allow successful and robust data fusion process. A successful integration system knows when and how to learn, adapt and discard information and keep the system from various failures; e.g. hardware failures, communication failures [13]. The tasks in multisensor integration can be broken down to sensor modeling, sensor registration, sensor processing, world model, sensor selection and system controllers [59].

3.3.3 Motivations and Advantages

With specific attention to mobile robot applications, I discuss the various advantages of multisensor data fusion here.

Uncertainty reduction: Range measurement sensors provide only an estimation of the range, which means we have to model uncertainty into the measurement to fully understand and make use of the data. Multisensor data fusion reduces the uncertainty in the presence of redundancy information at hand. With a proper modeling of sensing uncertainties and a robust mechanism of data selection/fusion, the accuracy of the system can be largely improved [49, 57]. Since

sensory information forms the basis of every advanced task in robotics operation such as localization and environment mapping, a robust sensor model provides the foundation and allow possible improved accuracy in robot localization and multi-view reconstructions [75, 40].

Complementary: Multisensor data fusion is a complementary process as it allows perceiving the information of different parts of the environment by different sensors [34].

Cost reduction: A common-processing module of multisensor data fusion process could reduce the overall system cost of an intelligent robot, while a single sensor needs several electronic modules to perform.

Improved reliability: A system relies on multiple sensors are more insensitive to single-point failures and disturbances [17].

Higher Resolution: A robust modeling and fusion algorithms allows imputation of missing data and enhance the map resolution.

Better map representation: Leveraging heterogeneous sensory information and representations allows a map to have semantic meanings and therefore allows decision-making module of the robot to make much mature decisions with a better understanding of its surrounding [62].

3.3.4 Limitations and Open problems

Great Progress has been made in multisensor data fusion in the last decade. However, sensor fusion does not stand for an omnipotent method as Fowler stated a criticism in 1979:

> One of the grabbiest concepts around is synergism. Conceptual application of synergism is spread throughout military systems but is most prevalent in the "multisensor" concept. This is a great idea provided the input data are a (sic!) good quality. Massaging a lot of crummy data doesn't produce good data; it just requires a lot of extra equipment and may even reduce the quality of the output by introducing time delays and/or unwarranted confidence. [. . .] It takes more than correlation and fusion to turn sows' ears into silk purses. [29]

This publication led a discussion in the academia on multisensor data fusion, and work like [74] has been made to provide a metric/performance measurement for intelligent robotic systems for

fields like object detection, tracking, and classification. And such metric is used in work like [21] to investigate the possible benefits of incorporating more data. However, such investigations are not typical as the fusion algorithms used cannot be guaranteed optimal. Similar work [73] has shown that a distributed system is sub-optimal in comparison to a completely centralized processing scheme concerning the communication effort. However, with recent research on the communication channel in the context of embedded system [61], we may need a more comprehensive analysis.

All in all, works mentioned above does imply the possible open questions in multisensor data fusion; robust fusion mechanism, unified representation of information and fail recovery from hardware and communication failures.

3.3.5 Multisensor data fusion Architectures

Sensor fusion models heavily depend on the applications, and there is no generally accepted model of multisensor data fusions. And research has shown that it is hard to come up a uniformly general solution for multisensor data fusion [35]. The most popular and frequently referred fusion model are JDL model, named after the US Joint Directors of Laboratories, Waterfall fusion process model and the LAAS architecture.

I presented the LAAS (Laboratoire d'Analyse et d'Architecture des Systemes) architecture here due to its close relation to mobile robots applications [1].

LAAS architecture can be broken down to the following levels [1]:

Logical robot level: The logical robot level builds a hardware-independent interface between the physical layers (sensors) and the functional level.

Functional level: The functional level describes how the robot makes sense of the sensory data and integrate them into useful information. At the same time, the functional level describes how the robot operates and defines its essential actions. Common functions included are image processing, obstacle avoidance, and control loops.

Execution control level: Acting as the executor of the robot, execution control level performs routines based on given tasks and coordinates the basic functionality provided by the functional level.

Decision level: Decision level performs the decision making based on the perception input and

can be further broken down to more fine-grained levels that provide different representation abstractions and have different temporal properties [28].

The LAAS architecture is depicted in Figure 3.4. The raw data and feature based fusion are abstracted into its functional level while the decision and symbol-level fusion are written into its decision level. While LAAS architecture provides a reasonable means to give a software-oriented breakdown of the sensor fusion system, it does not describe a good representation of the communication between the modules since the timing requirements are different decision level and the functional level. However, the breakdown allows the developers to implement reusable modules and speeds up the development of a data fusion system.

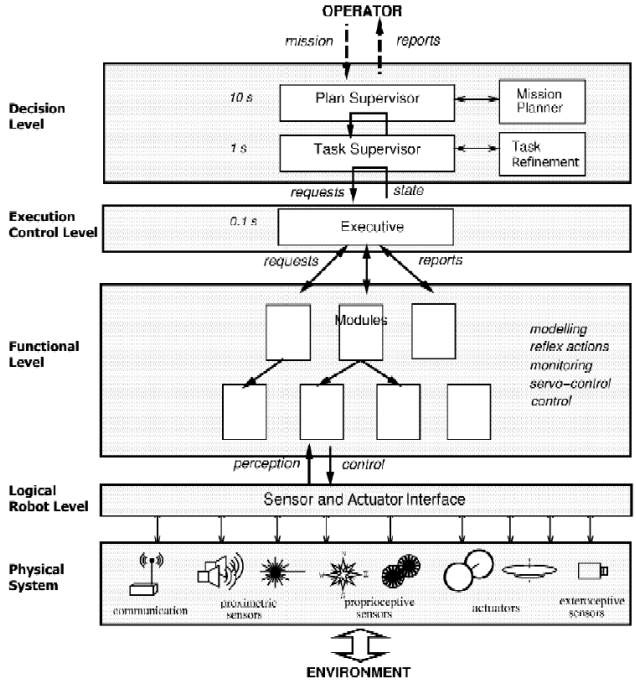


Figure 3.4: LAAS Architecture [1]

3.4 Multi-robot SLAM

As mentioned in the first part in this section, SLAM is the task where robots make sense of its environment and try to create an internal representation of their surrounding. One practical application of data fusion exists in the task where we distribute the computationally expansive across multiple robots, and divide the scene into smaller areas, each mapped by a different robot. This

approach has two variants: the centralized one, where robots build sub-maps and transfer the local information to a central station that performs inference [25, 65], and the decentralized one, where there is no central data fusion, and the agents leverage local communication to reach consensus on a common map. Nerurkar et al. [60] propose an algorithm for cooperative localization based on the distributed conjugate gradient. Araguez et al. [4] investigate consensus-based approaches for map merging. Knuth and Barooah [37] estimate 3D poses using distributed gradient descent. In Lazaro et al. [41], robots exchange portions of their factor graphs, which are approximated in the form of condensed measurements to minimize communication. Cunnigham et al. [18] use Gaussian elimination, and develop an approach, called DDF-SAM, in which each robot exchanges a Gaussian marginal over the separators (i.e., the variables shared by multiple robots). A recent survey on multi-robot SLAM approaches can be found in [69]. The literature shows a robust and general fusion model is in need for cooperative SLAM to allow practical deployment and insensitivity to potential problems in the communication channels between the robots.

4. Project Description and Our Progress

We built our robot using the most readily available resources and will upgrade them as we go. We approach the problem in this way due to the limited time frame and budget requirement. And we want a hardware platform as quickly as possible to start developing software. Therefore, in this section, many systems have versions associated with them. We divide the chapter into six sections: Hardware Setup, Circuits setup, Control Algorithm, Lidar SLAM, Visual SLAM, Multisensor Data Fusion, and Validation. And we give a detailed description of each section along with our progress so far. For Multisensor data fusion and validation, as we have not made progress on those topic and will give our planning schedule in the Future Work section.

4.1 Building The Vehicle

A test vehicle will serve as the mechanism for implementing new algorithms and techniques for SLAM and sensor fusion. At the same time, it will serve a role of benchmarking the performance of new sensor fusion algorithms. A considerable amount of design thinking was necessary when building a test vehicle. Our team tackled the problem with a leaning method; we prototyped an initial model of the test vehicle to find out any issues that couldn't be preemptively avoided, and then quickly integrated and fixed those errors on the vehicle design to save time and effort.

4.1.1 Constraints

There were some constraints to what our test vehicles should look like. First of all, its shape and appearance have to resemble an actual automobile as much as possible so that our findings become more meaningful when applied to real automobiles. This constraint includes having four wheels, two rear and two front, a cuboid shape and reasonable weight of the entire system. However, seeking convenience in controlling the vehicle's velocity, our team eventually decided to have two rears wheels to both drive and steer. This design allows easier manipulation of the vehicle since there are only two degrees of freedom to control: the two rear wheels. The two front wheels are simply omnidirectional wheels which serve as two trailing wheels with no influence on the speed and direction of the test vehicle's movement.

Another constraint taken into account was material. We decided to use acrylic over plastic beams manufactured for robotics purposes, which are usually expensive, due to the limited budget. Simultaneously, the material has to be sturdy enough to hold all the robot parts (batteries, on-board computer, etc.) without any deformation. Prevention of deformation is critical because the base platform for the vehicle must stay parallel to the floor so that sensors, especially a Lidar, can sense surrounding objects without breaking its internal assumption that it is placed on a flat surface.

