3D Perception of Outdoor and Dynamic Environment using Laser Scanner

Asma Azim

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Mathmatique, Informatique (Robotique)**

Arrêté ministérial : 7 août 2006

Présentée par

Asma AZIM

Thèse dirigée par **Olivier AYCARD**

préparée au sein du **Laboratoire d'Informatique de Grenoble**
et de **Mathmatiques, Sciences et Technologies de l'Information, Informatique**

3D Perception of Outdoor and Dynamic Environment using Laser Scanner

Thèse soutenue publiquement le **17 décembre 2013**,
devant le jury composé de :

M. Thierry FRAICHARD
Chargé de Recherche, INRIA Rhône-Alpes, Président
M. François CHARPILLET
Directeur de Recherche, INRIA Nancy, Rapporteur
M. Michel DEVY
Directeur de Recherche, LAAS-CNRS Toulouse, Rapporteur
M. Yassine RUICHEK
Professeur, Université de Technologie de Belfort-Montbéliard, Examinateur
M. Olivier AYCARD
MCF HDR, Université de Grenoble, Directeur de thèse

3D Perception of Outdoor and Dynamic Environment using Laser Scanner

To:

My family

No words to say thanks!

Abstract

With an anticipation to make driving experience safer and more convenient, over the decades, researchers have tried to develop intelligent systems for modern vehicles. The intended systems can either drive automatically or monitor a human driver and assist him in navigation by warning in case of a developing dangerous situation. Contrary to the human drivers, these systems are not constrained by many physical and psychological limitations and therefore prove more robust in extreme conditions.

A key component of an intelligent vehicle system is the reliable perception of the environment. Laser range finders have been popular sensors which are widely used in this context. The classical 2D laser scanners have some limitations which are often compensated by the addition of other complementary sensors including cameras and radars. The recent advent of new sensors, such as 3D laser scanners which perceive the environment at a high spatial resolution, has proven to be an interesting addition to the arena. Although there are well-known methods for perception using 2D laser scanners, approaches using a 3D range scanner are relatively rare in literature. Most of those which exist either address the problem partially or augment the system with many other sensors. Surprisingly, many of those rely on reducing the dimensionality of the problem by projecting 3D data to 2D and using the well-established methods for 2D perception.

In contrast to these approaches, this work addresses the problem of vehicle perception using a single 3D laser scanner. First contribution of this research is made by the extension of a generic 3D mapping framework based on an optimized occupancy grid representation to solve the problem of *simultaneous localization and mapping* (SLAM). Using the 3D occupancy grid, we introduce a variance-based elevation map for the segmentation of range measurements corresponding to the ground. To correct the vehicle location from odometry, we use a grid-based in-

cremental scan matching method. The resulting SLAM framework forms a basis for rest of the contributions which constitute the major achievement of this work. After obtaining a good vehicle localization and a reliable map with ground segmentation, we focus on the *detection and tracking of moving objects* (DATMO). The second contribution of this thesis is the method for discriminating between the dynamic objects and the static environment. The presented approach uses motion-based detection and density-based clustering for segmenting the moving objects from 3D occupancy grid. It does not use object specific models but enables detecting arbitrary traffic participants. Third contribution is an innovative method for layered classification of the detected objects based on supervised learning technique which makes it easier to estimate their position with time. Final contribution is a method for tracking the detected objects by using Viterbi algorithm to associate the new observations with the existing objects in the environment.

The proposed framework is verified with the datasets acquired from a laser scanner mounted on top of a vehicle moving in different environments including urban, highway and pedestrian-zone scenarios. The promising results thus obtained show the applicability of the proposed system for simultaneous localization and mapping with detection, classification and tracking of moving objects in dynamic outdoor environments using a single 3D laser scanner.

Keywords: Intelligent vehicles, 3D laser scanner, perception, SLAM, DATMO, classification, occupancy grid

Résumé

Depuis des décennies, les chercheurs essaient de développer des systèmes intelligents pour les véhicules modernes, afin de rendre la conduite plus sûre et plus confortable. Ces systèmes peuvent conduire automatiquement le véhicule ou assister un conducteur en le prévenant et en l'assistant en cas de situations dangereuses. Contrairement aux conducteurs, ces systèmes n'ont pas de contraintes physiques ou psychologiques et font preuve d'une grande robustesse dans des conditions extrêmes.

Un composant clé de ces systèmes est la fiabilité de la perception de l'environnement. Pour cela, les capteurs lasers sont très populaires et largement utilisés. Les capteurs laser 2D classiques ont des limites qui sont souvent compensées par l'ajout d'autres capteurs complémentaires comme des caméras ou des radars. Les avancées récentes dans le domaine des capteurs, telles que les capteurs laser 3D qui perçoivent l'environnement avec une grande résolution spatiale, ont montré qu'ils étaient une solution intéressante afin d'éviter l'utilisation de plusieurs capteurs. Bien qu'il y ait des méthodes bien connues pour la perception avec des capteurs laser 2D, les approches qui utilisent des capteurs lasers 3D sont relativement rares dans la littérature. De plus, la plupart d'entre elles utilisent plusieurs capteurs et réduisent le problème de la 3ème dimension en projetant les données 3D sur un plan et utilisent les méthodes classiques de perception 2D.

Au contraire de ces approches, ce travail résout le problème en utilisant uniquement un capteur laser 3D et en utilisant les informations spatiales fournies par ce capteur. Notre première contribution est une extension des méthodes génériques de cartographie 3D fondée sur des grilles d'occupations optimisées pour résoudre le problème de cartographie et de localisation simultanée (SLAM en anglais). En utilisant des grilles d'occupations 3D, nous définissons une carte d'élévation pour la segmentation des données laser correspondant au sol. Pour corriger les erreurs

de positionnement, nous utilisons une méthode incrémentale d'alignement des données laser. Le résultat forme la base pour le reste de notre travail qui constitue nos contributions les plus significatives.

Dans la deuxième partie, nous nous focalisons sur la détection et le suivi des objets mobiles (DATMO en anglais). La deuxième contribution de ce travail est une méthode pour distinguer les objets dynamiques des objets statiques. L'approche proposée utilise une détection fondée sur le mouvement et sur des techniques de regroupement pour identifier les objets mobiles à partir de la grille d'occupations 3D. La méthode n'utilise pas de modèles spécifiques d'objets et permet donc la détection de tout type d'objets mobiles. Enfin, la troisième contribution est une méthode nouvelle pour classer les objets mobiles fondée sur une technique d'apprentissage supervisée. La contribution finale est une méthode pour suivre les objets mobiles en utilisant l'algorithme de Viterbi pour associer les nouvelles observations avec les objets présents dans l'environnement.

Dans la troisième partie, l'approche proposée est testée sur des jeux de données acquis à partir d'un capteur laser 3D monté sur le toit d'un véhicule qui se déplace dans différents types d'environnement incluant des environnements urbains, des autoroutes et des zones piétonnes. Les résultats obtenus montrent l'intérêt du système intelligent proposé pour la cartographie et la localisation simultanée ainsi que la détection et le suivi d'objets mobiles en environnement extérieur et dynamique en utilisant un capteur laser 3D.

Mots clés: véhicules intelligents, capteur laser 3D, perception, SLAM, DATMO, apprentissage et classification, grille d'occupation

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Building the artificial beings to replace humans for performing exhausting and dangerous tasks has been an aspiration since a long time. Literature reflects the efforts by many ancient civilization such as Chinese, Greeks and Egyptians who attempted to build the self-operating machines including the artificial people (Needham, 1991; Rosheim, 1994). Motivation behind all these efforts has been the comfort, convenience and safety of human beings.

In the last couple of decades, field of robotics has seen a huge transformation both in scope and dimensions. The reason behind this evolution is the maturity of the field as well as the general development in technology in recent years. The upcoming robots appear to meet the challenges of cohabiting dependably with the humans, both indoor and outdoor, facilitating them with the services as well as entertainment (Burgard *et al.*, 2000; Clodic, 2005; Jensen *et al.*, 2005). Moreover these intelligent machines can be used to perform tasks which are inherently dangerous for human such as cleaning the explosive mines, rescue operations in the disaster sites etc. All these things are possible because machines are not constrained by human limitations like eating, sleeping, breathing, temperature and pressure constraints and psychological limitations.

This recent advancement in robotics has impacted a wide range of research areas including automotive engineering, haptics, neuroscience, surgery, virtual simulation and sensor technology. On the other hand, the advancement in these and other newly emerging areas provides the incentives for further exploration in the field of robotics. Thus, the onset of new technologies opens new horizons for the research in this field.

One of these technologies which is a fundamental requirement for all the robotic applications and highly impacts their capabilities is the sensor technology. In order to perform its tasks, a robot must have a clear knowledge about its environment. This knowledge, commonly known as *environment perception*, includes information about the infrastructure as well as the entities which are temporarily present in the environment. The previously built maps can help in this regard but they are definitely not sufficient to define the actions of the robot as they lack information about the temporarily present entities (e.g. cars, humans, animals etc.) and they may not be up-to-date. For example, a recent change in the infrastructure, which is not updated in the map, can lead to a huge difference in traversable area of the environment. Thus, the robot's knowledge about the environment should never rely on the pre-built maps alone and it must have a mechanism to gather a precise and up-to-date information. This mechanism is ensured through the sensors.

Every robot, whether operating autonomously or semi-autonomously, is equipped with one or multiple sensors. There is a wide range of sensors available in market which vary in their type, dimension (2D/3D), range, resolution and price. The choice of sensors depends upon the environment in which the robot has to operate and the tasks it requires to perform. For instance, in the robotic applications operating indoor, such as tour guides or domestic robots, a short range sensor provides sufficient information while the outdoor applications, such as intelligent cars or planetary rovers require the long range sensors instead.

Other than range, a very important characteristic of the sensor is the dimension of its output. In robotics, state-of-the-art techniques for perception have been relying on the use of two dimensional sensors. These sensors scan the environment and capture information about a 2D slice of the world. One of the landmark advancements in sensor technology during the last decade is the introduction of three dimensional sensors for environment perception. These are the sensors that provide dense 3D point clouds corresponding to the environment surrounding a robot.

A question that strikes the mind is that in the presence of so many successful robotic applications with the help of 2D sensors, why do we need the expensive 3D sensors at all? Imagine the situation when a mobile robot crashes into an open drawer just because it is higher than the scan plane of the robot's collision detection sensor. This makes us realize that we live in a 3D world. To make this belief firm, assume a robot heading towards a flight of stairs. The robot's horizontal 2D scan-

ner depicts the area in the front as unobstructed and navigable. As a result, it will possibly continue in its heading direction and crash down the stairs. Furthermore, the targeted environments in which the robots are supposed to operate now-a-days no longer remain limited to the structured dedicated spaces. They vary from the urban streets to the barren and unpredictable landscapes of Mars, as illustrated in Fig. 1.1. None of these environments can be modeled fully and robustly with a flat-world assumption of the 2D sensors.

Figure 1.1: Two different environments where a robot may evolve which contain very different kind of geometry: urban with a lot of poles and Mars with sand desert.

These and other similar cases show that the sensors which scan the environment in 2D are not sufficient to serve the purpose for a robot. An extensively exploited solution to this problem has been the use of multiple sensors. Many research groups have combined several sensors to achieve an improved and relatively complete perception of the environment (Baig, 2012; Darms *et al.*, 2008; Nedevschi *et al.*, 2009; Petrovskaya, 2011). The disadvantage of this approach is that it introduces an extra overhead of sensor calibration and fusion of the data from multiple sensors. Moreover, such multi-sensor systems have their own limitations regarding the range, density, field of view (FoV) etc. Thus, some of the research groups in the automation industry have recently turned their interest towards exploring the perception with 3D sensors (Leonard *et al.*, 2008; Moosmann and Stiller, 2011; Nüchter *et al.*, 2007a; Urmson *et al.*, 2008).

An intended field for the deployment of these sensors is the autonomous vehicle[1] driving. Autonomous vehicles are highly safety critical applications and require a sufficiently large amount of data to perceive the environment. This is the reason

[1]In this dissertation, the terms autonomous vehicle, intelligent vehicle and driver less cars are used in the same sense.

why all such applications have been using multiple sensors as the information provided by a single 2D sensor is never enough to navigate safely. However, there are as yet many challenges to overcome in this field, especially related to extraction, combination and interpretation of the useful information from sensors' data to understand the external world. The emergence of 3D sensors has opened a new corridor for the research in this field.

We have also used a 3D laser range sensor[2] in this work and proposed a solution for the challenging problem of environment perception in the context of intelligent vehicles. It is the only perception sensor that we utilized in contrast to some recent approaches which have used the 3D laser sensor in collaboration with the other sensors. As the targeted applications for our work are the intelligent vehicles therefore, in the rest of the chapter, we first provide an introduction to the area of intelligent vehicles and the sensors they commonly use for gathering information about their environment. Then we detail the problem of perception for intelligent vehicles, its importance and its components. The chapter ends with a list of contributions of this work and organization of the rest of the dissertation.

1.1 Intelligent Vehicles

In the modern society, cars (vehicles) have become an essential part of our life which provide transportation with convenience. Vehicle industry is competing to provide modern facilities to make journey safer and more enjoyable. At the same time, rapid growth of the number of vehicles on road has increased the ratio of traffic accidents considerably. Numerous factors such as traffic signs, speed, overtaking, arrogance, driving skills, mechanical faults, tailgating, pedestrians, animals, overloading, etc. are involved in the road accidents that need to be considered carefully to improve the road safety.

This problem has captured the attention of not only the competitive vehicle industry but also the educational and research institutes. Over the decades, researchers from different fields of automotive engineering, robotics and other associated domains are trying to develop the intelligent systems for modern vehicles to make the driving experience safer and more convenient. The aspired systems can be divided into two categories; i) the monitoring applications designed to assist human drivers

[2]The terms laser range sensor, laser range scanner and lidar are used interchangeably.

in navigation and ii) the fully autonomous vehicles.

The first type of intelligent vehicles, known as the *advanced driving assistance systems* (ADAS), have already hit the market with the latest automotive technologies, as illustrated in Fig. 1.2. Toyota was the first to introduce a radar-based adaptive cruise control (ACC) system on a commercial vehicle sold in Japan in 1998 which was further refined by adding "break control" in 2000. Soon after, in 1999, Daimler also introduced their radar-based ACC in the market with their premium class vehicles. The systems such as lane departure warning, collision warning, lane change assistance, automatic parking, traffic sign recognition, parking assistance, driver drowsiness detection etc. are also commercially available now in the modern vehicles.

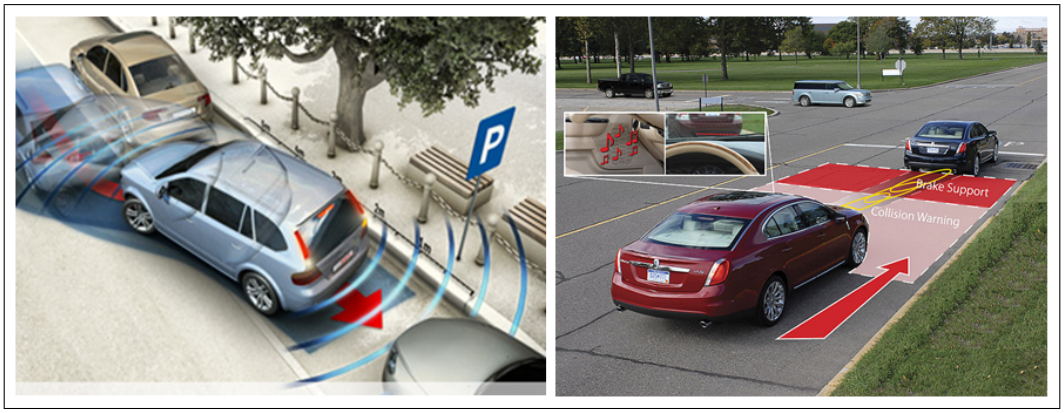


Figure 1.2: Advance Driving Assistance Systems (ADAS): Parking assistance system (left) and collision warning system (right).[3]

The second type of intelligent vehicles are the fully autonomous vehicles. The term fully autonomous is defined as "a mode of operation wherein the unmanned system (UMS) is expected to accomplish its mission without human intervention" (Huang *et al.*, 2005). A fully autonomous vehicle must possess the level of cognition to be able to reason about a designated mission and devise and execute a plan of action. However, building a vehicle with these capabilities is extremely difficult therefore fully autonomous vehicles have not been able to make to the market yet and they are still in the process of research. With the availability of very accurate positioning systems, and highly precise range-finders, such as the lasers, that can provide an abundant information about the surrounding of the vehicle at a sufficiently good frequency, the dream of achieving a fully autonomous vehicle does

[3]http://www.teach-ict.com/, http://rb-kwin.bosch.com/

not seem far from the sight. An illustration of this fact can be seen in the DARPA Urban Challenge[4] of 2007 where six of the participating autonomous vehicles successfully finished the entire course despite the challenging circumstances. These vehicles obeyed all the traffic rules as well as detecting and avoiding the other participants and moving obstacles on the road. The drawback of these vehicles that is a hindrance to their commercialization is the huge number of costly sensors that they used. Another addition to the newly developed, but not yet commercialized, autonomous vehicles has been the VIAC challenge of VisLab[5] and Google's self-driving car[6]. VisLab vehicles used vision cameras as main perception sensors to follow a man-driven car while the Google cars used multiple sophisticated sensors.

As of today, the intelligent vehicle systems mentioned above are not capable of dealing with all possible scenarios that can be encountered while driving, especially on busy city roads. A substantial hindrance to this capability is the required amount of information about the environment that may suffice for dealing with such situations successfully. To get that abundant information from the sensors is yet a challenge for the researchers. It is generally accepted that a single 2D sensor, due to its limitations and uncertainty (detailed in section 1.2), is not capable enough to provide all necessary information required for this complicated task. Using multiple sensors is a potential solution, however, combining information optimally from these sensors is a challenge (Durrant-Whyte *et al.*, 1990; Hall and Garga, 1999). Moreover, the individual limitations of the sensors further reduce the capability of these systems. The use of reliable 3D sensors seems to be an ideal solution due to the ample amount of information they provide at a reasonably high frequency. However, there are a number of problems that must be overcome to achieve the goal of perception with a 3D sensor. We have addressed those problems and proposed some possible solutions in this work.

In summary, we can say that the degree of autonomy of the intelligent vehicle depends on its capability of understanding the external environment using its sensors' data. Following section gives an overview of commonly used sensors with an analysis of their performance in the context of intelligent vehicles.

[4]http://archive.darpa.mil/grandchallenge/

[5]http://www.vislab.it

[6]http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/how-google-self-driving-car-works

1.2 Sensors

The sensors used in intelligent vehicle perception are generally divided into two categories: *vision-based sensors* (cameras) and *telemetry/range sensors* (lidar and radar). In this section, we discuss these sensors with their capabilities and limitations.

1.2.1 Vision-based Sensors

Vision-based sensors or cameras are the most widely used sensors owing to their high information content and low cost. Generally, in comparison to the range sensors, cameras provide high-resolution images at a high frame rate with low energy consumption. However, their biggest limitation is that they do not directly capture the distance to the objects. Another drawback is the high impact of lighting conditions, shadows and other ambient effects. Thus, unlike the range sensors, cameras do not provide useful information in the absence of sufficient lighting. In order to overcome these limitations, several special camera configurations are available. For instance, *stereo cameras* use two or more cameras to obtain binocular vision which generates the range data in addition to the images. This setup is very sensitive to calibration errors and has a small accuracy at larger distances. To avoid the impact of lighting conditions, *infrared cameras* are an alternative but they suffer from a high signal-to-noise ratio and a short range in comparison to the laser sensors.

1.2.2 Telemetry/Range Sensors

The commonly used telemetry sensors in intelligent vehicles are the radar and lidar.

Radar

Radar emits the radio signals and finds the distance to the obstacles by measuring the time delay or phase shift of the reflected signals. Radars are very precise for velocity estimation and perform well in bad weather conditions, however, they have narrow field of view and low resolution. Although they possess a long distance range but the measured range to the targets can suffer from echoes from the objects in the path of the wave thus corrupting the bearing to the objects. A possible solution to this problem is to use the vision sensors in addition to radars for improving the accuracy (Lundquist C, 2008; Richter E, 2008). Alternatively, the static returns

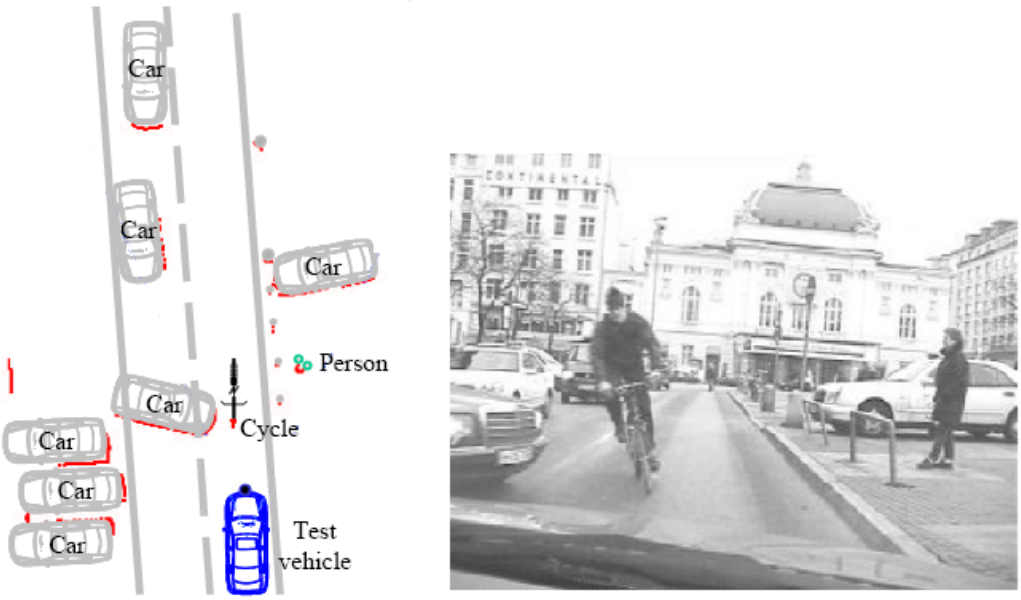


Figure 1.3: A goal of the vehicle perception with 2D lidar. Right: a real road scenario. Left: internal belief of the vehicle about it.

are filtered out to avoid the effect of echoes from the static targets and infrastructure but this reduces their suitability for the urban driving applications.

Lidar

Lidar uses the laser beams and measures their time-of-flight to retrieve the range. An advantage of this technique is that it works reliably in all lighting conditions. The available types of lidars are categorized as 2D, multilayer 2D, and 3D lidar.

2D Lidar: A 2D lidar uses a rotating mirror which deflects a laser beam into a plane. A receiver circuit detects this reflected beam and calculates the distance by computing the time elapsed between the emission and detection of the beam. It gives a very wide field of view (at least 120 degrees), however, the retrieved information is limited to a 2D slice of the world. Figure 1.3 is an illustration of the data produced by a 2D SICK lidar[7] mounted horizontally on a vehicle. The red dots represent the laser hits. A limitation of these sensors is that they scan the environment at a specific height. Any object above or below that height does not get detected. Hence, using a single 2D lidar proves insufficient for the purpose of vehicle perception.

Multilayer 2D Lidar: In order to avoid the problem mentioned above, multilayer

[7] http://www.sick.com/

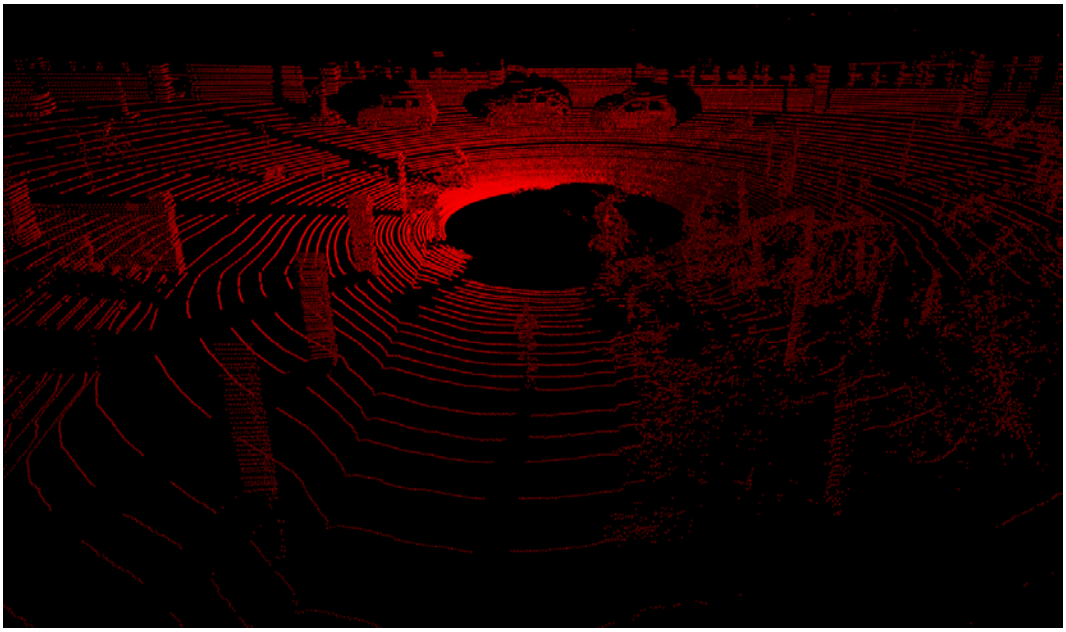


Figure 1.4: An illustration of the data provided by a single scan of Velodyne HDL-64E, a high definition lidar system for environment perception which consist of 64 laser beams rotating at 10Hz and covering a range of up to 120m.

2D lidars are introduced which produce more slices of information by using multiple laser beams and receiving circuits. For example, IBEO LUX[8] sensors produce four scan lines. These sensors can be used to construct an abridged 3D representation of the environment with a very low vertical resolution and density. To obtain a higher resolution and density, 3D lidars are a preferred choice.

3D Lidar: These sensors capture a dense 3D point cloud corresponding to the environment surrounding the vehicle. A highly appreciated example of such sensors is the Velodyne HDL-64E[9]. It retrieves the data at a frequency of 5-15Hz and a single scan contains more than 50,000 points (see Fig. 1.4). Thus it can provide the data at the rate of up to two million points per second. This sensor has all the aforementioned advantages of a range device, as illustrated in Table 1.1. It has 64 lasers aligned at different vertical angles mounted on a cylinder which rotates around the vertical axis. In each rotation, the lasers sweep the space and generate a 3D point cloud. This sensor was successfully used for vehicle detection in DARPA Urban Challenge 2007 for the first time. With the rich data generated by this sensor, the vehicle perception can be made more reliable in all scenarios. Moreover, 3D lasers

[8]http://www.ibeo-as.com/index.php/en/ibeproducts/sensorsv
[9]http://velodynelidar.com/lidar/hdlproducts/hdl64e.aspx

Table 1.1: Characteristics of different perception sensors

Sensor		3D Data	Range	Resolution	FoV	Light Impact	Cost
Camera	Mono	-	+	+	+	-	+
	Stereo	+	+	+	+	-	+
Radar		-	+	-	-	+	+
Lidar	2D	-	+	-	+	+	+
	Multilayer 2D	-	+	-	+	+	-
	3D (Velodyne)	+	+	+	+	+	-

provide data at high frame rates and with high information content. Unlike cameras, lasers give range information for every data point and can work in the dark.

In this work, We have used the 3D lidar for the environment perception for intelligent vehicles as it is far better in performance then the other sensors described above. The only question mark on the choice of this sensor is its cost but it becomes acceptable when we consider the criticality of the process of perception for an intelligent vehicle. In the following, we detail the problem of perception, emphasizing on its position and importance in the framework of an intelligent vehicle. We will also explain the different components of environment perception.

1.3 Perception for Intelligent Vehicles and the Challenges

Environment perception is an elementary task for reliable performance of the intelligent vehicle. It is one of the three components of fundamental robotic system architecture, also known as the *modern robotic paradigm* (Baum and Petrie, 1966). The three components of this architecture include *Perception*, *Decision* and *Action*, hence called the *PDA* architecture (see Fig. 1.5). This architecture implies that an intelligent system must be able to perform three main tasks:

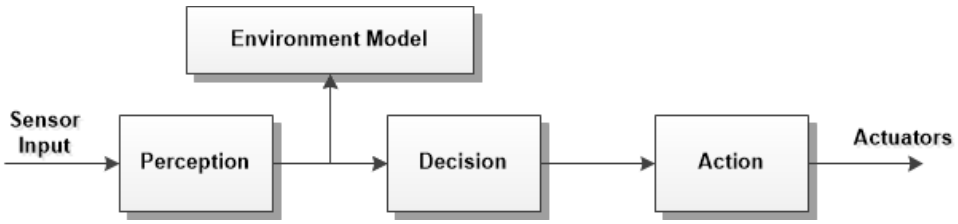


Figure 1.5: Modern robotic paradigm: Perception, decision and action. The decisions are based on the environment model provided by perception module, hence perception is the base of robotic decisions and actions in PDA architecture.

- Perceiving or modeling of the environment for scene understanding.
- Reasoning about the actions to perform, depending on perceived situation.
- Realizing those actions through the actuators.

Analogous to the humans, who perceive the surroundings using their senses, intelligent vehicles use their sensors to gather information about the environment. Human beings can assess the situation based on the sensory information thanks to their brains but the vehicles cannot do so. For instance, the raw information provided by a laser scanner is simply a set of ranges (red dots in Fig. 1.3) or a range image in the case of 3D sensor (Fig. 1.4). A human can possibly look at that image and identify different entities in it but for a machine, this information is useless until it is converted into some form which is understandable to it. Thus the purpose of perception systems in the intelligent vehicles is to transform that raw data acquired by the sensors into useful information and interpret it to understand the situation surrounding the vehicle.

The output of perception task is a representation of the scene surrounding the vehicle (e.g. position on the road, presence of obstacles, road signs, pedestrians and other vehicles etc.) which is further used to make the decisions (e.g. change of trajectory or speed over time). Those decisions finally become the base for the action for physically controlling the vehicle in order to navigate smoothly. Thus, the environment perception is the key to successful navigation of an intelligent vehicle and an erroneous and inaccurate environment model can lead towards wrong decisions consequently leading to the dangerous situations such as collisions.

1.3.1 Problem Statement

The perception problem is defined as the process which takes inputs from the sensors and outputs the internal state of the vehicle X, a static map of the environment M and a list of moving objects O around the vehicle (see Fig. 1.6). The sensor inputs include the measurements from perception sensors such as cameras or laser scanners as well as the measurements from motion sensors such as odometry or inertial measurements. The perception and motion measurements are denoted by Z and U respectively. Further, the vehicle state X comprises of the variables related to the vehicle such as its speed and relative pose with respect to the map M. The static map of the environment M consists of the information about the stationary objects

Figure 1.6: The general perception process. Z represents the perception measurements, U represents the motion measurements, X is the vehicle state, M is the map of stationary objects and O represents the states of moving objects.

and their locations in the map. The list of moving objects O consists of the information related to the dynamic objects including their locations as well as dynamic states such as direction of motion, velocity etc.

In order to represent the states which change with time, specific variables are used to show their values at certain times. For instance, x_t represents the vehicle state at time t. Thus the trajectory of the vehicle can be represented by:

$$X = x_{0:t} = \{x_0, x_1, ..., x_t\} \tag{1.3.1}$$

As the vehicle moves, its state x_t evolves, the motion sensors give the control measurement u_t for the evolved state whereas the perception sensors give the environment measurements z_t. These measurements up to time t are defined as:

$$Z = z_{0:t} = \{z_0, z_1, ..., z_t\} \tag{1.3.2}$$

$$U = u_{1:t} = \{u_1, u_2, ..., u_t\} \tag{1.3.3}$$

The static map M of the environment comprises of the stationary objects and the information related to them:

$$M = \{m^1, m^2, ..., m^k\} \tag{1.3.4}$$

Here, K is the total number of stationary objects, and m^k represents the location and properties of each object with $1 \leq k \leq K$. The list of moving objects O up to time t is denoted by:

$$O = O_{1:t} = \{O_1, O_2, ..., O_t\} \tag{1.3.5}$$

where O_t is the list of moving objects at time t consisting of a finite set of N objects:

$$O_t = \{o_t^1, o_t^2, ..., o_t^n\} \tag{1.3.6}$$

Here, each o_t^n with $1 \leq n \leq N$ contains the information regarding the locations and dynamic states of each moving object at time t.

Thus the perception problem is defined as the estimation of the vehicle state X, the static map M and the state of moving objects O given the sensor measurements Z and U over time. This problem is also called *Simultaneous Localization and Mapping* (SLAM) with *Detection and Tracking of Moving Objects* (DATMO).

A mathematical framework to solve SLAM with DATMO was introduced by Wang *et al.* (2003). The SLAM with DATMO problem was defined as a joint posterior over all unknown states (related to ego-vehicle, static and dynamic objects) given all sensor measurements (both perception and control measurements):

$$P(X, M, O|Z, U) \tag{1.3.7}$$

However, Wang *et al.* (2003) showed that estimating this joint posterior is computationally demanding and generally infeasible due to high dimension of the joint state variable. Therefore, they proposed to solve this problem by decomposing the measurements Z into static ($Z^{(s)}$) and dynamic ($Z^{(d)}$) measurements and thus decoupling the SLAM and DATMO problems such that:

$$P(X, M, O|Z, U) = P(X, M|Z^{(s)}, U)P(O|Z^{(d)}) \tag{1.3.8}$$

where $Z = Z^{(s)} + Z^{(d)}$. This decomposition of measurements corresponds to the detection of moving objects and the two terms in the posterior (1.3.8) correspond to SLAM ($P(X, M|Z^{(s)}, U)$) and moving object tracking ($P(O|Z^{(d)})$).

Maintaining separate posteriors for static and dynamic objects reduces the dimensionality of the problem of estimating SLAM with DATMO in (1.3.8) considerably as compared to the direct estimation of SLAM with DATMO in (1.3.7). We give a brief background in to the SLAM and DATMO problems in the next sub-sections.

1.3.2 Simultaneous Localization and Mapping

Simultaneous localization and mapping (SLAM), first introduced by Smith and Cheeseman (1986), allows robots to operate in an unknown environment and incrementally build a map of this environment and concurrently use this map to localize themselves. However, if the map is incomplete or the environment is very symmetrical then the robot cannot be certain of its position. One popular solution for outdoor applications is the use of Global Positioning System (GPS). GPS allows a robot to estimate its global position in the world easily and quite accurately . However, it suffers from problems with noise and errors caused by large structures, such as the buildings in urban areas.

In most SLAM applications, initial estimate of the pose is given by odometry or some other mechanism which is corrected using observations from the environment and the stored map. On the other hand, the corrected pose helps to update the map by aligning the previous and current information. Thus, classically, the problem of SLAM consists of two steps: environment representation and map association. There are three general approaches to the environment representation: grid-based (Elfes, 1989), feature-based (Dissanayake *et al.*, 2001), and direct methods (Cole and Newman, 2006). These approaches are further discussed in Chapter 2. When a 3D range sensor is used, a considerable amount of the acquired information corresponds to the ground (see Fig. 1.4). This introduces the demand for another crucial step for mapping, known as the ground identification and segmentation (B. Douillard, 2010; Himmelsbach, 2010).

In the last couple of decades, the SLAM problem has been explored immensely (Moosmann and Stiller, 2011; Nüchter *et al.*, 2007b; Wang and Thorpe, 2004), and SLAM techniques have become the base of many successful robotic applications (Nüchter *et al.*, 2007a; Thrun *et al.*, 2006). However, due to the static environment assumption, SLAM alone has proven to perform badly in the crowded urban environments and, as explained by Wang and Thorpe (2002), an intelligent vehicle must also be capable of detecting and avoiding the moving obstacles.

1.3.3 Detection, Classification and Tracking of Moving Objects

Many approaches for moving object tracking suppose that the measurements correspond uniquely to moving objects, however, most of the real applications include static objects and spurious elements in the measures. The approaches that assume the absence of static objects track all the objects in the environment, regardless of being static or dynamic. As a result, they suffer from an unneeded complexity in the system. To simplify the problem, some approaches filter out the objects that do not fit the expected criteria for moving objects. For instance, Labayrade *et al.* (2005) propose to remove the objects which are not on the road from the consideration for dynamic objects while Nashashibi and Bargeton (2008) ignore the objects which are too large. Despite reducing some objects, these approaches still have a high complexity, specially in the context of crowded environments. Therefore, the preferred approaches for the intelligent vehicles are the ones that discriminate the dynamic objects from static environment.

Differentiating between the static and dynamic objects is a critical aspect of a moving object tracking system. Moving object detection in crowded urban environments is not easy because of a wide variety of targets. Vision-based applications use feature-based or appearance-based recognition approaches to detect moving objects (Taleghani *et al.*, 2009; Viola and Jones, 2001). When laser is used, motion-based approaches (Vu, 2009; Wang *et al.*, 2003) are usually the preferred solution since both appearance-based and feature-based methods rely on prior knowledge of the targets. These approaches are further discussed in Chapter 3.

After moving objects are identified, multiple object tracking problem arises in order to estimate dynamic states of each object. It has to deal with the data association problem and maintenance of a list of objects currently present in the environment as well as their dynamic models (Bar-Shalom and Fortmann, 1988). In general, clutters, occlusions or missed detections from the detector could cause more challenging situations to the data association step. In addition, changing motion behaviors of moving objects make defining a suitable motion model of the tracked objects more difficult. Identifying the class of the objects can be of help in this regard for the choice of the appropriate dynamic model.

In the computer vision literature, algorithms for object classification have been attained in a mature stage, however, so far very few researchers have addressed this problem with laser sensors. Since all data returned by laser scanner are discrete points of impact on moving objects (see Fig. 1.3), it is hard to define distinguished features to identify different kinds of objects. The dimensions of the detected objects, line segments, or rectangular bounding boxes are the most commonly used features in the classification process in 2D. Typical length and width values are usually compared against the modeled bounding boxes to choose an object class (Petrovskaya, 2011; Vu, 2009). Moreover, in case of 2D, this step is mostly avoided by performing the object tracking directly after detection. A useful clue is often the use of estimated velocities of moving objects. However, in the case of 3D range data, it is not that implicit as the detected objects usually consist of the partial point clouds. These point clouds can neither be tracked directly nor can be classified by using the dimensions of the bounding box alone. Therefore, a more sophisticated approach for classification is required to identify the objects from 3D laser data. The existing approaches range from the vision-based classification from range images (Lee *et al.*, 2010) to the feature-based classification (Himmelsbach *et al.*, 2008) from raw 3D point clouds. The problem with the former method is that it converts

the data to 2D image, hence losing the valuable detail of the 3D point cloud. On the other hand, the latter methods usually consume a generous amount of time for feature extraction and representation from the raw point clouds. As a result, they become less interesting for the time critical application of object tracking for intelligent vehicles. Chapter 3 provides a detail about the existing approaches for classification and tracking of moving objects as well as a novel approach proposed for classification and tracking in 3D.

1.3.4 SLAM with DATMO

Since Wang (2004) combined the SLAM and DATMO into a single problem known as SLAMMOT[10] and demonstrated its feasibility using laser sensor, it has attracted extensive research interest in the robotics literature. It proves to be a basic component for the robotic systems exploring in the unknown and unstructured environments. Vu (2009) provided a promising framework to solve this problem using 2D laser sensor. Recently, the trend has been moving to using cameras in addition to laser. The successful illustrations include the work of Petrovskaya (2011) and Baig (2012) who used stereo-vision cameras with laser while Chavez-Garcia *et al.* (2012) complemented it with mono-vision and radar. Stereo-vision is also used to perform SLAM with DATMO in 3D (Nedevschi *et al.*, 2009) but a more attractive and emerging area in this regard is the use of 3D laser. Many recent works have utilized it with a combination of other sensors (Montemerlo *et al.*, 2008; Moosmann and Stiller, 2011; Nüchter *et al.*, 2007a).