On top of these constraints, there was one constraint that significantly influenced our design: Appending a Lidar and a stereo camera on to the vehicle. Each of the two sensors has a set of conditions that need to be met for its best performance. For example, it was important that Lidar is not blocked by any other parts for the robot so that it doesn't identify them as obstacles. As a result, it was concluded that the vehicle should have multi-levels and have Lidar be appended on the highest level. On the other hand, a stereo camera would not be sensing the rear view of the vehicle, so it was okay to have some vehicle parts behind the camera, but it was crucial that no parts are lying in front of the camera since unlike Lidar, the camera is sensing a wide angle.

Such constraints led to our initial design as shown below in Figure 4.1.

4.1.2 Prototyping

Solidworks was a necessary tool for the design process, as the vehicle's chassis needed laser cut and we need to calculate how much space each part component is going to take up and where to locate

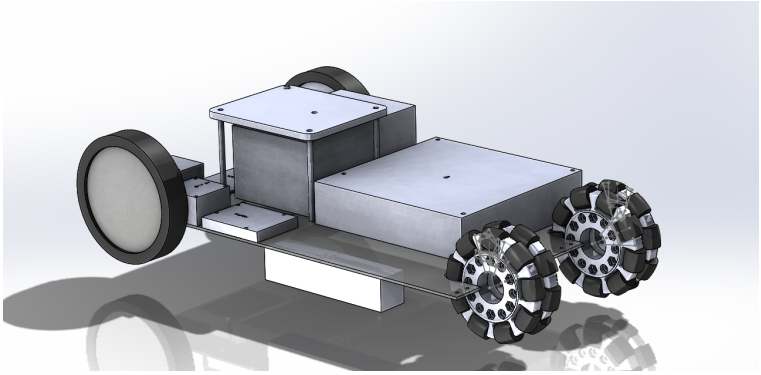


Figure 4.1: Assembly design in Solidworks before building

it. It was the very first step of prototyping our test vehicle. Then our next step to build the car with cheap and prototype material. We decided to use a 1/8" plywood board to be used for our vehicle's chassis, and 3D printed plastic joints for connecting the front two wheels, as shown in Figure 4.2.

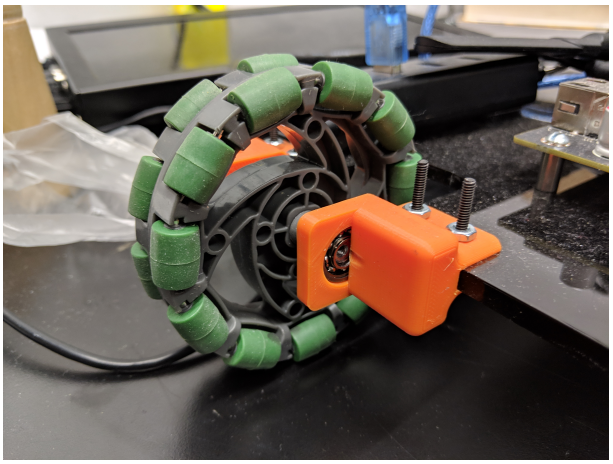


Figure 4.2: Omni-Directional Wheel with Custom Orange Mounting Piece

4.1.3 Sensing Components

RP Lidar2

Our team was able to luckily obtain an RP Lidar for free. RP Lidar is a sleek, indoor, 360 degree 2D Lidar which can take up to 8000 samples of laser ranging per second from its high rotation speed. The onboard system can perform 2D 360 degree scans within a range of 12 meters or 18 meters with a bit of firmware adjustment. The generated 2D point cloud data can easily be used for multiple purposes such as mapping, localization, as well as objective and environment modelling. The typical

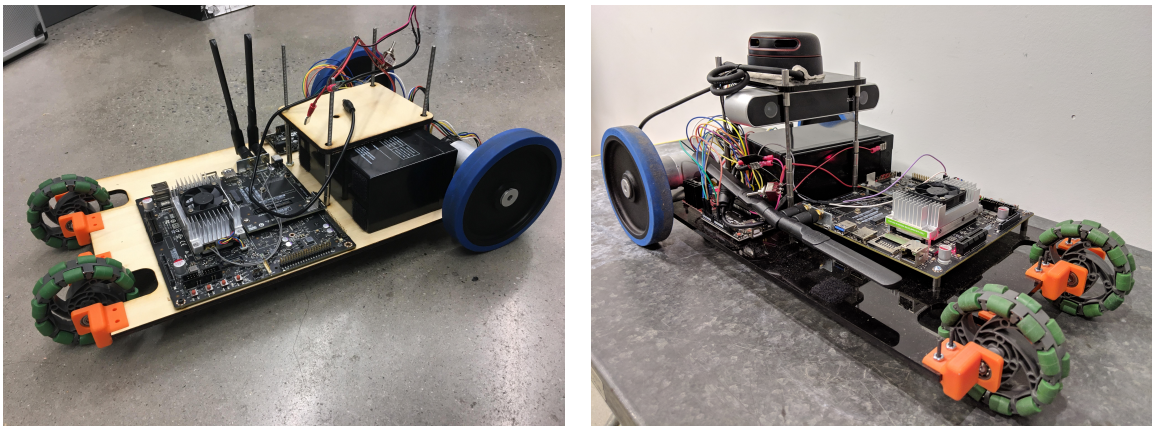


Figure 4.3: First Prototype Car and Current Version

scanning frequency of the scanner is 10hz (6000rpm) with an actual scanning frequency that can be adjusted in the range of 5-15hz. The Lidar is known for its excellence in performing multiple indoor and outdoor environments even with substantial light changes.

ZED Stereo Camera

ZED stereo camera is a sensor that uses an advanced sensing technology based on a human stereo vision principle which adds depth perception, positional tracking, and 3D mapping to any application. Simply put, a ZED camera perceives the world in three dimensions using binocular vision and high-resolution sensors. It can tell how far objects are around it from 0.5 to 20m at 100FPS, indoors and outdoors. The camera supports ROS, which enables us to access and manipulate the raw visual data easily.

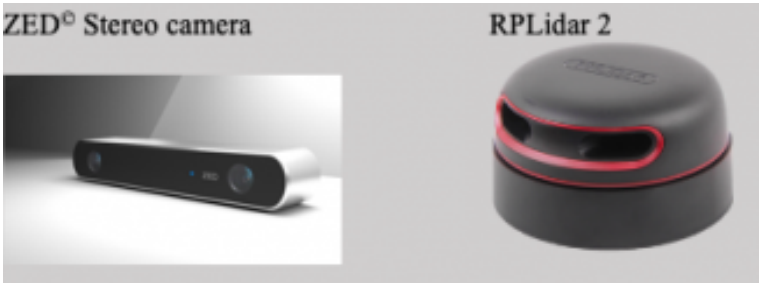


Figure 4.4: System sensors

JETSON TX2

Jetson TX2 can be thought of as the brain of the test vehicle. It is a fast, power-efficient embedded computing device which will be used onboard to compute any processes performed as the vehicle roams around a floor. It features a variety of standard hardware interfaces that make it easy to integrate it into a wide range of products and form factors. Moreover, it packs this performance into a small, power-efficient form factor that's ideal for intelligent edge devices like robots, drones, smart cameras, and portable medical devices.

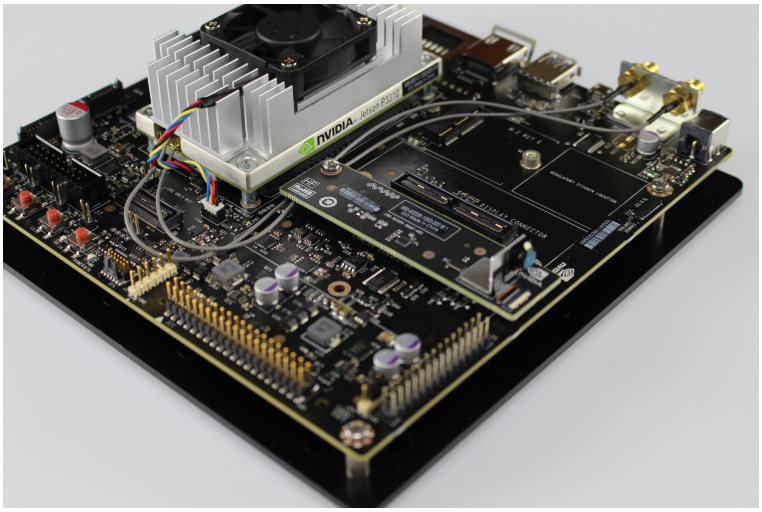


Figure 4.5: Jetson TX2

4.2 Circuits Setup

The electronics part follows the circuit architecture described in Figure 4.6. The Jetson TX2 was selected as the board for intense graphical computation packaged in a small form factor and with low power consumption. Furthermore, we chose Jetson TX2 for the educational discount price from Nvidia.

We recently also received the grant from Nvidia for Drive PX2. Drive PX2 is another embedded system aimed to handle the computation of multi-Lidar setups and is typically used for self-driving testing vehicle. We would fully use its potential if we eventually obtain the 16 line Lidar and have a multi-camera setup.

For lower level input and output (interrupt from encoders and PWM output), we chose to use an

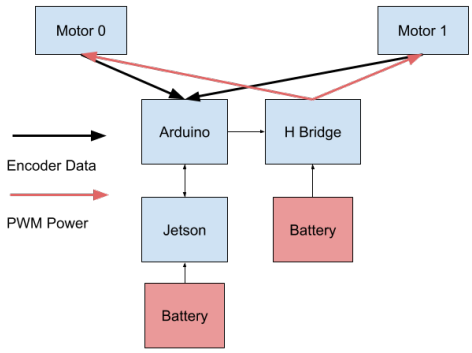


Figure 4.6: System Architecture

Arduino to simplify the process. We tested the Jetson TX2, and found out its GPIO pinout is not the easiest to work with. Therefore, an Arduino is used to bridge the gap between hardware and Jetson. The H-bridge we selected is L298N, Figure 4.7 is the photo of the Arduino and the H-bridge.

The placement of electronics was carefully arranged before building the car. We ensured both

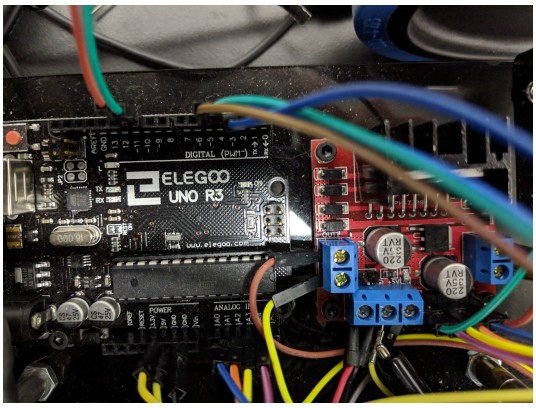


Figure 4.7: Arduino and H-bridge for Encoder and PWM Output

minimum sizes of the robot and the minimum length of the wires. After everything was built, we determined how to route the power from Arduino to the wheels. Furthermore, we made sure motors and Arduino easily plugs into the board we had. Therefore, we first designed a Protoboard and soldered wire onto it for fast prototyping. Design and picture can be found in Figure 4.8. Eventually,

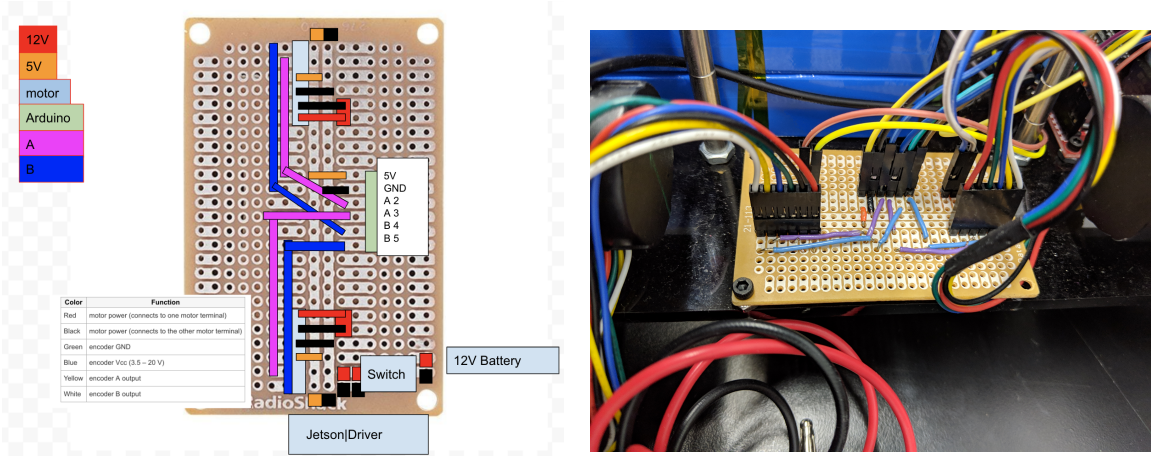


Figure 4.8: Protoboard Design and Picture

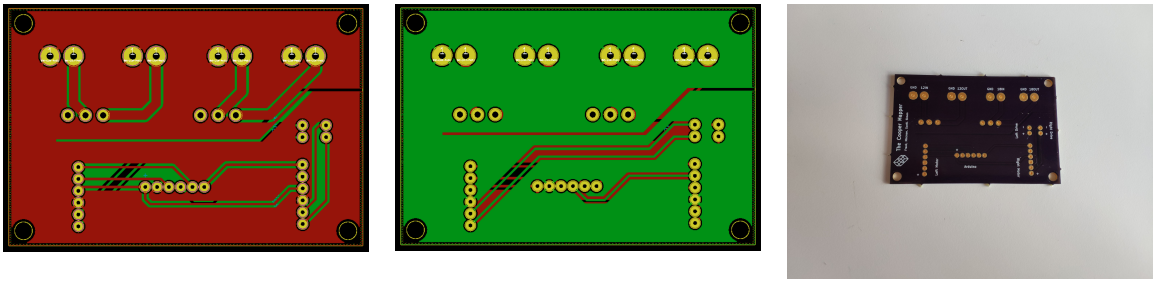


Figure 4.9: PCB Design

we designed a PCB board which does the same power routing. Furthermore, we placed two switches to control the power supply to the Jetson and the Motors respectively. The board design and product can be seen in Figure 4.9

4.3 Software Control

For the software stack, we started by using the ROS framework. ROS is an open source, meta-operating system. It provides the services one would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly used functionality, message passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers.[64] ROS framework which simplified our workflow. Many packages give us access to the driver otherwise very time-consuming

Figure 4.10: Robotics Operating System (ROS)

to build. In this section, the primary function we used is the communication modules in ROS that simplified the communication between the Jetson and Arduino. The package rosserial_arduino let us focus on the message content transferring in between the boards without the getting into details of communication protocols. The package differential_drive provides the PID controller for a differential drive robot so that we can drive each wheel independently with a target speed command. Furthermore, Arduino has libraries that provide quadrature encoder [1] that simplifies the hardware interaction. We also created a node that enables control from the keyboard. If the keyboard is connected to the USB port on the Jetson, the four keys W, S, A, D will command the robot to go forward, backward, turn left and turn right for one second.

Any other keys pressed during the process will be treated as a signal to stop the vehicle immediately. Furthermore, to gain control of the robot when it is running on the battery power, we set up ROS communication between multiple machines. ROS supports running nodes across different machines natively. Therefore, what we need to do first is setting up a WLAN ad-hoc hotspot on the Jetson and use a laptop to connect to the WLAN. Then using SSH, we can get control of the robot and emulate operating on the Jetson using a keyboard. Note that both the laptop and Jetson are running the ROS which can receive the sensor data and manipulate it.

To record the data that the sensors generate, ROS comes with a tool called rosbag. It is a software that records all data being sent to the ROS-framework with a time stamp. When someone wants to replay the test trial to test out a new version of the algorithm, they can open the recorded rosbag and test the algorithm as if it is running on the real-time data. We used this approach to ensure the testing environment variants to our algorithm stay the same. And it saves time by providing the same test data for multiple versions of the algorithm. Furthermore, we added utility functions to speed up the data output while playing the bag, to test our algorithm with more data frames and larger throughput.

[1] `https://www.pjrc.com/teensy/td_libs_Encoder.html`

4.4 Lidar-SLAM

To implement Lidar SLAM, we tested our vehicle with various open-source libraries including Fast SLAM, Hector SLAM, Gmapping. We finally decided to implement our algorithm with Gmapping. We first describe our test and then provide our implementation details of Lidar SLAM.

4.4.1 Preliminary Tests

We ran Fast SLAM, Hector SLAM and Gmapping on our trail run data collected with rosbag when we drove our vehicle around the 6th floor at 41 Cooper. The Hector SLAM and Fast SLAM performed poorly, with the reasons described below. Hector SLAM was developed for a system capable of autonomous exploration in Urban Research and Rescue (USAR) environment.[38] It serves as a general open-source algorithm, which needs only a little modification to fit on a specific platform. A remarkable feature of this algorithm is that it does not necessarily need the odometry data to support its operation. Another feature of Hector SLAM is elevation and cost mapping. **hector-elevation-mapping** module allows us to fuse the point cloud measurements produced by a stereo camera into an elevation map, resulting in a 2D grid with another variable 'height' stored in a corresponding variance for each cell. Odometry is notoriously known to be unreliable in an environment where there are many altitude changes (such as the uneven floor), so we decided to test Hector SLAM on our data. However, though Hector SLAM was able to create a map, the drift was large, and we realized we had to feed in the odometry data to the system so that the algorithm can make more informed estimations of its pose and give a more accurate map. Instead of entirely relying on fast Lidar data feature selection and scan-matching, Fast SLAM uses a particle filter method which use a lot of small particles to perceive a submap and then create a complete map by stitching those submaps together. The particles are generated randomly, and one's submaps are compared with others' to see if they agree with each other on how their environment should look like given their poses. In other words, particle's correctness is evaluated by consensus and inference based on others' submaps. Wrong particles are immediately discarded. Eventually, only the particles that can make sense of each other's submaps are kept and used to stitch together the whole map. However, the particle-filter-based method is relatively expansive regarding memory since

each particle needs to be kept in a joint state matrix and updated every frame and the comparison process takes huge computing power. The problem could be largely simplified with a prior map given as the particles' submaps could then be compared with the prior map and if they have too big a difference, the particle will be discarded. Thus the particle number would quickly decrease and converge to allow the construction of a complete map. It is especially useful in the re-localisation problem for self-driving automobiles where a prior High Definition(HD) map is built. Therefore, we decided to use Gmapping at the end, and our implementation details are described in the next section. [33]

4.4.2 Gmapping Implementation

In general we follow the implementation of Gmapping with the same problem formulation and code structure, which can be found here. [33]. Here, we provide a discussion of how to improve Gmapping as it essentially uses the Rao-Blackwellized particle filters (RBPF) method, which shares the same essential idea with the particle filter method introduced above. The key idea behind RBPF is to estimate a posterior about potential trajectories of the robot given its observation and its odometry measurements. Then, the posterior is used to compute a posterior over maps and trajectories and thus give a relative pose estimation. To do so, RBPF uses a particle filter in which an individual map is associated with every sample. The robot's trajectory changes over the robot's motion, so proposal distribution is chosen to be same as the probabilistic odometry motion model. One of the most common particle filtering algorithms is the Sampling Importance Resampling (SIR) filter. A SIR filter for mapping incrementally processes the observations and the odometry readings as they are available. This is done by updating a set of samples representing the posterior about the map and the trajectory of the vehicle.