1.4 Contributions

Intelligent vehicle perception in 3D is becoming a hot area of research. The 3D laser range scanner has turned out to be a sensor of choice for many mobile applications in the last few years. The first successful illustration of this technology was the DARPA Urban challenge 2007 where five of the six teams which managed to finish the complete course of the race were equipped with the Velodyne HDL-64E 3D laser scanner (Schwarz, 2010). A more recent example is the Google's driver-less car, which used the same sensor. A limitation for these vehicles is that they used a number of sensors (cameras, radars, 2D lidars, GPS etc.) in addition to the 3D

[10]Simultaneous localization and mapping with moving object tracking

lidar, which increase the cost as well as processing complexity. For instance, in the DARPA Urban challenge, the winning vehicle "Boss" of the team Tartan Racing (Urmson *et al.*, 2008) was equipped with a combination of 18 laser and radar sensors besides a Velodyne HDL-64E. The vehicle "Talos" of Team MIT (Leonard *et al.*, 2008) used a combination of 12 planar lidars, 12 radars and a Velodyne HDL-64E for environment perception. Similarly, the sensors on the Google cars include four radars, a camera and a GPS in addition to the 3D Velodyne laser scanner and high-resolution maps of the world[11].

In contrast to these approaches, we propose that a single 3D laser range scanner is sufficient to deal with the problem of vehicle perception. The use of other sensors, in addition, creates redundancy and workload. To address this problem, we started by exploring the methods of vehicle environment representation focusing on using 3D laser scanner as the main perception sensor. Further, we explored the methods for detection, classification and tracking of moving objects as well. We review the state-of-the-art approaches to SLAM and DATMO briefly in terms of the environment representations in this thesis. The main contributions of this research are as follows:

First contribution of this research is made by a baseline solution to the problem of SLAM in 3D based on an optimized occupancy grid to represent the environment. In order to extend a generic 3D mapping framework to use it for solving the SLAM problem, we introduce a variance-based elevation map technique for the detection and segmentation of the ground. To correct the vehicle location from odometry, we introduce a grid-based incremental scan matching method. Our approach extends the traditional Iterative Closest Points (ICP) algorithm to work reliably and efficiently in 3D environment. Experimental results on datasets collected from different scenarios demonstrate the validity of the method.

Second and more important contribution follows the first results. After obtaining a good vehicle localization and a reliable map with ground segmentation, we focus on the dynamic objects in the environment. We make the hypothesis about the dynamic voxels based on the inconsistencies between the constructed grid map and the newly received scans. These hypotheses are then refined by performing the density-based clustering of the voxels and the classification into the known object categories. We introduce a novel layered approach using boosting for the

[11] http://www.youtube.com/watch?v=YXylqtEQ0tk

supervised classification of the 3D dynamic objects. We test the proposed algorithm on real-life data of urban traffic and present promising results.

Third contribution is a method for robust tracking of dynamic objects. The position of the dynamic objects is predicted by using Kalman filter (KF). For data association, we have proposed to use the Viterbi Algorithm to find the most likely associations between the detected moving objects and the newly acquired observations. The algorithm is implemented and tested on real environment for the tracking of both single as well as multiple objects.

1.5 Thesis Organization

The remaining of this dissertation is organized as follows:

Chapter 2 starts with an overview of the background and state-of-the-art techniques used for simultaneous localization and mapping in the context of intelligent vehicles. Then we detail the 3D octree-based occupancy grid mapping approach we adopted for environment representation. Finally, we present our method to extend this approach for SLAM by introducing ground segmentation and localization.

Chapter 3 covers the topics of detection, classification and tracking of dynamic objects from the grid map. The detection is achieved through scan comparison and density-based clustering while the classification relies on converting the problem to 2D, splitting each dynamic cluster into the horizontal layers. Each layer is then used to compute the weak classifier. The resulting strong classifier is obtained from the combination of the weak classifiers through boosting. Finally we present our method for multiple object tracking using Viterbi data association with a description of the method for track maintenance.

Chapter 4 provides the implementation details and the experimental evaluations performed to test our framework for perception in dynamic outdoor environments. Both qualitative and quantitative results are presented for different components of the framework.

Chapter 5 concludes the dissertation with a summary of our work, and thoughts about the future perspectives of research.

Chapter 2

Simultaneous Localization and Mapping

2.1 Introduction

Simultaneous Localization and Mapping (SLAM), also known as *Concurrent Mapping and Localization* (CML), is the process of building a map of the environment from sensor data and at the same time using that map to determine the vehicle's location. The sensor data includes the information from both internal or *proprioceptive* sensors as well as external or *exteroceptive* sensors (see Fig. 2.1). As the term implies, SLAM is composed of two sub-problems: *localization* and *mapping*. Localization is the problem of estimating the pose of the vehicle given a map and a sequence of sensor measurements whereas mapping addresses the problem of building a map given the poses of the vehicle and a sequence of sensor measurements. Thus an accurate map is essential for reliable localization and an accurate localization is a prerequisite for constructing a consistent map. Consequently, in practice, these two problems cannot be solved independent of each other.

Figure 2.1: Simultaneous Localization and Mapping (SLAM).

In most applications, the initial position estimate is known using the propriocep-

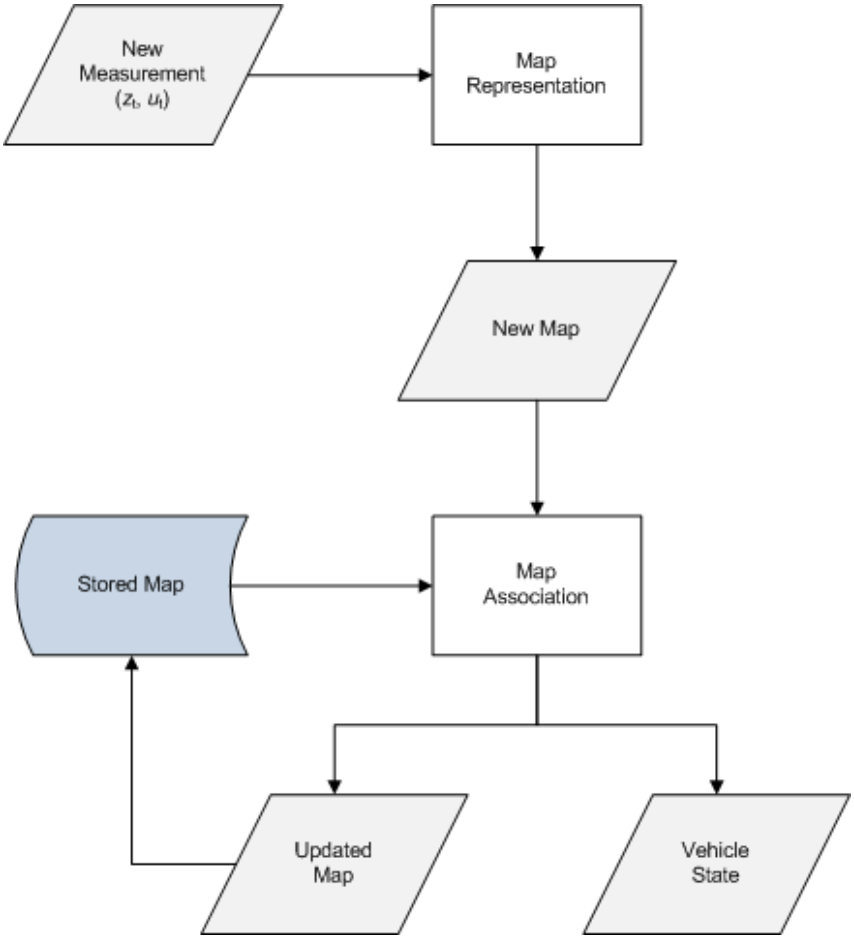


Figure 2.2: Simultaneous Localization and Mapping: Generic steps.

tive sensors. The vehicle starts the mapping process incrementally by translating its external sensor measurements into the map representation. In the next step, when the vehicle moves, it predicts its pose using the internal sensors and integrates the newly observed exteroceptive information into the existing map thus updating the map and the pose at the same time. This process is repeated for each new set of measurements as shown in Fig. 2.2. As explained in chapter 1, if the static measurements can be differentiated from the dynamic measurements, then SLAM and DATMO can be handled as independent processes. In this chapter, we assume that we have the static measurements only and address the problem of SLAM alone. The dynamic measurements and the problem of detection and tracking of moving objects will be addressed in chapter 3.

There are a number of methods developed to perform SLAM in both indoor and outdoor environments. The underlying methods to solve the SLAM problem de-

pend on the type of the sensor and the constraints imposed by the application scenario. Along with this, these methods also differ in their representation of the environment as well as their dimensionality (2D/3D). The first part of this chapter describes the important solutions proposed for the SLAM problem in last two decades. First, in section 2.2, we give the background and mathematical formulation of the SLAM problem. A review of the related work on SLAM follows in section 2.3 which provides a baseline for the contributions of this chapter. The second part of the chapter starts with a description of the approach that we have adopted for 3D mapping in this work. After providing an overview and mathematical formulation of the adopted framework in section 2.4, we discuss its merits as well as limitations. Section 2.5 gives a summary of our contributions in the context of SLAM based on the selected mapping framework. Our methods for ground segmentation and localization are detailed in section 2.5.2 and 2.5.3 respectively.

2.2 Background

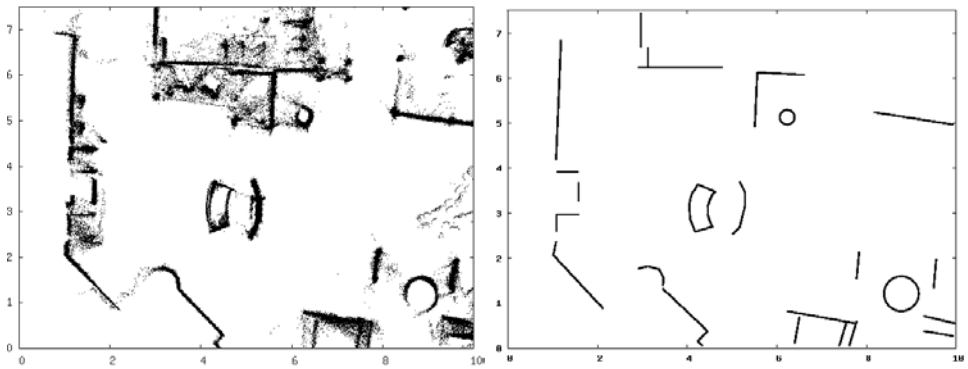
In this section we provide a background study of the SLAM problem. In section 2.2.1, we introduce the popular methods for map representation as they highly influence the choice of relevant method for SLAM. Next, we introduce the probabilistic framework for SLAM and its common implementations in section 2.2.2.

2.2.1 Map Representations

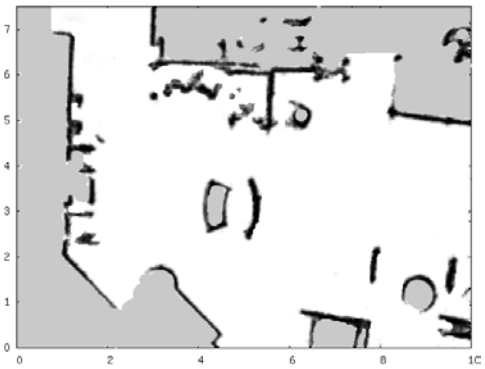
Th map representation methods are generally characterized in four categories – grid-based approaches (Elfes, 1989), feature-based approaches (Dissanayake *et al.*, 2001), direct approaches (Cole and Newman, 2006) and topological approaches (Choset and Nagatani, 2001). We give a brief introduction of each of these methods in the following.

Direct Representation

Direct approach represents the environment using raw data measurements without simplification or extraction of features. This representation is often used with the range sensors such as laser scanners. In this case, each data measurement, known as a *scan*, is a set of points (2D or 3D) which are impacts of laser beam with the obsta-



(a) Point cloud map **(b)** Feature map



(c) Grid map

Figure 2.3: Example of different map representations built from the same set of range measurements acquired from a 2D laser scanner.

cles in the environment. Data association between two scans is usually performed by using iterative closest point (ICP) algorithm or its variants (Lu and Milios, 1997; Nüchter *et al.*, 2005). The resulting representation is a point cloud map as shown in Fig. 2.3a. It is a simple approach, however, it requires significant amount of memory and processing power to handle the large size of point clouds. Further, it does not provide a direct mechanism to identify the objects.

Feature-based Representation

Feature or landmark-based approaches compress raw data into a set of features defined in accordance with the operating environment. The features are usually geometric primitives such as points, lines, circles etc. The common choice of features for indoor and structured outdoor environments are line segments due to the presence of many straight edges (Siadat *et al.*, 1997) while circles are often used to represent the tree trunks and tree-like objects, such as pillars, present in the semi-

structured, outdoor environments (Zhang *et al.*, 2003). Figure 2.3b is an illustration of the two-dimensional geometric feature map. A limitation of this approach is not being able to model complex outdoor environments as well as the spaces between the features. A method proposed to overcome this limitation is to use artificial landmarks and add recognizable features to the environment. This approach is not feasible in real life situations such as intelligent vehicles as it requires major changes in the infrastructure.

Grid-based Representation

Grid-based approaches discretize the environment into a a series of cells (or *voxels* in 3D), known as *occupancy grid* or *evidence grid* (Wang and Thorpe, 2004). Each cell is assigned a value based on its occupancy inferred from the sensor data. In a simple binary occupancy grid, the cell can be either occupied or free but in a more sophisticated approach, value of the cell represents the probability of occupancy. This probability is updated whenever the new sensor measurements arrive. In the literature, many methods are proposed for this update including Bayesian filtering (Elfes, 1992; Thrun *et al.*, 2005), Dempster-Shafer theory (Gambino *et al.*, 1997; Pagac *et al.*, 1998) and Fuzzy Logic (Oriolo *et al.*, 1997). Figure 2.3c is an example of an occupancy grid map representation, free cells are shown in white and occupied cells in black. A big merit of this approach is the compensation for sensor uncertainty by probabilistic modelling of the occupancy. Moreover, sensor fusion is also straightforward as multiple sensors can be used to update the occupancy of a cell.

Topological Representation

Topological maps represent the environment as adjacency-graph (either 2D or 3D) composed of a finite number of nodes and links. Nodes represent the particular locations in the environment, called *distinctive places*, that the system can recognize while the links represent the paths between those nodes and contain procedural information to go from one node to another (Choset and Nagatani, 2001). The assumption is that the distinctive places can be locally distinguished from the surrounding area. These maps are usually developed on top of the grid-based or feature-based maps by partitioning them into coherent regions, and thus termed structured feature maps.

2.2.2 Mathematical Formulation

As the data provided by the sensors is inherently uncertain and corrupted by noise, uncertainty modelling is the principal problem for any SLAM algorithm. Different sensors have different noise and accuracy characteristics which must be modelled appropriately. This can be done by using probabilistic approaches. Csorba (1997) developed a probabilistic method to solve the problems of localization and mapping simultaneously. He proposed to explicitly model the spatial-relationships between landmarks in an environment while simultaneously estimating the pose of a robot. Since its introduction, a probabilistic approach is considered as the standard method for modelling the SLAM problem and has stimulated a considerable amount of research.

In the probabilistic form, the goal of a SLAM algorithm is to estimate the following probability distribution:

$$P(x_t, M|z_{0:t}, u_{1:t}) \tag{2.2.1}$$

This probability distribution, known as the *SLAM posterior*, describes the joint probability density of the map M and vehicle state x_t at time t given the measurements $z_{0:t}$ and control inputs $u_{1:t}$ up to time t. As the measurements sequentially arrive over time, a recursive solution to SLAM is desirable. Bayesian filtering (Anderson and Moore, 1979), also called *Bayesian Sequential Estimation*, provides this solution by recursive estimation of the state of a system and it is the foundation of most of the SLAM approaches. In this section, we will provide the basic formulation of Bayesian filter as derived by Montemerlo *et al.* (2007), and then discuss its major implementations practically used in the SLAM applications.

Bayesian Filter

Bayesian filter is a general probabilistic framework to estimate the dynamic states of a system evolving in time given sequential observations about that system. If the general state space model is defined by the dynamic states as hidden variables $\mathbf{h}_t = \{h_0, h_1, ..., h_t\}$ and observed variables $\mathbf{o}_t = \{o_0, o_1, ..., o_t\}$, the goal is to perform inference on the hidden variables given the observed variables. This inference relies on the joint posterior distribution $P(\mathbf{h}_t|\mathbf{o}_t)$. The graphical model corresponding to this problem, called the *Bayesian belief network*, is shown in Fig. 2.4.

Two important assumptions of this framework are that the states follow a first order

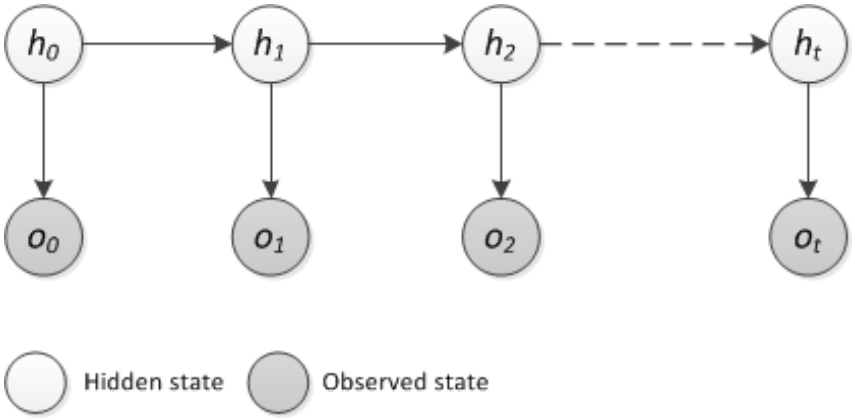


Figure 2.4: A graphical model for generic Bayesian filter, where the observed states o_i are dependent on the hidden states h_i.

Markov process (i.e. the current state depends only on the previous state) and the observations are independent of the given states. Thus, assuming an initial distribution of the hidden variables $P_0(h_0)$ and a state transition model $P(h_t|h_{t-1})$, we can use the Bayesian filter to derive a recursive expression for the posterior:

$$P(\mathbf{h}_t|\mathbf{o}_t) = P(\mathbf{h}_{t-1}|\mathbf{o}_{t-1})\frac{P(o_t|h_t)P(h_t|h_{t-1})}{P(o_t|\mathbf{o}_{t-1})} \tag{2.2.2}$$

In case of the SLAM problem, hidden variables are the vehicle position x_t and the map M, while the observed variables are the perception sensor measurements z_t and motion measurements u_t. This leads to the SLAM posterior (2.2.1):

$$P(\mathbf{h}_t|\mathbf{o}_t) = P(x_t, M|z_{0:t}, u_{1:t}) \tag{2.2.3}$$

In order to derive the recursive expression for this posterior, we expand it by using Bayes' rule:

$$P(x_t, M|z_{0:t}, u_{1:t}) = \eta P(z_t|x_t, M, z_{0:t-1}, u_{1:t})P(x_t, M|z_{0:t-1}, u_{1:t}) \tag{2.2.4}$$

Here, η is the normalization constant ensuring that Eq. 2.2.4 correctly represents a probability distribution.

As the perception measurement z_t is independent of the prior perception measurements $z_{0:t-1}$ and the motion measurements $u_{1:t}$ given the pose x_t and the map M, Eq. 2.2.4 can be simplified to:

$$P(x_t, M|z_{0:t}, u_{1:t}) = \eta P(z_t|x_t, M)P(x_t, M|z_{0:t-1}, u_{1:t}) \tag{2.2.5}$$

Using the Total Probability Theorem,

$$P(x) = \int P(x|y)P(y)dy \tag{2.2.6}$$

the rightmost term of Eq. 2.2.5 can be conditioned on the prior pose by integrating over all poses at time $t-1$. Thus:

$$P(x_t, M|z_{0:t}, u_{1:t}) = \eta P(z_t|x_t, M) \int P(x_t, M|x_{t-1}, z_{0:t-1}, u_{1:t}) P(x_{t-1}|z_{0:t-1}, u_{1:t}) dx_{t-1} \tag{2.2.7}$$

Using the definition of conditional probability to expand the first term in the integral, we get:

$$P(x_t, M|x_{t-1}, z_{0:t-1}, u_{1:t}) = P(x_t|M, x_{t-1}, z_{0:t-1}, u_{1:t}) P(M|x_{t-1}, z_{0:t-1}, u_{1:t}) \tag{2.2.8}$$

Moreover, according to our assumption, x_t only depends on x_{t-1} and u_t, therefore:

$$P(x_t|M, x_{t-1}, z_{0:t-1}, u_{1:t}) = P(x_t|x_{t-1}, u_t) \tag{2.2.9}$$

and Eq. 2.2.8 can be rewritten as:

$$P(x_t, M|x_{t-1}, z_{0:t-1}, u_{1:t}) = P(x_t|x_{t-1}, u_t) P(M|x_{t-1}, z_{0:t-1}, u_{1:t}) \tag{2.2.10}$$

Now, using the definition of conditional probability:

$$P(M|x_{t-1}, z_{0:t-1}, u_{1:t}) P(x_{t-1}|z_{0:t-1}, u_{1:t}) = P(x_{t-1}, M|z_{0:t-1}, u_{1:t}) \tag{2.2.11}$$

Replacing Eq. 2.2.10 and 2.2.11 in Eq. 2.2.7:

$$P(x_t, M|z_{0:t}, u_{1:t}) = \eta P(z_t|x_t, M) \int P(x_t|x_{t-1}, u_t) P(x_{t-1}, M|z_{0:t-1}, u_{1:t}) dx_{t-1} \tag{2.2.12}$$

The last motion measurement u_t does not provide any information about x_{t-1} without the perception measurement z_t. By eliminating it from the above relation, we get the recursive formulation of SLAM posterior similar to the recursive Bayesian filter equation (Eq. 2.2.2):

$$\underbrace{P(x_t, M|z_{0:t}, u_{1:t})}_{\text{posterior at } t} = \eta \underbrace{P(z_t|x_t, M)}_{\text{measurement model}} \int \underbrace{P(x_t|x_{t-1}, u_t)}_{\text{motion model}} \underbrace{P(x_{t-1}, M|z_{0:t-1}, u_{1:t-1})}_{\text{posterior at } t-1} dx_{t-1} \tag{2.2.13}$$

The posterior density, as given in Eq. 2.2.13, is described by the following three terms:

- **Previous posterior** $P(x_{t-1}, M|z_{0:t-1}, u_{1:t-1})$**:** For the first iteration, this probability needs to be defined as an initial distribution known as *prior*. For each subsequent iteration, it is deduced from the posterior of the previous iteration.

- **Motion model**[1] $P(x_t|x_{t-1}, u_t)$**:** It describes where the vehicle might be at time t, given that it was previously at location x_{t-1}. This model strongly depends on the information provided by the odometry measurements for the estimation.

- **Measurement model**[2] $P(z_t|x_t, M)$**:** This model describes the likelihood of making observation z_t given that the vehicle is at location x_t in map M. It is usually considered as a property of the given sensor technology. It depends on the type of the sensor and captures its error characteristics.

Any implementation of the Bayesian filter needs to explicitly define these models according to the capabilities of the vehicle and its environment. Moreover, the Bayesian filter in Eq. 2.2.13 contains an integral over all previous vehicle positions which is usually intractable to compute and requires to be approximated. The most popular approximations involve using Kalman filters (Kalman *et al.*, 1960) and particle filters (Arulampalam *et al.*, 2002).

Kalman filter is the most widely used variant of Bayesian filters which uses the Gaussians to represent the posteriors (Bar-Shalom *et al.*, 2001). It relies on the assumption that the motion model and measurement model are linear functions with added Gaussian noise and the initial distributions are also Gaussian (Kalman *et al.*, 1960). As most systems are not strictly linear, there are variations of Kalman filters proposed to accommodate the non-linearities from the real world. The Extended Kalman Filter (EKF) (Gordon *et al.*, 1993) is an extension which linearizes the system using first-order Taylor series expansions of the non-linear distributions. In case of highly non-linear and non-Gaussian systems, particle filters (Arulampalam *et al.*, 2002) are a better choice than Kalman filters in terms of accuracy. Particle filters represent the posterior distribution with a set of samples that have weights associated to them.

[1] Also known as *state transition function* and *dynamics model*

[2] Also known as *likelihood function*, *sensor model* and *perceptual model*

2.3 Related Work

Simultaneous localization and mapping is a crucial component of a mobile robot. It allows the robot to navigate from an unknown location, building the map of an unknown environment from the measurements of perception sensors and concurrently estimating its pose with respect to that map. The general problems of mobile robot navigation were summarized by Leonard and Durrant-Whyte (1991a). The navigation strategies started with *line following* in 1970s and evolved to *beacon-based systems* in 1990s. Those systems were extremely limited in the context of their dependence on the specific features introduced in the infrastructure and could not be used in unknown and unstructured environments. The development of SLAM frameworks has made the mobile robots capable of overcoming these limitations.

In the early work, researchers addressed the localization and mapping as two independent problems. The systems thus introduced assumed that some part of SLAM was performed manually. For instance, the localization algorithms were usually based on the pre-built maps while the mapping was done using known or guided localization. The amount of human effort required for such systems can be assessed by the example of creating a map of the museum in Bonn for the robot RHINO which, according to Thrun (1998), took one week of hard work. The major breakthrough in this regard was provided by the seminal work of Smith, Self, and Cheeseman (1988). They proposed a feasible solution for SLAM by establishing a powerful statistical framework for probabilistic treatment of uncertainty in geometric relationships and parametrization. Other pioneering work is considered to be that of Leonard and Durrant-Whyte (1991b). Since then, the field of SLAM has been generally dominated by probabilistic techniques.

The underlying difficulties of SLAM have been addressed relentlessly in last couple of decades and several solutions are proposed. At a theoretical level, SLAM can be considered a solved problem even in a populated environment (Hahnel *et al.*, 2002). However, the issues still remain in realizing the general SLAM solutions practically and building perceptually rich frameworks. The proposed methods differ in various aspects therefore it is hard to cluster and categorize them. Map representation is an attribute which is common to all the proposed methods therefore, in this section, we review them on the basis of commonly used map representations in mobile robot navigation (A slightly outdated classification on the basis of map representation is provided in (Thrun, 2002)). Although there is an ample amount of research

and literature relevant to SLAM in robotics community, in this section a few of those will be considered in connection to the approach presented in this thesis.

2.3.1 Direct mapping SLAM

Direct mapping is an approach proposed by Lu and Milios (1997) to avoid the issue of feature detection for localization. Lu and Milios (1997) used the odometry measurements and raw scan data acquired by a 2D laser scanner to build a sparse network of constraints between vehicle poses which are solved with batch-iterative optimization methods. Gutmann and Konolige (1999) extended this batch method by allowing incremental updates and updating the map with every new sensor input. They also introduced the method to close the loops in indoor environments.

Some recent works have shown the success of direct mapping with the sensors such as 3D laser scanner and stereo cameras (Cole and Newman, 2006; Nüchter *et al.*, 2007a). These sensors provide dense point clouds which are stored directly to represent the occupied space in the environment. The point clouds can be aligned pairwise using Iterative Closest Points (ICP) algorithm or its variants (Besl and McKay, 1992). This algorithm is applied sequentially to the scans as they are collected through the course of the robot (Nüchter *et al.*, 2007a). Despite being simple to describe, dense alignment algorithms have demonstrated impressive performance by producing accurate results under normal conditions.

The main idea of ICP is to search for the pairs of closest points in two different scans and iteratively calculate their optimal transformation by minimizing the error function. Thus it performs the localization in addition to scan registration. For minimizing the error function, four direct methods are listed in literature (Eggert *et al.*, 1997): singular value decomposition (SVD) based method (Arun *et al.*, 1987), quaternion method (Horn, 1987), an algorithm using orthonormal matrices (Horn *et al.*, 1988) and a calculation based on dual quaternions (Walker *et al.*, 1991). Eggert *et al.* (1997) have shown that all four methods demonstrate a similar performance and stability in the context of noisy data.

The major issue with the ICP based algorithms is that they require an initial estimate to initialize the algorithm. Thus the quality of alignment is highly dependent on the initial estimate. In general, this estimate is provided by the odometry of the vehicle and then refined iteratively through ICP. The odometry measurements are

usually uncertain over long distances thus making the initial estimate unreliable.

An obvious solution to limit this uncertainty is to reduce the spacing between the scans which leads to the incremental mapping scenario. King *et al.* (2005) proposed to improve the quality of the initial estimate by using a feature-based method for initial alignment and dense point cloud for further refinement. They use an intensity camera associated with the laser scanner to obtain the intensity images and extract a combination of interest points from them. These interest points are used as features to generate transformation hypothesis. Then a region growing variant of ICP is used to align the two scans. A similar alignment approach incorporating point features for initial estimate, called *coarse-to-fine scan registration* is used by Brenner *et al.* (2008) for outdoor urban environments.

Another issue with direct alignment of the point clouds arises for the large datasets where dense data can prove to be computationally intractable (Pulli, 1999). Fairfield *et al.* (2010) use the segmentation of point cloud into metric sub-maps and a topological map of the relationships between the sub-maps to limit the growth of computational requirements. In order to resolve the inconsistencies between the pairwise associations of the points, Borrmann *et al.* (2008) present a global relaxation technique by distributing the error over the entire map. Though this approach resolves the inconsistencies, however, the performance is still dependent on the initial estimate.

2.3.2 Feature-based SLAM

Feature-based approaches were the first methods used in the early days of SLAM. Features were extracted from the sensor data and combined with motion controls using an Extended Kalman Filter (EKF) approach (Smith *et al.*, 1990). Moutarlier and Chatila (1990) used the line-segment features extracted from the laser scans while Leonard and Durrant-Whyte (1991a) used the geometric features extracted from the sonar data and both used EKF for localization of the robot. A major advantage of using EKF is that it ensures the association between the features by maintaining a complete covariance matrix and mean vector for all the features.

The choice of features depends on the sensor as well as the environment in which the robot operates. In indoor and structured outdoor environments, line segments are a popular choice due to many straight edges. There are three major techniques

for line extraction; *Successive Edge Following*, *Line Tracking* and *Iterative End Point Fit* (IEPF) (Siadat *et al.*, 1997). According to the comparison made by Siadat *et al.* (1997), IEPF generates a better environment representation than the others. Nguyen *et al.* (2005) compare a variant of IEPF, known as *Split and Merge* algorithm, and the *Line Tracking* algorithm against some other methods such as Hough transform, Expectation Maximization (EM) algorithm and Random Sample Consensus (RANSAC) algorithm and conclude that the Split and Merge performs best in the context of SLAM applications using 2D laser scanner.

Another popular feature in SLAM applications are circles. Zhang *et al.* (2003) propose algorithms to extract circles, in addition to line-segments, for representing trunks of the trees and pillars in semi-structured outdoor environments. The algorithm clusters the points using an EKF to estimate the position of the next point as long as the difference between estimated and actual position of the next point is within a threshold. Then a line is fit to the points in the cluster and fitting error is calculated. If the error is greater than the threshold, then a circle is fit. Adams *et al.* (2004) use the same approach to extract circles and associate them in the consecutive time frames using nearest neighbour search. The position of the vehicle is estimated by a particle filter and compared to the estimate made by the association of features for validation.

Garulli *et al.* (2005) perform the SLAM using linear features extracted from the range scans whereas they compute the corresponding covariance matrices from the statistical properties of the raw data. Again, EKF is used for performing simultaneous update of the pose and estimation of line feature associations.

Weingarten and Siegwart (2006) perform feature-based 3D SLAM using a rotating laser scanner. The selected features for this approach are planar segments composed of plane model parameters. The associated planar segment information is considered to be a set of polygons corresponding to each plane. They also use EKF for incrementally building 3D map of the environment with high detail and localizing the robot in that map.

2.3.3 Grid-based SLAM

The occupancy grid representation was proposed by Elfes (1989) for 2D mapping of the environment. Later, Moravec (1996) extended this representation to discretize

the environment in 3D. Elfes had used sonar for the validation of his approach while Moravec used a stereo vision system to generate 3D data. Moreover, in contrast to the Bayesian filtering approach of Elfes to update the probability of occupancy of a cell, Moravec proposed a new method based on evidence theory of Dempster-Shafer. The 3D grid hence produced was called *evidence grid* where each cell contained the value indicating the evidence of its occupation. An alternative representation of the grid was presented by Oriolo *et al.* (1997). The authors proposed that the uncertainty based perception and planning problems can be formulated and solved using the set theory or *fuzzy logic*. The map is defined as a *fuzzy set* in which each point is associated to a real number quantifying the possibility that the map pertains to an obstacle. A merit of this approach, as illustrated by Noykov and Roumenin (2007), is that it can use different types of fuzzy operators for uncertainty modelling and fusion of information from different sensors. A comparison of the three approaches mentioned above is given in (Ribo and Pinz, 2001) for building a sonar-based occupancy grid in indoor environments.

Another variation of the approach, called MUltiple Representation Independent Evidence Log (MURIEL) was proposed by Konolige (1997) which is a mathematical refinement of the method proposed by Elfes (1989). This method addressed the problems intrinsic to the sonar such as reflection multiples and redundant readings. A recent variation to the approach was proposed by Yguel *et al.* (2006) which addresses the problems of representation and data storage for large maps. They introduce a compact multi-scale occupancy grid representation based on wavelets, termed *Wavelet Occupancy Grids* (WavOGs).

In 2D, occupancy grid is a well-established and explored framework which is a base of some of the most impressive navigation systems including the ones described in (Eliazar and Parr, 2003; Grisettiyz *et al.*, 2005; Murphy, 1999; Vu, 2009). The direct extension of 2D occupancy grid to 3D, as proposed by the early work of Moravec (1996) and Roth-Tabak and Jain (1989), is to discretize the mapped area into a grid of equally sized cubic volumes, known as *voxels*. These approaches modelled the environment using rigid grid of voxels which required to be initialized according to the size of the mapped area. This implementation is impractical in outdoor scenarios due to an immense requirement of memory and computational power that increases with the size and resolution of the map. Moreover, the extent of the mapped area should be known in advance.

Triebel *et al.* (2006) and Rivadeneyra *et al.* (2009) utilized a relatively compact data structure called *multi-level surface maps* (MLS) for outdoor mapping. These surface maps use a horizontal grid and represent the 3D structures as height values in that grid, allowing for the storage of vertical objects. In this representation, although the memory requirement is reduced than a pure 3D occupancy grid, but a drawback is that only positive sensor data is recorded and the occupancy values of the objects cannot decrease thus effecting the updatability of the map. As a result, a false positive recorded on the map in one scan cannot be removed in the subsequent scans and stays on the map.

An optimization of the 3D grid is possible by using a tree-based approach such as octrees. As a data structure, octrees are a popular choice for the applications such as storing dense point clouds (Elseberg *et al.*, 2011), compression of point clouds (Kammerl *et al.*, 2012) and rendering in computer graphics (Botsch *et al.*, 2002; Laine and Karras, 2011). Using octrees for 3D grids overcomes the problem of rigid grid structure as it delays the initialization of the voxels until a measurement needs to be integrated. Thus we do not need to know the size of the mapping area in advance. Also, the memory is required to store only those voxels which are already measured (either occupied or free). Another advantage of this method is that it can be used for a multi-resolution environment representation by incorporating the pruning of tree at any level. Octrees were first used by Meagher (1982) for geometric modelling of the objects. Payeur *et al.* (1997) introduced the octrees for adapting probabilistic occupancy grid mapping from 2D to 3D. Recently, many researchers have used this technique for mapping and exploration of the environment (Fairfield *et al.*, 2007; Fournier *et al.*, 2007; Pathak *et al.*, 2007; Wurm *et al.*, 2010).

Fournier *et al.* (2007) presented an approach for generating local 3D models of the environment stored in octree and updated using ray tracing. A similar approach was used by Pathak *et al.* (2007) with a reformulated forward sensor model instead of the commonly used inverse sensor models. Both the approaches do not explicitly deal with the issues of data compression and map adaptability in dynamic environments. Wurm *et al.* (2010) addressed these issues by providing a method for lossless compression and bounded confidence by clamping update policy of Yguel *et al.* (2008). However, all the mentioned approaches are limited to mapping without localization.

An important contribution to octree-based SLAM is proposed by Fairfield *et al.*

(2007). The approach is based on a map structure called *Deferred Reference Counting Octree* and a particle filter based SLAM algorithm which allows for efficient updates in the map. A drawback of their approach is that they use lossy maximum-likelihood compression periodically to ensure the compactness of the map which results in discarding the probability information for future updates. The approach does not allow for the multi-resolution queries as well.

2.3.4 Synthesis

Despite being a simple method, the direct approach has the ability to represent all kinds of environments. However, a major drawback of all the direct mapping approaches is that they provide information about the occupied areas only and do not differentiate between free space and unknown areas. Consequently, they are suitable only for highly precise sensors and in static environments where unknown areas are not needed to be modelled. Further, there is no direct mechanism to deal with the sensor noise as well as identifying the temporarily present dynamic objects. Also, generally in this approach each point is treated independently and not grouped implicitly to represent the objects. Moreover, the memory consumption for storing and processing large amounts of data in the point clouds requires considerable amount of resources and lacks efficiency in a large environment. The resulting maps are usually very complex as they consist of as many as several million data points. This becomes a hindrance to the applicability of these maps for high level robotic tasks such as scene understanding or path planning.

Feature-based approaches perform data compression and generally require less storage space in comparison to the direct or grid-based approaches. It is an advantage of these approaches but it also makes the representation sparse. Further, the free space is not represented in this approach which makes it difficult to detect dynamic objects directly as well as to perform path planning. Moreover, if bad features are selected, it affects the consistency of the map by introducing error into it as well as affecting the pose estimate. A solution is to use artificial landmarks which is only feasible in indoor or limited structured outdoor environments. Thus the feature-based approaches are limited to be used in the environments in which the observed objects can be depicted by geometric feature models which is not the case in large-scale outdoor environments. In these environments, objects can consist of arbitrary curves instead of the distinct points, lines or circles. For such

environments, parametric feature models must be described instead of the simple geometric models for identifying the objects.

Occupancy grid mapping, in general, has several advantages over its counterparts. By probabilistic modelling of occupancy, it can compensate for the sensor uncertainty as well as ensuring the fusion of information from different sensors. Further, it can represent any object and can be used in indoor as well as outdoor environments. A considerable advantage as compared to the other methods is that it explicitly models the free areas of environment which serves as a basis for the identification of moving objects. Moreover, the resolution of the grid can be adjusted according to the sensor and environment thus providing the level of detail accordingly while some approaches even provide a multi-resolution representation as well (Wurm *et al.*, 2010; Yguel *et al.*, 2007). The trade-off between the resolution and computational complexity is an important consideration for the occupancy grid (Bailey, 2002). A small grid resolution corresponds to a fine and detailed environment representation as well as accurate pose estimation but a high computational cost and storage requirement. Thus, a method to model the environment at different levels of granularity is a huge advantage for this case. Despite the research efforts dedicated to improving this representation in recent years, amount of data required to store and process the grid still remains a major drawback of this approach as compared to some other methods such as feature-based approaches.

Regardless of this drawback, occupancy grid has still become a preferred representation due to its advantages over other representations specially in the context of outdoor environments (Baig, 2012; Fairfield *et al.*, 2007; Pathak *et al.*, 2007; Vu, 2009; Wang, 2004). Furthermore, when the sensors such as 3D laser scanners are used which generate a huge amount of data, normally an imperative expectation is to have a full and perceptually rich representation. Consequently, the required processing time and memory are expected to be high. Occupancy grids provide a full and rich representation for any environment as compared to feature-based methods, as well as the compactness with optimized implementation as compared to the direct approaches. Our approach to 3D perception is also based on occupancy grid representation. We have used the octree-based probabilistic occupancy grid approach proposed by Wurm *et al.* (2010) for environment representation which is a generic mapping approach and does not address the problem of localization. It assumes that the trajectory of the robot is known.

In order to perform localization with mapping, Kalman filter is the most common choice. The main attraction of Kalman filter SLAM lies in the estimation of a fully correlated posterior over feature maps and robot poses. However, as mentioned earlier, KF makes a strong assumption that the motion model and measurement model are linear functions with Gaussian noise and the initial distributions are also Gaussian. Moreover, KF SLAM is practically effective only with the feature-based environment representation which is not an ideal choice for unstructured outdoor environments. Another popular approach for SLAM is the *incremental maximum likelihood* (ML) method which incrementally builds a single map from the sensor measurements instead of performing a full posterior estimation over all poses and maps. As a result, it is a simple and computationally effective method which can be used with any map representation. The disadvantage of this paradigm is that due to keeping the most likely map at each time step, it is unable to perform loop closing in cyclic environments. This problem can be addressed by using an alternative method called *FastSLAM* which maintains the full posterior using particle filter instead of Kalman filter to represent non-linear models and non-Gaussian noise. This method is applicable to both feature-based and grid-based mapping and it is much faster as compared to the classical KF SLAM, however, it still becomes too costly in the context of dense 3D maps with 6D poses. Consequently, a periodical lossy maximum-likelihood compression is often used in order to ensure the tractability of the algorithm and compactness of the map which results in discarding the probability information. In practice, this approach is suitable for the applications in which a consistent global map is mandatory such as those constructing the accurate maps for later use. However, in the context of applications where an instantaneous map serves the purpose such as the applications focusing on obstacle avoidance, maximum likelihood SLAM is a better choice.

We describe the approach that we adopted for environment representation in the next subsection and propose how to extend it with ground segmentation (section 2.5.2)and localization (section 2.5.3) in order to adapt it to the problem of simultaneous localization and mapping in 3D.

2.4 Adopted Approach for Map Representation: Probabilistic 3D Occupancy Grid based on Octrees

In this work, we have used the octree-based occupancy grid approach presented in (Wurm *et al.*, 2010) for environment representation. Its implementation is available as an open-source library, named OctoMap[3], to generate volumetric 3D environment models. It is purely a mapping approach and does not perform localization. The assumption is that the trajectory of the robot is known.

In the following subsection, we provide an overview of the OctoMap and its mapping methodology. Then we describe the merits of this approach for mapping in section 2.4.2 with its limitations in section 2.4.3 which provide a context for our contributions in this chapter.

2.4.1 OctoMap 3D Occupancy Grid Mapping – Overview

OctoMap (Wurm *et al.*, 2010) is an open source library for probabilistic occupancy grid modeling of three dimensional environments based on octrees. It provides a multi-resolution representation of arbitrary environments as full 3D models including occupied, free and unknown areas. The tree-based implementation provides maximum flexibility regarding the area and resolution of the map as well as allowing for efficient probabilistic update of the grid. The memory consumption is also improved by a great deal as compared to a rigid grid. The following subsection provides an introduction to the octrees and how they are used for occupancy grid mapping in OctoMap. Next, we describe the method for updating the occupancy probability of the grid map given the sensor inputs followed by the sensor model used for computation of occupancy.

Octrees for Occupancy Grid Mapping

An occupancy grid map is a uniform discretization of the space in cells where each cell stores a probabilistic estimate of its occupancy (Moravec, 1996). In 3D, the cells are cubic units of volume, called *voxels*. Assuming that the occupancy states of individual cells are independent, the grid can be considered as a simple Markov Random Field. The accumulated occupancy probabilities of the cells are estimated

[3] http://octomap.github.com

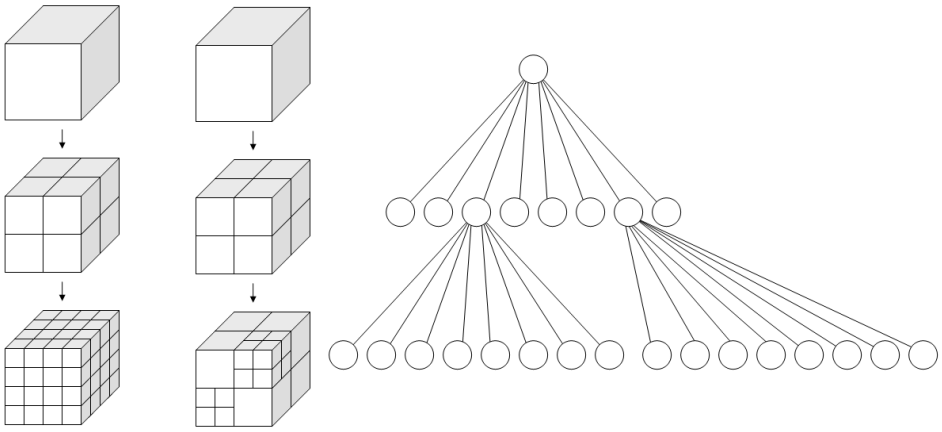


Figure 2.5: Example of an octree structure (center) with the corresponding tree representation (right) in comparison to a uniform 3D grid (left). While a uniform grid (left) must represent every cell, each level of an octree divides the remaining volume into eight octants (center) and the octree can efficiently represent sparse volumes because the tree structure (right) does not have to be fully expanded.

whenever new data is received and inserted in the grid. In addition to inserting new information, querying to simulate measurements and copying the map are the primary operations performed on a grid. The main issues in 3D occupancy grids arise from the cost of these operations and the storage requirements which increase with the map size. In a regular grid, the whole space is subdivided and stored regardless of the fact that it contains any information or not. To solve these issues, we require an efficient data structure which may store only those parts which contain relevant information besides ensuring efficient query and update. Octree is a suitable data structure for this purpose.

An octree is a tree data structure in which each internal node has exactly eight children. It is a generalization of binary trees and quad trees in one and two dimensions respectively. Octrees are used to partition a three dimensional space by recursively subdividing it into eight octants or cubic volumes (*voxels*), as shown in Fig 2.5. This subdivision continues recursively until a desired minimum voxel size is reached which determines the resolution of the octree. Being a hierarchical data structure, the tree can be cut at any level to obtain a coarser subdivision. This ensures that the subdivision is not applied in the empty nodes, that is the nodes that do not contain any data. Thus a large contiguous part of the environment which is empty, can be efficiently represented by a single node.

In occupancy grid mapping, octree is used to model probabilistic occupancy of a

volume based on the sensor measurements. If a certain volume in the grid is measured as *occupied*, the corresponding node in the octree is initialized. The uninitialized nodes can correspond to *free* as well as *unknown* areas. To resolve this ambiguity, free voxels are explicitly represented as free nodes in the tree. These are the voxels identified by ray-casting between the sensor and the measured end point of the ray. The remaining uninitialized nodes implicitly correspond to the unknown areas only. The next subsection describes the method for updating the occupancy probability of the grid in this approach.

Updating Occupancy Grid

The goal of mapping algorithm is to estimate the distribution of maps M given the history of sensor measurements $z_{1:t}$ and vehicle locations $x_{1:t}$ i.e $P(M|x_{1:t}, z_{1:t})$. In occupancy grid formulation, as mentioned earlier, the occupancy states of individual cells are assumed to be independent. Thus the problem of estimating the distribution over maps is decomposed into estimating the occupancy state of each voxel m of the grid, i.e the leaf node of the octree, given by $P(m|x_{1:t}, z_{1:t})$.

For each sensor reading, the occupancy probability is recursively estimated using Bayes' theorem:

$$P(m|x_{1:t}, z_{1:t}) = \frac{P(z_t|x_{1:t}, z_{1:t-1}, m)P(m|x_{1:t}, z_{1:t-1})}{P(z_t|x_{1:t}, z_{1:t-1})} \tag{2.4.1}$$

This update equation is further simplified using the important assumption that the sensor measurements are conditionally independent of each other and the previous vehicle positions $x_{1:t-1}$ given the cell m. That is:

$$P(z_t|x_{1:t}, z_{1:t-1}, m) = P(z_t|x_t, m) \tag{2.4.2}$$

Further, using Bayes' theorem:

$$P(z_t|x_{1:t}, z_{1:t-1}, m) = P(z_t|x_t, m) = \frac{P(m|x_t, z_t)P(z_t|x_t)}{P(m|x_t)} \tag{2.4.3}$$

Substituting this in Eq. 2.4.1, we get:

$$P(m|x_{1:t}, z_{1:t}) = \frac{P(m|x_t, z_t)P(z_t|x_t)P(m|x_{1:t}, z_{1:t-1})}{P(m|x_t)P(z_t|x_{1:t}, z_{1:t-1})} \tag{2.4.4}$$

where $P(m|x_t)$ is the prior probability of occupancy of the cell m and is independent of the current position, i.e. $P(m|x_t) = P(m)$. Hence:

$$P(m|x_{1:t}, z_{1:t}) = \frac{P(m|x_t, z_t)P(z_t|x_t)P(m|x_{1:t}, z_{1:t-1})}{P(m)P(z_t|x_{1:t}, z_{1:t-1})} \tag{2.4.5}$$

Again using Bayes' theorem to expand $P(m|x_{1:t}, z_{1:t-1})$:

$$P(m|x_{1:t}, z_{1:t-1}) = \frac{P(x_t|x_{1:t-1}, z_{1:t-1}, m)P(m|x_{1:t-1}, z_{1:t-1})}{P(x_t|x_{1:t-1}, z_{1:t-1})} \tag{2.4.6}$$

Substituting it in Eq. 2.4.5:

$$P(m|x_{1:t}, z_{1:t}) = \frac{P(m|x_t, z_t)P(z_t|x_t)P(x_t|x_{1:t-1}, z_{1:t-1}, m)P(m|x_{1:t-1}, z_{1:t-1})}{P(m)P(z_t|x_{1:t}, z_{1:t-1})P(x_t|x_{1:t-1}, z_{1:t-1})} \tag{2.4.7}$$

Equation 2.4.7 gives the probability for an occupied cell. By analogy, the probability of a free cell is given by:

$$P(\overline{m}|x_{1:t}, z_{1:t}) = \frac{P(\overline{m}|x_t, z_t)P(z_t|x_t)P(x_t|x_{1:t-1}, z_{1:t-1}, \overline{m})P(\overline{m}|x_{1:t-1}, z_{1:t-1})}{P(\overline{m})P(z_t|x_{1:t}, z_{1:t-1})P(x_t|x_{1:t-1}, z_{1:t-1})} \tag{2.4.8}$$

Dividing Eq. 2.4.7 by Eq. 2.4.8:

$$\frac{P(m|x_{1:t}, z_{1:t})}{P(\overline{m}|x_{1:t}, z_{1:t})} = \frac{P(m|x_t, z_t)}{P(\overline{m}|x_t, z_t)} \frac{P(\overline{m})}{P(m)} \frac{P(x_t|x_{1:t-1}, z_{1:t-1}, m)}{P(x_t|x_{1:t-1}, z_{1:t-1}, \overline{m})} \frac{P(m|x_{1:t-1}, z_{1:t-1})}{P(\overline{m}|x_{1:t-1}, z_{1:t-1})} \tag{2.4.9}$$

As the current position x_t is known, $P(x_t|x_{1:t-1}, z_{1:t-1}, m) = P(x_t|x_{1:t-1}, z_{1:t-1}, \overline{m})$. Therefore, Eq. 2.4.9 reduces to:

$$\frac{P(m|x_{1:t}, z_{1:t})}{P(\overline{m}|x_{1:t}, z_{1:t})} = \frac{P(m|x_t, z_t)}{P(\overline{m}|x_t, z_t)} \frac{P(\overline{m})}{P(m)} \frac{P(m|x_{1:t-1}, z_{1:t-1})}{P(\overline{m}|x_{1:t-1}, z_{1:t-1})} \tag{2.4.10}$$

Usually the probability values are very small and division makes them even smaller therefore the *logOdds* values are used rather than the raw probabilities due to a better numerical representation. Moreover, the Bayesian update of *logOdds* for a particular voxel becomes a simple addition (Moravec, 1996). Using the definition of *Odds* ratio:

$$Odds(x) = \frac{P(x)}{1 - P(x)} = \frac{P(x)}{P(\overline{x})} \tag{2.4.11}$$

Equation 2.4.10 can be written as:

$$Odds(m|x_{1:t}, z_{1:t}) = Odds(m|x_t, z_t)Odds(m)^{-1}Odds(m|x_{1:t-1}, z_{1:t-1}) \tag{2.4.12}$$

Taking *log* of Eq. 2.4.12:

$$\begin{aligned} &\log Odds(m|x_{1:t}, z_{1:t}) \\ &\quad = \log Odds(m|x_t, z_t) - \log Odds(m) + \log Odds(m|x_{1:t-1}, z_{1:t-1}) \end{aligned} \tag{2.4.13}$$

In this equation, the first term on the right-hand-side corresponds to the inverse sensor model and the second to the map prior. Considering a uniform prior probability leads to $P(m) = P(\overline{m}) = 0.5$. Therefore, $Odds(m) = 1$ and $\log Odds(m) = 0$ and Eq. 2.4.13 simplifies to:

$$\log Odds(m|x_{1:t}, z_{1:t}) = \log Odds(m|x_t, z_t) + \log Odds(m|x_{1:t-1}, z_{1:t-1}) \quad (2.4.14)$$

This formulation ensures that the update of each voxel can be reduced to simply adding the $\log Odds$ of inverse sensor model to the $\log Odds$ of previous belief of the voxel. In case of precomputed sensor models, the logarithms do not need to be computed during the update step. Computation of the sensor model for the used sensor is detailed in section 2.4.1.

For a 3D map, a threshold is applied on the occupancy probability and a voxel is considered to be occupied when the threshold is reached and free otherwise. Equation 2.4.14 indicates that, in order to change the state of a voxel (e.g. from free to occupied), as many observations should be integrated as have been used to define its state previously. This implies that if a voxel is observed free for k times, it must be observed occupied for at least k times to be considered as occupied. In dynamic environments, this property effects the consistency of the map in case of temporarily appearing objects as the map should adapt to the changes quickly. This adaptability is ensured in OctoMap using the *clamping update* policy of Yguel *et al.* (2008). This policy defines an upper and lower bound on the occupancy estimate. The occupancy is updated using the following equation, instead of Eq. 2.4.14:

$$\begin{aligned} &\log Odds(m|x_{1:t}, z_{1:t}) \\ &\quad = \max(\min(\log Odds(m|x_t, z_t) + \log Odds(m|x_{1:t-1}, z_{1:t-1}), l_{max}), l_{min}) \end{aligned} \quad (2.4.15)$$

Where l_{min} and l_{max} are the predefined lower and upper bounds on the $\log Odds$ values. In addition to ensuring the reversibility for the states of voxels, clamping update policy enables to compress the neighboring voxels by pruning. Whenever the $\log Odds$ value of a voxel reaches either the bound l_{min} or l_{max}, the voxel is considered to be stable, and thus measured free or occupied with high confidence. If all children of a node are stable leafs with the same occupancy state, then they can be pruned leading to a considerable reduction in the total number of nodes. In case of the future measurements which contradict the occupancy state of the node, its children are regenerated and updated accordingly. This compression leads to

the loss of information only close to $P(m) = 0$ and $P(m) = 1$ while the probabilities are preserved for all other values in between those.

Each voxel of the OctoMap stores log *Odds* value instead of its raw occupancy probability. However, the original probabilities can be recovered from these values using:

$$P(x) = \frac{Odds(x)}{1 + Odds(x)} \tag{2.4.16}$$

Using this law in Eq. 2.4.12 leads to:

$$P(m|x_{1:t}, z_{1:t}) = \left[1 + \frac{1 - P(m|x_t, z_t)}{P(m|x_t, z_t)} \frac{P(m)}{1 - P(m)} \frac{1 - P(m|x_{1:t-1}, z_{1:t-1})}{P(m|x_{1:t-1}, z_{1:t-1})}\right]^{-1} \tag{2.4.17}$$

This is the generic equation for updating the occupancy probability of a voxel. Here, $P(m)$, as described above, is the prior occupancy probability (which is set to 0.5) and $P(m|x_{1:t-1}, z_{1:t-1})$ is the previously estimated occupancy probability value of the voxel. The term $P(m|x_t, z_t)$ is the current belief, known as inverse sensor model (Thrun, 2003), which is specific to the sensor used.

Sensor Modeling

As measurements are received, the evidence they provide about the occupancy of each voxel is integrated into the grid according to the inverse sensor model, $P(m|x_t, z_t)$. This model is called *inverse* since it maps from the measurement to the physical world. It specifies the probability that a voxel m is occupied given a single sensor measurement z_t at the location x_t.

OctoMap can be used with any type of sensor knowing the corresponding sensor model. In our case, a laser scanner is used as the main perception sensor. For this type of sensors, OctoMap employs a beam-based inverse sensor model which assumes that the endpoints of a measurement correspond to the obstacles whereas the line of sight between the sensor origin and endpoint contains no obstacles. To insert the new measurements given by the laser scanner, a *ray tracing* is performed to determine the voxels along a beam from the sensor origin to the measured point. The beam is approximated by an efficient 3D variant of the classic Bresenham algorithm (Amanatides *et al.*, 1987) (Fig. 2.6). Each voxel which falls within the beam needs to be updated with the value of the sensor model at that point according to

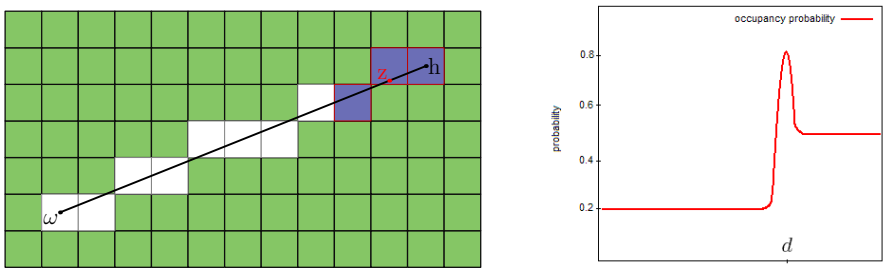


Figure 2.6: Sensor modelling for laser scanner. (Left) Ray tracing technique to update an occupancy grid using Bresenham algorithm. (Right) Inverse sensor model showing the occupancy probability of voxels along a beam measuring a distance of 4m.

the Bayesian update equation (Eq. 2.4.15) using:

$$\log Odds(m|x_t, z_t) = \begin{cases} l_{occ} & \text{if beam is reflected within the volume} \\ l_{free} & \text{if beam traversed the volume} \end{cases} \tag{2.4.18}$$

Here, l_{occ} and l_{free} are the log Odd values of occupied and free voxels respectively. In practice, we have set $l_{occ} = 1.39$ and $l_{free} = -1.39$ corresponding to the probabilities $P_{occ} = 0.8$ and $P_{free} = 0.2$ respectively. The clamping thresholds are set to $l_{min} = -2$ and $l_{max} = 3.5$ corresponding to minimum probability of 0.12 and maximum 0.97.

An illustration of the OctoMap representation is presented in Fig. 2.7 which shows the occupied as well as free voxels. In the following subsections, we present an evaluation of this technique describing both its advantages as well as limitations.

2.4.2 Advantages of OctoMap

OctoMap has certain advantages as a 3D mapping approach. In this section, we summarize the main advantages of this approach.

Probabilistic Representation

In order to navigate, the robot relies on its sensors to provide the 3D range measurements. These measurements are inherently uncertain and noisy. This sensor uncertainty and noise is compensated by probabilistic modelling of the occupancy in OctoMap. Further, multiple uncertain measurements are integrated to obtain a robust estimate of the actual state of the environment. Moreover, the probabilistic representation allows for the multiple sensor fusion to combine the data from

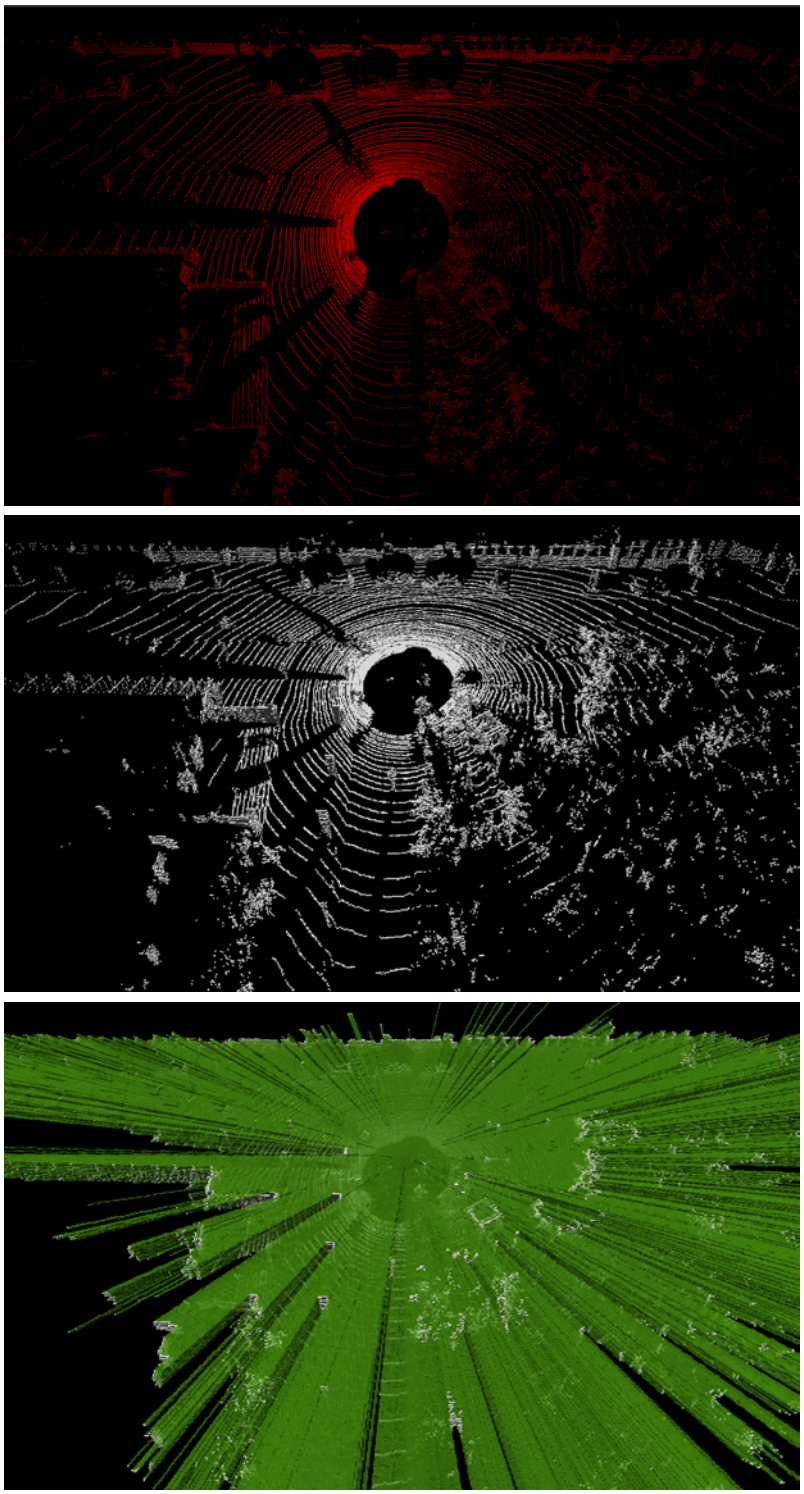


Figure 2.7: Octree-based Occupancy grid representation. (Top) A point cloud generated by Velodyne HDL-64E laser scanner. (Center) The occupancy grid representation corresponding to the point cloud, showing the occupied voxels only. (Bottom) Occupancy grid showing both occupied and free voxels in grey and green colors respectively.

multiple sensors as well as multiple robots.

Modeling of Free Areas

A considerable advantage of this approach is that it explicitly models the free areas of the environment in addition to the occupied areas. In autonomous navigation tasks, a robot can plan collision-free paths only for those areas that are detected to be free. The unknown areas need to be avoided and therefore the map has to represent these areas. In this approach, the areas about which no information is available are differentiated from the ones which are observed as free. This distinction helps in navigation and path planning for autonomous exploration of the environment along with serving as a basis for the identification of moving objects.

Efficiency

The map is a fundamental component of a robotic system as most of its capabilities such as path planning, collision avoidance etc depend on the consistency of the map. Thus, the map should be efficient both in access time as well as memory usage. In 3D representations, memory usage is often considered as the main impediment. In this approach, the tree-based method reduces the access time in addition to the pruning based on bounded per-voxel confidence which ensures a compression scheme leading to substantial reduction in the memory consumption.

Flexibility

Finally, another considerable advantage of this approach, in comparison to the rigid grid approaches, is that the extent of the map does not need to be known in advance. Rather, the map is dynamically expanded depending on the observed area which makes it suitable for large outdoor environments.

2.4.3 Limitations of OctoMap

The main limitation of this approach, as mentioned above, is that it is only a mapping approach and not a complete perception framework. In the context of SLAM as well as that of a complete perception framework, following are the limitations of this approach that need to be addressed.

Localization

An important assumption in this approach is that the vehicle positions are known at all instances. In reality, this is not always true. Although the vehicle is usually equipped with various sensors to measure its own motion (e.g. odometer, wheel encoder etc.) as well as position (e.g. GPS) but the information provided by these sensors, known as *dead reckoning*, is inevitably noisy and uncertain. Thus, each pose estimate has a component of error which accumulates with the integration of scan measurements. As a result, the uncertainty in the pose estimate increases monotonically with time and eventually the pose estimate becomes so erroneous that it can not serve any useful purpose. However, the dead reckoning can still be used as a supplementary source of information in conjunction with another method for localization.

In order to correct the dead reckoning pose estimate errors in parallel to constructing a consistent map of the environment, we need to perform simultaneous localization and mapping (SLAM). Thus, this approach lacks a method for localization essential for real robot applications.

Ground Segmentation

As this approach uses 3D range data, a 3D laser scanner can be considered a sensor of choice to be used in this context. When a 3D range sensor is used, a considerable amount of the acquired information corresponds to the ground (see Fig. 1.4) which influences the process of localization. Moreover, in the dynamic environments, this ground information affects the detection, classification and tracking processes too. This introduces the demand for another crucial step for mapping, known as the ground identification and segmentation (B. Douillard, 2010; Himmelsbach, 2010). Although the OctoMap approach provides the heights of the voxels but that is not sufficient for ground identification in real outdoor environments due to unevenness of the surface. Thus, a sophisticated method must be provided for ground segmentation.

In literature, there are three categories of methods for ground identification from 3D point clouds: cell-based methods (B. Douillard, 2010; Thrun *et al.*, 2006), line-based methods (Himmelsbach, 2010), and fitting plane methods (Douillard *et al.*, 2011; Lam *et al.*, 2011; Vasudevan *et al.*, 2011; Vosselman *et al.*, 2004). Cell-based

methods are widely used in recent applications where the point cloud is divided in a 2D Cartesian grid and height for each cell of the grid is computed. Commonly, the height value is either computed as the difference between the minimum and maximum height of the returns falling in each cell of the grid (Thrun *et al.*, 2006) or as the mean of the heights (B. Douillard, 2010). In line-based methods, the point cloud is divided in a polar grid and line extraction algorithms such as incremental algorithms, ANSAC algorithm or Hough transform are used to represent the ground surface (Himmelsbach, 2010). Fitting plane methods assume the ground to be flat and determine the planar surfaces from the point cloud by using the algorithms such as plane ANSAC algorithm (Lam *et al.*, 2011) and 3D Hough transform (Vosselman *et al.*, 2004).

Detection of Moving Objects

In natural outdoor scenarios, dynamic objects constitute an important part of the environment. For the mobile robots that require to operate in such scenarios, it is imperative to identify and track the motion of these objects. In addition to the limitations of the OctoMap approach as a SLAM system, it does not provide any information about the dynamic objects as well.

Object Classification

Another important remark is that in the mentioned approach, the environment is represented by individual voxels. There is no method to group those voxels in order to represent and classify the relevant objects. Object classification is very important for extracting perceptually rich information and also for tracking of moving objects.

2.5 Contributions: 3D Occupancy Grid SLAM

In this section, we present the method that we have developed for the grid-based maximum likelihood SLAM using 3D laser scanner. We have used the OctoMap approach described in section 2.4 as a base for the environment representation and adapted it for the SLAM problem by introducing a 3D grid-based scan matching method for localization along with ground identification and segmentation. In the

next section, we give an overview of our SLAM framework followed by the approach for ground segmentation in section 2.5.2 and scan matching in section 2.5.3.

2.5.1 General Architecture

Scan matching approaches align consecutive scans taken by the vehicle at different locations and thereby they estimate the trajectory of the vehicle as well as creating a consistent map of the environment. In case of the data acquired by a 3D laser scanner, the general problem for scan matching is the consistent alignment of overlapping 3D point clouds into a complete map. This process is known as *registration*. In other words, registration is the process of finding the relative position and orientation of one 3D scan (known as the *scene D*), to another (known as the *model scan M*). A famous method for solving this problem is the iterative closest points algorithm (ICP). This algorithm searches for the pairs of closest points in two different scans and iteratively minimizes their relative transform. In this work, we use a fast variant of the ICP algorithm for 3D grid-based scan matching.

An important observation in the case of 3D laser scanner, as mentioned in previous section, is that the lower layers of the scanner give the points corresponding to the ground. These ground points do not provide any useful information for the task of registration and rather affect the performance of scan matching methods. Thus, it is necessary to identify these points by performing ground segmentation before scan matching. Another contribution of this chapter is our method for an efficient segmentation of the ground using the variance-based elevation maps.

Figure 2.8 illustrates the general flow of our method. The octree occupancy grid map m_t generated from the new scan measurement is first projected to $2\frac{1}{2}$D grid map. Then the features are computed for each cell in the $2\frac{1}{2}$D grid to identify the ground cells. These ground cells are then used to determine the ground voxels in the original 3D grid of current scan measurement. This results in the separation of the voxels into ground and non-ground. The non-ground voxels are then passed to the localization module which generates the point cloud corresponding to those. In the next step, ICP based scan matching is performed between the sub-sampled point cloud and the stored occupancy grid map M_{t-1} constructed from the previous measurements. This gives the estimated transformation which is applied on the occupied voxels to obtain the updated map and vehicle state. The ground voxels are also transformed to integrate them in the updated map.

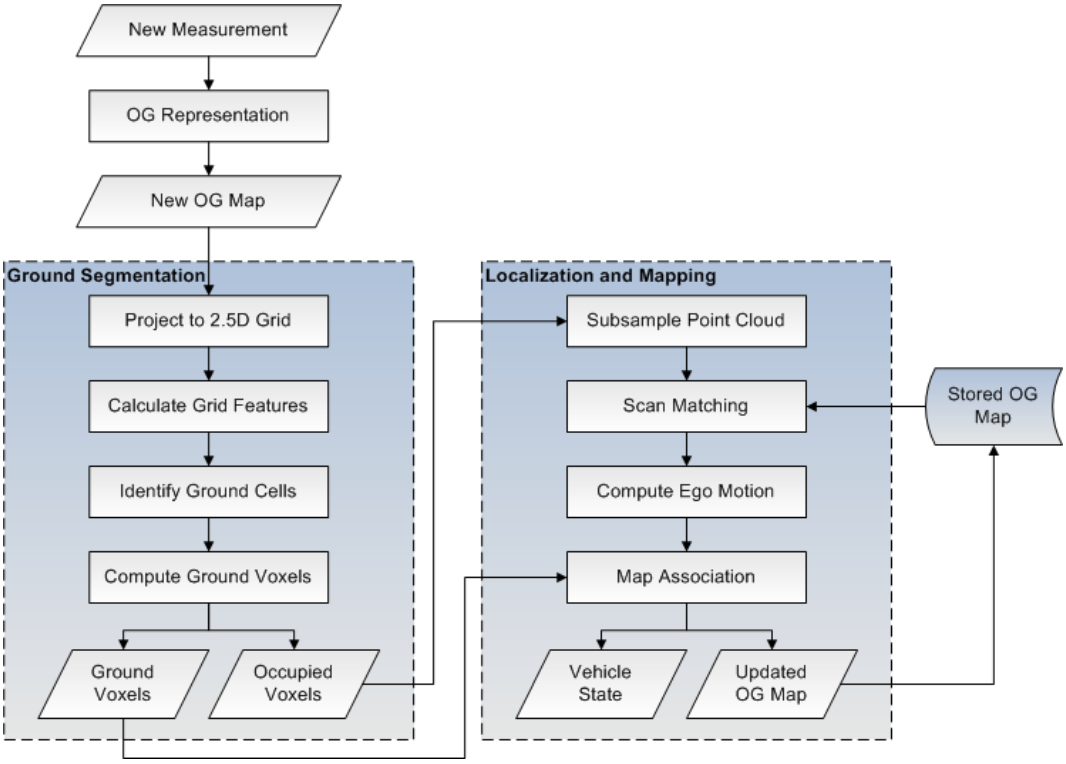


Figure 2.8: Architecture of proposed method for SLAM.

Next section describes the proposed method for ground segmentation followed by the grid-based scan matching for localization explained in section 2.5.3.

2.5.2 Ground Segmentation

The simplest solution for ground segmentation problem, when using a 3D laser scanner, is to use a height window and mark all the points inside that window as ground. This window is computed by using the height of the vehicle h as the standard reference. If the vertical component of a 3D point is between $-h - \triangle$ and $-h + \triangle$, then this point is considered as ground point. We used this technique as a preliminary method for ground segmentation, however, we observed that this method is not useful if the surface of the ground is sloped or rough where the vehicle does not move smoothly and the height of 3D sensor relative to the ground is not constant.

Alternatively, as described in section 2.4.3, there are three categories of methods for ground identification from 3D point clouds: cell-based methods, line-based methods and fitting plane methods. In this work, we propose a variance-based ground

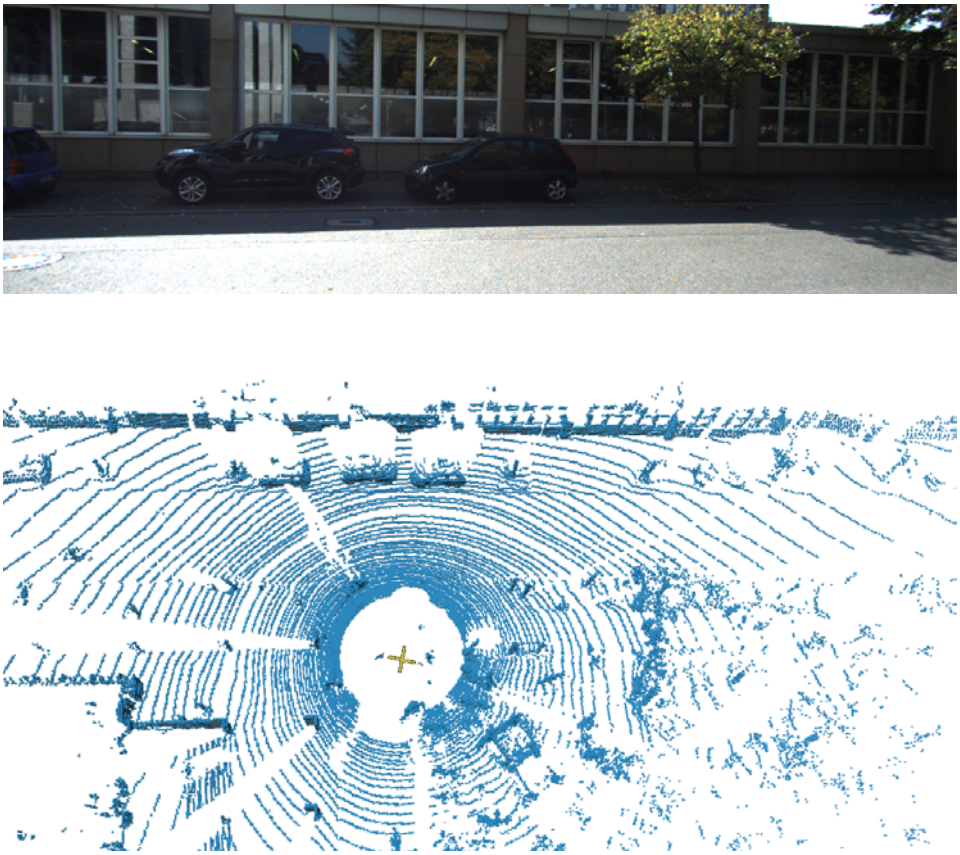


Figure 2.9: 3D Occupancy grid representation (bottom) of an outdoor scenario (top) that we will use to illustrate our approach for ground segmentation

segmentation method which can be categorized in the cell-based methods. The approach most similar to our work is used for object segmentation in (B. Douillard, 2010) from the data acquired by a stationary Riegl sensor. The main difference, other than using a moving sensor, is that we have used the variance as the defining feature for ground points as it indicates the tendency of variable dispersion in comparison to the mean used by Douillard. Moreover, our approach makes use of the octree-based occupancy grid which is a sub-sampled form of the original point cloud. Consequently, we need to identify the ground voxels instead of all the ground points which makes it more efficient. Figure 2.9 is an example of the occupancy grid representation of an outdoor environment that we will use to illustrate our approach for ground segmentation. The image on the top shows an outdoor scenario with some parked cars, trees and buildings. The image on the bottom is the octree-based occupancy grid representation showing the occupied voxels in blue. The ego-vehicle is represented by a mustard '+' sign in the middle. The free

Algorithm 1 Ground segmentation

Input: *newOG*, *mapRes*, *varianceThresh*, *meanThresh*, *heightThresh*
Output: *groundVoxels*

elevationMap ← `Generate_Elevation_Map` (*newOG*, *mapRes*)
for all cells c_i in *elevationMap* **do**
    σ_i = `Compute_Height_Variance` (c_i)
    assign σ_i to (c_i)
end for
clusters ← `Cluster_Flat_Cells` (*elevationMap*, *varianceThresh*, *meanThresh*)
largestCluster ← `Find_Largest_Cluster` (*clusters*)
groundHeight ← `Get_Mean_Height` (*largestCluster*)
groundClusters ← `Get_Ground_Clusters` (*clusters*, *groundHeight*, *heightThresh*)
groundVoxels ← `Retrieve_Voxels` (*newOG*, *largestCluster*, *groundClusters*)

voxels are not shown for the purpose of a clear representation.

In order to separate the voxels representing the ground from all occupied voxels, we first convert the 3D occupancy grid into a $2\frac{1}{2}$D grid, known as *elevation map*, constructed such that we keep track of the heights of the voxels falling in each cell of the $2\frac{1}{2}$D grid. Then we compute variance (σ) for each cell of the elevation map as it indicates the tendency of variable dispersion which is a good feature to determine the flatness of a surface. In the next step, neighboring cells are clustered on the base of the variance and mean height of the returns falling therein. Finally, we find the largest cluster and consider it to correspond to the ground. The other locally flat surfaces that do not belong to the ground are then filtered out. This process is illustrated in Algorithm 1 and detailed in the following.

Variance-based Elevation Map

Elevation maps are a compact $2\frac{1}{2}$-dimensional representation of the environment which allow the storage of the 3D data in a 2D grid while preserving a certain amount of information corresponding to the third dimension. An elevation map can be generated from a 3D occupancy grid map by projecting its occupancy information to 2D. The idea is to consider the 3D grid as a 2D grid consisting of columns of voxels along the z-axis. Each column corresponds to a single cell in the eleva-

tion map whose state is computed from the values of the voxels in that column. In a simple elevation map, each cell of the grid contains a single value which may correspond to the maximum height of the voxels falling in that cell (contained in the corresponding column), the difference between the maximum and minimum height or the mean of all the heights.

A standard approach for computing the elevation map is to take the mean or average of the heights of the returns falling in each cell of the grid and save it as the value of that cell. The map generated by this method is thus called a *mean elevation map*. We propose to use a an alternative approach for generating elevation map using the variance of the heights of the returns falling in each cell instead of the mean. The reason for preferring variance over mean is that it is a rich feature concerning the dispersion and identification of the flatness of a cell.

For our implementation of the elevation map, instead of storing the *mean* of the heights only, we keep track of all the heights corresponding to each grid cell. In order to do so, we first define a horizontal 2D grid corresponding to the existing 3D occupancy grid with the same resolution as that of the 3D grid and populate it by using a flood-fill algorithm. In our representation of the 3D occupancy grid, each voxel is defined by its center (x, y, z) and the length l of its side. For computing the 2D grid, we consider the centers of all the voxels in the 3D occupancy grid as the sub-sampled input point cloud. We put all these 3D points into a 2D occupancy grid data structure aligned to the (X, Y)-axis of the coordinate system. We associate an additional field with each of these cells that consists of the heights z of all the voxels in 3D that correspond to that cell in 2D.

In the next step, we compute the variance (σ) of height for the points in each cell using:

$$Var(z) = E[z^2] - (E[z])^2 \tag{2.5.1}$$

where $E[z] = \mu$(mean) is the expected value of the variable z. If the variance of the cell is below a certain threshold, it is considered as a flat cell. These flat cells are candidates for the ground but they might correspond to the locally flat surfaces too. Therefore, in the next step, we eliminate such locally flat surfaces and segment the ground cells from them. Figure 2.10(top) represents the variance-based elevation map of the occupancy grid in Fig. 2.9. The color of each cell in this elevation map represents the value of the variance in that cell. The lower variance values are represented by blue while the higher are represented by green, yellow and red.