We applied an improved algorithm for RBPF by computing an improved proposal distribution on every particle so that information obtained from the sensors can be used while generating the particles. This algorithm brings two advantages: First, the algorithm draws the particles more effectively; computing accurate proposal distribution handles not only the movement of the robot but also the most recent observation, which causes the uncertainty in robot's pose in prediction step to decrease. Second, the highly accurate proposal distribution allows the system to utilize the number

of effective particles as a robust indicator to decide whether or not a resampling has to be carried out. This effect further reduces the particle depletion problem, which refers to the scenario where no particle is valid at all.

Computing the Improved Proposal Distribution

In the prediction step, samples from a proposal distribution have to be drawn, and the generic algorithm does not specify how the proposal distribution is computed. According to previous researches done on RBPF, the odometry motion model has been chosen as the proposal distribution. However, when modelling a mobile robot equipped with a laser range finder, this choice can be suboptimal since the accuracy of the laser range finder leads to extremely peaked likelihood functions. Our solution makes an approximation of trajectory estimation given previous trajectories and locally approximate the distribution around the maximum of the likelihood function by a Gaussian approximation. Then importance weights are computed under the proposal distribution. The proposal distribution based on our likelihood function enables a robot equipped with a laser range finder to draw samples in a very accurate way. Thus, the resulting densities have a much lower uncertainty compared to situations in which the odometry motion model is used. To illustrate this fact, Figure 4.11 depicts typical particle distributions obtained with our approach. In detail, Figure 4.11 (a) shows the resulting distribution when sampling from the raw motion model. Figure 4.11 (b) illustrates the robot reaching the end of such a corridor. As can be seen, the uncertainty in the direction of the corridor decreases and all samples are centered around a single point. In case of a straight featureless corridor, the samples are typically spread along the main direction of the corridor as depicted in Figure 4.11 (c). As one can see from the figure, the solution results in much lower uncertainty than when the odometry motion model is utilized and greatly decreases the number of particles.

Selective Resampling

A further aspect that has a major influence on the performance of a particle filter is the resampling step. Resampling step preserves only a finite number of particles to use by replacing particles have low weight $w^{(i)}$ with ones have a higher weight. But this process can delete good samples from the sample set, causing particle depletion. Thus, it is crucial to determine when to perform resampling.

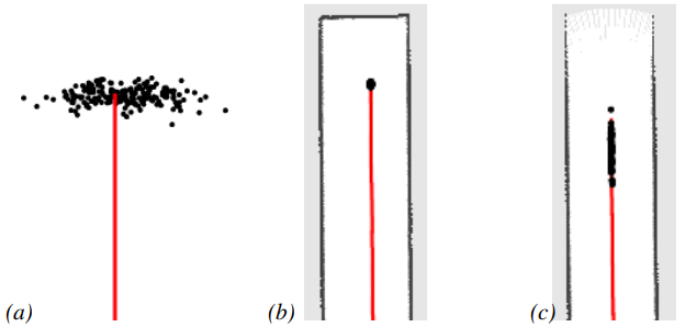


Figure 4.11: Proposal distributions typically observed during mapping. In a featureless open space the proposal distribution is the raw odometry motion model (a). In a dead end corridor the particles the uncertainty is small in all of the directions (b). In an open corridor the particles distributes along the corridor (c).

[77] To do so, a variable called the effective number of particles (N_{eff}) is introduced to estimate how well the current particle set represents the true posterior. [33] proposed the formula as

$$N_eff = \frac{1}{\sum_{i=1}^{N} (w^{(i)})^2}$$

The intuition behind N_{eff} is as follows. If the samples were drawn from the true posterior, the importance weights of the samples would be equal to each other. The worse the approximation, the higher the variance of the importance weights. Since N_{eff} can be regarded as a measure of the dispersion of the importance weights, it is a useful measure to evaluate how well the particle set approximates the true posterior. And we resample each time N_{eff} drops below a given threshold of N/2 where N is the number of particles. This as a result drastically reduces the risk of replacing good particles because the number of resampling operations is reduced and resampling operations are only performed when needed.

Algorithm

The full algorithm is summarized in Algorithm 1. Each time a new measurement tuple (u_{t-1}, z_t) is available, the proposal is computed for each particle individually and is then used to update that particle.

Algorithm 1 Improved RBPF for Map Learning

Require:
$\mathcal{S}_{t-1}$, the sample set of the previous time step
z_t, the most recent laser scan
u_{t-1}, the most recent odometry measurement
Ensure:
$\mathcal{S}_t$, the new sample set

$\mathcal{S}_t = \{\}$
for all $s_{t-1}^{(i)} \in \mathcal{S}_{t-1}$ **do**
$< x_{t-1}^{(i)}, w_{t-1}^{(i)}, m_{t-1}^{(i)} >= s_{t-1}^{(i)}$

// scan-matching
$x_t'^{(i)} = x_{t-1}^{(i)} \oplus u_{t-1}$
$\hat{x}_t^{(i)} = \text{argmax}_x \, p(x \mid m_{t-1}^{(i)}, z_t, x_t'^{(i)})$

if $\hat{x}_t^{(i)} =$ **failure then**
$x_t^{(i)} \sim p(x_t \mid x_{t-1}^{(i)}, u_{t-1})$
$w_t^{(i)} = w_{t-1}^{(i)} \cdot p(z_t \mid m_{t-1}^{(i)}, x_t^{(i)})$
else
// sample around the mode
for $k = 1, \ldots, K$ **do**
$x_k \sim \{x_j \mid |x_j - \hat{x}^{(i)}| < \Delta\}$
end for

// compute Gaussian proposal
$\mu_t^{(i)} = (0, 0, 0)^T$
$\eta^{(i)} = 0$
for all $x_j \in \{x_1, \ldots, x_K\}$ **do**
$\mu_t^{(i)} = \mu_t^{(i)} + x_j \cdot p(z_t \mid m_{t-1}^{(i)}, x_j) \cdot p(x_t \mid x_{t-1}^{(i)}, u_{t-1})$
$\eta^{(i)} = \eta^{(i)} + p(z_t \mid m_{t-1}^{(i)}, x_j) \cdot p(x_t \mid x_{t-1}^{(i)}, u_{t-1})$
end for
$\mu_t^{(i)} = \mu_t^{(i)} / \eta^{(i)}$
$\Sigma_t^{(i)} = \mathbf{0}$
for all $x_j \in \{x_1, \ldots, x_K\}$ **do**
$\Sigma_t^{(i)} = \Sigma_t^{(i)} + (x_j - \mu^{(i)})(x_j - \mu^{(i)})^T \cdot$
$p(z_t \mid m_{t-1}^{(i)}, x_j) \cdot p(x_j \mid x_{t-1}^{(i)}, u_{t-1})$
end for
$\Sigma_t^{(i)} = \Sigma_t^{(i)} / \eta^{(i)}$
// sample new pose
$x_t^{(i)} \sim \mathcal{N}(\mu_t^{(i)}, \Sigma_t^{(i)})$

// update importance weights
$w_t^{(i)} = w_{t-1}^{(i)} \cdot \eta^{(i)}$
end if
// update map
$m_t^{(i)} = \text{integrateScan}(m_{t-1}^{(i)}, x_t^{(i)}, z_t)$
// update sample set
$\mathcal{S}_t = \mathcal{S}_t \cup \{< x_t^{(i)}, w_t^{(i)}, m_t^{(i)} >\}$
end for

$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} (\tilde{w}^{(i)})^2}$
if $N_{\text{eff}} < T$ **then**
$\mathcal{S}_t = \text{resample}(\mathcal{S}_t)$
end if

4.4.3 Results

Figure 4.12 shows a test run of Lidar SLAM using Hector SLAM modules. This test run was completely done on the test vehicle we built, powered by on-board batteries and remote controlled using a Wi-Fi connection between Jetson TX2's and a remote Linux laptop. The test vehicle was manually controlled by a human driver remotely and was driven around the 6th floor of 41 Cooper Square for about 10 minutes. The result shows that Lidar SLAM successfully ran and created a complete, closed map of the 6th floor.

Comparing Figure 4.12, a 2D map created by our research vehicle, with Figure 4.13, an actual 2D floor plan of the 6th floor of 41 Cooper Square, it is clear that the vehicle successfully created an accurate representation of its environment using Hector SLAM algorithm. However, although Lidar SLAM was able to create a detailed and accurate 2D map of its environment, it can lack critical information about what the environment actually is consisted of and what they actually are. This deficiency in semantics can be compensated with visual SLAM, which is introduced in the next section.

Figure 4.12: Lidar SLAM output

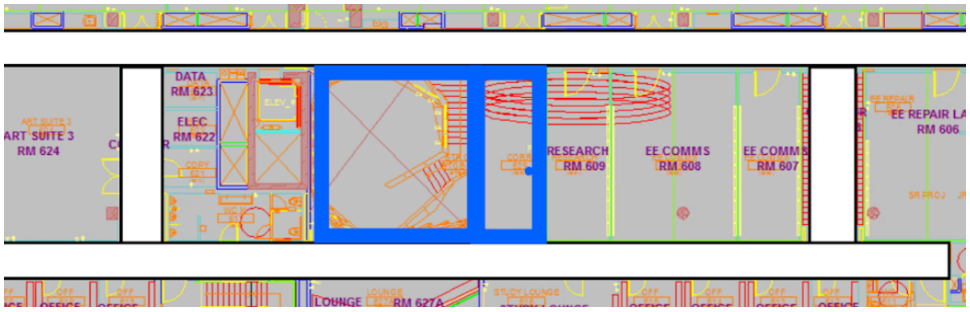


Figure 4.13: 6th Floor Floor Plan

4.5 Visual-SLAM

4.5.1 How does it work?

In this section, we give an introduction to each module of ORB-slam and present the result by implementing it on our vehicle. Note that most visual SLAM algorithms will work the same way. The input to the algorithm is a series of images(either grey-scale or coloured), and the output is camera pose(location and orientation) in each image, a sparse 3D point-cloud with colour (if the input is coloured), and other more detailed information.[54] Figure 4.14 shows the stereo image pipeline and Figure 4.15 shows the whole algorithm pipeline, this section explains the most important features of ORB-SLAM [53] and ORB-SLAM2 [55].

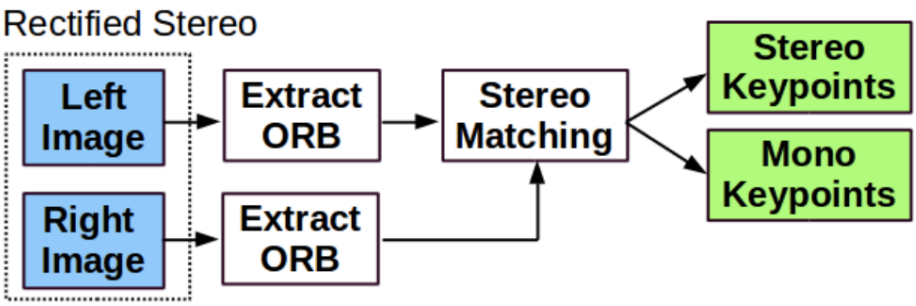


Figure 4.14: Stereo Camera Pipeline

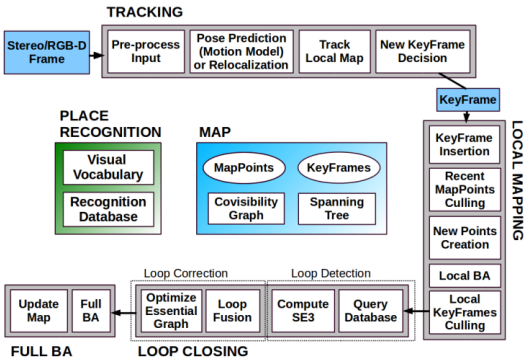


Figure 4.15: ORB-SLAM2 Pipeline

4.5.2 Feature Extraction

The first step of SLAM is the extraction of features. Features are the basis for comparison. Let us treat a teddy bear as an example of features. (Real features are smaller) When the teddy is viewed in the front, no matter the angle of the viewpoint, the system should recognize that it is looking at the same thing and make a connection saying that they are the same Teddy Bear.

In ORB-SLAM, the feature extraction algorithm is called ORB. ORB algorithm can extract features from an image in 33 ms, achieving almost real time processing. ORB starts with a FAST (feature from accelerated segmentation test) feature extraction. The algorithm first performs corner detection using the contrast difference in the image. In figure 4.16, it finds the window to be much darker than the frame. Next, from the relative pixel brightness of the surrounding pixels from the point P (the real world location of the feature), it generates a displacement vectors (representing the difference in illumination) for each pixel around it. From this feature, the algorithm inputs the displacement vectors into a BRIEF(Binary Robust Independent Elementary Features) feature by transforming the displacement vectors to a 256 bits binary string that serves as a unique identifier of the feature to ensures quick comparison and invariance in viewing the pixels from different perspectives. This whole process is called generating an ORB feature: Corners are turned into 256 bits binary strings for comparison between frames. One big advantage of ORB-SLAM is that the single type of feature is used for mapping, tracking, relocalization and loop detection. This simplified ORB-slam with respect to other algorithms, which have to keep more than one type of feature.

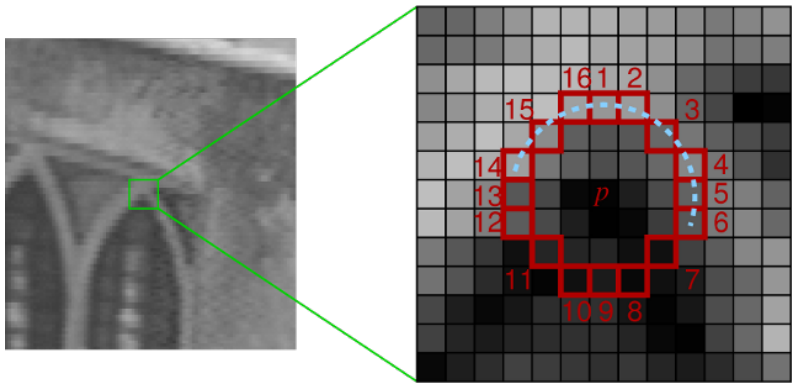


Figure 4.16: Example of a Feature Extractionn

4.5.3 Generating Map-points and Localization

Now that in consecutive frames, the features can be extracted and their relative locations are marked by 2D coordinates on each frame, how would the algorithm generate 3D locations from 2 sets of 2D coordinates?

Using Stereo Vision, the task of generating map points is simpler than using One camera. First, the stereo vision data is fed through feature extraction and frame comparison. Using Epipolar geometry like in Figure 4.17, we know that for the same point P, if the pose(location, orientation) of camera O1 and O2 are both known, and the pixel p and p' on the respective images are recognized as the same feature, then we can infer the approximate location of feature P in 3D space. Conversely, if we know location P, location O1, p, and p', location O2 can also be determined. Therefore, the pose estimation could be gained. For the stereo camera, the relative difference between its two "eyes"(left and right camera) are fixed, (same orientation and distance between cameras), known as the baseline, so the location of each feature P can be determined from each stereo frame. With the relative poses known, we can project the images on the relative position they should be, which is inferred from the pose estimation, to stitch the image together and reconstruct the 3D world we are trying to map. Note that when the camera moves, the algorithm uses only the left camera for localization: since the location of P is known in 3D from previous stereo frame, location of camera position O1 is known before, use the same feature in p' in the second frame, we can infer the location of O2, thus localization is achieved. Connecting the localization dots will also give us a trajectory. Figure 4.18 shows an example of a data association between two frames, where the features in the two consecutive frames are extracted and connected.

4.5.4 Keyframes

If this map stitching task is performed on every consecutive frame, the memory cost and computation power required would be tremendous. To reduce the amount of work and achieve real-time performance, keyframes are used.

Keyframes are selected from one of a few consecutive frames to represent the collection of those frames since they often overlap with each other regarding features. Keyframes are the frames that

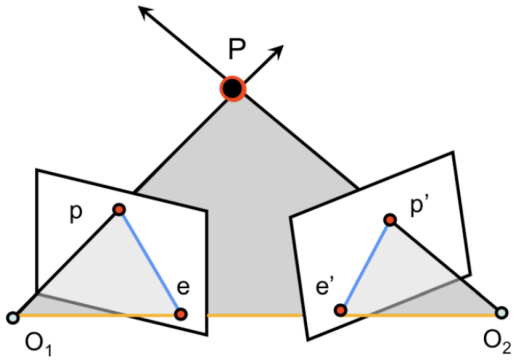


Figure 4.17: Epipolar Geometry

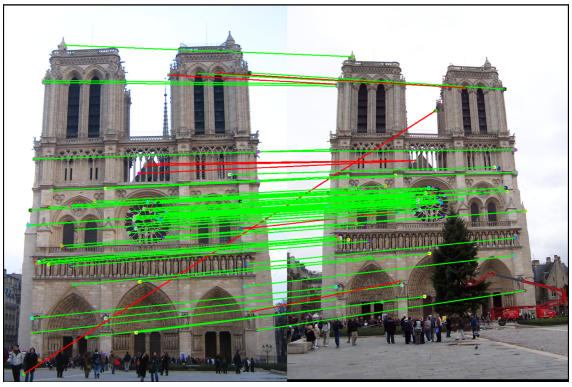


Figure 4.18: Two pictures that have matched features, green ones are correct matches and red are incorrect matches

are used to generate the map. If there are too many keyframes, there is not enough room to store the data and computation will take longer to go through many images. If there are too few keyframes, the map generated may be incomplete. There are various ways to select keyframes. The algorithm implemented here uses a generous policy when creating keyframes, while a very exigent culling mechanism to detect redundancy. [53] If a keyframe has 90% features same as three other keyframes have, that keyframe is immediately discarded.

4.5.5 Relocolization

Relocalization refers to the case where the system loses track of its pose due to an abrupt change in the environment. When relocalization or loop detection occurs, the algorithm needs to search in all keyframes for the frame that have the closest match.