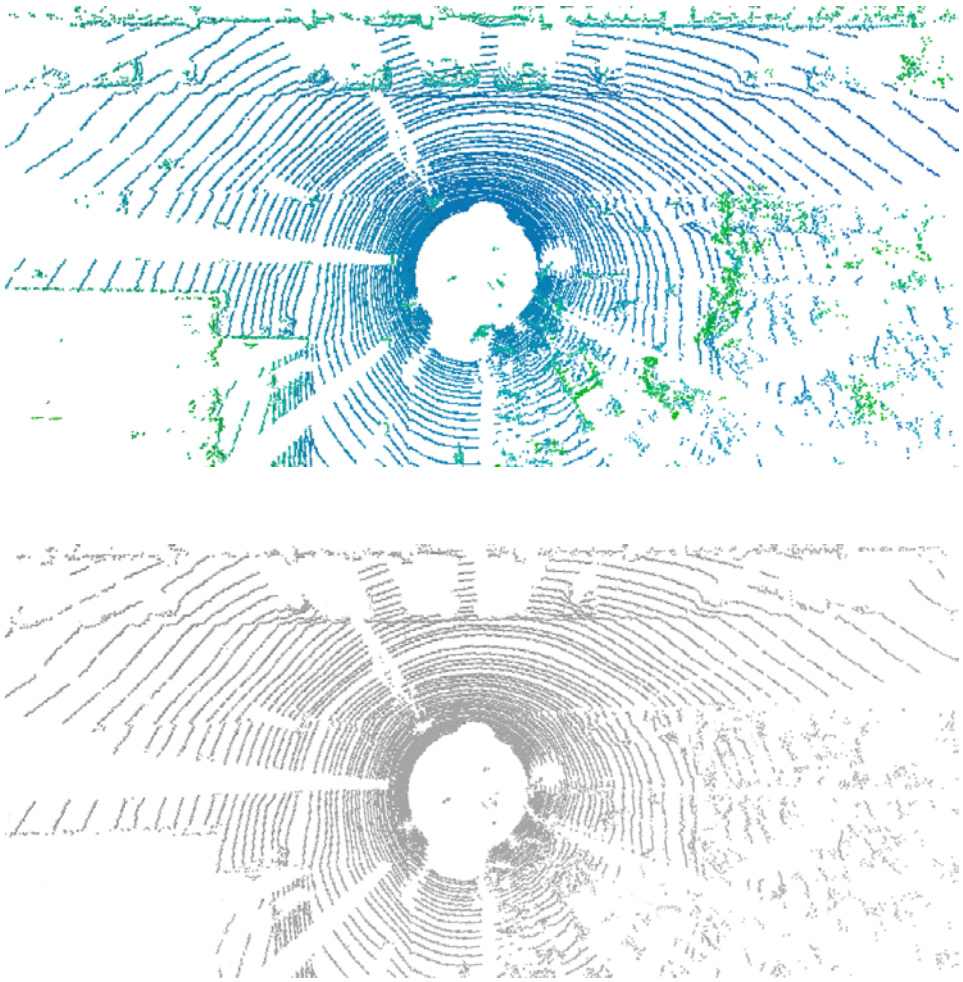


Figure 2.10: The image on the top represents the variance-based elevation map of the grid shown in Fig. 2.9 while the image in the bottom shows the same variance map after applying threshold. After thresholding and clustering, the remaining cells belong to the ground.

Ground Identification

After computing the variance for each cell of the elevation map and applying a threshold on it, the neighboring cells identified as flat are connected to form clusters of flat cells. An additional criterion, other than the neighborhood, is the mean height of each cell. For clustering the cells, the 2D grid is traversed and the same cluster ID is assigned to the flat cells which are neighboring to each other and their mean heights are within a specified threshold.

Next, the largest cluster is identified which is considered to be the reference for the ground. The mean height of this cluster is considered as a reference height of the ground. This height is used to decide if the other smaller clusters belong to the

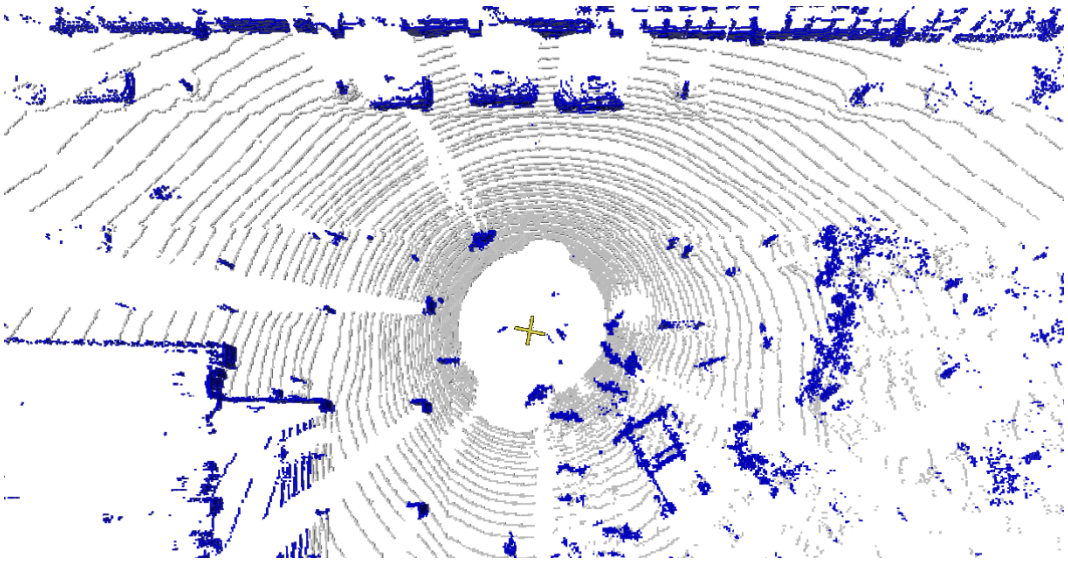


Figure 2.11: The final output of ground segmentation. All ground voxels are displayed in grey while all other occupied voxels are in blue.

ground or not. If the height of a cluster is within a threshold of the height of the ground cluster, then it is marked as ground and ignored otherwise. The results of this step are shown in Fig. 2.10(bottom).

Finally, all the voxels in the 3D occupancy grid whose centres fall in the cells identified as ground in the elevation map are retrieved. These voxels are maintained separately as ground voxels and removed from the occupied voxels. Figure 2.11 shows the result of ground segmentation where the ground voxels are represented in grey and all other occupied voxels in blue.

After ground segmentation, we perform the maximum likelihood based simultaneous localization and mapping to correct vehicle location from odometry. We explain this process in the following subsection.

2.5.3 Grid-based SLAM

We have implemented occupancy grid-based *incremental maximum likelihood* SLAM approach for solving the problem of simultaneous localization and mapping. The main idea of this algorithm is to incrementally build a single map of the environment based on the set of measurements received at each time step. It does not keep track of any residual uncertainty in the vehicle pose. The incremental maximum likelihood method consists in maximization of the marginal likelihood of t-th pose

and map relative to the $(t-1)$-th pose and map for the estimation of a sequence of poses $\hat{x}_1, \hat{x}_2$... while updating the maps $\hat{M}_1, \hat{M}_2$... incrementally. Thus the pose $\hat{x}_t$ is defined as:

$$\hat{x}_t = \arg\max_{x_t} \{P(z_t|x_t, \hat{M}_{t-1}) P(x_t|\hat{x}_{t-1}, u_t)\} \tag{2.5.2}$$

In this equation, the first term $P(z_t|x_t, \hat{M}_{t-1})$ represents the measurement model which defines that how probable a sensor measurement z_t is given the vehicle pose x_t and the map $\hat{M}_{t-1}$ constructed from the previous measurements $\{z_1, ..., z_{t-1}\}$ at their corresponding poses $\{\hat{x}_1, ..., \hat{x}_{t-1}\}$ which were estimated earlier. The term $P(x_t|\hat{x}_{t-1}, u_t)$ represents the motion model of the vehicle which is defined as the probability of the vehicle being at position x_t knowing that it was at position $\hat{x}_{t-1}$ at the previous time step and executed the action u_t.

The $\hat{x}_t$ computed from Eq. 2.5.2 is then used along with the corresponding set of measurements z_t for generating the new map $\hat{M}_t$ using the incremental process for map update described in subsection 2.4.1 (Eq. 2.4.17):

$$\hat{M}_t = \hat{M}_{t-1} \cup \{\langle \hat{x}_t, z_t \rangle\} \tag{2.5.3}$$

Maximizing Eq. 2.5.2 is equivalent to finding the vehicle pose x_t which satisfies the motion model of the vehicle according to which the measurement z_t best fits the previously constructed map M_{t-1}. In this context, the maximum likelihood SLAM approach is also termed as *scan matching SLAM*.

The method for scan matching depends on the map representation. Generally, in the outdoor environments where the features are difficult to define and extract, direct scan matching approaches such as Iterative Closest Point algorithm (ICP) are used. Our approach for scan matching is also based on the classical ICP algorithm and we use different known methods for improving its performance. Our solution to the SLAM problem maintains a single robot pose estimate at each time step. The reason for maintaining only one pose hypothesis is that combining 3D ICP scan matching and multiple poses in a multi-hypotheses approach is computationally very demanding.

Grid-based Scan Matching with ICP

In this section we detail our implementation of the ICP algorithm for scan matching in the context of a 3D occupancy grid with the optimizations that we achieve.

The ICP algorithm is commonly used for registering two sets of points in a common coordinate system. The input to the algorithm is two independently generated sets of 3D points M and D corresponding to a single shape where M is called the *model set* and D is called the *data set*. The purpose of ICP is to compute an optimal transformation between the two sets which minimizes the cost function:

$$E(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^{N_m} \sum_{j=1}^{N_d} w_{i,j} \parallel \mathbf{m}_i - (\mathbf{R}\mathbf{d}_j + \mathbf{t}) \parallel^2 \tag{2.5.4}$$

Here, $(\mathbf{R}, \mathbf{t})$ defines the transformation consisting of a rotation $\mathbf{R}$ and a translation $\mathbf{t}$. N_m and N_d are the number of points in the set M and D respectively, i.e. $N_m = |M|$ and $N_d = |D|$. $w_{i,j}$ is the weight function which is assigned 1 if i-th point of M describes the same point in space as the j-th point of D and 0 otherwise. In order to minimize the error function, we need to compute two things: the point correspondences and transformation $(\mathbf{R}, \mathbf{t})$ which minimizes the function $E(\mathbf{R}, \mathbf{t})$ based on the computed point correspondences.

The point correspondences are calculated iteratively by the ICP algorithm. In each iteration, the closest points in the two sets are selected as correspondences and a transformation $(\mathbf{R}, \mathbf{t})$ is computed to minimize Eq. 2.5.4. It is assumed that in the last iteration the exact point correspondences are achieved. However, in our case, the exact correspondences are not possible as the two sets of points can overlap only partially. Therefore we use a maximum tolerable distance d_{max} as a termination condition for the iterations.

The basic ICP algorithm that we have described above for registering 3D scans is known as *pairwise matching*. In this method, each new scan is aligned with a previously acquired scan. However, we need to register each new scan with the existing map constructed from the previous sets of scans. In order to do so, we use an *incremental matching method* in which each new scan is registered with the union of previously received and aligned scans. This union of scans which constitutes the map of the environment on the previous time step $\hat{M}_{t-1}$ is also called the *metascan*.

As in our case the scans to be registered are the 3D range measurements converted to occupancy grid maps therefore our method is directly based on aligning the grid maps. In order to register a sequence of scans, we construct an occupancy grid map for each scan, separate the ground measurements and align it with the previously constructed map using incremental ICP algorithm. As we have mentioned

before, our optimized octree-based representation of the 3D occupancy grid maps considerably reduces the size of the data which is further reduced by our method for ground segmentation. Whenever a new scan arrives, we convert it to the 3D occupancy grid representation. The center of each voxel of the 3D occupancy grid is used as a point to generate a sub-sampled point cloud. This point cloud is then registered with the existing occupancy grid map formed from all previous scans. For initial estimate of the transformation, we use the odometry information provided by the proprioceptive sensors of the vehicle.

One of the major problems in ICP is to efficiently find the point correspondences in order to minimize the cost function. For this purpose, we used an approximate nearest-neighbor search algorithm proposed by Arya and Mount (1997) which is available as an open source library named ANN[4]. This algorithm is implemented with k-d trees to accelerate the nearest neighbor search. Moreover, instead of searching for the *true* nearest neighbors, this algorithm searches for the *approximate* nearest neighbors within a clipping threshold ϵ i.e. it allows for the distance between two approximate neighboring points to exceed the distance between the true nearest neighbors by a factor of $(1+\epsilon)$. Adjusting the value of ϵ can further accelerate the search process. Setting the value of ϵ to 0 corresponds to the true nearest neighbors.

With the calculation of the point correspondences, the optimal transformation $(\mathbf{R}, \mathbf{t})$ can be computed which minimizes Eq. 2.5.4. The error function in Eq. 2.5.4 can be reduced to:

$$E(\mathbf{R}, \mathbf{t}) \propto \frac{1}{N} \sum_{i=1}^{N} \parallel \mathbf{m}_i - (\mathbf{R}\mathbf{d}_i + \mathbf{t}) \parallel^2 \tag{2.5.5}$$

where $N = \sum_{i=1}^{N_m} \sum_{j=1}^{N_d} w_{i,j}$. Moreover, the point correspondences can be described as a vector $\mathbf{v}$ which contains the point pairs:

$$\mathbf{v} = ((\mathbf{d}_1, \mathbf{m}_{f(\mathbf{d}_1)}), (\mathbf{d}_2, \mathbf{m}_{f(\mathbf{d}_2)}), ..., (\mathbf{d}_N, \mathbf{m}_{f(\mathbf{d}_N)})) \tag{2.5.6}$$

Here, $\mathbf{d}_i$ and $\mathbf{m}_i$ are the corresponding points in set D and M respectively and $f(\mathbf{d}_i)$ is the search function which yields the closest point. Thus by the definition of ICP, it is assumed that in the last iteration, the point correspondences, and thus the vector $\mathbf{v}$, are correct.

In order to minimize $E(\mathbf{R}, \mathbf{t})$ in (2.5.5), we have used the singular value decomposition (SVD) based method due to its robustness and ease of implementation. This

[4]http://www.cs.umd.edu/ mount/ANN/

method enforces the orthonormality of the rotation matrix $\mathbf{R}$. In order to do so, the first step is to use the centroids of the points in the matching to decouple the calculation of rotation $\mathbf{R}$ from the translation $\mathbf{t}$. We compute the centroids $\mathbf{c}_m$ and $\mathbf{c}_d$ of the two sets M and D by:

$$\mathbf{c}_m = \frac{1}{N}\sum_{i=1}^{N}\mathbf{m}_i, \mathbf{c}_d = \frac{1}{N}\sum_{i=1}^{N}\mathbf{d}_i \tag{2.5.7}$$

In the next step, we compute the sets of points M' and D' by subtracting the centroids from each point in their corresponding set M and D, i.e.:

$$M' = \{\mathbf{m}'_i = \mathbf{m}_i - \mathbf{c}_m, i = 1, ..., N\}, D' = \{\mathbf{d}'_i = \mathbf{c}_i - \mathbf{c}_m, i = 1, ..., N\} \tag{2.5.8}$$

Next, we replace Eq. 2.5.7 and 2.5.8 in Eq. 2.5.5 and the error function $E((\mathbf{R}, \mathbf{t})$ reduces to:

$$E(\mathbf{R}, \mathbf{t}) \propto \sum_{i=1}^{N} \parallel \mathbf{m}'_i - \mathbf{R}\mathbf{d}'_i \parallel^2 \text{ with } \mathbf{t} = \mathbf{c}_m - \mathbf{R}\mathbf{c}_d \tag{2.5.9}$$

Now, the optimal rotation $\mathbf{R}$ is calculated by $\mathbf{R} = \mathbf{V}\mathbf{U}^T$ where the matrices $\mathbf{V}$ and $\mathbf{U}$ are derived by the singular value decomposition (SVD) of a 3×3 correlation matrix $\mathbf{H}$ i.e. $\mathbf{H} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$. The matrix $\mathbf{H}$ is defined as:

$$\mathbf{H} = \sum_{i=1}^{N}\mathbf{d}'_i\mathbf{m}'^T_i = \begin{pmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{pmatrix} \tag{2.5.10}$$

where $S_{xx} = \sum_{i=1}^{n} m'_{ix}d'_{ix}$, $S_{xy} = \sum_{i=1}^{n} m'_{ix}d'_{iy}$, $S_{xz} = \sum_{i=1}^{n} m'_{ix}d'_{iz}$ etc. After calculating rotation, we calculate the translation $\mathbf{t}$ by using $\mathbf{t} = \mathbf{c}_m - \mathbf{R}\mathbf{c}_d$.

Once we find the optimal rotation $\mathbf{R}$ and translation $\mathbf{t}$, we apply it to the occupancy grid voxels generated from the current scan including the ground voxels thus registering them with the existing grid map. As a by product, we obtain the corrected pose of the vehicle too. Figure 2.12 is an illustration of the results we obtain after matching a scan with a grid map.

Our approach has reduced the computing time for ICP by two methods. First, we use the octree occupancy grid for scan matching which is a sub-sampled version of the original point cloud. After segmenting the ground voxels, the remaining voxels (which serve as the point cloud for scan matching) are further reduced by a large factor. Second, we employ a fast approximation of the point correspondences in ICP

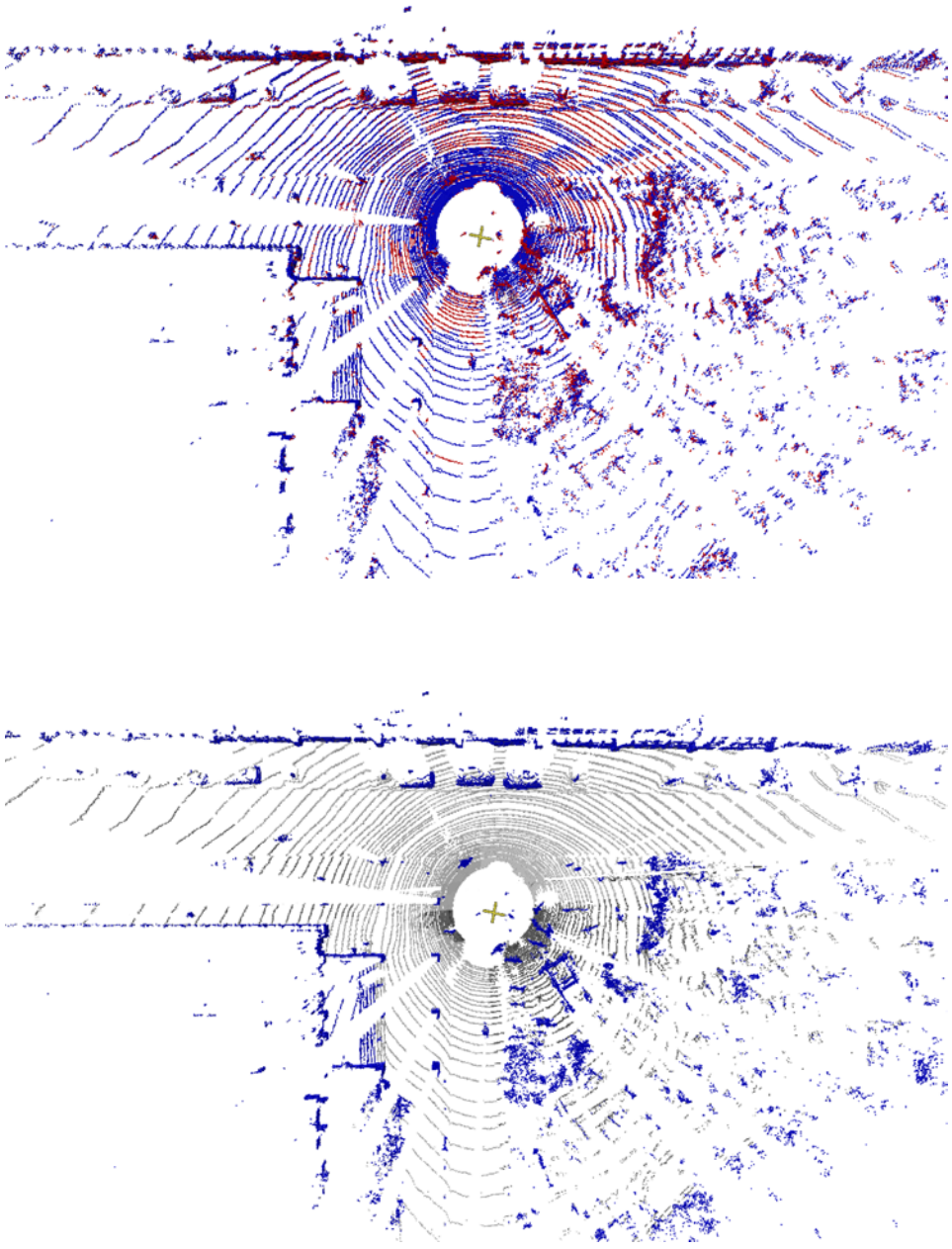


Figure 2.12: An illustration of our grid-based scan matching method. We consider two consecutive scans. (Top) The 3D occupancy grid corresponding to the first scan, represented in red, is used as the *meta scan* or the existing map M_{t-1}. The next scan is also converted in to the occupancy grid map, represented by blue voxels to distinguish from the previous map. It is transformed using the odometry information to illustrate the error in the registration. (bottom) The two maps are aligned by our ICP based approach resulting in the updated map M.

algorithm by using kd-tree based implementation for approximate nearest neighbor search. We down-sample the original point clouds to approximately 30,000 points and adapt the clipping threshold for optimization. This improves the performance of the ICP and makes it feasible to be used in our framework. Moreover, we use the singular value decomposition (SVD) method for minimizing the mean square error (mse) function. The reason for the choice of SVD is that it is robust and easy to implement as compared to the other methods of minimization. As a result, we obtain an implementation of ICP which yields an acceptable performance baseline for localization. However, since we consider the odometry as the initial estimation for the ICP scan matching, it is likely that further specialization in calculating a heuristic initial estimation can make the framework more robust.

2.6 Conclusions

In this chapter, we have presented a background of the SLAM problem and the related work in last few years. In addition to that, we presented a method to perform SLAM with ground segmentation in 3D. In the proposed method, we assumed that the static measurements are separated from the dynamic measurements and we addressed the problem of SLAM only. This separation of measurements is done by the moving object detection step which will be detailed in the next chapter. For SLAM, we used a 3D grid-based representation of the environment and grid-based scan matching is performed which allows estimating vehicle location as well as building a consistent map of the surrounding. As a preliminary step, the ground voxels are detected and separated from the occupied voxels in the grid. In order to perform the scan matching, we used an efficient variant of incremental ICP algorithm and applied it on the reduced point cloud obtained from the occupied voxels other than the ground voxels. As a result, we obtained the updated estimate of the position as well as an updated grid map with a precise differentiation between the ground and non-ground voxels. The ground identification precisely divides the different objects such as walls, parked cars and trees into separate groups of voxels. These groups of voxels can later be clustered and classified to identify the corresponding objects. Our method for ICP based localization requires the odometry for initially estimating the transition and it can be improved by using a heuristic based initial estimation. However, in the given circumstances, it provides a reasonably good performance baseline.

Chapter 3

Detection, Classification and Tracking of Moving Objects

3.1 Introduction

In Chapter 2, we described the modelling of the static environment, focusing on the first perception problem of SLAM with ground segmentation. The outdoor environments cannot be modelled with a static world assumption therefore the intelligent vehicles operating in these environments must be able to perceive the dynamic objects, such as other vehicles, pedestrians and bicyclists, in addition to the static environment. For each dynamic object, the goal is to determine its state, which commonly includes the location and the velocity, in order to predict its position in the nearby future for path planning and obstacle avoidance. Thus, in contrast to the static entities, *detection* of the dynamic objects is not sufficient; one must also *track* them where tracking refers to the process of estimating their trajectories to be able to predict their future locations. This gives rise to the second main problem of intelligent vehicle perception referred to as *Detection and Tracking of Moving Objects* (DATMO) (Fig. 3.1).

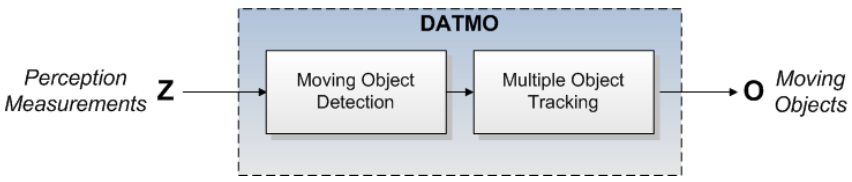


Figure 3.1: Detection and Tracking of Moving Objects (DATMO).

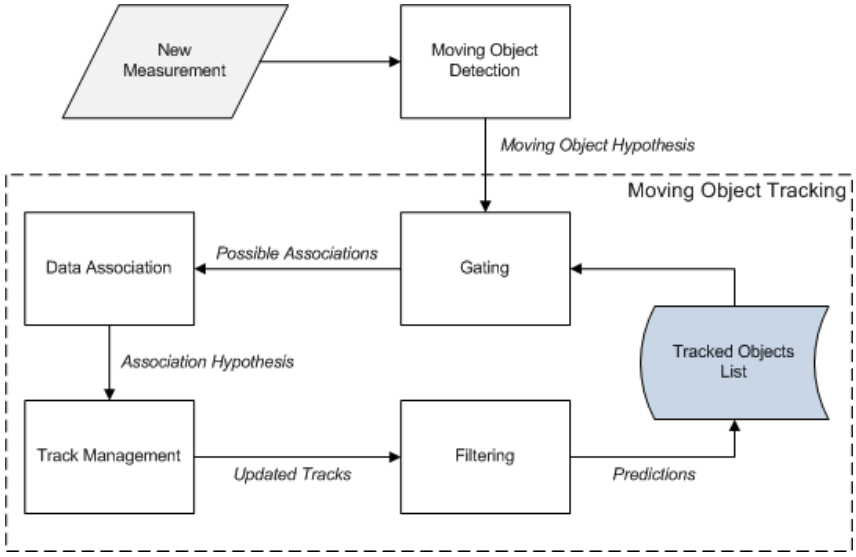


Figure 3.2: Detection and Tracking of Moving Objects: Generic steps.

The two subtasks of DATMO, *detection* and *tracking*, are closely related. Although there are some exceptions such as the *tracking-before-detection* approaches mentioned in (Davey *et al.*, 2008), most of the successful applications at present follow the *tracking-by-detection* paradigm in which the output of *moving object detection* serves as an input for the *tracking* part. A generalized framework for DATMO is outlined in Fig. 3.2 where the tracking part is further subdivided into four steps: *gating*, *data association*, *track management* and *filtering*. *Gating* defines the area around the predicted position of a track where the new observations from the detector are searched to be associated to the track while *data association* is the process in which the actual observation-to-track association hypotheses are computed within the gate area. *Track management* is the step in which the association hypotheses computed in the previous step are used to update the list of tracks. Based on these hypotheses, new tracks are initialized while the existing ones are maintained or deleted. Finally, *filtering* is performed to recursively estimate the state of the tracks from given observations over time. A more general architecture for tracking considers only two steps, *filtering* and *data association*, considering that *gating* is controlled by the predictions made by *filtering* while *track management* depends on the selected technique for *data association* (Bar-Shalom and Fortmann, 1988).

In the next section, we provide the mathematical formulation of DATMO followed by a description of the solutions presented in literature in section 3.3. Section 3.4 gives a summary of our contributions and an overview of the system we developed

in the context of DATMO in 3D using laser scanner. Our approach for detection, classification and tracking of moving objects is detailed in the sections 3.5, 3.6 and 3.7 respectively. A brief summary is provided in section 3.8 to conclude.

3.2 Mathematical Formulation

Considering the SLAM derivations provided in chapter 2, we believe to have a reasonably precise self-localization and a map of the environment. Moreover, as described in the previous section, the detection and tracking can be handled as two separate problems. In this section we provide the basic mathematical formulation for tracking of multiple moving objects assuming that we know how to separate from the sensor data all the measurements originated by the moving objects.

Although there exist some non-probabilistic algorithms for tracking (Kluge *et al.*, 2001; Krishna and Kalra, 2002; Lindstrom and Eklundh, 2001), most of the common and successful algorithms are based on statistical methods to deal with the uncertainty in the measurements (Bar-Shalom and Fortmann, 1988; Burlet *et al.*, 2007; Ess *et al.*, 2010; Montemerlo *et al.*, 2008; Nashashibi and Bargeton, 2008; Petrovskaya, 2011; Schulz *et al.*, 2003; Vu, 2009; Wang *et al.*, 2007; Yu *et al.*, 2008). The idea is, usually, to implement the general recursive Bayesian filter for keeping track of moving objects. Implementations of Bayesian filtering for tracking include the parametric methods (such as Kalman filter (KF), extended Kalman filter (EKF), unscented Kalman filter (UKF)) and non-parametric methods (such as particle filter (PF)) (Arulampalam *et al.*, 2002). Parametric filters are considered to be more efficient and their computational cost is polynomial in the dimensionality of the state vector. However, they lack in representing complex beliefs arising due to ambiguities in the data. On the other hand, non-parametric methods are able to represent arbitrary beliefs but suffer from the curse of dimensionality i.e. their computational cost increases exponentially with the dimensionality of the state (MacKay, 1998).

For the moving object tracking problem, at a specific time t, the system consists of a set of moving objects $\{o_t^1, o_t^2, ..., o_t^{k_t}\}$. The number of objects k_t can be different at different time steps due to the objects entering and leaving the scene. In the following, we first describe the basic formulation of Bayesian filter for single object tracking followed by the tracking of multiple objects.

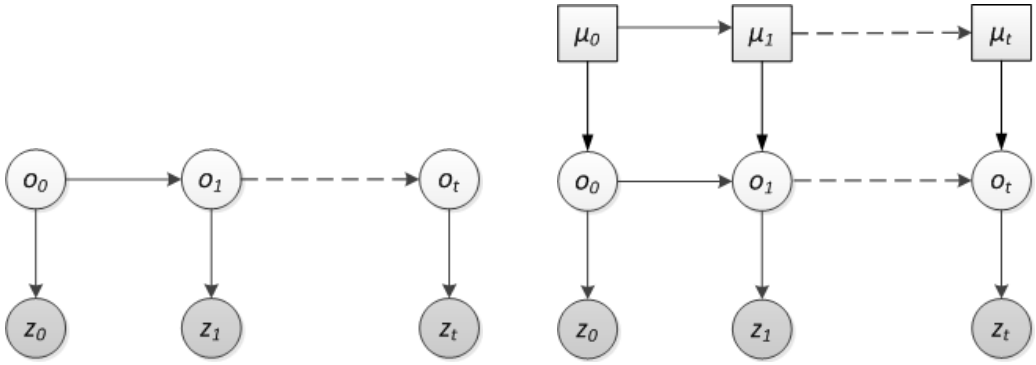


Figure 3.3: Graphical model representation of single object tracking using single motion model (left) and multiple motion models (right). The clear squares represent a hidden discrete variable μ_t that describes the motion of the object at each time step.

3.2.1 Single Object Tracking

The moving object tracking problem for a single target can be formalized probabilistically as a posterior using the recursive update equation of the Bayesian filter (Eq. 2.2.13) derived in section 2.2.2:

$$P(o_t|z_{1:t}) = \eta P(z_t|o_t) \int P(o_t|o_{t-1}) P(o_{t-1}|z_{1:t-1}) do_{t-1} \tag{3.2.1}$$

Here, o_t is the state vector describing the quantities to be estimated/tracked for an object. These quantities are usually kinematic comprising of the position, velocity and sometimes the acceleration of the object and can be constrained by the limits such as a maximum speed or acceleration. The other quantities may include the identity, class (e.g. pedestrian, bicycle, car etc.) or other features (e.g. motion characteristics) of the object. $z_{1:t}$ is the set of measurements received up to time t. Note that the state o_t is inferred only from the previous state o_{t-1} and exteroceptive measurements $z_{1:t}$ as we do not have access to the control inputs of the tracked object. Thus $P(o_t|z_{1:t})$ is a probability distribution that describes our belief about the state of the tracked object at time t given the measurements up to time t. The *dynamic Bayesian network* corresponding to the tracking of a moving object is shown in Fig. 3.3. The three terms describing the posterior, analogous to the SLAM posterior (Eq. 2.2.13), are motion model of the moving object ($P(o_t|o_{t-1})$), measurement model ($P(z_t|o_t)$) and the posterior at time $t-1$ which becomes the prior at time t ($P(o_{t-1}|z_{1:t-1})$). The prior $P(o_0)$ describes the initial knowledge about the state of the tracked object represented by a stochastic process. The sample paths of this process correspond to the possible target paths through the state space. The other

two terms are detailed in the following.

Motion Model

The *motion model* $P(o_t|o_{t-1})$ describes how the state of the object evolves over time. As there is no *a priori* information about the trajectory as well as the control inputs of the moving object, a stochastic model is used to predict the possible motion of the object. In tracking literature, different motion models are used including *Brownian motion*, *constant velocity* and *constant acceleration* model. The simplest model is the *Brownian motion model*. It is often used for tracking pedestrians as they can change their velocity and direction of motion rapidly. In this model, pose of the object is predicted with an addition of zero-mean Gaussian noise whose variance grows with time. Though it does not assume any knowledge about the future evolution of the object but the predicted distribution can be confined by including physical constrains such as maximum velocity.

A more commonly utilized motion model is the *constant velocity model* which estimates the position and velocity of the object using an acceleration noise to model the changes in velocity. In this model, the pose usually evolves via *linear motion model* which is often used when the exact dynamics of tracked objects are not known. In this case, the motion is estimated by perturbing the orientation of the object by $\Delta\theta_1$, then translating forward by a distance $v_t\Delta t$ where v_t is the current velocity and finally, making an adjustment to the orientation by $\Delta\theta_2$. Here, $\Delta\theta_1$ and $\Delta\theta_2$ are either sampled uniformly from $[-d\theta_{max}\Delta t, d\theta_{max}\Delta t]$ or from a normal distribution $\mathcal{N}(0, d\theta_{max}\Delta t)$. A more complete but rather complex model for pose evolution is *bicycle motion model*. This model assumes a constant angular velocity in addition to the linear velocity which defines estimated trajectory of the object in the form of an arc.

In practice, the objects can change their dynamics behaviors over time (e.g. stopped, moving, accelerating, turning etc.) and the motion models for each of these behaviors are quite different. This consideration leads to the requirement for a more complex method such as *Interacting Multiple Models* (Bar-Shalom *et al.*, 2001). This method makes the assumption that, at a given time, the object moves according to one model from a predefined set of models. In this case, at each time step, we need to estimate the corresponding motion model in addition to the state of the object. Figure 3.3 graphically illustrates the difference between single motion model

and multiple motion model approach for solving the object tracking problem. The discrete variables μ_t are the switches to select appropriate motion model at each point in time. Although using multiple models improves the prediction of the pose with a more powerful representation but it increases the complexity of the system (Burlet *et al.*, 2007; Vu, 2009; Wang, 2004).

Measurement Model

Another important term in the posterior (3.2.1) is the *measurement model*, $P(z_t|o_t)$ which defines the measurement process probabilistically. In context of object tracking, the measurement model describes the probability to obtain a specific set of measurements z_t given a state o_t of the tracked object. Measurement models are specific to the sensors used and capture the uncertainty or un-modeled effects of the sensor as noise in the measurements.

There exist two generic approaches in DATMO to model the measurements: using raw data provided by the sensor (*physical sensor models*) and extracting higher order features from the data (*virtual sensor models*). In case of raw data, the main idea is to model the operation of physical sensor directly by taking into account the physical phenomena which occur when the sensor interacts with the world. An example is the modeling of laser beams which is often done using either *independent beam model* (IB) or *likelihood field model* (LF) (Thrun *et al.*, 2005).

The second methods enhance the raw data by performing some sort of preprocessing techniques and use the enhanced data for modeling the sensor virtually. The resulting preprocessed data could range from the extracted features to the centroids of the measurements corresponding to the moving objects (Schulz *et al.*, 2003). In case of range data, the common approaches for preprocessing include sub-sampling the rays, readjusting origin point of rays, projecting from 3D to 2D, filtering out ground as well as segmenting the measurements specific to the moving objects. However, the preprocessing or segmentation to extract the measurements specific to the moving objects can produce false positives therefore it is necessary to ascertain this process in more than one time steps. For vision sensors, the common approaches include converting the image to grey scale, applying Gaussian blur and obtaining 3D point clouds corresponding to the pairs of images (for stereo-vision) (Labayrade *et al.*, 2005).

3.2.2 Multiple Object Tracking

The framework described in the previous section can be used to track several moving objects at the same time. For this, we can directly extend the state vector to include all moving objects. However, there are two additional problems to solve: motion model for several moving objects and data association between the measurements and objects to identify which measurement is generated by which object.

Considering the set of k_t objects $\{o_t^1, o_t^2, ..., o_t^{k_t}\}$ at time t, a state vector O_t is defined for all the objects. The dynamic model for this state vector, denoted by $P(O_t|O_{t-1})$, must define the joint evolution of all the objects. In order to simplify this problem, it is often assumed that the motion of different objects is independent. Thus the joint distribution can be computed by:

$$P(O_t|O_{t-1}) = \prod_{i=1}^{k_t} P(o_t^i|o_{t-1}^i) \tag{3.2.2}$$

where $P(o_t^i|o_{t-1}^i)$ is the motion model for object i. In practice, the independence assumption is not always true as the motion of the objects is not completely independent especially in the cluttered environments. For example, there can be situations when the objects change their trajectories as they approach each other. Consequently, there exist the motion models which take into account the interaction between the objects (del Blanco *et al.*, 2011; Frank *et al.*, 2003; Khan *et al.*, 2005) but those models involve very complex computations as compared to the simple relation described above.

Regarding the measurement model, it again depends on the type of the sensor and the information provided by it. If the sensor provides information about the identity of the object such as its color, texture etc, then the objects can be distinguished from the sensor measurements directly. But, in general, it is often not possible to perfectly identify or distinguish the objects from each other. Thus arises the problem of data association which makes it difficult to directly extend the solution for single object tracking to the problem of tracking multiple objects. Figure 3.4 is an illustration of the data association problem in the simple case of two objects (o^1 and o^2) and two observations (z^1 and z^2) received at a particular time step. There can be two possibilities for data association in this case: ($o^1 - z^1$, $o^2 - z^2$) or ($o^1 - z^2$, $o^2 - z^1$). In case of N objects and N observations, the possibilities of association can be as much as $N!$. Moreover, the number of observations may not directly correspond to the number of existing tracks. In this case, there must exist a method for

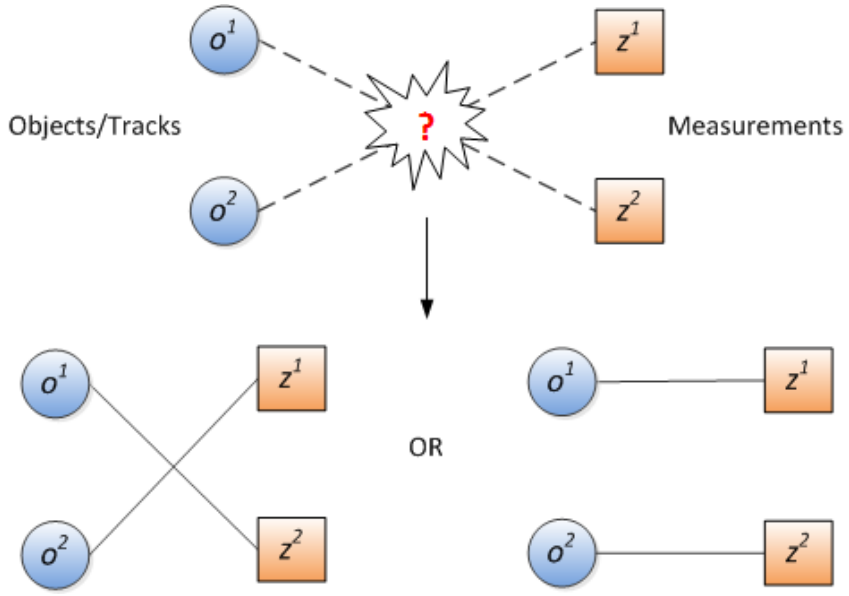


Figure 3.4: Data association problem. Given N existing tracks and N new measurements, there exist $N!$ possible ways to associate the measurements with tracks.

track management to ensure the correct associations as well as initialization and termination of the tracks. A considerable amount of research is done in this regard and there exist several methods to solve the problem of data association such as global nearest neighbor (GNN) (Blackman and Popoli, 1999), multiple hypothesis tracking (MHT) (Reid, 1979) and joint probabilistic data association (JPDA) (Bar-Shalom and Fortmann, 1988). These methods are further discussed in section 3.3.