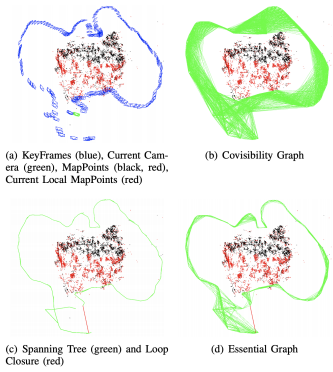
It first uses a bag of words approach (each dimension is a feature, and the number of this feature

is the length in that dimension) to compare frames with the current perceived visual data. In the multidimensional space, if the algorithm finds multiple close points, it will examine in detail the relative location of the features and finds the frame with the best match.

4.5.6 Loop Closing and Essential Graph

Mapping does not require a graph. However, when trying to correct loop closing error by distributing that error along each frame along the loop, an undirected graph is helpful for the system to determine the smallest loop to distribute the error.

Dense connections between keyframes require too much computation, while sparse connections do not provide enough information to detect the smallest loop to correct the error. The tradeoff again lies in the computation time and accuracy. Therefore, ORB-slam uses an Essential Graph, shown in Figure 4.19 (d), to solve this issue. First, it finds the keyframes that share a minimum of 10 features and constructs a Covisibility Graph. Then, it finds the minimum spanning tree by cutting the edges that have a lower number of shared features. And It adds the strong connections by adding the keyframes that share 100 features. At the end, the graph generated is called an Essential Graph. This graph will help distribute errors along the loop with a proper connection density, creating a map with acceptable error.



(a) KeyFrames (blue), Current Camera (green), MapPoints (black, red), Current Local MapPoints (red)

(b) Covisibility Graph

(c) Spanning Tree (green) and Loop Closure (red)

(d) Essential Graph

Figure 4.19: Essential Graph

4.5.7 Progress

We have successfully implemented Visual SLAM largely based on ORB-SLAM2 [56] using ZED's API, and a demo of the reconstruction of the laboratory room 601 in Cooper 41 can be viewed online.

[2] We are at the stage of moving the algorithm into the ROS operating system and deployed it on the mobile robot base. And a visual map is expected to be created with this algorithm running on the vehicle.

[2] https://p3d.in/e/x1fMl

5. Status and Future Work

We have implemented Visual SLAM and Lidar SLAM independently. Though the map is not yet perfect and we are still open to the opportunities to try new SLAM algorithms, we are ready to move on to the next stage of fusing the two maps.

5.1 Video-SLAM Fusion-SLAM and More

In this section, we dive into the actual algorithms for combining the two SLAM methods and take into account the different accuracy and noise levels of each sensor. We aim to come up with a general and robust framework which need little tuning for different sensors and has a relatively good accuracy regardless of the available computing platforms. Here, we first give a quick overview of a selected list of sensor fusion algorithms we are planning to test with. Then we give a Gantt chart explaining the future work.

5.1.1 Multisensor data fusion algorithms

This section gives quick introductions to various methods and models for multisensor data fusion. Generally, data fusion methods can be classified as an estimation method, a classification method, an inference method, or an artificial method. We are planning to test on these methods on our model and are trying to come up with something new out of them as a general fusion framework.

Kalman Filter: is a mathematical model that divides into two components, move estimation and measurement correction. It can be seen as a Hidden Markov Model (HMM). It is well defined in a set of mathematical equations and built under the assumption of Gaussian distributed errors. The mathematical equations provide an efficient way to perform pose estimation of the robot [9]. Kalman filter can be used to fuse different sensors with different modalities, as long as their modalities have well-defined mathematical models. To cope with the nonlinearity in the estimated parameters, Extended Kalman Filter [24] has been designed and served as the primary approach to map dynamic environment for the last several years. However, the Kalman filter approach suffers two well-known shortcomings; quadratic complexity, and the sensitivity to failures in data association [10]. As Kalman filter needs to include the new landmark features into its internal matrix representation of the map, it suffers from substantial computation time proportional to the square of the number of landmarks in the environment.

Dempster-Shafer: In DS theory, the weight of conflict metric and the enlargement of the frame of discernment are the two components used to measure the amount of consensus between different sensors. The certainty and confidence can be improved when the robot actively acquire more information about the ambiguous regions when the corresponding grid lacks a consensus between different sensors [5, 50].

Artificial Neural Networks: With the progress of artificial neural networks in fields like computer vision and natural language processing, neural networks has been used to map the occupancy grid. The result has been proven to be robust and adaptive to the environmental changes [22]. proposed back-propagation training of multi-layer perception [67]. The neural network is trained to perform the correct conversion of range information into occupancy grid. However, conventional neural network and deep learning models take huge time to train, and it is difficult to deploy them in a computationally-limited system to perform online-learning. Online learning and adaption remain an open problem before neural networks can be applied to industrial robot applications.

Fuzzy Logic based sensor fusion: relates to the artificial intelligence class of multisensor data fusion. This method can also be considered as a probabilistic approach in the sense that the

method does not assign probabilities to the propositions but it assigns the membership values to proposition [63]. There is vast flexibility to perform the fusion of multisensory information under the special rule of combination of fuzzy values.

5.2 Future Improvement

While attempts have been made to develop robust fusion algorithms for complex environments, challenges remain on each of the algorithms. The ANN-based approach needs to tackle its long training time while DS methods need to make sense of the information conflicts between the data. And problems alike exist in some ways which are not mentioned here. To make the mapping and localization robust there is need of pre-processing and post-processing of the sensory information and resultant internal representation in the form of the map. Our work aims to leverage the existing methods and create a generic and robust fusion framework.

5.3 Ghantt Chart

As presented on the Gantt Chart, we will try to come up with a fusion algorithm and then move forward to use autonomous navigation and planning as a metric to evaluate how our algorithm performs.