3.3 Related Work

Tracking of moving objects is a critical component of perception which provides the situational awareness for the intelligent vehicles to make safe decisions in dynamic environments. The breakthrough exploratory work on DATMO for intelligent vehicles emerged from the projects such as PROMETHEUS[1] launched in 1986 by the then Daimler-Benz AG in cooperation with several European partners from both industry and academia (Hellaker, 1990). This research project was followed by a number of initiatives in United States and Japan (Bertozzi *et al.*, 2000; Sun *et al.*, 2006). Since then, a large number of DATMO approaches have been developed, especially in the context of DARPA Grand and Urban Challenges (Darms *et al.*, 2008;

[1]Short for "Program for European Traffic with Highest Efficiency and Unprecedented Safety"

Leonard *et al.*, 2008; Montemerlo *et al.*, 2008; Stiller *et al.*, 2008; Urmson *et al.*, 2008).

Traditional tracking methods, initially proposed for radar and sonar applications for air and ocean surveillance systems, operated on the point observations and assumed that all the observations correspond to moving targets uniquely (Fortmann and Baron, 1978; Singer and Stein, 1971). However, this is not the case in real road scenarios where, together with the dynamic objects, static objects and spurious elements are in abundance. This makes the discrimination between static and dynamic objects, namely *moving object detection*, a critical requirement for a tracking system.

Generally, DATMO approaches follow the *standard tracking pipeline*, illustrated in Fig. 3.2, with the three major stages: *object detection*, *data association* and *Bayesian filter update* (as described in section 3.1). This pipeline separates the tracking stage from the detection stage and thus has the advantage of reduced computational complexity. Further, it allows to use various sensors in parallel. However, it requires a reliable and reproducible detection stage. This is the reason why many of the tracking works use object class specific detectors and hence track only the objects of a certain class e.g. cars (Petrovskaya, 2011; Wender and Dietmayer, 2008) or pedestrians (Schulz *et al.*, 2003; Spinello *et al.*, 2008) which may result in ignoring other important objects such as a cyclist. There exist a few vision-based generic approaches for detection but they require the objects to be relatively large and well separated from the background (Fardi *et al.*, 2006; Schamm *et al.*, 2008). These conditions may not be satisfied for all the scenarios in autonomous driving. An alternative method is to combine the object tracking with SLAM to detect and track generic objects. These approaches often use the 2D occupancy grids and track the generic objects by taking into account the parts of grid which change their occupancy (Vu *et al.*, 2008; Wang, 2004). The results presented by these approaches are highly promising but are limited to the 2D environments only.

An alternative paradigm to the *standard tracking pipeline* considered by a few researchers recently is the *track before detect* methodology (Davey *et al.*, 2008). It postpones the detection stage and quantizes the sensor data to use for tracking directly. Examples in 2D include the *stixels* quantization of images at fixed columns (Pfeiffer and Franke, 2010) and *Bayesian occupancy filtering* for tracking the cells of an occupancy grid (Brechtel *et al.*, 2010). The mentioned approaches only provide information about the occupied areas of environment and do not provide an object level representation. Moosmann and Fraichard (2010) use this methodology for

segmentation of the objects from the range images corresponding to the 3D laser range data and track those segments. A drawback of this paradigm is that it does not differentiate between the static and dynamic objects and therefore all the detected objects or segments need to be tracked. Due to this fact, the *standard tracking pipeline* still remains the popular methodology for DATMO.

Recent research works emphasized on the need of identifying the object class as an important step for DATMO in urban environments where there are many kinds of objects with considerably different motion dynamics (Ess *et al.*, 2010; Himmelsbach, 2010; Vu *et al.*, 2008). They claim that in such environments the object state vector must include the object semantics (whether the tracked object is a car, bicycle or a pedestrian) in order to achieve reliable tracking. This classification allows to make use of appropriate motion models for different types of objects.

In the context of sensors, many of the proposed DATMO approaches focus on using the vision exclusively (Ess *et al.*, 2010; Katz *et al.*, 2008; Xu *et al.*, 2011) while others use laser scanners (Moosmann and Fraichard, 2010; Nashashibi and Bargeton, 2008; Petrovskaya, 2011; Schulz *et al.*, 2003; Streller and Dietmayer, 2004; Vu *et al.*, 2008; Wang *et al.*, 2007) sometimes in addition to vision (Baig, 2012; Fayad and Cherfaoui, 2007; Labayrade *et al.*, 2005; Spinello *et al.*, 2008; Wender and Dietmayer, 2008). Vision-based methods, considered to be the most frequently used methods for DATMO, are discussed and summarized in (Yilmaz *et al.*, 2006). The authors give an overview of different object representations and features along with the frequently used techniques for object detection including point detectors (e.g. Harris point detector (Harris and Stephens, 1988) or SIFT detectors (Lowe, 2004)), background subtraction and supervised learning (e.g. AdaBoost (Freund and Schapire, 1995) or Support Vector Machines (Boser *et al.*, 1992)). The main disadvantage of pure vision-based approaches is that they do not provide the depth information directly. Stereo-vision can be used for obtaining the extrapolated depth information but it does not match the accuracy of a laser scanner as well as not being robust to the changing light conditions. Consequently, the vision-based systems are sometimes augmented by using 2D laser scanners which introduces an additional overhead of sensor fusion (Wender and Dietmayer, 2008). Moreover, visual classification does not help to distinguish between the moving objects and stationary objects (e.g. a moving car and a parked car). Many approaches for DATMO rely on 2D laser scanner alone (Streller and Dietmayer, 2004; Vu *et al.*, 2008; Wang, 2004) but it provides only a limited amount of information about the objects being tracked.

The field of 3D DATMO is relatively new as the fully three-dimensional laser scanners have been introduced only recently. Some recent works, including most of the successful teams in the DARPA Urban challenge, used a rotating 3D lidar but, surprisingly, many of these reduce the dimension of the 3D data to 2D by either projecting it to a ground plane or to a virtual image plane called *range image*. In the first case, data is projected to an estimated ground plane combined with an occupancy grid and then DATMO is performed using established 2D methods (Montemerlo *et al.*, 2008; Stiller *et al.*, 2008; Urmson *et al.*, 2008). Density of the points within a grid cell is considered to be its occupancy value and the objects are extracted by clustering the cells on the base of this occupancy value (Himmelsbach *et al.*, 2008). This approach assumes the ground to be level and the objects to be vertical, therefore it is not suitable for the outdoor environments with slopes and non-vertical structures. Moreover, a considerable amount of information is discarded due to projecting from 3D to 2D. In the second method, 3D data is projected onto a cylinder with axis same as the rotational axis of the scanner (Moosmann and Fraichard, 2010). The resulting projection is a range image in which the pixel values correspond to the original distance measurements. The detection and tracking can then be performed on this image using the vision-based methods. This approach, again, lacks the detail of the full 3D data and gives a 2D representation only.

In the following, we describe some recent solutions proposed for the detection, classification and tracking of moving objects with a laser scanner.

3.3.1 Detection of Moving Objects

Moving object detection algorithms are often classified into two groups: model-based and model-free methods. Model-based methods detect the objects using explicit models and are tuned for the specific object classes. Model-free methods perform more generic detection and are further categorised into *appearance-based*, *feature-based* and *motion-based* methods. *Appearance-based* methods are used with the vision sensors and not directly applicable to the laser data.

Feature-based methods extract the features from the 2D or 3D scan data and use them for the classification of data points into known classes (Arras *et al.*, 2007; Schulz *et al.*, 2001). The extracted features can be motion features or geometric features including lines, circles or a combination of both. Moreover, the features can be predefined or learned. For example, Schulz *et al.* (2001) use the predetermined features

of local minima in the distance profile of the range scan to detect the people in indoor environments. On the other hand, Arras *et al.* (2007) use the approach to learn the features from the hand-labeled data. With a 3D laser scanner, as mentioned above, a common practice is to use the 2D projection into range images and apply computer vision techniques. Stiene *et al.* (2006) use this approach to apply a silhouette extraction method to the range images. They use a fast eigen-CSS method to extract the features and support vector machines (SVM) to perform the supervised learning and classification. The results are compared to the standard methods for object detection from range images. A more recent example of the application of computer vision algorithms to the range image is proposed in (Moosmann and Fraichard, 2010). The goal was to develop a class-independent approach similar to optical flow in intensity images. They used the range image to segment the data with a local convexity criterion and then use the corresponding 3D data to track the motion of those segments. A similar approach is used in (Morris *et al.*, 2008) with a combination of 2D and 3D scans instead of the range images. They use 2D scans to generate the hypothesis for moving vehicles and 3D scans for further examining those hypotheses. A linear support vector machine is used to discriminate the vehicles from the background.

Motion-based methods are often used with the range sensors where the main idea is to detect the moving objects by detecting occupation of the previously unoccupied space. A popular approach is to construct an occupancy grid map incrementally and apply the techniques similar to the background subtraction method of computer vision (Burlet *et al.*, 2007; Vu *et al.*, 2008; Wang *et al.*, 2003). The constructed occupancy grid serves as the background modeling process. At each time step, new data is compared with the occupancy grid map and the inconsistencies between them are identified. These inconsistencies give the individual measurements possibly corresponding to the moving objects which are further clustered together to represent the objects. An alternative approach proposed in (Yu *et al.*, 2008) is to first cluster the measurements obtained from the 2D laser scanner using k-nearest neighbors and then compute the movement parameters of the clusters by local grid map matching.

Vu *et al.* (2008) and Petrovskaya and Thrun (2009) have discussed the problems posed by the model-free detection approaches in the context of laser scanners. For instance, an object can be divided into several segments due to partial occlusions as well as glassy or black surfaces. Moreover, at any specific time, only parts of the ob-

ject facing the sensor are visible which makes it difficult to identify and track these objects correctly. They state that a geometric model of the objects can improve the object detection and consequently the tracking results by handling disjoint point clusters. Vu *et al.* (2008) use fixed models to represent four different classes of moving objects namely bus, car, bike and pedestrian with 2D laser data. Petrovskaya and Thrun (2009), though use 3D laser data but reduce it to a 2D projection, which they call a *virtual scan*, to apply the efficient and well-established 2D methods. They perform the detection only for the vehicles using flexible box model which adapts to the size of the vehicle during tracking. Presence of a digital map of the road is also assumed which reduces the search area for the new tracks but limits the approach only to the structured environments. An extension to the unstructured environments is proposed in (Wojke and Haselich, 2012) which uses the temporal and geometric clues to separate the vehicles from the background. However, as mentioned before, all these motion-based detection methods either use the 2D laser data or the 2D projection of 3D laser data and none of them uses the 3D data directly.

3.3.2 Tracking of Moving Objects

As mentioned in section 3.1, multiple object tracking problem consists of two main steps: *filtering* and *data association*. The most common tracking methodologies use Kalman filter for estimating new state of the tracked objects from given observations over time (Fayad and Cherfaoui, 2007; Mendes *et al.*, 2004; Moosmann and Fraichard, 2010; Nashashibi and Bargeton, 2008; Premebida and Nunes, 2006; Stiller *et al.*, 2000; Streller and Dietmayer, 2004; Streller *et al.*, 2001). The estimated positions of the objects are compared to the observations through *gating* process using a distance measure (Fayad and Cherfaoui, 2007; Lindstrom and Eklundh, 2001; Stiller *et al.*, 2000; Yu *et al.*, 2008) or a validation region (Premebida and Nunes, 2006). Some tracking approaches augment the Kalman filter method by using interacting multiple models (IMM) for describing multiple motion modes of the objects where a separate Kalman filter is defined for each motion model (Burlet *et al.*, 2007; Vu *et al.*, 2008; Wang *et al.*, 2007).

Although tracking literature is dominated by the variants of Kalman filter, there exists a significant amount of literature using particle filters too which is capable of estimating the evolution of non-linear models with non-Gaussian noise (Arulam-

palam *et al.*, 2002). Särkkä *et al.* (2007) use a Rao-Blackwellized particle filter for tracking unknown number of targets. A similar approach is used in (Vihola, 2007) with the finite set statistics for multi target tracking. Rao-Blackwellization is used to improve the efficiency of particle sampling. However, its application for the high dimensionality of 3D tracking problem is limited due to the large number of simulations required. Recently, some methods are proposed for GPU implementation of the particle filter tracking to overcome this limitation (Brown and Capson, 2012).

Filtering estimates the state of exactly one object and requires observations to update the estimated state. Association between the estimated state and observations, considered as the most challenging stage of DATMO, has been studied extensively in last couple of decades. It is a complex task as the observations generally do not correspond to the number of objects. Moreover, the number of objects is difficult to estimate since an object might be temporarily occluded or unobserved. The data association for multi-target tracking comprises of deducing the number of true objects and identifying if each observation corresponds to a tracked object, a spurious measure or a new object in the scene. The complexity of the problem increases exponentially with the number of objects in the scene.

The simplest data association method is *global nearest neighbor* (GNN) (Blackman and Popoli, 1999) which associates the measurement to the nearest object according to a given metric such as Euclidean or Mahalanobis distance. A validation gate is used to limit the maximum distance for an association. The main advantage of this method lies in its smallest cost of calculation. However, it is a purely sequential method and the decision of the association is immediate and irrevocable: the hypothesis having highest probability is considered as true (Rong Li and Bar-Shalom, 1996). Thus it makes a hard decision at each point in time. Once an association is established, it gets fixed and unchangeable.

A possible solution for this problem is to use the suboptimal Bayesian data association such as the *joint probabilistic data association* (JPDA) filter (Bar-Shalom and Fortmann, 1988). JPDA is a method that considers all possible associations but maintains a single filter. At each time step, instead of finding a single best association between observations and existing tracks, JPDA estimates the probability of each possible association. The state of each target is estimated by a filter which takes into account all possible associations weighted by their probability. Though this approach was originally proposed with the Kalman filter (Bar-Shalom and Fort-

mann, 1988) but it can also be used in the sample-based frameworks (Schulz *et al.*, 2003). Recently, Svensson *et al.* (2011) proposed an optimization of the JPDA ignoring the identity of the tracked objects. They assume that the only important thing is the existence of the track and not its identity (or labeling). This assumption can not be valid in the applications where the object class is used to deduce the dynamics of the tracked objects. In comparison to GNN, JPDA has proven to be more efficient in cluttered environments but still the possibility of erroneous decision exists as only a single scan is considered for association. Moreover, as in the case of GNN, the association made in the past is not reversible.

A more robust technique for association is *multiple hypothesis tracking* (MHT). Originally proposed by (Reid, 1979), it is affluently used and still proves to be an excellent approach for multiple object tracking (Blackman, 2004; Burlet *et al.*, 2007; Koller and Ulmke, 2005; Lau *et al.*, 2010; Ryoo and Aggarwal, 2008; Thomaidis *et al.*, 2010). Unlike the previous strategies, MHT is a multi-scan algorithm which retains all possible association hypotheses until there is enough information to resolve the ambiguities which occurred in the previous time steps (Thomaidis *et al.*, 2010). The main disadvantage of this approach is its complexity in the pure form due to the exponential growth of association hypotheses in the presence of ambiguities. Various methods exist which implement the prune and merge operations to reduce the number of hypotheses but they result in compromising the *maximum a posteriori* (MAP) property of the method (Blackman, 2004).

Another solution to preserve the MAP property in data association is to use an optimisation technique. The Viterbi algorithm (VA) (Forney Jr, 1973) is an established optimisation method for discrete Markovian systems which has been extensively used in speech recognition. It is actually a batch algorithm, although in practice it may be used in a fixed-lag processing mode due to merging of paths in the trellis (Forney Jr, 1973). The application of Viterbi algorithm to the data association problem in the single target case was proposed in (Gad and Farooq, 2003). The essential idea is to create a trellis based on the measurements rather than the states. Any path through the trellis corresponds to a sequence of data associations. The Viterbi algorithm is harnessed to determine the shortest or lowest cost path through the trellis. This technique is not optimal for tracking because the sequence of data associations conditioned on the estimated states is not Markov. Nonetheless, convincing results have been achieved over existing single-scan approaches such as JPDA (Gad and Farooq, 2003). The so-called Viterbi data association (VDA) approach has been

used for multiple object tracking for widely separated targets (Pulford, 2006) and for pedestrians in a cluttered environment (Azim and Aycard, 2010).

3.3.3 Classification of Moving Objects

Classification of moving objects is an interesting addition to the DATMO process which often becomes essential in the case of 3D range data. Each detected object is compared against a possible list of classes, consisting of the frequently observed objects (e.g. cars, pedestrians, bicycles) and the appropriate class is assigned to it. The dynamics of the assigned class are used for filtering which improves the performance of tracking. Most of the methods use vision sensors (Premebida and Nunes, 2006), sometimes coupled with the laser scanners (Premebida *et al.*, 2007), to perform classification of objects with well-established methods of computer vision.

With 2D laser range data, the simplest methods for classification fit the bounding boxes to the clusters corresponding to the detected objects. The dimensions of these boxes are then used to identify the class of the object (Dietmayer *et al.*, 2001; Petrovskaya and Thrun, 2009; Vu *et al.*, 2008). Dietmayer *et al.* (2001) used the prior knowledge about the typical road users for the classification of detected objects. Predefined length and width values for each object class are compared against the bounding box of each detected object and appropriate class is selected. An additional verification phase is introduced to verify the class assignment using expected dimensions and dynamic constraints of that class. If the probability of a different class increases, then the class assignment is changed. Mendes *et al.* (2004) proposed a similar approach with additional features used for classification which contribute a weighted vote towards a class. The class with the highest score is assigned to the object. However, these approaches can not be directly extended in the context of 3D data as the point cloud corresponding to the object varies highly with its position relative to the range scanner (Mendes *et al.*, 2004). Moreover, the approaches that use the object dimensions for classification suffer from the problem of occlusion. The dimensions for partially occluded objects can not be calculated correctly. Nashashibi and Bargeton (2008) address this problem by considering the occlusion while computing the votes. The probabilistic formulation of this approach is provided in (Zhao *et al.*, 2006). Vu *et al.* (2008) and Petrovskaya and Thrun (2009) use model-based detection to handle the partial occlusion which is again limited to the case of 2D data only.

A hybrid approach for classification uses laser scanner to detect the objects and to specify a region of interest (ROI) in the camera image. The ROI reduces the area to be searched in the image and consequently the processing time. Premebida *et al.* (2007) apply an AdaBoost classifier to the ROI in the image and a Gaussian mixture model classifier to the laser data. The results of the two classifiers are combined using a sum decision rule to generate the final classification.

Some of the existing classification approaches using 3D range data alone tend to label the scene directly into regions corresponding to the object classes, possibly including a background class. These methods use the repetitive instances of similar objects to define a class. For instance, Anguelov *et al.* (2005) use a supervised approach based on Markov random fields (MRF) to compute local features from individual data points and produce globally consistent class labels. Triebel *et al.* (2010) employ an unsupervised method by using conditional random fields (CRF) to build a neighbourhood graph of planes segmented from the 3D point clouds where each plane is considered to be an object part. Subsequently, the combinations of segments (object parts) which occur multiple times are searched and assigned a class label. These methods work on 3D indoor-data which makes the segmentation relatively simple. An unsupervised approach for outdoor scenarios is presented by Moosmann and Sauerland (2011) where the objects are first segmented by a region growing algorithm and a feature vector is calculated for each segment. The similar feature vectors are then clustered to form groups of segments which have similar geometric properties. Finally, the clusters are analysed in order to define the object classes. These approaches provide a good segmentation of the data along with the labeling of repeated instances of objects without predefined classes or object models. However, they are only capable of discovering frequently occurring geometric structures and none of them provides a semantic information about the type of the objects (e.g. car, bicycle, tree, wall, etc.). Thus they are not suitable to be used for tracking of moving objects directly.

Alternatively, some other approaches for 3D range data assume that either the point clouds corresponding to the objects have already been segmented (Teichman *et al.*, 2011) or can be segmented from the ground projection of the data (Douillard *et al.*, 2011; Himmelsbach *et al.*, 2008) and focus only on classification. Himmelsbach *et al.* (2008) build a ground projection based 2D occupancy grid from 3D data and cluster the grid cells not belonging to the ground to represent the object hypotheses. In the next step, the 3D point clouds corresponding to the segmented objects are

back-projected which are then used for feature extraction and classification by a support vector machine (SVM). The SVM is trained with the hand-labeled data and the objects are classified into two classes namely *vehicles* and *non-vehicles*. Another approach for segmentation based on ground identification is proposed by (Douillard *et al.*, 2011).

A different approach of classification in conjunction to tracking is presented in Teichman *et al.* (2011) which classifies the complete tracks instead of the individual segments of the objects. The first step, similar to the above mentioned approaches, is the 2D grid-based local ground plane removal and segmentation by connected components clustering of the remaining points. The segments are then tracked by Kalman filter based tracker. The track classification is done by two boosting classifiers: one based on the shape of the object at each frame, and second on motion descriptors of the entire track. The two predictions are combined by a discrete Bayes filter. The approach assumes a perfect segmentation and the discrimination between the static and dynamic objects is only made on the base of tracking.

3.3.4 Synthesis

The problem of DATMO has been extensively studied by the robotic community for several decades but accomplishing this task from a ground vehicle in outdoor environments still remains difficult. Proposed strategies use different sensors where the vision-based approaches have been most prevalent thanks to the greater field of view and lower cost of the cameras. Moreover, the additional color information used for identifying the specific objects outweighs the disadvantages of low precision in depth information. In order to improve the precision, some approaches use a combination of camera and laser range finder. However, this might strongly affect the computational complexity by introducing additional problem of fusion between the sensors. Furthermore, the field of view and range of the two sensors is not the same and therefore not all the information available can be utilised. Although there are well-known methods for DATMO using 2D laser scanners, approaches using a 3D range scanner are relatively rare in literature. Most of those which exist, rely on reducing the dimensionality of the problem by projecting 3D data to 2D. Others which utilize the full 3-dimensional data, usually depend on a segmentation of the point cloud in to individual objects, regardless of distinguishing between the static (e.g. walls, trees, fences etc) and dynamic objects (e.g. cars,

bicycles, pedestrians etc). This differentiation is later made either on the base of classification using known models of the objects of interest, for instance other road users, or by tracking the possible motion of each object. For the first case, the classification yields all the instances of a specific class (e.g. cars) whether they are moving or not. Moreover, the classification considers all the object hypothesis generated by segmentation including the static objects which can amount to a large number in unstructured outdoor scenarios. Consequently, the classification needs to handle too many of the objects which are not of a specific interest for DATMO. Same is the problem for the tracker in the second case. A possible solution for 3D DATMO can be a hybrid approach using motion-based detection and then classification into the predefined objects of interest. A motion-based approach for detection, similar to the ones which exist for 2D data, will return only the objects which are possibly moving at a specific time and thus reduce the number of objects to consider. Then, those detected objects can be classified for tracking. In the context of multiple object tracking, we have observed that MHT is considered to be the best option. However, it suffers from the problem of computational complexity due to exponentially growing number of hypotheses. The prune and merge methods which attempt at reducing the number of hypotheses result in the loss of the MAP property of MHT. A possible solution is to use an optimisation method such as Viterbi algorithm. Our approach for DATMO with a 3D laser scanner uses motion-based detection and density-based clustering for segmenting the moving objects from the 3D occupancy grid. As established in some other recent works, we also posit that in order to successfully navigate in outdoor dynamic scenarios and to correctly estimate the objects' motion paths and future locations, an environment model must include the object semantics i.e. whether a moving object is a car or a pedestrian. Therefore, a key component of our approach is the classification of detected objects into known categories namely: pedestrian, bicycle, car and bus. We propose a novel approach to classification in 3D range data based on supervised learning technique. The detected object is divided into layers to define a set of weak classifiers for different height levels of the object. The division of the objects to layers allows to apply 2D methods to each layer for defining the weak classifier. Subsequently, a strong classifier is trained from the features extracted at each layer by applying AdaBoost which returns the eventual classification of the objects. Finally, we present a new solution to the problem of multiple objects tracking using the Viterbi data association.

3.4 Contributions: 3D Occupancy Grid DATMO

In this section, we present the method that we have developed for the grid-based detection and tracking of moving objects. The detected objects are classified into four classes: pedestrian, bicycle, car and bus and tracked using Viterbi data association.

3.4.1 General Architecture

Motion-based approaches detect the moving objects by detecting occupation of the previously unoccupied space. In this work, we have used an occupancy grid based approach for motion detection which is often used with 2D range sensors. The basic idea is to use the octree-based Occupancy Grid representation of the environment that we presented in chapter 2 and to detect moving objects based on inconsistencies between the scans. The proposed method for discrimination between moving and stationary objects without a priori knowledge of the targets is the first main contribution of this chapter. Second main contribution is our method for classification of the detected moving objects into four classes: pedestrian, bicycle, car and bus. Since we have knowledge of the object classes of interest, and the set of those classes is comparatively small, we apply a supervised approach for classification. Moreover, a preliminary binary classification is already done by the detection step which separated the possibly dynamic data from the rest. As a final contribution, we present an extension of the Viterbi algorithm to solve multiple objects tracking in cluttered environments. The complete end-to-end framework of DATMO is illustrated in Fig. 3.5. Next section describes the proposed method for grid-based detection of moving objects. The classification and tracking of those objects are explained in section 3.6 and 3.7 respectively.

3.5 Detection of Moving Objects

In this section, we describe our approach for detection of moving objects from the octree occupancy grid map presented in chapter 2. We first describe the method for detecting the voxels which possibly belong to the dynamic objects in section 3.5.1. Section 3.5.2 describes our method for clustering the individual dynamic voxels to represent the objects.

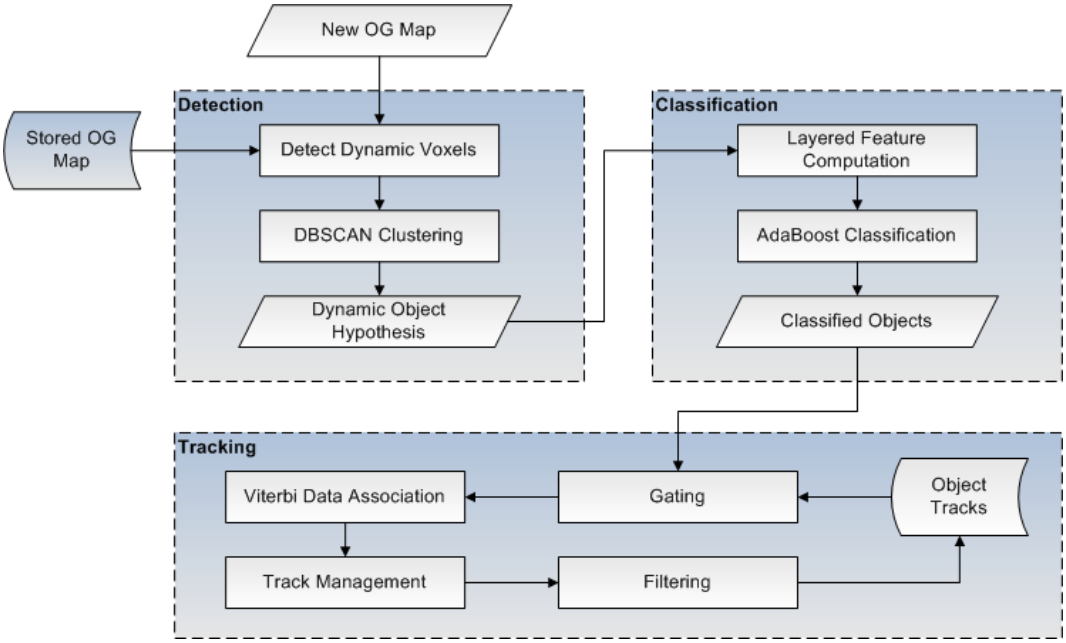


Figure 3.5: Architecture of proposed method for DATMO.

3.5.1 Motion-based Detection

After the construction of a consistent local map of the vehicle from SLAM, moving objects can be detected when new measurements arrive. The principal idea of our approach for the hypothesis of a moving object is based on the inconsistencies between observed free space and occupied space in the local grid map. This method borrows idea from background-subtraction methods in computer vision. Our process for the detection of moving objects is carried out in two steps: detection of dynamic voxels and there segmentation into individual dynamic objects.

The first step is to detect the voxels that might be containing measurements obtained from dynamic objects. This can be considered as a background modeling process. In this step, we construct a 3D occupancy grid map incrementally from laser measurements, as explained in chapter 2, and based on the constructed grid map we are able to make a hypothesis about the voxels of the grid occupied by moving objects when new measurements arrive. For this, we maintain a list of voxels whose states are inconsistent between the current and previous scan.

Let S_{t-1} and S_t be the states of a voxel in previous scan and current scan respectively. If the transition between these two states for a specific voxel of the grid is such that $S_{t-1} = free$ and $S_t = occupied$ then this is the case when an object is

detected on a location previously seen as $free$ space and it is possibly a moving object. We add it to the list of possible dynamic voxels. In contrary, if $S_{t-1} = occupied$ and $S_t = free$, it means that the location which was previously being observed as $occupied$ is $free$ now. This can possibly be caused by a missed detection by the sensor or it was a voxel occupied by a dynamic object which may have displaced now. We search this voxel in our list of dynamic voxels maintained from previous scans. If it is found, we wait for the next few scans instead of removing it from the dynamic voxels list immediately. If it is observed as $free$ in next scans as well, then we delete it from the list.

If $S_{t-1} = occupied$ and $S_t = occupied$, it means that an object is observed on a location previously $occupied$ then it probably is static. If an object appears in a previously not observed location, then we can say nothing about that object. For such measurements, a priori we will suppose that they are static until later evidences come. As a result of this step, all the inconsistencies between the two measurements are identified as dynamic voxels. These include a large number of sparsely situated voxels generated as noise. Further steps deal with this issue by differentiating between the noise and measurements corresponding to the possible dynamic objects.

Once we have maintained the list of all possible dynamic voxels, the next step consists of the segmentation of detected dynamic measurements into regions. It is carried out by clustering these dynamic voxels into separate groups where each group represents a single object. The criterion used for deciding whether the voxels belong to the same cluster is the Euclidean distance between their centers.

3.5.2 Density-based Clustering

We can intuitively expect that all voxels belonging to a specific cluster are neighboring or at least spatially very close to each other. Thus, we do not require to exhaustively compare the voxels pairwise to check whether they belong to a cluster or not. The clustering can be performed using an approach similar to a region-growing algorithm. In this approach, we examine the neighboring voxels of initial *seed point* voxel and determine whether the neighbors should be added to the region or not. The *seed points* are selected from the detected dynamic voxels randomly.

In this work, we have used a highly cited algorithm for clustering known as *density-*

based spatial clustering of applications with noise (DBSCAN) (Ester *et al.*, 1996). DBSCAN defines a cluster based on the notion of *density reachability*. In order to explain the term *density reachability*, we use the concept of ε-neighborhood $N_\varepsilon(p)$ of a point p which consists of all the points q which are within a distance ε of p, i.e.:

$$N_\varepsilon(p) = \{q \in D \mid dist(p,q) \leq \varepsilon\} \tag{3.5.1}$$

The point q is defined to be *directly density-reachable* from the point p if it is a part of the ε-neighborhood of p and p has sufficient number of points in its ε-neighborhood to be considered as part of a cluster. On the other hand, if q is not within the ε-neighborhood of p, however, there exists a sequence $p_1, \ldots, p_n$ of points with $p_1 = p$ and $p_n = q$ such that each p_{i+1} is *directly density-reachable* from p_i, then q is called *density-reachable* from p.

It is important to note that the *density-reachability* is not a symmetric relation. For instance, the point q may lie at the edge of a cluster and it may not have sufficient number of neighbors. If the recursive process of clustering starts from this point then it will stop immediately as it will not find enough density in the neighborhood. However, if the process starts from the point p, it would lead to the point q and would stop there. Thus q would be *density-reachable* from p but not the vice versa. This asymmetric relation leads to another notion of *density-connectedness*: two points p and q are *density-connected* if there exists a point o such that both p and q are *density-reachable* from o. All these notions are illustrated by an example in Fig. 3.6.

Thus, by definition of DBSCAN, a cluster must satisfy two constraints:

- All points in the cluster should be *density-connected* to each other.
- If a point is *density-connected* to any point of a specific cluster, it is part of that cluster as well.

The main concept of this algorithm is to continue expanding a cluster as long as the density of the ε-neighborhood is greater than a threshold. Thus the neighborhood distance threshold (ε) and the minimum number of points ($minPts$) required to define a cluster are the two main parameters of this algorithm.

In our implementation of the grid-based DBSCAN, as described above, all possible dynamic voxels are stored in a data list. Our clustering algorithm starts with step-

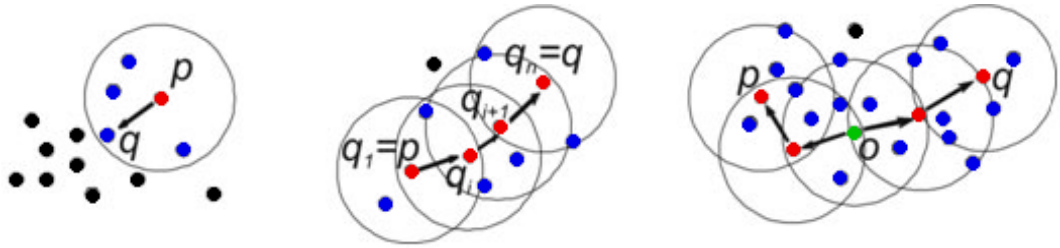


Figure 3.6: Examples of *directly density-reachable* (left), *density-reachable* (center) and *density-connected* (right) in DBSCAN. Assume that the minimum number of points required to form a cluster is 3 i.e. $minPts = 3$. The dots represent the points to be clustered, and the black circles define the area of radius ε around the points in red, the arrows denote the relation of *direct density-reachability*. In the figure on left, point p is the *core point*, while q is *directly density-reachable* from p, similar to all the other blue points. In the figure in center, point q is *density-reachable* from the point p. In figure on right, point q is *density-connected* to p whereas o (the point in green) is a point such that both p and q are *density reachable* from o.

ping through this list. A voxel is defined by the position of its center and the length of its side. The centers of the detected dynamic voxels define the set of points for which we have to perform clustering. We start with an arbitrary dynamic voxel in the list which is not yet assigned to any cluster. The ε-neighborhood of this voxel is retrieved and if it contains at least as many dynamic voxels as the $minPts$, we initialize a new cluster. In other case, we label that voxel as noise. However, this voxel might later be discovered in a sufficiently sized ε-environment of a different voxel and hence be included in a cluster. If a voxel is a dense part of a cluster then its ε-neighborhood is also included in that cluster. Thus the search continues with all those voxels recursively. This process continues until the complete cluster is discovered. Afterwards, a new unvisited voxel is retrieved from the list and processed to start a new cluster or to be marked as noise. The details of our method for clustering of dynamic voxels are given in Algorithm 2 and 3.

At the end, all the dynamic voxels are either assigned to a cluster or marked as noise. The noise voxels are discarded which results in getting rid of the false alarms and spurious elements in the environment which were wrongly identified as sparse dynamic object voxels. The remaining dynamic voxels in the list have a higher possibility of corresponding to moving objects which is further improved in the next section of classification. The clusters corresponding to those voxels are maintained as the dynamic object hypotheses which are forwarded to the classification system to assign the appropriate classes.

Algorithm 2 DBSCAN for clustering dynamic voxels

Input: List of dynamic voxels D, ε, $minPts$
Output: Cluster id assigned to each voxel (or NOISE)

mark all voxels as UNVISITED
$ClusterId \leftarrow 1$
for all voxels v_i in D **do**
    if v_i is UNVISITED **then**
        mark v_i as VISITED
        if `Expand_Cluster`$(D, v_i, ClusterId, \varepsilon, minPts)$ **then**
            $ClusterId \leftarrow ClusterId + 1$
        end if
    end if
end for

Figure 3.7, 3.8 and 3.9 are an illustration to explain the two steps described above for the detection of moving objects in the same scenario at two different instances. Figure 3.7 shows the occupancy grid map for the point cloud corresponding to a situation where the vehicle is moving on the road having dynamic objects around it. The ground voxels are shown in blue while the other occupied voxels are in grey. This local static map of the environment is incrementally constructed from the point clouds after each scan. Figure 3.8a and 3.9a show the voxels for dynamic object hypothesis detected on the base of the inconsistencies between the scans displayed in red. Here, you can see the amount of noise detected as the sparse red points. Figure 3.8b and 3.9b represent the result of clustering of the dynamic voxels. Applying the DBSCAN with an appropriate value of the minimum acceptable density for defining a cluster has eliminated quite a large number of wrongly detected dynamic voxels. However, there remain some clusters that do no belong to the dynamic objects which will be handled in the next step. Figure 3.10 provides a closer look at the detection results shown in Fig. 3.9a to show the different types of objects detected in the surrounding.