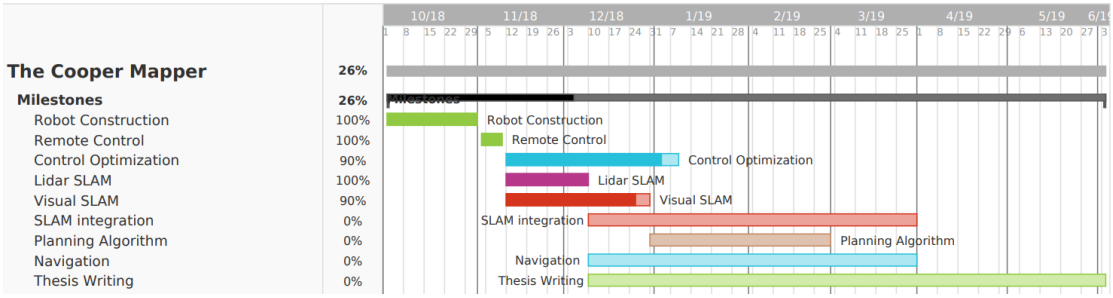
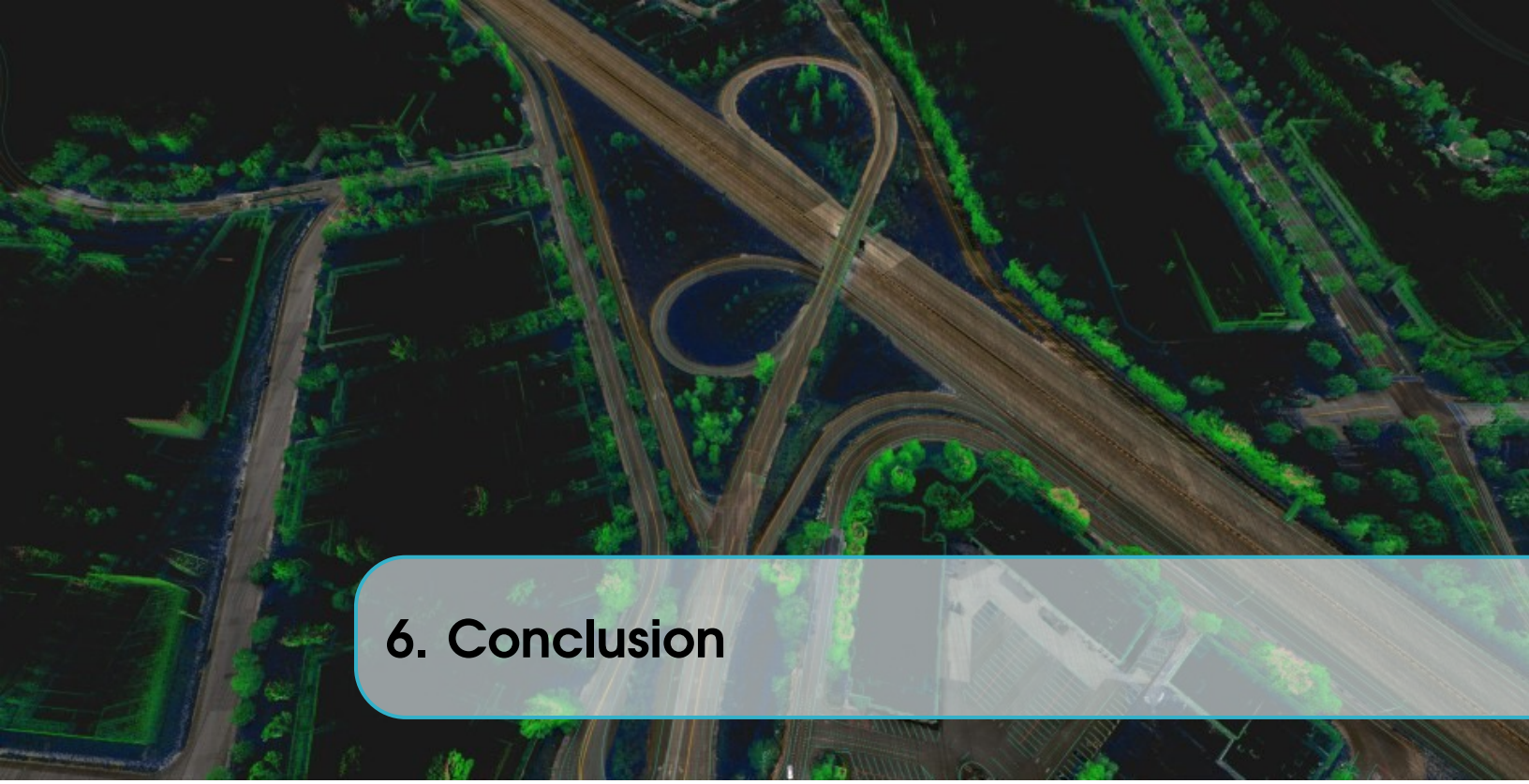
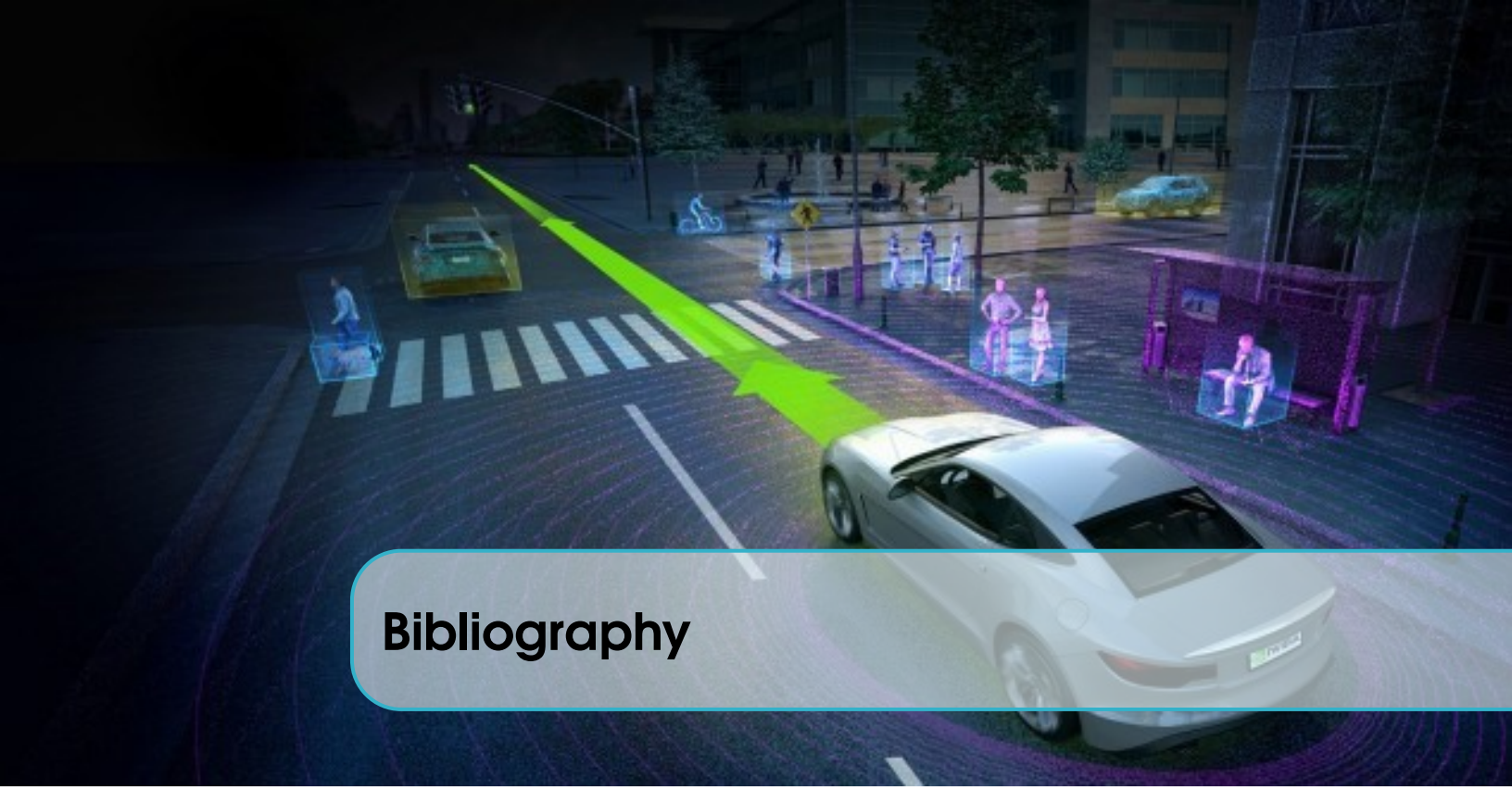


Figure 5.1: Ghantt Chart, the filled part in the bar suggests for completeness as a percentage



6. Conclusion

Though SLAM is a heavily researched area, its industrialization on autonomous vehicles is far from perfect. As we are exploring topics like Dempster-Shafter, fuzzy logic, and HIMM in work, we have realized that the future of long-term SLAM lies in multisensor data fusion; a robust fusion mechanism would not only reduce uncertainties in various aspects like sensory information, robot localization, and recognition algorithms, but also allow multi-level representations of the map and semantic understanding of the environment. With our current progress on SLAM algorithm implementations and multisensor data fusion research, we are confident to move forward to working on the sensor fusion part next semester and build a relatively accurate map for smooth autonomous navigation and path planning.



Bibliography

[1] Rachid Alami, Raja Chatila, Sara Fleury, Malik Ghallab, and Félix Ingrand. An architecture for autonomy. *The International Journal of Robotics Research*, 17(4):315–337, 1998.

[2] Juan Andrade-Cetto and Alberto Sanfeliu. Concurrent map building and localization on indoor dynamic environments. *International Journal of Pattern Recognition and Artificial Intelligence*, 16(03):361–374, 2002.

[3] GC Anousaki and Kostas J Kyriakopoulos. Simultaneous localization and map building for mobile robot navigation. *IEEE Robotics & Automation Magazine*, 6(3):42–53, 1999.

[4] Rosario Aragues, Jorge Cortes, and Carlos Sagues. Distributed consensus on robot networks for dynamically merging feature-based maps. *IEEE Transactions on Robotics*, 28(4):840–854, 2012.

[5] Rui Araújo and Aníbal T De Almeida. Learning sensor-based navigation of a real mobile robot in unknown worlds. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(2):164–178, 1999.

[6] Minoru Asada, Yasuhito Fukui, and Saburo Tsuji. Representing global world of a mobile robot with relational local maps. *IEEE transactions on systems, man, and cybernetics*, 20(6):1456–

1461, 1990.

[7] Nicholas Ayache and Olivier D Faugeras. Building, registrating, and fusing noisy visual maps. *The International Journal of Robotics Research*, 7(6):45–65, 1988.

[8] Mordechai Ben-Ari and Francesco Mondada. *Robotic Motion and Odometry*, pages 63–93. Springer International Publishing, Cham, 2018.

[9] L Conde Bento, Urbano Nunes, Fernando Moita, and António Surrecio. Sensor fusion for precise autonomous vehicle navigation in outdoor semi-structured environments. In *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE*, pages 245–250. IEEE, 2005.

[10] Johan Bijker and Willem Steyn. Kalman filter configurations for a low-cost loosely integrated inertial navigation system on an airship. *Control Engineering Practice*, 16(12):1509–1518, 2008.

[11] Dolores Blanco, Beatriz L Boada, Luis Moreno, and Miguel Angel Salichs. Local mapping from online laser voronoi extraction. In *Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, volume 1, pages 103–108. IEEE, 2000.

[12] Ömür Bozma and Roman Kuc. Building a sonar map in a specular environment using a single mobile sensor. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (12):1260–1269, 1991.

[13] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[14] Stephan K Chalup, Craig L Murch, and Michael J Quinlan. Machine learning with aibo robots in the four-legged league of robocup. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(3):297–310, 2007.

[15] Raja Chatila and Jean-Paul Laumond. Position referencing and consistent world modeling for mobile robots. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 138–145. IEEE, 1985.

[16] ZJ Chong, B Qin, T Bandyopadhyay, T Wongpiromsarn, ES Rankin, MH Ang, E Frazzoli, D Rus, D Hsu, and KH Low. Autonomous personal vehicle for the first-and last-mile transportation services. In *Cybernetics and Intelligent Systems (CIS), 2011 IEEE 5th International Conference on*, pages 253–260. IEEE, 2011.

[17] Howie M Choset, Seth Hutchinson, Kevin M Lynch, George Kantor, Wolfram Burgard, Lydia E Kavraki, and Sebastian Thrun. *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

[18] Andrea Cohen, Christopher Zach, Sudipta N Sinha, and Marc Pollefeys. Discovering and exploiting 3d symmetries in structure from motion. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1514–1521. IEEE, 2012.

[19] James L Crowley. World modeling and position estimation for a mobile robot using ultrasonic ranging. In *ICRA*, volume 89, pages 674–680, 1989.

[20] Paolo Dario, Eugenio Guglielmelli, Vincenzo Genovese, and Maurizio Toro. Robot assistants: Applications and evolution. *Robotics and autonomous systems*, 18(1-2):225–234, 1996.

[21] Belur V Dasarathy. More the merrier... or is it? sensor suite augmentation benefits assessment. In *Information Fusion, 2000. FUSION 2000. Proceedings of the Third International Conference on*, volume 2, pages WEC3–20. IEEE, 2000.