Algorithm 3 *Expand_Cluster*

Input: List of dynamic voxels D, $v_i \in D$, $ClusterId$, ε, $minPts$
Output: *true* if a new cluster is created, $false$ otherwise

$SeedVoxels \leftarrow Retrieve\_Neighbors(v_i, \varepsilon)$
if then$|SeedVoxels| < minPts$ ▷ Insufficient voxels in neighborhood
    mark v_i as NOISE
    return false
else ▷ v_i is a core point of $ClusterId$
    for all voxels v_j in $SeedVoxels$ **do**
        assign v_j to $ClusterId$
    end for
    delete v_i from $SeedVoxels$
    ▷ Identify density-reachable voxels for v_i
    for all v_j in $SeedVoxels$ **do**
        mark v_j as VISITED
        $NeighbourVoxels(v_j) \leftarrow Retrieve\_Neighbors(v_j, \varepsilon)$
        if $|NeighbourVoxels(v_j)| \geq minPts$ **then** ▷ v_j is also a core point
            for all v_k in $NeighbourVoxels(v_j$ **do**
                if v_k is UNVISITED or NOISE **then**
                    if v_k is UNVISITED **then**
                        add v_k to $SeedVoxels$
                    end if
                    assign v_k to $ClusterId$
                end if
            end for
        end if
        delete v_j from $SeedVoxels$
    end for
    return true
end if

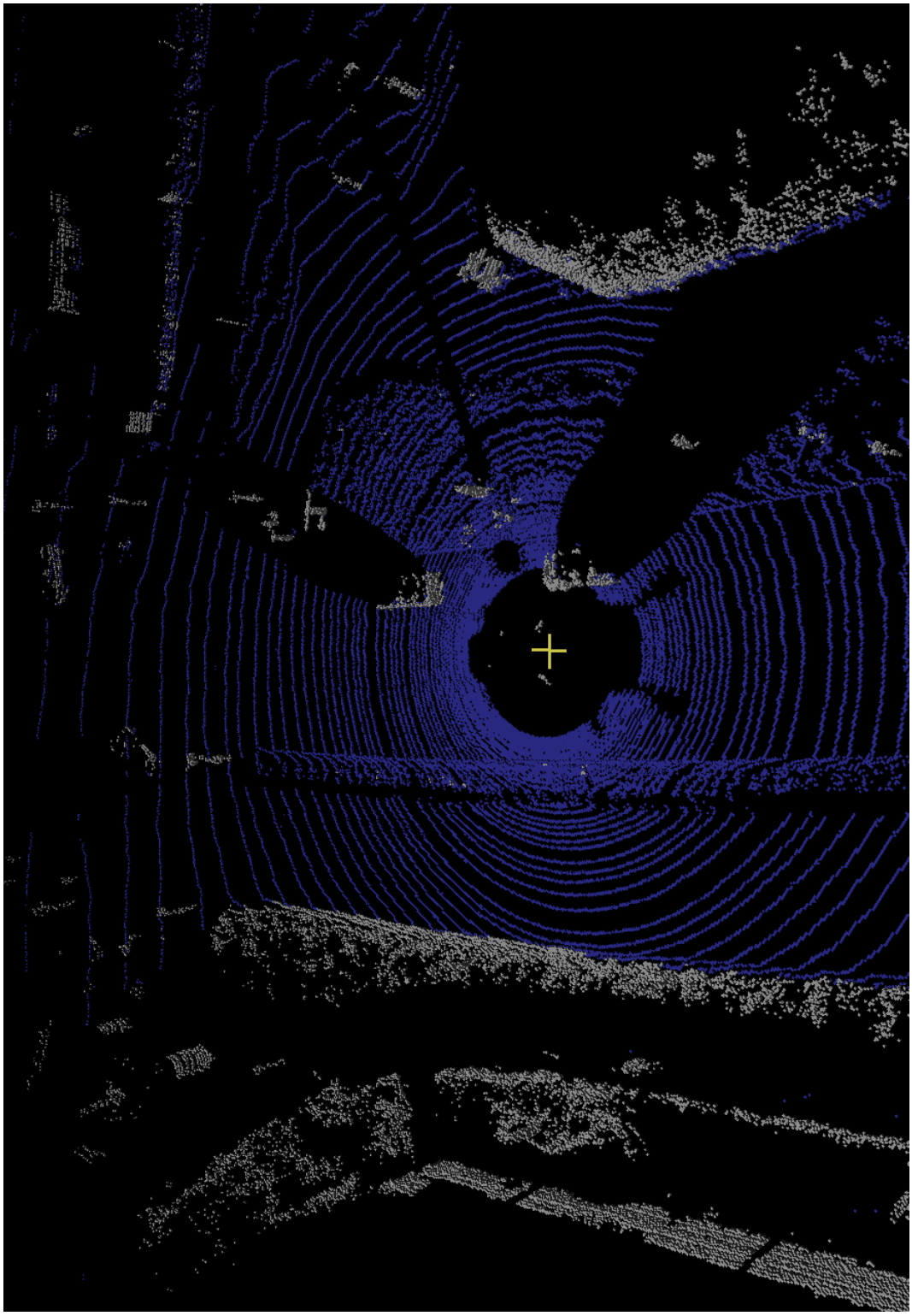
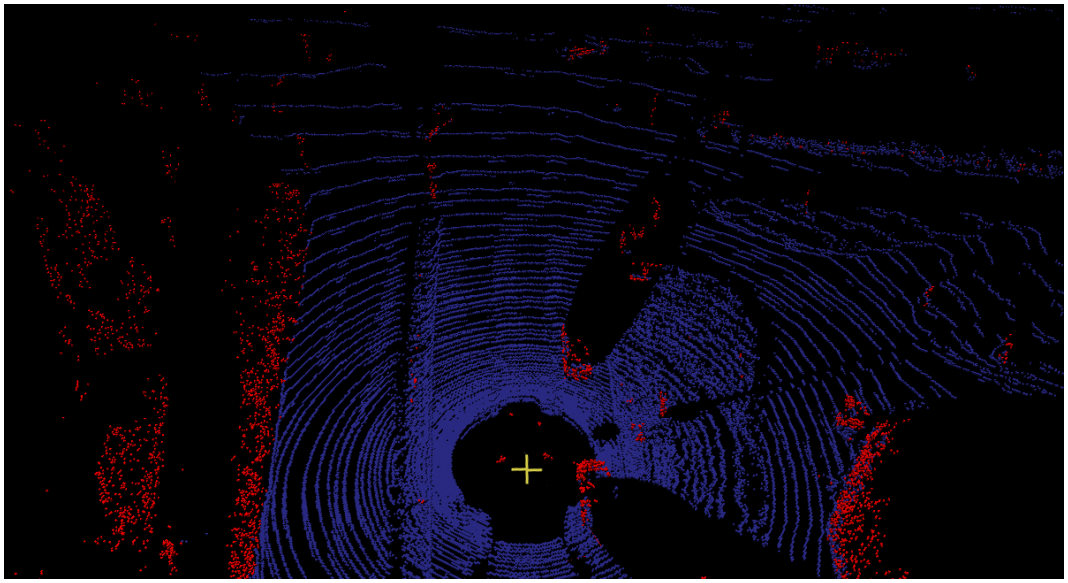
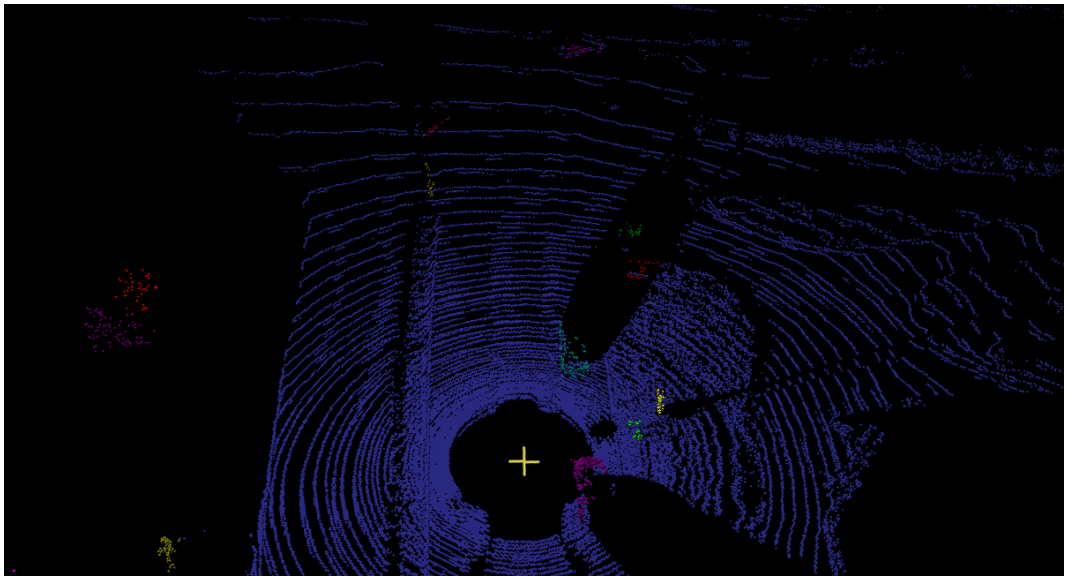


Figure 3.7: Occupancy grid map corresponding to a scan obtained by the vehicle in an urban road scene with multiple moving objects. The static occupied voxels are shown in grey and ground voxels in blue. The ego-vehicle is represented by the '+' symbol in mustard color.

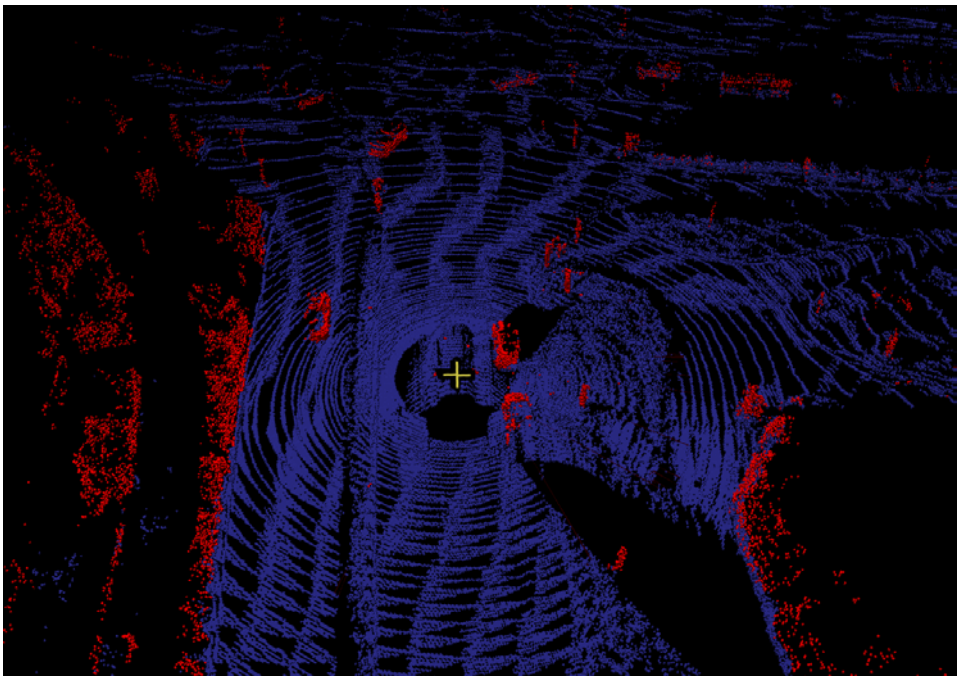


(a) Results of detection after two scans: Identification of dynamic voxels.

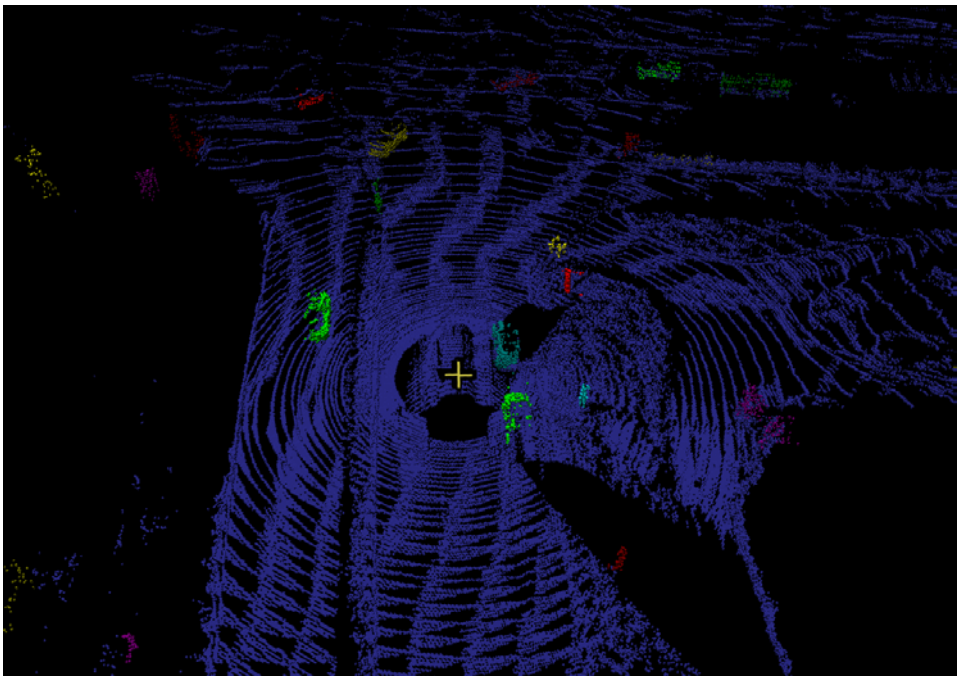


(b) Results of detection after two scans: Clustering of dynamic voxels.

Figure 3.8: An illustration of detection of moving objects after two scans. All the dynamic voxels detected from inconsistencies between the scans are shown in red (3.8a). After clustering, different dynamic object hypotheses are shown in different colors (3.8b). The static occupied voxels are hidden for better visibility of the dynamic voxels.



(a) Results of detection after ten scans: Identification of dynamic voxels.



(b) Results of detection after ten scans: Clustering of dynamic voxels.

Figure 3.9: An illustration of detection of moving objects after ten scans. In Fig. 3.9a, detected dynamic voxels are accumulated over ten previous scans while in Fig. 3.9b, many of those voxels are ignored as noise at each time step when clustering is performed. Static occupied voxels are hidden for better visibility of the dynamic voxels.

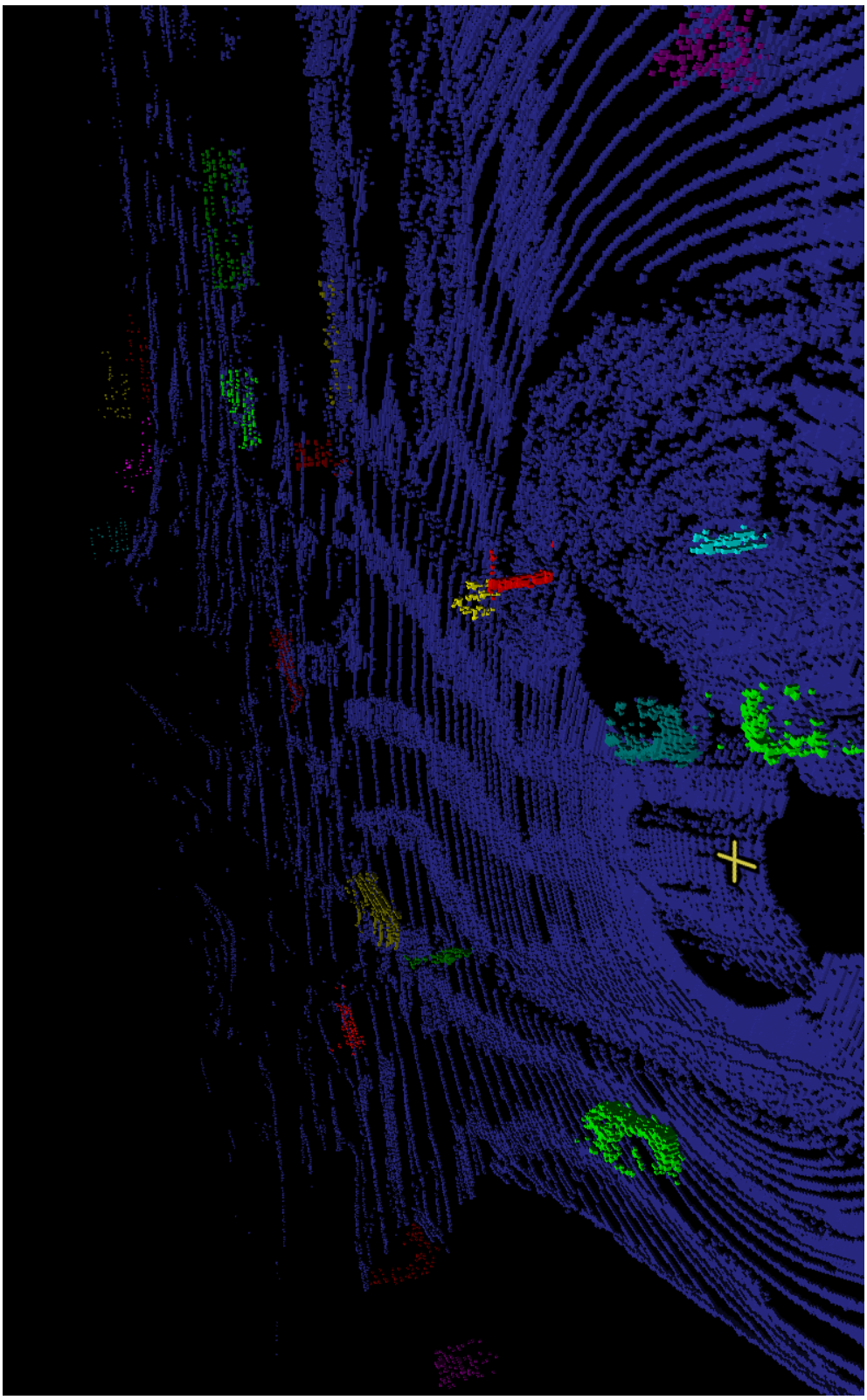


Figure 3.10: The results of detection of moving objects provided from Fig. 3.9: a closer view.

3.6 Classification of Moving Objects

After detecting all possible hypotheses of the dynamic objects, we address the problem of classification to identify different kinds of objects in order to infer about their characteristics such as motion dynamics. In general, there exist numerous kinds of objects in traffic scenes, however, only a few of those are considered to be of interest in the context of intelligent vehicles. These are the objects which occur most frequently in these scenarios. We consider four such classes namely *pedestrian*, *bike*, *car* and *bus*. Generally, each of these objects vary in their sizes and structures. Thus the simplest approach, which we employed as a naive classification method in (Azim and Aycard, 2012), is to use the size of the object to identify its class. The idea was to use a cluster of dynamic voxels provided by the detection step and fix a 3D bounding box to it. The classification was done based on the properties of this box such as the ratio between its length, width and height. A benefit of this method is that it is very simple and time efficient as compared to any complex classification technique, however, it requires the object to be visible in such a way that its size may be correctly estimated. This is usually not the case with the objects detected by the range sensors. For example, over a period of time, an object might be observed from the front, side or the back and thus its visible size may differ largely from the actual size. Therefore, we require a more sophisticated and robust method for classification which can deal with these situations.

In this section, we present a novel layered approach for 3D classification of moving objects based on supervised learning. A description of our methods for preprocessing of the data, feature extraction, training and classification is given in the following.

3.6.1 Approach Overview

Figure 3.11 provides an overview of our classification and learning method. The input data are the clusters of 3D voxels corresponding to the object hypotheses detected in the previous step. The main idea is to divide the 3D object into different height levels, termed as *layers*. For example, the layered representation of a car is illustrated in Fig. 3.12 where different colors represent different layers. These layers serve as the 2D slices of the object. In the next step, simple features are extracted from each layer. In generating the training dataset, where the class of the object

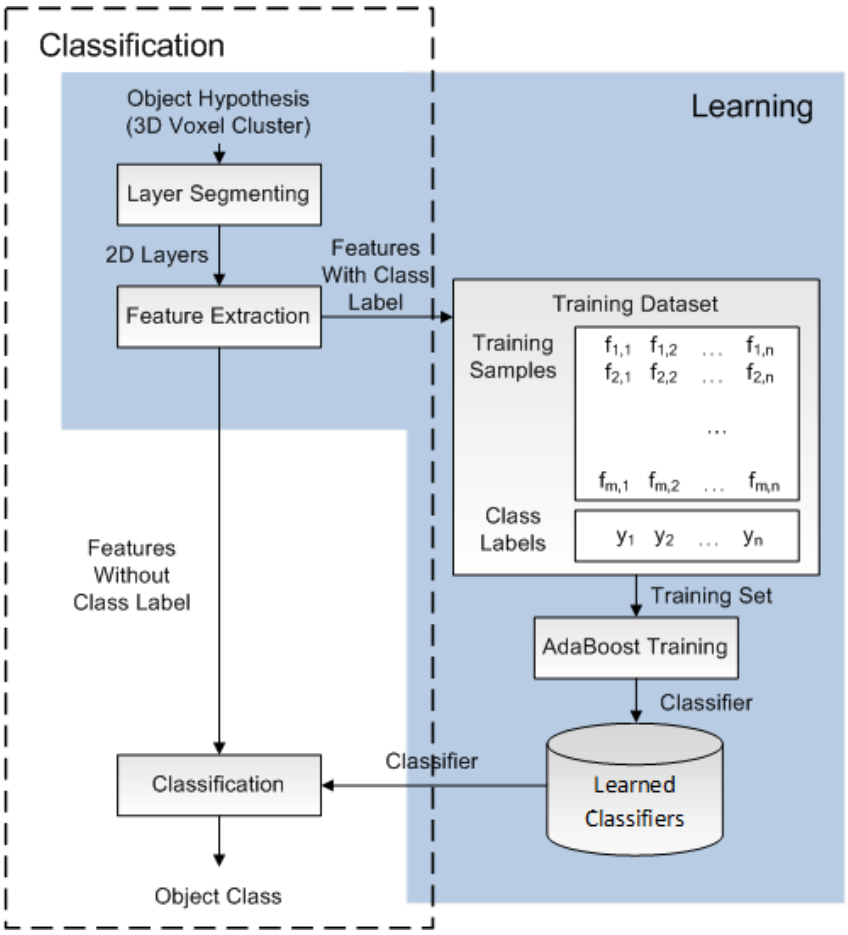


Figure 3.11: Classification Algorithm: The shaded box represents learning and the dashed box represents the classification.

is known or hand-labeled, the extracted features are added as new sample to the training set along with their class label. Otherwise, these features are used as input for the classification process where trained classifiers are used to infer about the class of the object. In the training step, different classifiers are trained and stored using a supervised approach based on AdaBoost, a known method to train a strong classifier from the extracted features. These learned classifiers are applied to the features of the detected objects and the appropriate classes are assigned to them. Different stages of our association algorithm are detailed in the following subsections.

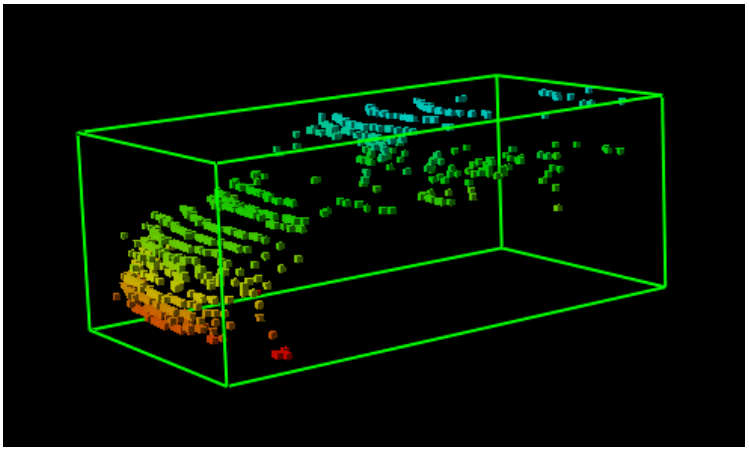


Figure 3.12: The layered representation of a car. Each layer is represented by a different color.

3.6.2 Object Segmentation in Layers

As outcome of dynamic object detection, we have a cluster of 3D voxels corresponding to each object hypothesis. As described earlier, each voxel of the occupancy grid is defined by its center and the length of its side. Thus we can consider each cluster of dynamic voxels as a sub-sampled 3D point cloud formed by the centers of the voxels. Further, this 3D point cloud can be considered as a collection of 2D points arranged in *layers* or *slices*.

As a first step in our approach for classification, we divide each cluster into a set of 2D layers at different heights. We aim to characterize the 3D shape of the object by computing features in each of its 2D layers. We select this subdivision method as the 2D layers at different heights can describe the local shape properties of the objects. Moreover, it provides a more flexible class representation where occluded objects can also be identified. We consider an object X as consisting of the layers $L_i = \{\mathbf{x}_j^i\}$ where $\mathbf{x}_j^i$ are the voxels in the i-th layer and each voxel in our map is represented by its center, (x, y, z). For a specific layer L_i, height of the layer is computed by averaging the heights of all the voxels contained in that layer. As a result, the voxels in the layer are represented by their x and y components only, $\mathbf{x}_j^i = (x_j^i, y_j^i)$. A layered representation of a pedestrian is shown in Fig. 3.13 along with an illustration of the voxels in a layer at a specific height.

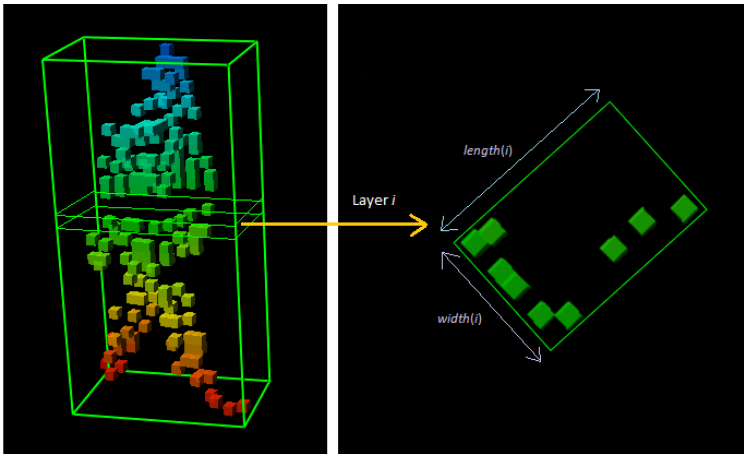


Figure 3.13: The layered representation of a pedestrian (left) and the illustration of a specific layer (right).

3.6.3 Feature Extraction

After segmenting the individual object into layers, we compute the features for each layer which describe its appearance. A feature f is defined as a function $f : L_i \rightarrow \mathbb{R}$ which takes a layer L_i containing n_i voxels and returns a real number. We consider two types of features which are often used in the context of classification with 2D range sensors: *geometric* and *statistical* (Arras *et al.*, 2007). Most of the features we compute are *simple* single-valued features instead of the high-level features required for 3D classification.

For each layer L_i, we first compute a minimum, axis-aligned 2D bounding box and then compute the following features:

- *Number of voxels*: This feature represents the cardinality of L_i defined as $n_i = \mid L_i \mid$.
- *Height of the layer*: It is defined by the average height of all the voxels in that layer: $h_i = \frac{1}{n_i}\Sigma_j z_i^j$.
- *Width*: This feature measures the length of the shorter side of the 2D bounding box corresponding to the voxels in that layer.
- *Length*: This feature measures the length of the longer side of the bounding box.
- *Area*: This features describes the area A_i of the bounding box.

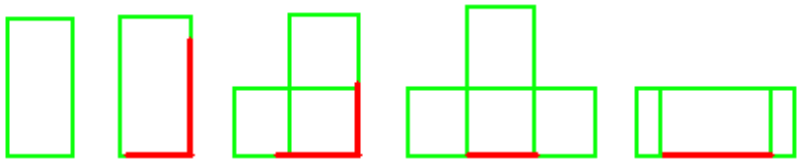


Figure 3.14: Illustration of *L*-shaped and *I*-shaped layers with object box models used by (Vu, 2009).

- *Density*: It is the density of the voxels with respect to the area of the 2D bounding box: $\rho_i = \frac{n_i}{A_i}$

- *Standard deviation*: This feature represents the compactness of the voxels in the layer. The smaller is the standard deviation σ, the more compact is the layer. It is computed by:

$$\sigma = \sqrt{\frac{1}{n_i}\Sigma_j \parallel \mathbf{x}_j - \overline{x} \parallel^2} \tag{3.6.1}$$

  where $\overline{x}$ is the center of gravity of the layer L_i.

- *Mean deviation from median*: This feature is often used for shape analysis as it represents the data compactness more robustly than σ and it is less sensitive to the outliers. The median $\tilde{\mathbf{x}}$ is the numerical value which separates the higher half of a distribution from its lower half. For a 2D distribution, it can be computed by $\tilde{\mathbf{x}} = (\tilde{x}, \tilde{y})$. The mean deviation from median is then computer by:

$$\varsigma = \frac{1}{n}\Sigma_j \parallel \mathbf{x}_j - \tilde{\mathbf{x}} \parallel \tag{3.6.2}$$

- *Shape*: We have defined this feature following the 2D model-based classification method of (Vu, 2009) (Fig. 3.14). The idea is that at a specific time instant, maximum two sides of an object can be observed by the range sensor. Considering the sides of 2D bounding box, the layer is classified as *L*-shaped if both its sides are longer than a threshold, *I*-shaped if one side is longer than threshold and *mass-point* otherwise. We represent these three shape hypotheses as numerical values of 2, 1 and 0 for the *L*-shape, *I*-shape and *mass-point* respectively.

This collection of features constitutes a profile of each layer and all the layers together define a profile of the object.

3.6.4 Supervised Learning of Classifier

Once the features are extracted, they can be used to classify the objects using a classifier. Our approach to define the classifier is based on supervised learning. For the learning process, we used the labeled object samples to generate a training dataset and trained the classifiers offline. The process is shown in Fig. 3.11.

As a first step, we generate the training dataset by extracting the features from the positive as well as negative examples of a class. For instance, in case of the *car* class, clusters corresponding to a car are considered as positive examples while those corresponding to all other objects such as pedestrians, bikes, bus, trees, fences, walls and unknown objects as negative examples. The extracted features are stored as a training vector. These training vectors and their labels are then used to create a classifier. Our approach for supervised training and classification is based on a *boosting* method.

Boosting is a general method which creates an accurate *strong* classifier by linearly combining the performance of many *weak* classifiers. The weak classifiers, also known as *base* classifiers, are required to be slightly correlated to the true classification. In other words, the accuracy of each weak classifier only needs to be better than a random guess thus they can be very simple and computationally efficient.

We use the popular method for boosting known as AdaBoost which is detailed in Algorithm 4. It takes a set of labeled training examples $(x_1, y_1), ..., (x_N, y_N)$ as input where $x_i \in X$ is an example and $y_i \in Y$ is the corresponding class label. The original AdaBoost is a binary classifier which means that $Y = \{+1, -1\}$ where $+1$ corresponds to a positive example and -1 to a negative example. All the training examples are initially assigned the weights according to a distribution D. Subsequently, in a series of iterations $1, ..., T$, AdaBoost repeatedly selects a weak classifier $h_t(x)$ using the weighted distribution over the examples. It takes the best weak classifier which has a small classification error at each step and adds it to the final set of selected classifiers. It modifies the weight distribution D_t on each iteration to assign more importance to the examples which were incorrectly classified by the previously selected weak classifier. The final strong classifier H is a weighted majority sum of the T weak classifiers selected in each iteration. Finally, the good weak classifiers are assigned larger weights as compared to the poor ones.

We use the AdaBoost algorithm presented in (Viola and Jones, 2001) in which each

Algorithm 4 AdaBoost algorithm for classifier learning

Input: A set of labeled training examples $(x_1, y_1), ..., (x_n, y_n)$ where x_i is an example and y_i is the class label which is equal to $+1$ or -1 for positive and negative examples respectively.

Initialize weights $w_{1,i} \leftarrow \frac{1}{2m}, \frac{1}{2l}$ for the positive and negative examples of the object respectively where m and l are the total number of positive and negative examples in the training set respectively.

for $t \leftarrow 1$ to T **do**

  Normalize the weights: $w_{t,i} \leftarrow \frac{w_{t,i}}{\Sigma_{j=1}^{n} w_{t,j}}$

  for each feature f_j **do**

    Train a weak classifier h_j which uses a single feature.

    Determine the error ϵ_j of h_j with respect to the weights $w_{t,1}, ..., w_{t,n}$:

$$\epsilon_j \leftarrow \Sigma_i w_{t,i} |h_j(x_i) - y_i|$$

  end for

  Select the best classifier h_t for this iteration with the lowest error ϵ_t.

  Update the weights for next iteration:

$$w_{t+1,i} \leftarrow w_{t,i} \gamma_t^{1-e_i}$$

  where $e_i = 0$ if the example x_i is correctly classified by h_t and 1 otherwise.

  Also, $\gamma_t = \frac{\epsilon_t}{1-\epsilon_t}$.

end for

The final strong classifier is a linear combination of the T selected classifiers given by:

$$H(x) = \begin{cases} 1 & \text{if} \quad \Sigma_{t=1}^{T} \alpha_t h_t(x) \leq \frac{1}{2} \Sigma_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where $\alpha_t = \log(\frac{1}{\gamma_t})$.

single-valued feature computed from the layers is used as a weak classifier. The weak classifiers are defined as:

$$h_j(x) = \begin{cases} +1 & \text{if } p_j f_j(x) < p_j \theta_j \\ -1 & \text{otherwise} \end{cases} \tag{3.6.3}$$

where $p_j \in \{+1, -1\}$ and θ_j is a threshold. In each iteration, AdaBoost learns the values of p_j and θ_j in order to minimize the classification error in the weighted training dataset. The resulting strong classifier $H(x)$ is obtained by a weighted combination of the selected weak classifiers. These selected classifiers actually correspond to the best (most highly voted) possible features for classification. As the features in the feature vector are arranged according to the layers from which they were extracted, we can get information about the best features corresponding to each layer as well as the layers which contribute the most towards classification by providing most of the best features. For example, in the case of pedestrians, we observe that the layers corresponding to the shoulders provide more of the selected features as compared to those corresponding to the feet or head. This can help to extend our approach by extracting a mix of different features from different layers.

The output of this algorithm is either $+1$ (for positive example) or -1 (for negative example) as it is a binary classifier. In our case, we need to categorize the objects into four known classes. For this, we use the two-class algorithm described above and extend it to multi-class by employing *one-against-all* approach. In this approach, we train a separate binary classifier for each class of the objects. Thus we obtain four classifiers corresponding to each class which distinguish between the samples of one class and the samples of all other classes. In the following, we explain our method for inferring about the class of a detected object using these classifiers.

3.6.5 Sequential Multi-class Classification

After training the individual classifier for each object class, we use them in our system to infer the class of the detected objects. Our approach is based on a sequential implementation of AdaBoost classifiers. This method performs the multi-class classification by arranging several binary classifiers sequentially. In our case with four possible classes $Y = \{pedestrian, bike, car, bus\}$, we construct a decision list with four binary classifiers which are trained individually, as described in the previous section. Each of these classifiers provides a binary hypothesis $H_k(x)$ about the classification of the object, with $k = 1, ..., 4$. The detected object is passed to each of

Algorithm 5 Sequential classification of dynamic objects

Input: List of dynamic object hypotheses O_t, set of class labels $Y = \{y_k\}$, set of binary classifiers $C = \{c_k\}$ with $k = 1, ..., 4$
Output: Class labels assigned to each object hypothesis

```
for all object hypotheses o_i in the list do
    for k ← 1, ..., 4 do
        if h_k(o^i) = 1 then
            assign label y_k to o_i
            break
        end if
    end for
    if No label is assigned to o_i then
        assign label UNKNOWN to o_i
    end if
end for
```

the classifiers in order. If any of the classifiers returns a positive result, such that $H_k(x) = 1$ then the label of that classifier is assigned to the object. Otherwise, it is passed to the next classifier in the list. If none of the classifiers returns a positive result then the object is considered to be an *unknown* object. Algorithm 5 gives the general procedure used for classifying an object after detection.

The order in which the classifiers are arranged is of real importance in this implementation and it can affect the performance of the overall classification system. The recommended approach is to arrange the classifiers according to their estimated error. The classifiers with the lower error (higher accuracy) should be arranged before the ones with higher error in order to keep the overall error low. Another check that we use for the selection of classifiers is the size of the cluster. For example, we remark that if the bounding box of a cluster is larger than a specific size than it is not a good candidate for a pedestrian. As a result, it is not tested with the pedestrian classifier and it is directly passed to the next class, if any.

After performing this step, all of the detected objects are either assigned a valid class label or marked as unknown. These objects are then tracked in order to estimate their position sequentially.

3.7 Tracking of Moving Objects

The tracking of multiple moving objects is a complex problem which, as described earlier, is generally divided into two parts: *filtering* and *data association*. Filtering is the sequential estimation of the state of a dynamic object. It is usually performed using Bayesian filters which require a specific motion model for tracked objects to predict their positions in environment. After predicting the positions of existing tracks, next we perform data association to assign the observations to the existing tracks. In the following, we explain the method we adopted for filtering and data association for tracking multiple objects.

3.7.1 Object Representation and Dynamic Models

The detection and classification steps presented in the previous sections produce dynamic object hypotheses along with their class labels. These hypotheses can be further exploited to infer different geometric parameters of the objects. We compute the centroid of each object cluster and its principal orientation to represent each detected dynamic object in the environment. Generally, we represent the models of the objects parametrized by $M = \{c, x, y, z, \theta\}$ where c is the class of the object, (x, y, z) is the centroid of the cluster corresponding to the object and θ is its principal orientation. The other attributes such as *velocity* of the object can be derived from these parameters over time.

In order to track these objects, we need to estimate their state sequentially. The class information of the objects helps us in selecting the appropriate motion parameters for the state estimation and consequently selecting the appropriate motion model. In this work, we use the *constant velocity model* for *car*, *bus* and *bicycle* assuming that their velocities remains constant for the duration of each time interval from $t-1$ to t. However, this assumption is not made in the case of pedestrians as their motion may change abruptly at any time. Therefore we use the *Brownian motion model* for tracking of *pedestrians*. The state of the object is updated with the Kalman filter.

The next subsection defines our method for data association followed by the track management for multiple object tracking.

3.7.2 Viterbi Data Association

The Viterbi algorithm (Forney Jr, 1973) is a recursive algorithm that provides a solution to the discrete linear optimization problem. It is used for finding the most likely sequence of hidden states – called the Viterbi path – that results in a sequence of observed events, especially in the context of hidden Markov models.

The implementation of the Viterbi algorithm is based on a trellis. A trellis diagram is a type of directed graph (N, A) that consists of a set of nodes N and a set of directed arcs A. The nodes n_i^k are partitioned into ordered sets with the k-th set being denoted as $N(k)$, where k represents stages in the trellis $(k = 1, 2, ...T)$. The number of nodes at each stage is denoted by n_k. An important assumption underlying the use of the trellis diagram is that the state can be modeled by a Markov process. Hence, in dealing with the trellis diagram, the set of directed arcs A is a collection of ordered pairs $\{n_i(k-1), n_i(k)\}$ where $k = 2, ..., T$. A path P is a collection of directed arcs that connects an element at stage 1 to an element at stage T. Each directed arc is associated with a metric or a distance label $a_{ij}(k)$. A path metric is defined as the sum of the metrics of all the arcs contained in the path P as:

$$d(P) = \sum_{k=2}^{T} a_{ij}(k); \{n_i(k-1), n_i(k)\} \in P \tag{3.7.1}$$

where $d(P)$ is the total metric of the path P.

Starting from the initial stage, the Viterbi algorithm successively labels all the nodes in the trellis until the final stage is reached. The optimal state sequence in the trellis is then retrieved by backtracking, starting from the node in the final stage with the smallest metric. An illustration of the trellis diagram for Viterbi algorithm is shown in Fig. 3.15 where the Viterbi path is represented by the solid arrows.

Single Object Tracking

In order to use Viterbi algorithm to resolve the data association problem for single target tracking, we assume that each node in the trellis represents an observation (Gad and Farooq, 2003). The collection of measurements at k-th scan Z_k corresponds to the set of nodes at k-th stage of the trellis. Arcs of the trellis are defined as the metric d on the basis of which we can associate the observation to the corresponding track.

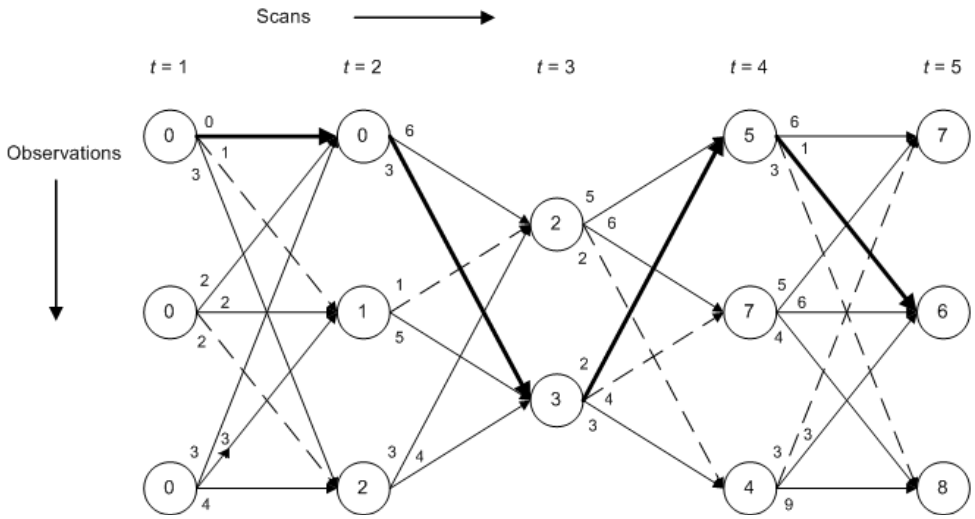


Figure 3.15: Trellis diagram of Viterbi Algorithm.

Using the notations defined above, we can summarize the Viterbi algorithm for single object tracking as follows:

Step 1 – Initialization: Assign a value of zero to the label of each node in first stage:

$$d_i(1) = 0, \quad 0 \leq i \leq n_1 \tag{3.7.2}$$

$$\psi_i(1) = 0, \quad 0 \leq i \leq n_1 \tag{3.7.3}$$

Step 2 – Recursion: Repeat the following steps for each stage k, where $k = 2, .., T$:

- For each node $i = 0, .., n_{k-1}$ (at stage $k-1$), calculate the predicted position using Kalman filter:

$$\widehat{x}_i(k/k-1) = \phi \widehat{x}_i(k-1/k-1) \tag{3.7.4}$$

$$P_i(k/k-1) = \phi P_i(k-1/k-1)\phi^T + Q(k-1) \tag{3.7.5}$$

$$S_i(k/k-1) = HP_i(k-1/k-1)H^T + R(k-1) \tag{3.7.6}$$

- For each node $j = 0, .., n_k$ (at stage k), calculate the distance metric $a_{ij}(k)$ of the arc joining nodes $n_i(k-1)$ and $n_j(k)$.

- Assign node $n_j(k)$ with the smallest label as follows:

$$i^* = arg\{\min_{0 \leq i \leq n_{k-1}} \{d_i(k-1) + a_{ij}(k)\}\} \tag{3.7.7}$$

$$Score_j(k) = d_{i^*}(k-1) + a_{ij}(k) \tag{3.7.8}$$

$$\psi_j(k) = i^* \tag{3.7.9}$$

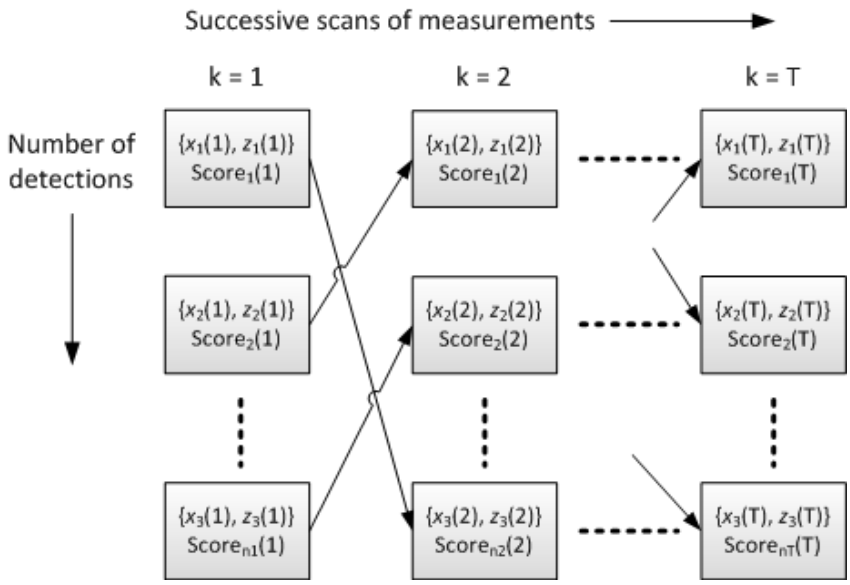


Figure 3.16: Trellis diagram of Viterbi data association for single target tracking.

- Update the target state at each node in stage k:

$$K_j(k) = P_{i^*}(k/k-1)H^T[HP_{i^*}(k/k-1)H^T + R(k)]^{-1} \tag{3.7.10}$$

$$P_{i^*}(k/k) = [I - K_j(k)H]P_{i^*}(k/k-1) \tag{3.7.11}$$

$$\widehat{x}_j(k/k) = \widehat{x}_{i^*}(k/k-1) + K_j(k)[z_j(k) - H\widehat{x}_{i^*}(k/k-1)] \tag{3.7.12}$$

Step 3 – Final selection: Determine the node with the minimum score in the final stage.

$$i^* = arg\{\min_{0 \le i \le n_T} \{d_i(k)\}\} \tag{3.7.13}$$

$$\widehat{x}(T) = \widehat{x}(T/T) \tag{3.7.14}$$

Step 4 – Backtracking: Recover the measurement sequence that terminates with the minimum node score in the final stage. For each stage k:

$$i^*(k-1) = \psi_{i^*(k)}, \quad k = T, T-1, ..., 2 \tag{3.7.15}$$

The notation used in the above algorithm are as follows:

- $d_i(k)$ is the metric of node $n_i(k)$
- $\psi_j(k)$ is the predecessor function of the node $n_j(k)$

- i, j are indices of elements in $N(k)$
- $d^*(T)$ is the metric of the shortest path in the trellis diagram.

Each of the nodes in our trellis contains the information about the observations. For the first scan, we obtain the first set of observations $z(1)$. We create a node for each of these observations and assign a value of zero to its accumulated distance metric. For all the later scans (k), we use Kalman filter to calculate the predicted position for each of the observations in the previous state $(k-1)$ and calculate the Euclidean distance between this predicted position and the new observations obtained in the current scan. This is the distance metric for each of the arcs in the trellis. We find the previous observation node $n_i(k-1)$ for which this distance metric is minimum and add this metric to the accumulated distance label of the current node $n(k)$. If the current scan is the final scan $(k = T)$, we find the minimum of the distance labels in this stage. The final state of the target is the observation associated with the minimal node. After obtaining this final state, we start to backtrack through the trellis to recover the measurement sequence that ends at this state. This sequence is the Viterbi path or the Viterbi track for the observed object.

Multiple Object Tracking

In order to implement Viterbi algorithm for multiple target tracking, we create a separate instance of Viterbi data association for each of the objects. Thus there is a separate Viterbi trellis for each object. At each time step, we use Kalman filter to estimate and update the positions of the tracks. In each scan, when we receive a new set of observations, we first perform the gating to find the likely observations to be associated with the existing tracks. We have used Mahalanobis distance between the predicted positions of the track and the newly received observation as a measure for gating. Moreover, as we perform classification of moving objects before tracking, we use the classification information as an additional criterion to find the likely associations between the tracks and the observations. For all existing tracks, we have the class labels corresponding to them. When we perform gating, we consider only those observations which have the same class label as the corresponding track with which the association is to be made.

After applying gating, the observations that fall outside the prescribed gate can possibly be the potential candidates for new tracks or the false alarms. We perform

the data association for each track with all the observations within the prescribed gates by using Eq. 3.7.13 and 3.7.14. The observation that gets associated to any of the tracks is marked *associated* and the remaining observations that are not associated to any of the existing tracks are marked *un-associated*. These *un-associated* observations might correspond to a new object or a false alarm. All such observations are handled in the track maintenance described in section 3.7.3. For the *associated* tracks, we use the observation node with the minimum metric to update the position of the track. An important observation is that our approach inherently handles the temporary occlusions as the paths in different trellises can traverse through the same nodes.

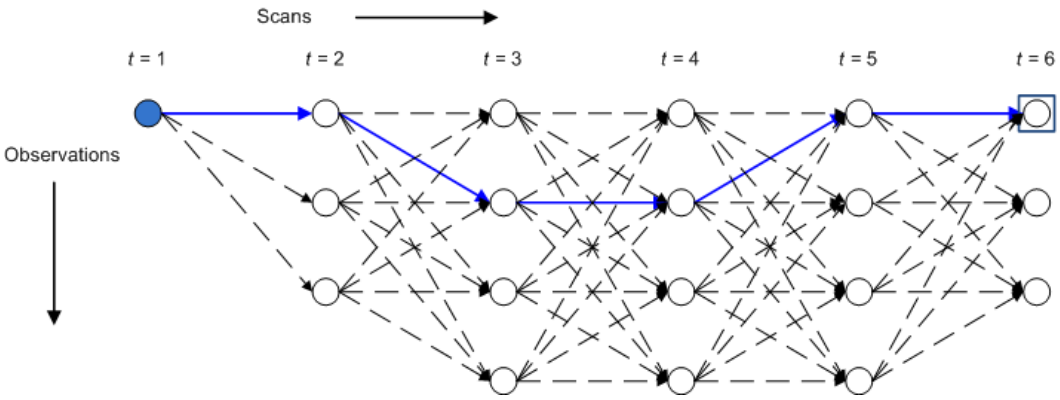
Figure 3.17, 3.18 and 3.19 are an illustration of the VDA for multiple object tracking. The example consists of four tracks which were initialized at $t = 1, 2, 3$ and 4 respectively. Figure 3.17, and 3.18 show the separate trellises maintained by each of the four tracks. The Viterbi path for each track is shown by the solid arrows in different colors corresponding to the tracks. The colored nodes represent the start of a track while the nodes surrounded by the squares represent the termination of a track. Figure 3.19 shows the combined trellis for the four tracks.

3.7.3 Track Maintenance

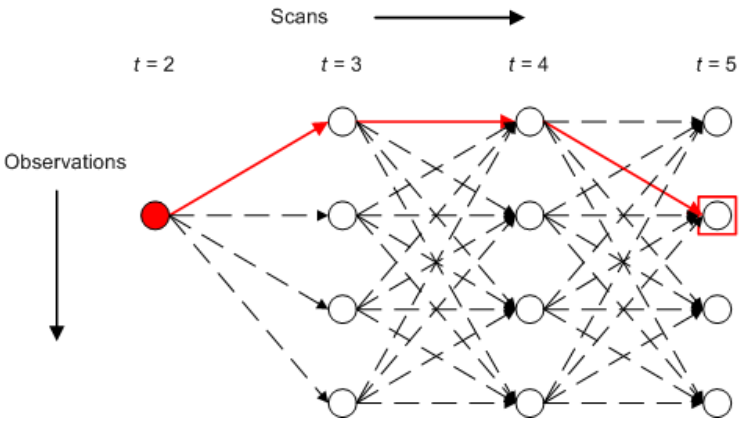
As the number of observations may not always correspond directly to the number of existing tracks in a dynamic environment, we have implemented the mechanism for the creation, suppression and maintenance of the tracks. We have also dealt with the classical problem of split and merge for tracking in cluttered environments.

Track Continuation

We illustrate the Viterbi data association technique for multiple objects tracking using the following example. In the start, we received two observations for moving objects and initialized two instances of VDA, VDA_1 and VDA_2 corresponding to each of these observations. In the next scan, we received 2 observations again. We made the associations of both the tracks VDA_1 and VDA_2 with each of these observations. We first associated VDA_1 to each of the observations and computed corresponding distance metrics for each association to find the best association at this time step. Then we associated VDA_2 to each observation and found the best

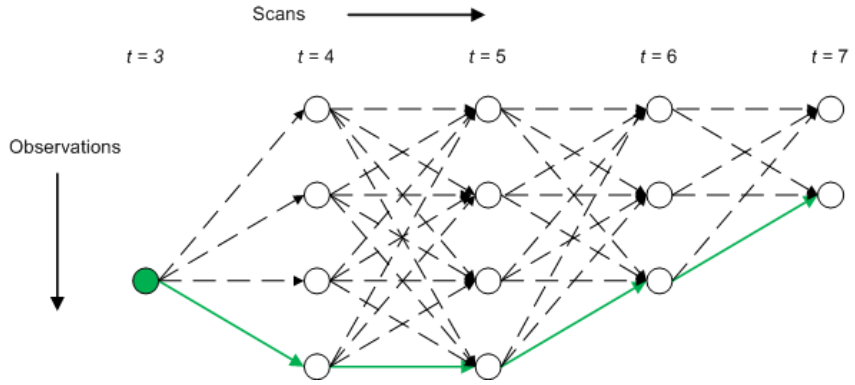


(a) Track 1.

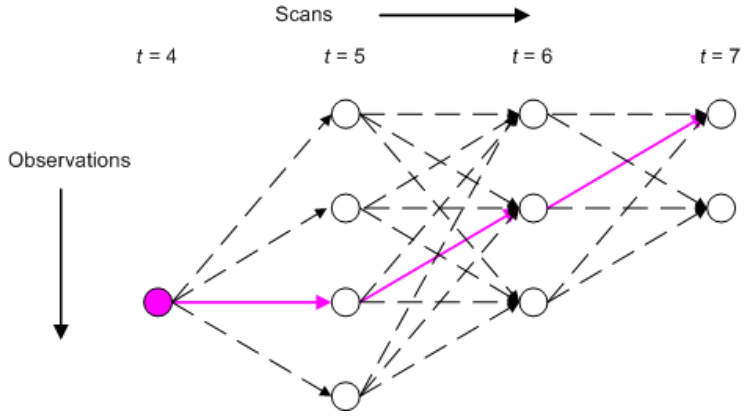


(b) Track 2.

Figure 3.17: Multiple object tracking with Viterbi data association: Trellis for track 1 and 2.



(a) Track 3.



(b) Track 4.

Figure 3.18: Multiple object tracking with Viterbi data association: Trellis for track 3 and 4.

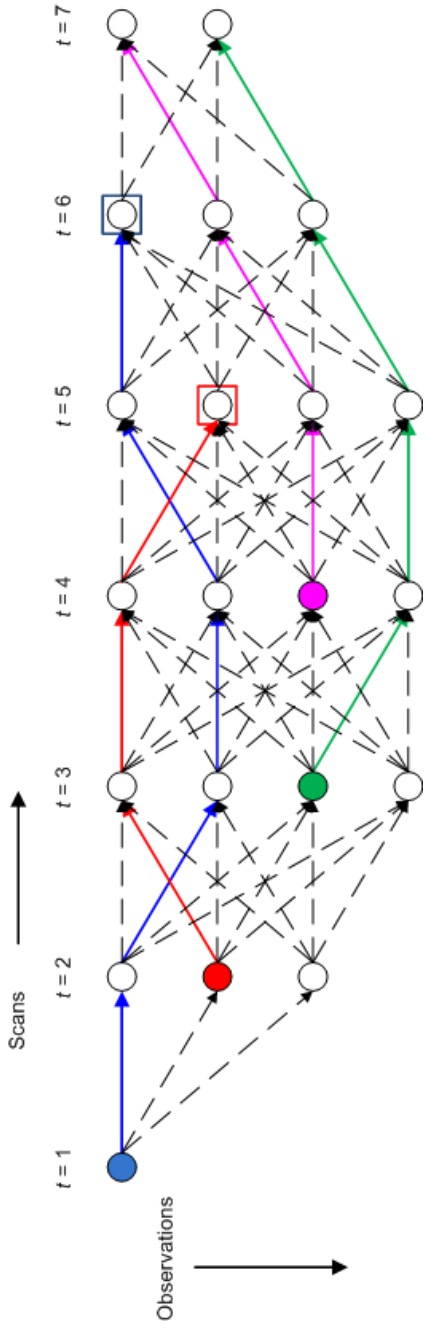


Figure 3.19: A complete trellis showing the paths for all four tracks in Fig. 3.17 and 3.18.

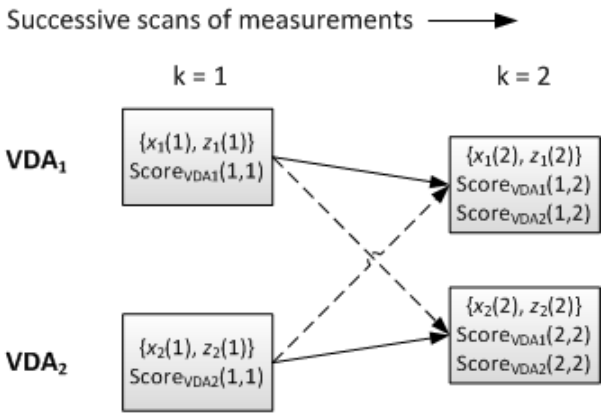


Figure 3.20: Multiple object tracking with Viterbi data association: Track continuation.

association for that as well. This process is shown in Fig. 3.20 where the Viterbi association is represented by the solid arrows and the others by dashed arrows.

Track Creation/Initialization

After iterating the new observations for all the existing tracks, we look for the observations which are not associated to any track yet. These unused observations are potential candidates for new tracks. We create a new track for each of these unused observations but those tracks are not confirmed yet. If those newly created tracks get associated to the observations for 3 consecutive scans then those are marked as confirmed objects and are displayed on the occupancy grid. In the other case, those observations are assumed to be false alarms and are suppressed subsequently. Using the same example as in the previous section, in the next few time steps, we assume that we received 2 observations each and associated them with the current tracks similarly. After a few scans we received 3 observations instead of two at the k^{th} stage. We associated each of the tracks VDA_1 and VDA_2 with these observations. Two observations got associated to the existing tracks while the third observation remained *un-associated*. We created a third instance of VDA, track VDA_3 for this observation but this newly created track is tentative and not confirmed yet.

Figure 3.21 illustrates this process. We have omitted the initial stages and shown the previous and current stage only to make the figure look clearer. For the next scans, we associate all the tracks, including VDA_3, with each of the observations received. If there are observations which are associated to VDA_3 for 3 consecutive

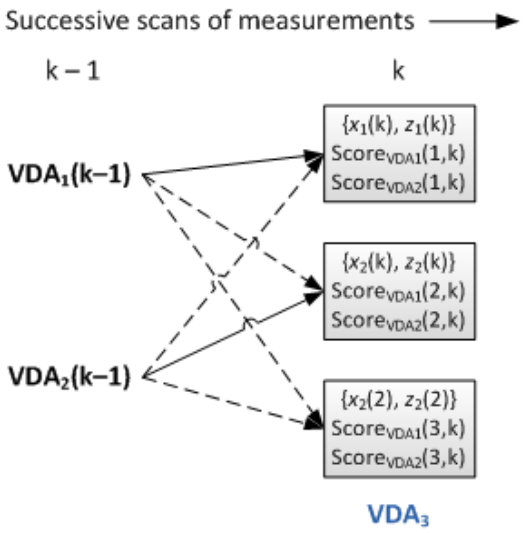


Figure 3.21: Multiple object tracking with Viterbi data association: Track creation.

scans, as shown in Fig. 3.21, then we confirm its status that it is the track for an object. Otherwise, if no observations are received for this track in next 3 scans, it is considered as a false alarm and suppressed by discarding the VDA instance.

Track Deletion

Similar to the track creation mechanism, a track deletion or suppression mechanism is also implemented in our approach. If at any stage, we find no observation associated to a track, we mark it as *not observed*. We predict the position of the object from its previous observations. This predicted position is considered to be the current position of that object. If the object is *not observed* for 3 consecutive scans, we assume that it has moved out of the scanning area and its track is deleted.

Continuing with the same example, we assume that after a few more scans, we did not receive any observation associated to track VDA_2. We estimated the position of VDA_2 using Kalman filter and considered this estimated position to be actual position of this track and marked it as *not observed*. The reason for doing so is that the object to which this track corresponds may be temporarily occluded by some other object and may become visible again after a short while. But, if we do not receive the observations associated to VDA_2 for 3 consecutive scans, we conclude that this object is no more in the viewing area therefore we delete the track VDA_2 as shown in Fig. 3.22.

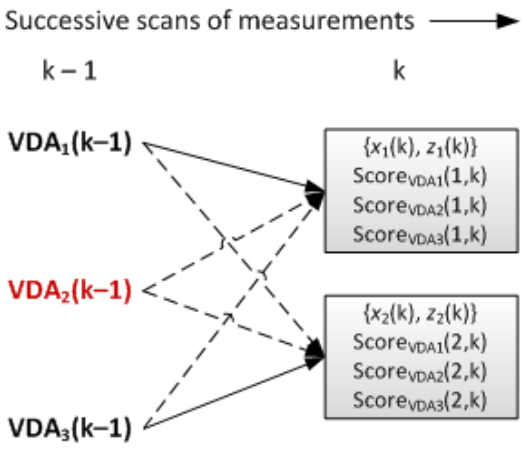


Figure 3.22: Multiple object tracking with Viterbi data association: Track deletion.

Track Merge and Split

The problems of track merge and split usually occur in the case of pedestrians as they may come so close to each other that they may not be distinguished. We assume that initially we have two independent tracks for a couple of objects. Then, at some later stage, they come so close to each other that we receive only one observation for both of them. In this case, keeping the previous record, and using the filtering technique for predicted positions, we associate the single observation with both the tracks thus to get a merged track for two objects. Figure 3.23 is an illustration of this problem. In scan $k-1$, we had tracks for two objects, VDA_1 and VDA_2. In scan k, we received only one observation $z_1(k)$. We used Viterbi algorithm to make the associations and found that both VDA_1 and VDA_2 got associated to $z_1(k)$. Then we found the predicted positions for both the objects. In the next scan, we again received a single observation and associated it with both tracks similarly. Thus we obtained a merged track for the two objects.

Splitting of tracks is the case opposite to merging. In the situations when there are two objects, such as pedestrians, moving very close to each other such that one might be hiding the other, we receive only one observation for both. As a consequence, only one track is created for both. Later, when they split their paths, we start receiving separate observations for them. In this case, we split their track in to two and the previous observations are associated to history of both tracks. Figure 3.24 is an illustration of the split problem. At scan $k-1$, we had a single observation

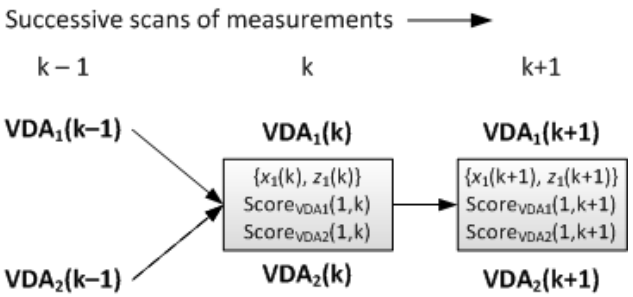


Figure 3.23: Multiple object tracking with Viterbi data association: Track merge.

for the two objects with merged tracks VDA_1 and VDA_2. At scan k, we received a couple of observations $z_1(k)$ and $z_2(k)$ and found the best associations between the existing tracks and those new observations. We found that VDA_1 got associated to the observation $z_1(k)$ and VDA_2 to $z_2(k)$. Thus we split the merged tracks for two objects as illustrated in Fig. 3.24.

A relatively complex situation of track splitting arises in the case when both the objects are moving in the same direction and with same speed from the very start. In this case, we receive a single observation for both the objects from the first scan thus we create only one instance of VDA corresponding to that observation. It carries on to be a single instance as long as we keep on receiving a single observation. But, at the instance when at least one of the objects (or both) deviates from its path, we get two separate observations corresponding to them. Then we calculate the best association for the existing track and the other observation is used to create a new instance of VDA. Thus, in this case, we have the track for the second object only from the instance we received a separate observation for it. We used the same example to illustrate this with the help of Fig. 3.25.

3.8 Conclusion

In this chapter, we have presented an introduction to the DATMO problem and the related work in last few years. We also presented a new method to perform DATMO with classification of moving objects in 3D based on the occupancy grid representation of the environment. Whenever a new measurement arrives, the moving object hypotheses are detected by finding the inconsistencies between the

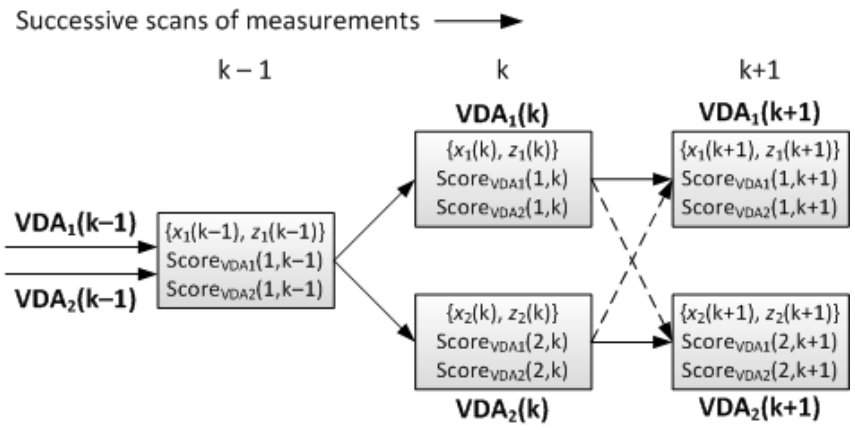


Figure 3.24: Multiple object tracking with Viterbi data association: Track split.

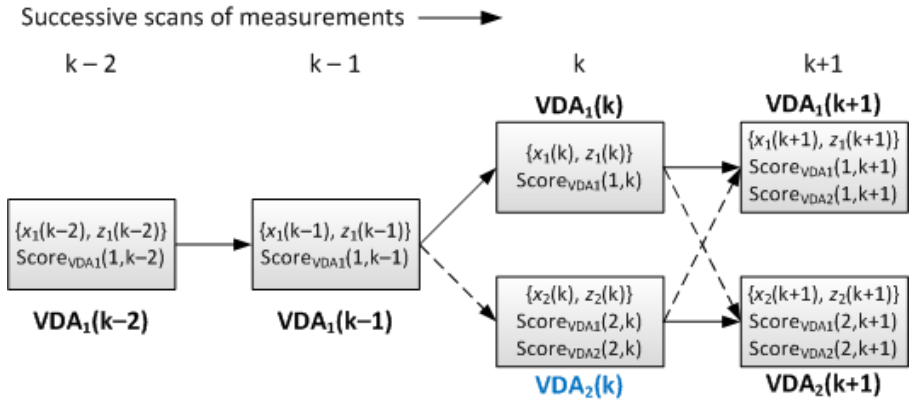


Figure 3.25: Multiple object tracking with Viterbi data association: Track split with creation of a new track.

existing grid map and the new measurement. The voxels in the grid which change from free to occupied are considered as the possible dynamic voxels. We use the DBSCAN algorithm to cluster the individual inconsistent voxels into groups. Each of these groups represents a possible dynamic object hypothesis. However, in order to identify the corresponding dynamic object, it is necessary to perform classification in addition to clustering. Thus the clustered voxels are further classified by a supervised learning approach into four predefined classes. Our algorithm for classification divides the object into layers of different heights and extracts features from each layer. In comparison to other approaches for object classification, this method does not require the pre-defined routines to extract high-level features. Instead, we extract the single-valued features from each layer and use the AdaBoost algorithm to boost these simple features to strong classifiers for object identifica-

tion. Moreover, using a supervised learning method results in assigning the semantic labels to the objects corresponding to the pre-defined classes of road users with known dynamics. Finally, we track each of those objects by using the Viterbi data association which results in robust tracking of the objects even in complex and crowded scenarios. An important aspect of this approach is the handling of temporary occlusions or missed detections as well as the problems of track split and merge. A detailed evaluation of the proposed framework will be presented in the following chapter.

Chapter 4

Experimental Results

4.1 Introduction

This chapter presents the results obtained from the implementation of methods described in chapter 2 and chapter 3 with a set of experiments in real environments. The approach presented in this work can be used with different types of range sensors in different environments (both indoor and outdoor), however, we have focused on the problem of perception for outdoor environments with a 3D laser scanner. The proposed methods are verified by using the datasets acquired by a vehicle equipped with multiple sensors. The only information that we used, other than the odometry of the vehicle, is the 3D data provided by the laser scanner mounted on top of the vehicle. The scenarios in which the datasets are acquired range from the highways to the urban and pedestrian zone environments. In section 4.2, we describe the sensor system used for data acquisition. The experimental results are then arranged in two groups based on the two main components of perception: SLAM and DATMO. First we describe the results obtained for SLAM with ground segmentation in section 4.3. These results serve as the base for our method to detect, classify and track the moving objects. Section 4.4 describes the related results that we obtained for DATMO with classification of moving objects in dynamic outdoor scenarios. Finally, the chapter ends with a summary of the results in section 4.5.

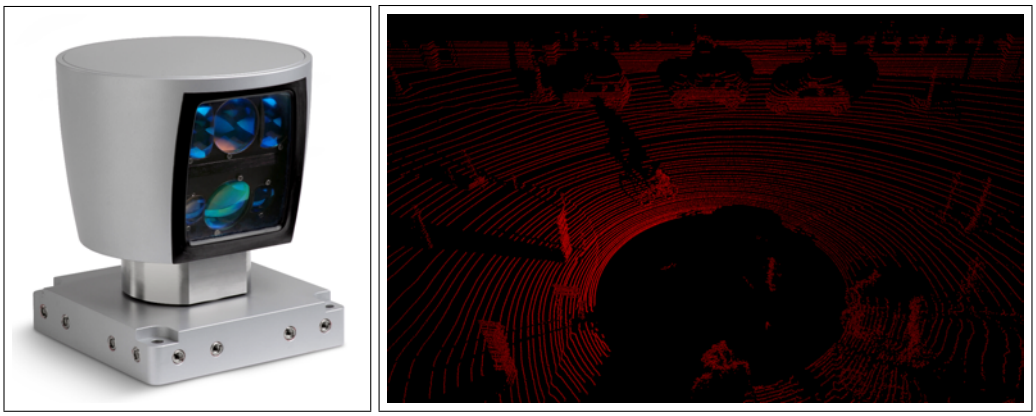


Figure 4.1: Velodyne HDL-64E sensor (left) and a sample of the data (right).

4.2 Sensor System

The data used for the evaluation of the approach presented for perception in 3D is provided by *Velodyne HDL-64E*, a high definition laser scanner (Schwarz, 2010). It is a sensor designed for obstacle detection and navigation for ground vehicles as well as marine vessels[1]. Soon after its introduction, this sensor attracted immense interest specially in the context of the DARPA urban challenge 2007 where it was used by five of the six teams which managed to finish the complete course of the race (Schwarz, 2010). However, those teams had used this sensor in combination with several other sensors including multiple 2D lidars, radars and cameras. Our approach differs in this respect by using the data acquired by a Velodyne alone.

Velodyne HDL-64E is a rotating laser scanner which consists of a column of 64 single lasers, covering a pitch range of approximately 28°. Each laser emits a beam in a specific direction allowing the measurement of reflected light upto a range of $120m$. The scanner rotates at a rate of $10Hz$ sweeping the complete horizontal ground plane and producing up to 180000 points per turn. Each chunk of data generated in a 360° turn is treated separately as a *scan*. Thus the complete scanning system has a 360° horizontal FoV, 28° vertical FoV and $120m$ maximum distance range. The sensor is shown in Fig. 4.1 with a sample of the data acquired in a single scan.

The data acquired by the sensor consists of the range measurements which can be represented as a 3D point cloud where each point (x, y, z) corresponds to a range

[1] http://www.velodyneLiDAR.com/LiDAR/hdlproducts/hdl64e.aspx

Figure 4.2: Sample of the Velodyne data stored in PNG distance image.

measurement. In addition to the 3D point cloud, Velodyne HDL-64E also provides the reflection intensity for each measurement. This intensity represents the strength of the reflected laser beam thus providing information about the surface of obstacle (e.g. shiny/matte, smooth/rough etc) from which it is reflected. Though useful for object identification, this information is not as precise as the range measurement itself therefore we have not considered it in this work.

4.3 SLAM Results

In this section, we present the results obtained for simultaneous localization and mapping with ground segmentation. For this section, we assume that the environment consists of the static entities only. Section 4.3.1 describes the used datasets followed by the details of the parameters used for implementation and illustration of the results in section 4.3.2.

4.3.1 Datasets

In order to evaluate the framework for SLAM with ground segmentation, which subsequently served as a base for the detection, classification and tracking of moving objects, we used the publicly available dataset consisting of two complex outdoor scenarios (Moosmann and Stiller, 2011). This dataset was captured with an experimental vehicle named *AnnieWay* which was equipped with a Velodyne laser scanner. The vehicle was manually driven in the streets of the residential area in Karlsruhe, Germany to obtain the data. Each 360° revolution of the Velodyne scanner was stored as a 16 bit PNG distance image as shown in Fig. 4.2. The scanner turned clockwise and filled the image from the leftmost to the rightmost column, where the leftmost and rightmost columns give the view behind the ve-

Table 4.1: SLAM Dataset

Dataset	Duration in Seconds	Number of Scans
Scenario 1	251	2513
Scenario 2	190	1900

hicle. The distance values can be converted from the image pixels to meters by using a setup provided in the configuration file "img.cfg". In addition to the point cloud, the odometry information is also provided for each dataset as a configuration file "imu.cfg". Each row in this file gives the position of the center of rear axis of the vehicle in 6D. Both the datasets are predominantly static with a very few dynamic objects. Moreover, the odometry information provided with the data is fairly inaccurate which makes it suitable to use for testing a localization method in comparison to the other datasets that we used for evaluation of DATMO where the odometry information is quite accurate. Details of the data are given in Table 4.1.

4.3.2 SLAM with Ground Segmentation

In order to test our algorithm for SLAM with ground segmentation, we experimented with different parameters and thresholds to achieve the desirable results. The first and most important parameter is the resolution of the 3D occupancy grid. We experimented with different resolutions to find the optimal voxel resolution. Figure 4.3, 4.4 and 4.5 illustrate the results obtained for the grid resolutions of 0.1m, 0.2m and 0.3m respectively. Here, the voxels identified as ground are displayed in blue while the other occupied voxels (non-ground voxels) are displayed in grey. The current location of the vehicle is represented by the '+' sign in mustard color.

Comparing the performance for different grid resolutions, we observe that larger resolutions (such as 0.3m) have an advantage of better speed but lack in the environment representation and the level of detail required for outdoor scenarios. We obtained sufficiently good results at the resolution of 0.2m with a differentiation between the ground and non-ground areas as shown in Fig. 4.4. However, the grid resolution of 0.1m was found to be most precise in the context of visual representation of the environment as well as the robustness to detect the occupied areas at a cost of the processing speed and storage requirement for the grid map. As we do not expect the system to be real time at the moment, therefore, for most of the experiments, we preferred to use the grid with 0.1m resolution due to its robustness.

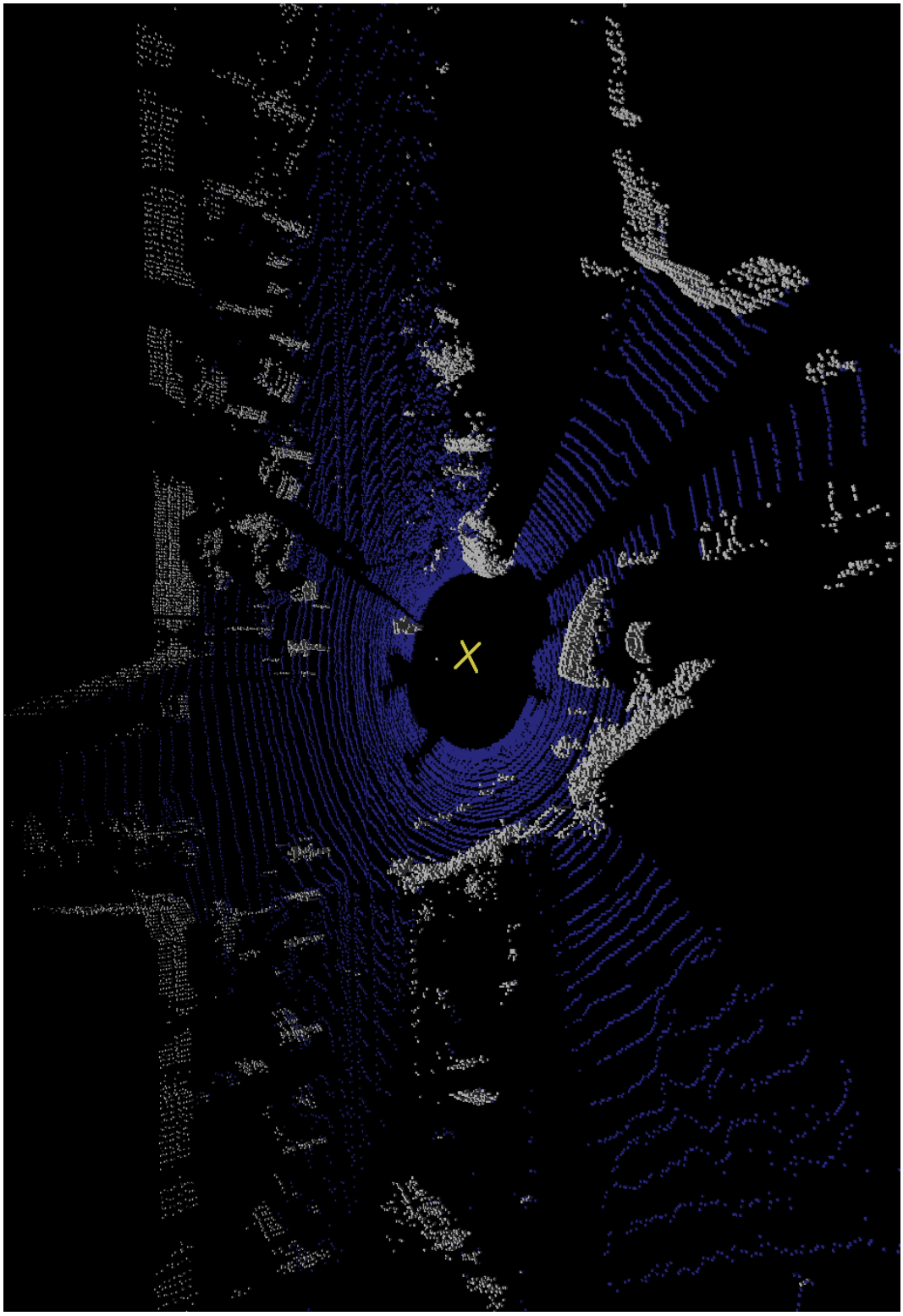


Figure 4.3: Ground segmentation results for a single scan in an outdoor scenario with a grid resolution of 0.1m. Ground voxels are displayed in blue and other occupied voxels in grey. The '+' symbol represents the position of the laser scanner.

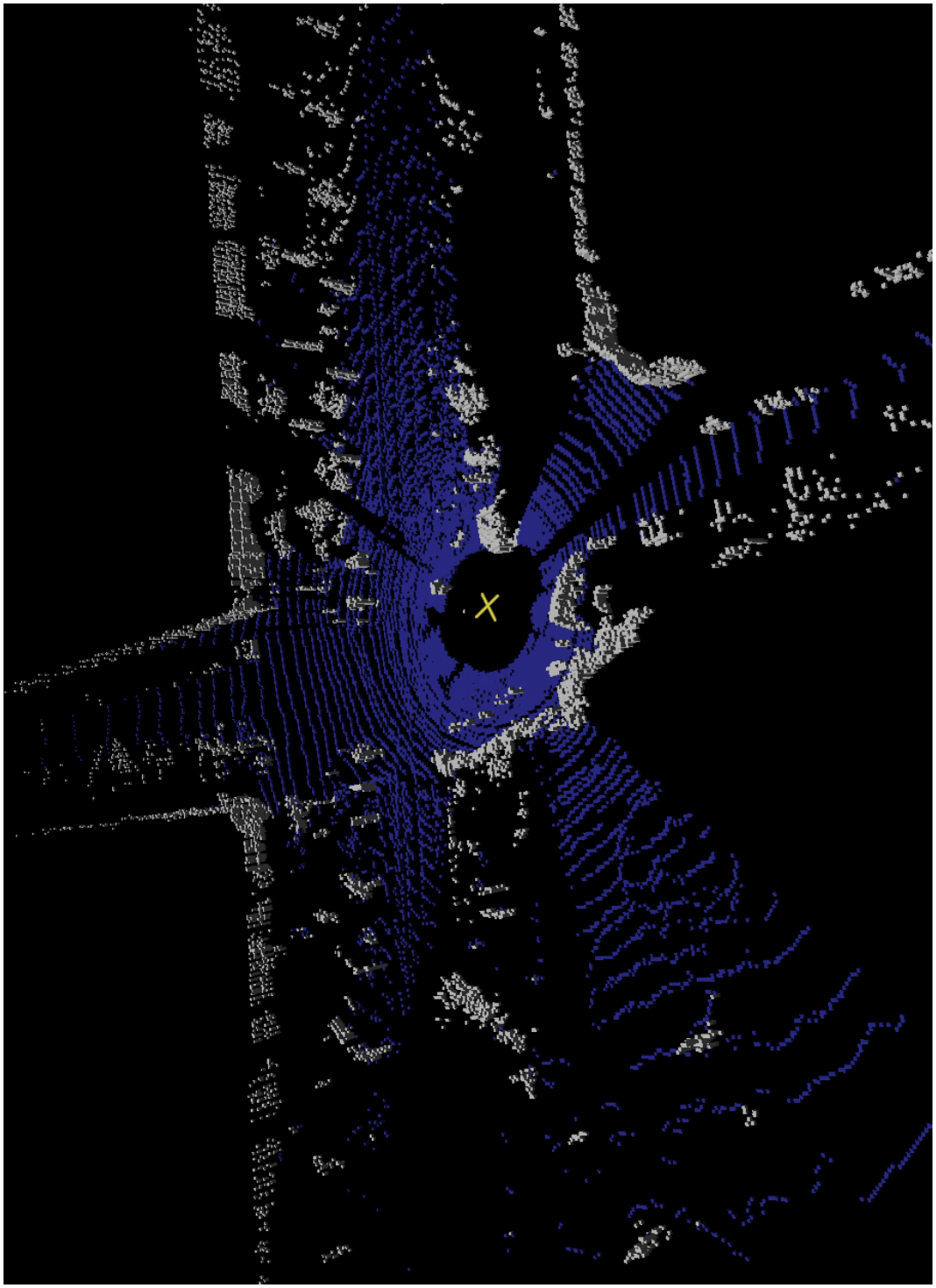


Figure 4.4: Ground segmentation results for a single scan in an outdoor scenario with a grid resolution of 0.2m.