[22] Ian Lane Davis and Anthony Stentz. Sensor fusion for autonomous outdoor navigation using neural networks. In *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 3, pages 338–343. IEEE, 1995.

[23] Varuna De Silva, Jamie Roche, and Ahmet Kondoz. Robust fusion of lidar and wide-angle camera data for autonomous mobile robots. *Sensors*, 18(8):2730, 2018.

[24] Rached Dhaouadi, Ned Mohan, and Lars Norum. Design and implementation of an extended kalman filter for the state estimation of a permanent magnet synchronous motor. *IEEE Transactions on Power Electronics*, 6(3):491–497, 1991.

[25] Jing Dong, Erik Nelson, Vadim Indelman, Nathan Michael, and Frank Dellaert. Distributed real-time cooperative localization and mapping using an uncertainty-aware expectation maximization approach. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 5807–5814. IEEE, 2015.

[26] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[27] Hugh F Durrant-Whyte. Uncertain geometry in robotics. *IEEE Journal on Robotics and Automation*, 4(1):23–31, 1988.

[28] Wilfried Elmenreich. An introduction to sensor fusion. *Vienna University of Technology, Austria*, 502, 2002.

[29] Charles A Fowler. Comments on the cost and performance of military systems. *IEEE Transactions on Aerospace and Electronic Systems*, (1):2–10, 1979.

[30] Udo Frese. Interview: Is slam solved? *KI-Künstliche Intelligenz*, 24(3):255–257, 2010.

[31] Santiago Garrido, Luis Moreno, and Dolores Blanco. Exploration and mapping using the vfm motion planner. *IEEE Transactions on Instrumentation and Measurement*, 58(8):2880–2892, 2009.

[32] Paolo Gaudiano, Eduardo Zalama, and Juan López Coronado. An unsupervised neural network for low-level control of a wheeled mobile robot: noise resistance, stability, and hardware implementation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(3):485–496, 1996.

[33] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE transactions on Robotics*, 23(1):34–46, 2007.

[34] Kang Hou, Hanxu Sun, Qingxuan Jia, and Yanheng Zhang. An autonomous positioning and navigation system for spherical mobile robot. *Procedia Engineering*, 29:2556–2561, 2012.

[35] Moshe Kam, Xiaoxun Zhu, and Paul Kalata. Sensor fusion for mobile robot navigation. *Proceedings of the IEEE*, 85(1):108–119, 1997.

[36] Bahador Khaleghi, Alaa Khamis, Fakhreddine O Karray, and Saiedeh N Razavi. Multisensor data fusion: A review of the state-of-the-art. *Information fusion*, 14(1):28–44, 2013.

[37] Joseph Knuth and Prabir Barooah. Collaborative localization with heterogeneous inter-robot measurements by riemannian optimization. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1534–1539. IEEE, 2013.

[38] Stefan Kohlbrecher, Johannes Meyer, Thorsten Graber, Karen Petersen, Uwe Klingauf, and Oskar von Stryk. Hector open source modules for autonomous mapping and navigation with rescue robots. In *Robot Soccer World Cup*, pages 624–631. Springer, 2013.

[39] T Lang and G Hayes. Information fusion 2007 10th international conference on. *Exploitation of Bistatic Doppler Measurements in Multistatic Tracking*, 2007.

[40] Gisbert Lawitzky, Wendelin Feiten, and Marcus Möller. Sonar sensing for low-cost indoor mobility. *Robotics and Autonomous Systems*, 14(2-3):149–157, 1995.

[41] Maria Teresa Lazaro, Lina María Paz, Pedro Pinies, Jose A Castellanos, and Giorgio Grisetti. Multi-robot slam using condensed measurements. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1069–1076. IEEE, 2013.

[42] John Leonard, Jonathan How, Seth Teller, Mitch Berger, Stefan Campbell, Gaston Fiore, Luke Fletcher, Emilio Frazzoli, Albert Huang, Sertac Karaman, et al. A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, 25(10):727–774, 2008.

[43] John J Leonard and Hugh F Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on robotics and Automation*, 7(3):376–382, 1991.

[44] John J Leonard, Hugh F Durrant-Whyte, and Ingemar J Cox. Dynamic map building for an autonomous mobile robot. *The International Journal of Robotics Research*, 11(4):286–298, 1992.

[45] Martin Liggins II, David Hall, and James Llinas. *Handbook of multisensor data fusion: theory and practice*. CRC press, 2017.

[46] Ren C Luo, Chih Chia Chang, and Chun Chi Lai. Multisensor fusion and integration: Theories, applications, and its perspectives. *IEEE Sensors Journal*, 11(12):3122–3138, 2011.

[47] Ren C Luo and Michael G Kay. Multisensor integration and fusion in intelligent systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):901–931, 1989.

[48] Ren C Luo and Chun C Lai. Enriched indoor map construction based on multisensor fusion approach for intelligent service robot. *IEEE Transactions on Industrial Electronics*, 59(8):3135–3145, 2012.

[49] Ren C Luo, Chih-Chen Yih, and Kuo Lan Su. Multisensor fusion and integration: approaches, applications, and future research directions. *IEEE Sensors journal*, 2(2):107–119, 2002.

[50] José A Malpica, María Concepcion Alonso, and María A Sanz. Dempster–shafer theory in geographic information systems: A survey. *Expert Systems with Applications*, 32(1):47–55, 2007.

[51] Larry Matthies, Yalin Xiong, R Hogg, David Zhu, A Rankin, Brett Kennedy, Martial Hebert, R Maclachlan, Chi Won, Tom Frost, et al. A portable, autonomous, urban reconnaissance robot. *Robotics and Autonomous Systems*, 40(2-3):163–172, 2002.

[52] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, et al. Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597, 2008.

[53] Montiel J. M. M. Mur-Artal, Raúl and Juan D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[54] 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.

[55] 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*, 33(5):1255–1262, 2017.

[56] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[57] Robin Murphy, Robin R Murphy, and Ronald C Arkin. *Introduction to AI robotics*. MIT press, 2000.

[58] Robin R Murphy. Dempster-shafer theory for sensor fusion in autonomous mobile robots. *IEEE Transactions on Robotics and Automation*, 14(2):197–206, 1998.

[59] KS Nagla, Moin Uddin, and Dilbag Singh. Multisensor data fusion and integration for mobile robots: A review. *IAES International Journal of Robotics and Automation*, 3(2):131, 2014.

[60] Esha D Nerurkar, Stergios I Roumeliotis, and Agostino Martinelli. Distributed maximum a posteriori estimation for multi-robot cooperative localization. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 1402–1409. IEEE, 2009.

[61] Peng Ning, Yueh-Hsun Lin, Stephen E McLaughlin, Michael C Grace, Ahmed M Azab, Rohan Bhutkar, and Yong Choi. Authorized control of an embedded system using end-to-end secure element communication, January 25 2018. US Patent App. 15/584,892.

[62] Georgios Paraskevopoulos, Giannis Karamanolakis, Elias Iosif, Aggelos Pikrakis, and Alexandros Potamianos. Sensory-aware multimodal fusion for word semantic similarity estimation.

[63] Alfredo Plascencia. *Sensor fusion for autonomous mobile robot navigation*. Aalborg Universitet, 2008.

[64] Morgan Quigley, Ken Conley, Brian P. 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*, 2009.

[65] Luis Riazuelo, Javier Civera, and JM Martínez Montiel. C2tam: A cloud framework for cooperative tracking and mapping. *Robotics and Autonomous Systems*, 62(4):401–413, 2014.

[66] Søren Riisgaard and M Blas. A tutorial approach to simultaneous localization and mapping. *Massachusetts Institute of Technology (MIT), Mobile Robotics course materials*, 2012.

[67] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

[68] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[69] Sajad Saeedi, Michael Trentini, Mae Seto, and Howard Li. Multiple-robot simultaneous localization and mapping: A review. *Journal of Field Robotics*, 33(1):3–46, 2016.

[70] Dirk Schulz and Wolfram Burgard. Probabilistic state estimation of dynamic objects with a moving mobile robot. *Robotics and Autonomous Systems*, 34(2-3):107–115, 2001.

[71] Stephen Se, David G Lowe, and James J Little. Vision-based global localization and mapping for mobile robots. *IEEE Transactions on robotics*, 21(3):364–375, 2005.

[72] Randall C Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The international journal of Robotics Research*, 5(4):56–68, 1986.

[73] Robert R Tenney and Nils R Sandell. Detection with distributed sensors. *IEEE Transactions on Aerospace and Electronic systems*, (4):501–510, 1981.

[74] A Theil, LJHM Kester, and E Bosse. On measures of performance to assess sensor fusion effectiveness. In *Information Fusion, 2000. FUSION 2000. Proceedings of the Third International Conference on*, volume 2, pages THB5–3. IEEE, 2000.

[75] Sebastian Thrun. Learning occupancy grid maps with forward sensor models. *Autonomous robots*, 15(2):111–127, 2003.

[76] Sebastian Thrun et al. Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, 1(1-35):1, 2002.

[77] Rudolph Van Der Merwe, Arnaud Doucet, Nando De Freitas, and Eric A Wan. The unscented particle filter. In *Advances in neural information processing systems*, pages 584–590, 2001.

[78] Wang Yaonan, Yang Yimin, Yuan Xiaofang, Zuo Yi, Zhou Yuanli, Yin Feng, and Tan Lei. Autonomous mobile robot navigation system designed in dynamic environment based on transferable belief model. *Measurement*, 44(8):1389–1405, 2011.