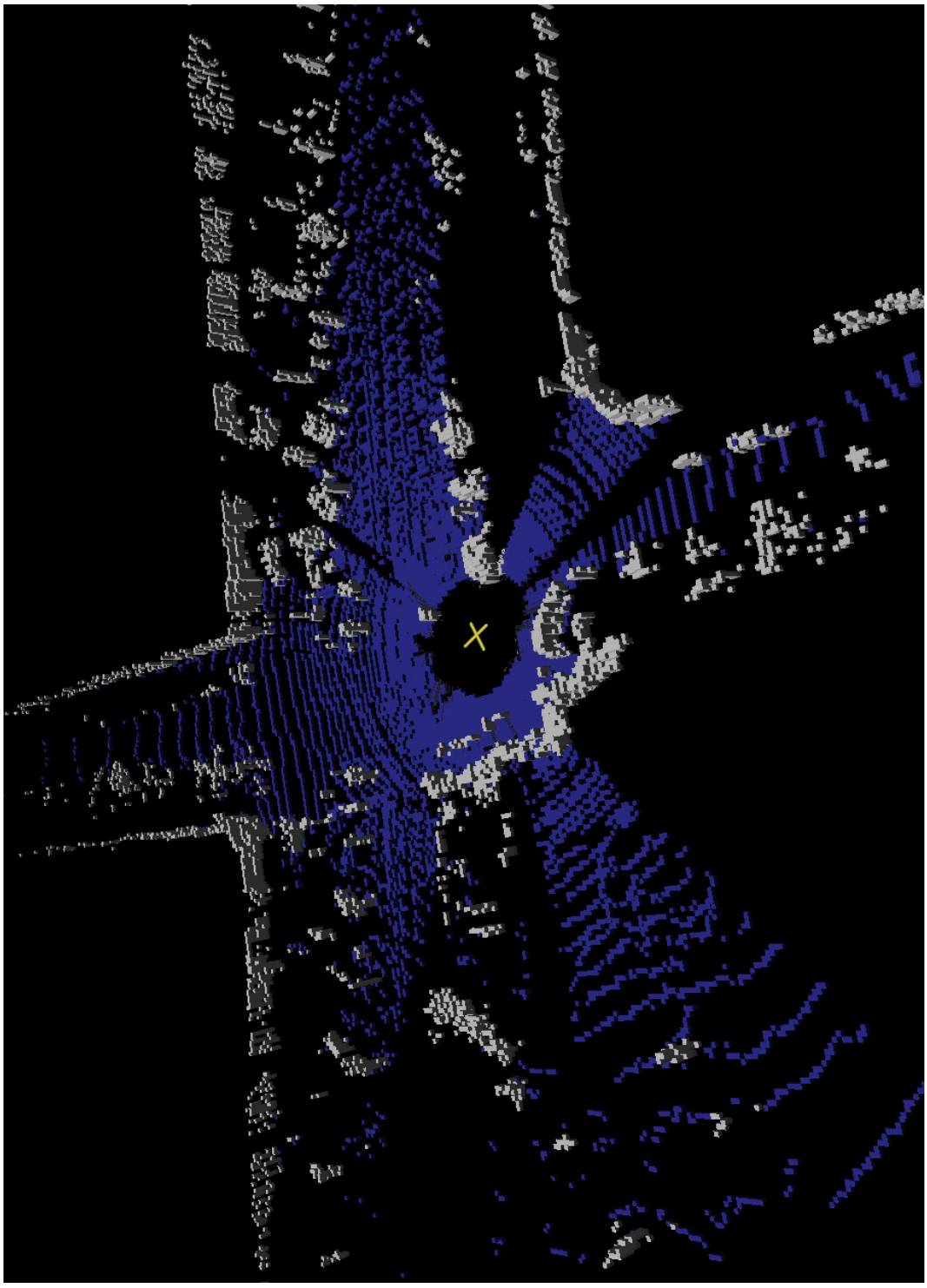


Figure 4.5: Ground segmentation results for a single scan in an outdoor scenario with a grid resolution of 0.3m.

Table 4.2: Map statistics before and after ground segmentation

Dataset	Average Number of Points per Scan	Average Number of Voxels			Percentage of Ground
		Total	Ground	Non-Ground	
Scenario 1	55680	27171	7799	19371	28.7
Scenario 2	55680	30818	10764	20053	34.9

As mentioned in chapter 2, the first step we perform after acquiring each scan is the ground segmentation using elevation map with variance. The two important parameters for this step are the resolution of the elevation map and the variance threshold used for detecting the flatness of the cell. For elevation map, we used the 2D occupancy grid with the same resolution as that of the 3D occupancy grid while the variance threshold was calculated experimentally. For the grid resolutions of 0.1m and 0.2m, the variance threshold is set to 0.015 and 0.023 respectively.

After finding the flat cells in the elevation map, we identify the largest connected area consisting of these cells and use it as the ground reference. Then we compare it with the other locally flat regions based on their mean height. We set a height threshold of 0.5 which compensates for the slight slopes or unevenness of the ground.

An interesting observation that we have after this step is that a considerable percentage of the voxels correspond to the ground. This percentage varies with the type of the environment. For the two scenarios that we used for validation, the average percentage of the ground voxels is computed to be 29% and 35% of all the occupied voxels with a grid resolution of 0.1m as shown in Table 4.2. However, both these datasets are gathered from the environments consisting of the urban streets with many objects around them. We observe that this percentage can grow to as large as 60% in the scenarios with less structures and objects which is usually the case on highways. Thus excluding these voxels, the number of the occupied voxels reduces considerably.

For scan matching, we use odometry information of the vehicle provided by its inertial measurement unit (IMU) as the initial transformation between the new scan and the grid map constructed from the previous scans. As the used sensor system has a high scan rate, the inaccuracy in the odometry information is not very high which makes it an acceptable choice for the initial transformation estimate. After constructing the 3D grid representation of each new scan and removing the ground

Table 4.3: Comparison of scan matching results with odometry

Dataset	Difference between Scan Matching Results and Odometry		
	Minimum	Average	Standard Deviation
Scenario 1	0.43	3.55	2.74
Scenario 2	0.32	2.17	1.41

voxels, we perform the grid-based ICP based scan registration of the grid map of new scan with the existing grid map. The point cloud used for ICP registration is a sub-sampled version of the original point cloud due to the grid representation and ground segmentation. Table 4.2 provides an example of the level of reduction in the number of points achieved by these two steps. The average number of points in each scan of the two datasets used in this section is 55680. Originally, a single scan of the Velodyne laser scanner generates as many as 125000 points. However, in these datasets Moosmann and Stiller (2011) have used a PNG image of 870×64 pixels to store each scan of data as a range image. The data points are stored as the intensity values of the pixels. Thus each scan has a maximum of $870 \times 64 = 55680$ points. When this scan is converted to the grid representation, the number of points which now correspond to the centers of the voxels in the grid are reduced by almost half of the original number. Further, the ground segmentation reduces these points to almost one-third of the point cloud.

We use these sub-sampled 3D point clouds for incremental scan matching. For ICP, we set the maximal allowed point-to-point distance threshold to 0.5m. Moreover, we set a convergence threshold where the change in translation becomes less that 0.01m or the change in rotation becomes less than 0.01rad. For the nearest neighbor search, we use an efficient approximate nearest neighbour library called ANN.

Figure 4.6 shows an example of the results obtained by this method for a subset of the scans from Scenario 2 of the dataset. The red line shows the estimated trajectory of the vehicle. The mapped area is of $170 \times 134\text{m}^2$ consisting of the urban streets with many cars parked on the sides. In the absence of the ground truth information, we have compared the results of our localization method with the pose obtained from IMU. Table 4.3 and Fig. 4.7 give the quantitative results obtained by this comparison. As shown in Fig. 4.7, the difference between the pose estimates is very small in the start but it increases significantly with time. This indicates the importance of a localization method to correct the pose obtained by the proprioceptive sensors of the vehicle.

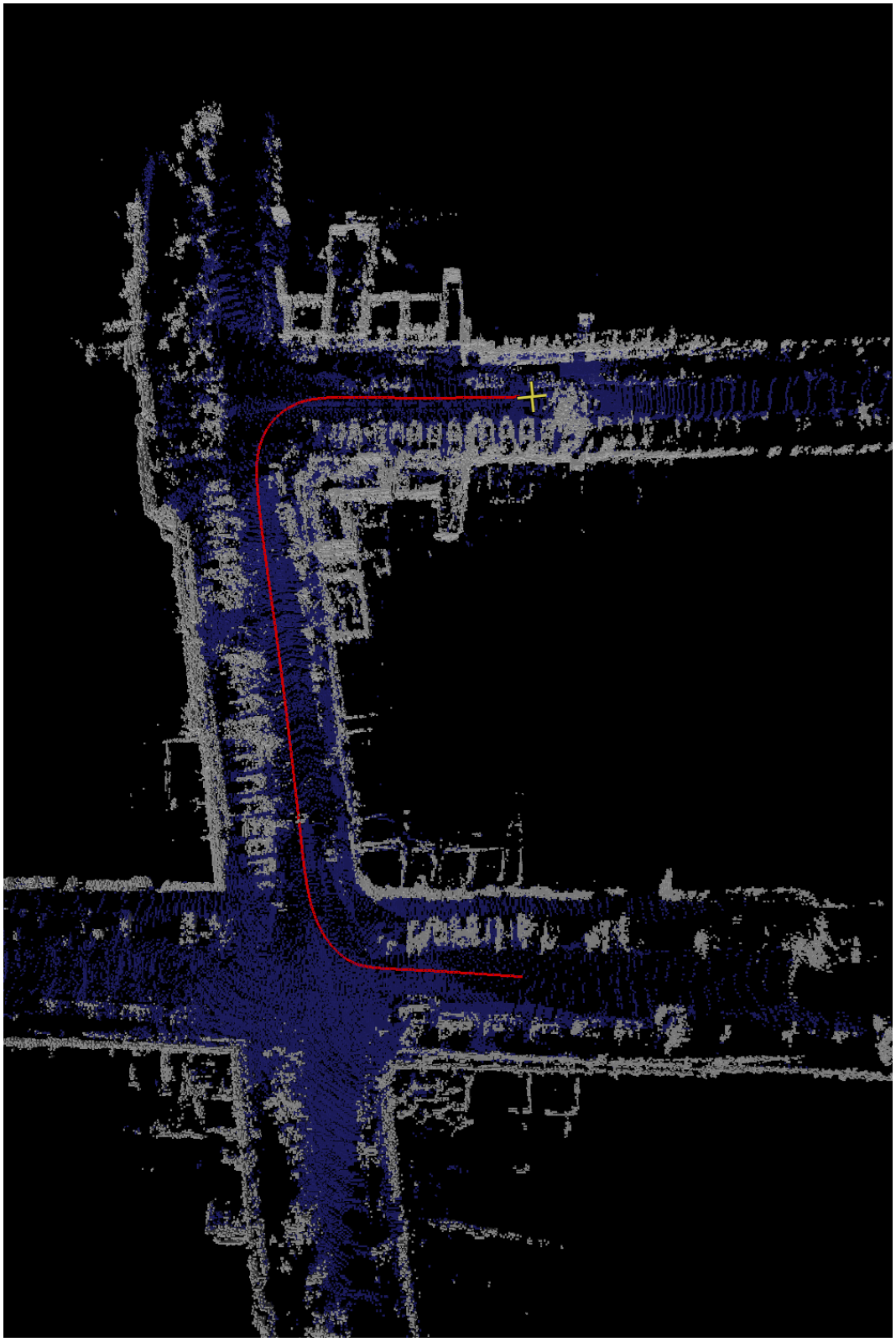
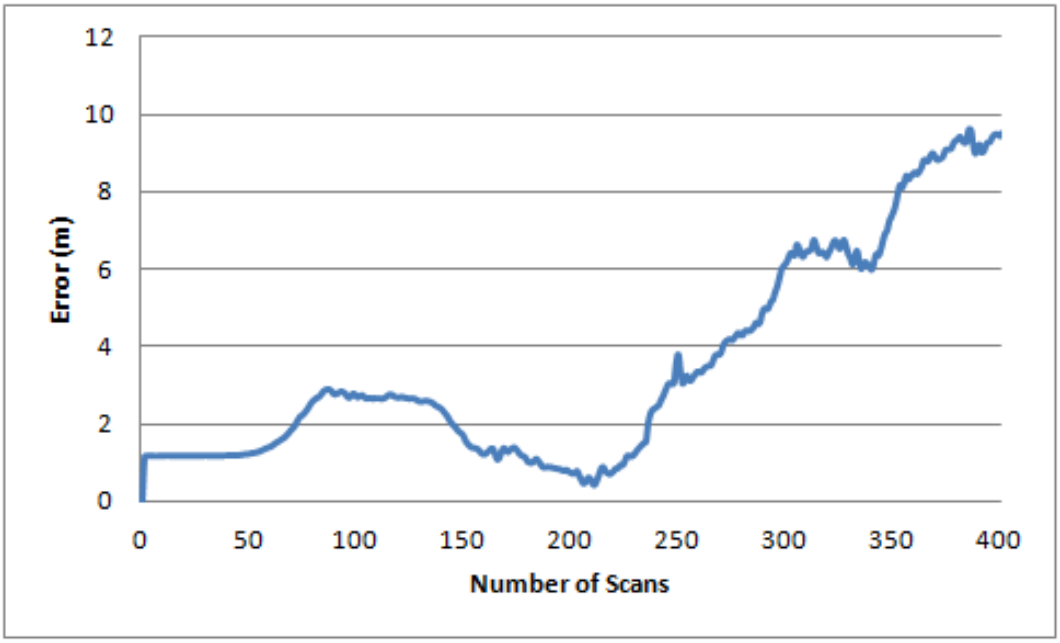
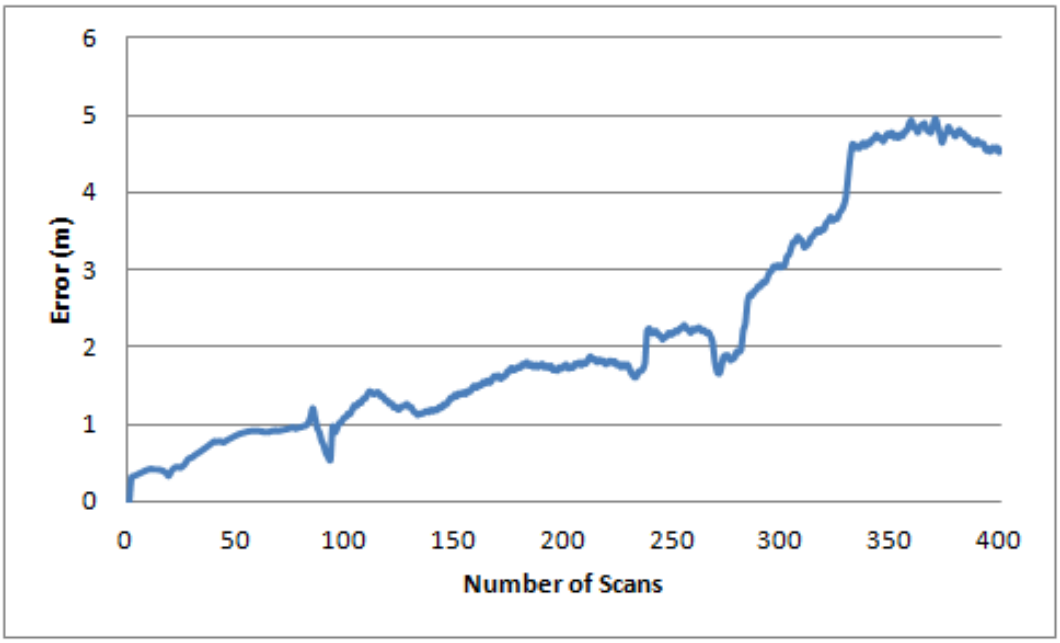


Figure 4.6: SLAM with ground segmentation results for first 400 scans in Scenario 2.



(a) Scenario 1.



(b) Scenario 2.

Figure 4.7: A comparison of the scan matching results with odometry.

4.4 DATMO Results

This section presents the experimental results we obtained for detection, classification and tracking of moving objects in dynamic outdoor scenarios. After introducing the datasets used for the evaluation (section 4.4.1), we describe the results obtained for detection of moving objects by inconsistencies between existing map and the new scan (section 4.4.2). We illustrate the effect of using different parameters for clustering the individual voxels into objects. After detection, the next step is to perform the classification of the objects by a supervised learning method. This method requires to train the classifiers using labeled datasets and then use them to distinguish between different types of objects. The experimental details are provided in section 4.4.3. Finally, the detected objects are tracked to estimate their positions with time. Section 4.4.4 provides the illustration of the tracking results in different scenarios.

4.4.1 Datasets

In order to evaluate our method for detection, classification and tracking of moving objects, we used a recently published dataset (Geiger *et al.*, 2012) which is publicly available at http://www.cvlibs.net/datasets/kitti/. The dataset, called *The KITTI Vision Benchmark Suite*, is obtained from a vehicle moving in different scenarios including highways, urban areas as well as pedestrian zones. The vehicle is equipped with multiple sensors including cameras, 2D laser scanners and a 3D Velodyne laser scanner. In this work, we have performed the evaluation only using the raw data from the Velodyne scanner, however, we have also used the camera images to manually label the data for training the classifiers as explained in section 4.4.3. The individual datasets consist of the sequences of data obtained by driving the vehicle in difference scenarios at different times. The number of frames in the sequences vary from 25 to 1000, however, most of the sequences consist of 200 to 300 frames. The 3D point cloud obtained in each frame consisting of a 360° scan of the Velodyne sensor is stored in a separate file as a binary float matrix which needs to be parsed using C++ or MATLAB code. In this file, each point is stored with four coordinates (x, y, z, r) where (x, y, z) give the position of the point and r gives the reflectance value of the laser at that point. We have only used the position and not the reflectance information in this work. Contrary to the dataset described in sec-

tion 4.3.1, here the number of points in a scan is not constant. The average number of points in a scan is approximately 120,000. In addition to the 3D point cloud, we have used the odometry information provided with the dataset. This information is obtained by an advanced precision inertial and GPS navigation system called OXTS RT3003[2] which combines the best attributes of inertial navigation system (INS) with a high quality GPS to provide the accurate positioning information. For each frame, 30 values of GPS/IMU are stored in a separate text file including position, orientation, velocities, accelerations, angular rates and satellite information. The position is stored in the form of latitude, longitude and altitude. We first convert it to the (x, y, z) coordinates and then transform it from the IMU coordinate frame to Velodyne coordinate frame by using the calibration matrix given with each sequence of data. For the evaluation of different components of our system, we have used the sequences of data obtained from three different types of scenarios: highway, urban and pedestrian zone. The obtained results are detailed in the following subsections.

4.4.2 Detection of Moving Objects

After constructing the 3D grid map, we detect the possibly dynamic voxels from the inconsistencies between the grid map of the new scan and that of the previous scans. The detected dynamic voxels are then clustered using DBSCAN to represent the individual dynamic object hypotheses. As described in section 4.2, the Velodyne laser scanner has a range upto 120m. However, in our implementation for the DATMO, we set a maximum range of 50m as the points become very sparse at further distances. Those sparse points, and thus the sparse voxels, cannot be clustered together to represent the objects and to perform classification.

For DBSCAN, we experimented with the different values of minimum acceptable density to start a cluster as well as different sizes of ϵ-neighborhood. For the choice of ϵ, it is important to note that the distance between the points in a scan increases with the distance from the sensor. Thus the objects closer to the sensor have denser clusters in comparison to those far away. Therefore, in order to cluster the distant objects correctly, we need to have a larger value of ϵ. In our implementation, we use an ϵ-neighborhood value of 0.65m. A disadvantage of this choice is that it results in an over-segmentation of the dynamic voxels where many of the clutter points which are close to the sensor are clustered together to form the object hy-

[2]http://www.oxts.com/products/rt3000-family/

Table 4.4: Average number of dynamic voxels before and after clustering

Dataset	Average Number of Dynamic Voxels		Percentage of Dynamic Voxels
	Before Clustering	After Clustering	After Clustering
Highway Scenario	8728	1971	23%
Urban Scenario	6315	2422	39%
Pedestrian Zone Scenario	7377	2808	38%

pothesis. However, in the safety critical applications such as autonomous driving, over-segmentation is considered safer than the under-segmentation as it results in avoidance of all the risky situations by detecting all potential objects. Moreover, in our case, this problem is dealt with in the next step of object classification.

We illustrate the results of our method for detection of moving objects from the scenario in Fig. 4.8. The vehicle is stopped at an intersection with the moving objects around it. The moving objects, represented by the green rectangles, include four pedestrians crossing the road in front of the vehicle, a car passing from left to right on the front road and two bicyclists coming from behind the vehicle. Figure 4.9 shows the same scenario after two scans. The detected dynamic voxels are displayed in red while the static and ground voxels are displayed in grey and blue respectively. Figure 4.10 is another representation of the same results where we have hidden the static occupied voxels for a clearer representation. It can be seen that the detected voxels contain a large amount of clutter in addition to the actual dynamic objects. After clustering, the remaining objects are shown in Fig. 4.11 with different colors representing the individual clusters.

Figure 4.12 and 4.13 are the illustration of the results of dynamic object detection from the same scenario after ten scans. Figure 4.12 shows all the dynamic voxels accumulated over the previous scans. However, when we perform clustering at each time step, the voxels which are detected as noise are discarded from the dynamic voxels and therefore they do not appear in the subsequent time steps. This reduces the number of wrongly detected dynamic voxels by a huge amount as illustrated in Table 4.4. Despite this fact, the results contain many of the clusters which do not belong to the dynamic voxels as illustrated in Fig. 4.13. To identify the real dynamic objects from these clusters of voxels, we perform the classification of moving objects.

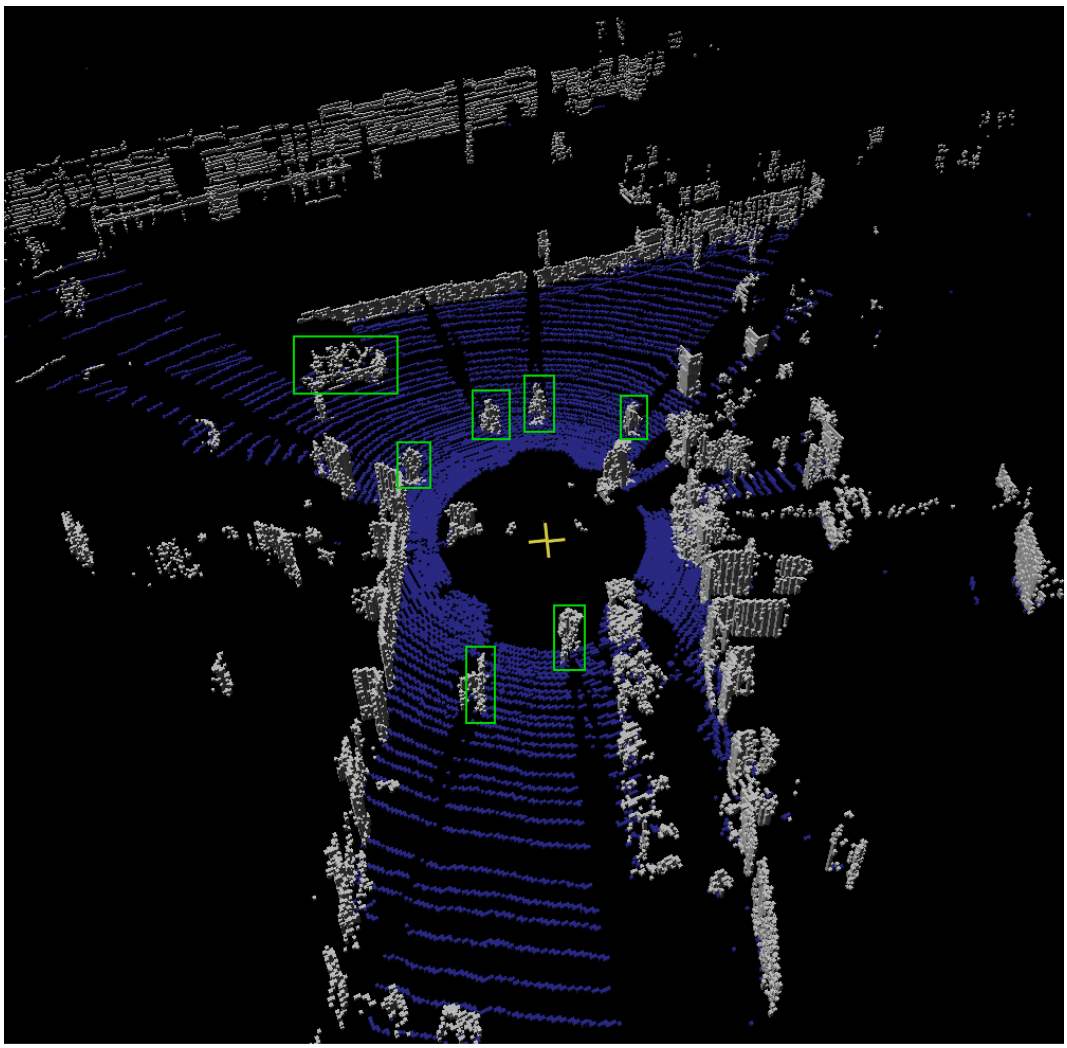


Figure 4.8: Detection of moving objects. The scene at the start of the sequence, $t = 0$ s, consisting of multiple moving objects represented by green rectangles.

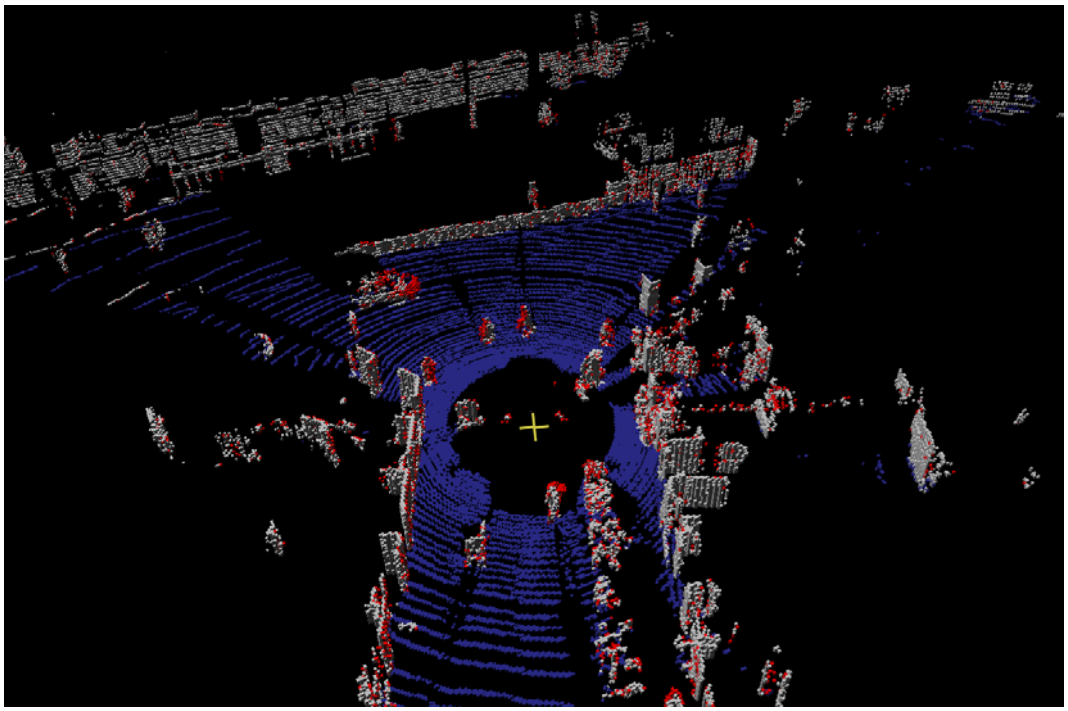


Figure 4.9: Detection of moving objects. Results after two scans. All the dynamic voxels detected from inconsistencies between the scans are shown in red and all other occupied voxels in grey while the ground voxels are in blue.

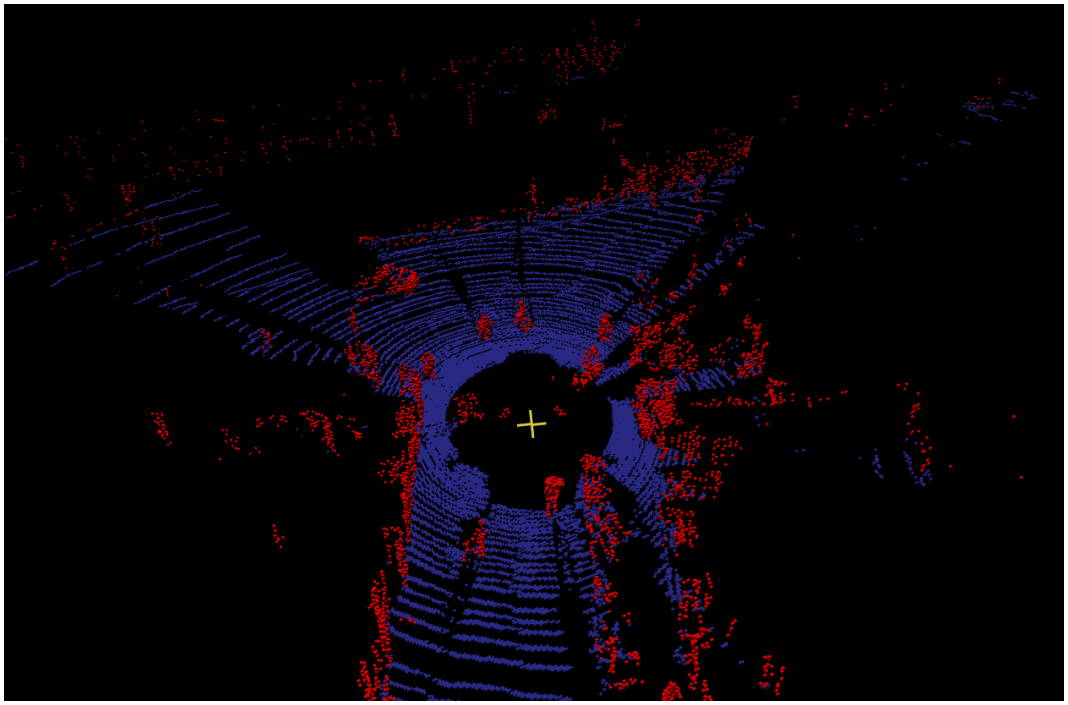


Figure 4.10: Detection of moving objects. Results after two scans. Only dynamic occupied voxels are shown (in red) for a clear representation.

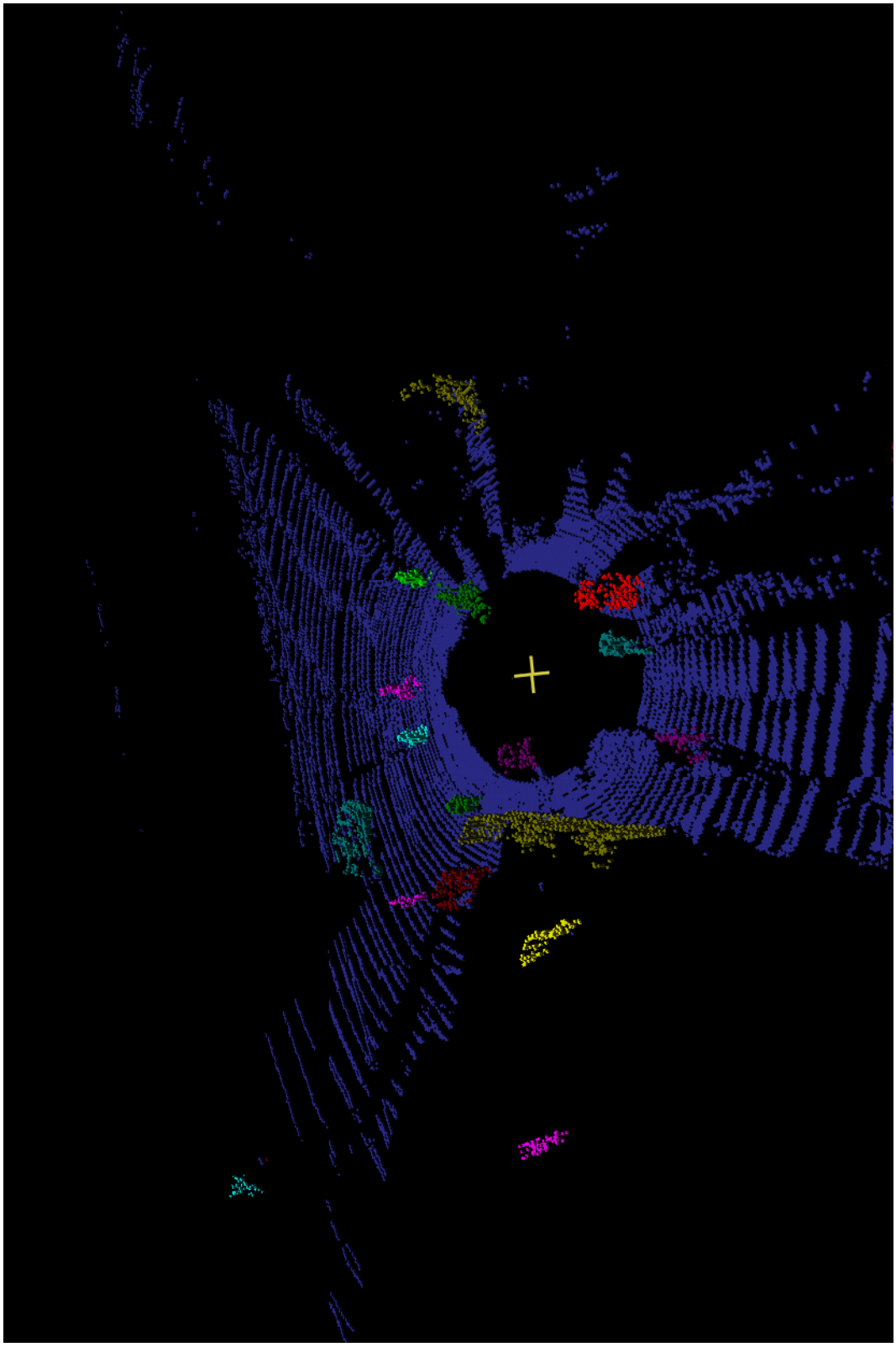


Figure 4.11: Detection of moving objects. Results after two scans with clustering of the dynamic voxels. Different dynamic object hypotheses are shown in different colors.

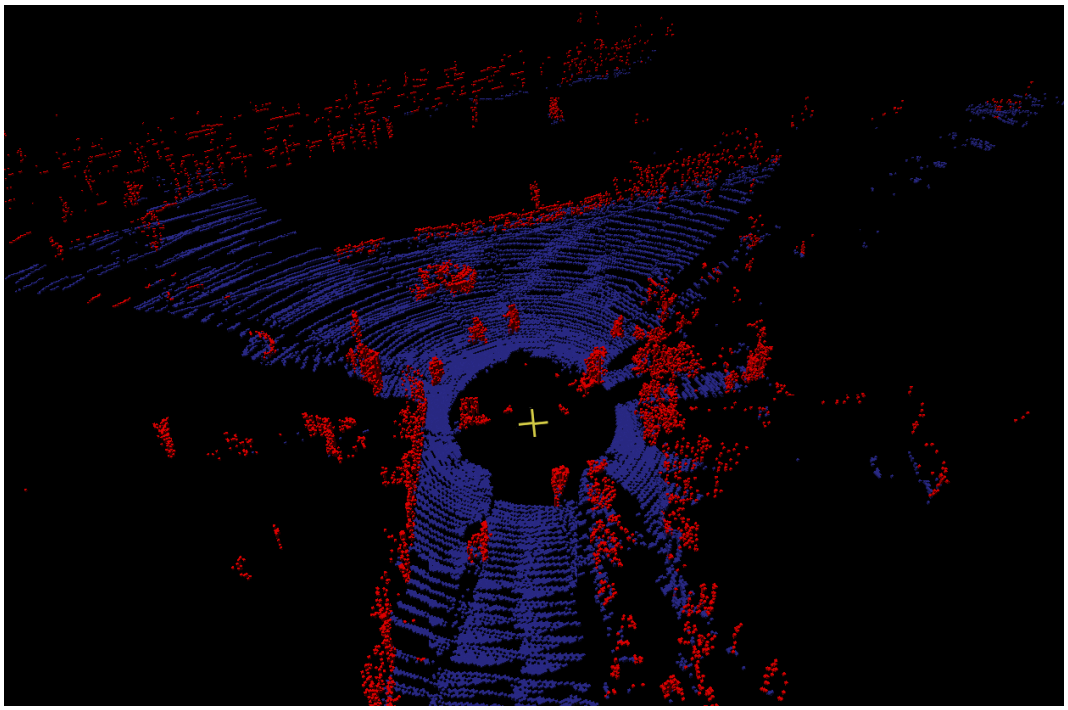


Figure 4.12: Detection of moving objects. Results after ten scans, showing all detected dynamic voxels accumulated over the ten previous scans.

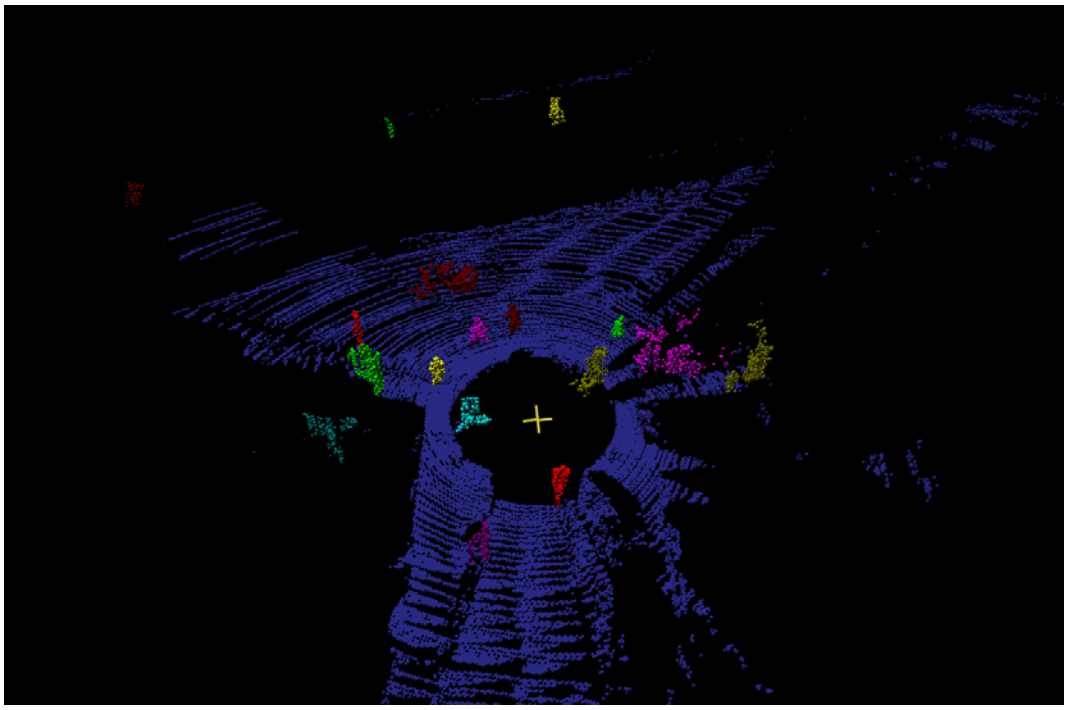


Figure 4.13: Detection of moving objects. Results after ten scans showing different object hypotheses as clusters of different colors. Some of these object hypotheses are not dynamic objects such as the clusters in cyan and mustard colors near the vehicle.

4.4.3 Classification of Moving Objects

The task is to classify the detected objects in to the following four classes: *pedestrian*, *bike*, *car* and *bus*. Here, *bike* class contains both bicycles and motorbikes, *car* contains cars and vans and *bus* contains buses and trucks. In order to test the object classification, we first trained four separate classifiers for each object class. For this purpose, we required the labeled datasets which are quite rare in the context of the used sensor. Therefore, we introduced a method to manually label the data to train the classifiers. In the following section, we detail the different datasets that we used for training as well as the process used to label the data. We then demonstrate the results obtained by our method for object classification in outdoor environments.

Data Labeling and Training Datasets

Our approach for classification is based on four separate classifiers corresponding to the four object classes. As explained in chapter 3, we used the supervised Adaboost algorithm to define each classifier. Adaboost requires the features from a set of positive and negative examples of the class as an input to train the classifier. Therefore, we first create the training dataset for each class which consists of both positive as well as negative examples of that class. For example, to train the *car* classifier, the set of positive examples consists of the clusters of voxels representing the cars while the set of negative examples contains all other clusters. We trained each classifier by computing several features from these clusters as described in section 3.6.3. The same training algorithm is used for all four classes with the only difference being the dataset used for training the classifiers.

Due to the unavailability of sufficient labeled data, we relied on defining our own databases corresponding to the examples of each class. We first tried to use the synthetic data generated from the object models in Google's 3D Warehouse[3], as described in (Lai and Fox, 2010). The idea is to use a simulated laser scan for generating the data in the form of 3D point cloud corresponding to the object model. The merit of using this publicly available dataset was the variety and the large number of objects that it contains. However, the main problem in this context was that the point clouds generated from these objects were quite different than those produced by a laser scanner in the real environments. As an illustration, Fig. 4.14 shows

[3]http://sketchup.google.com/3dwarehouse/

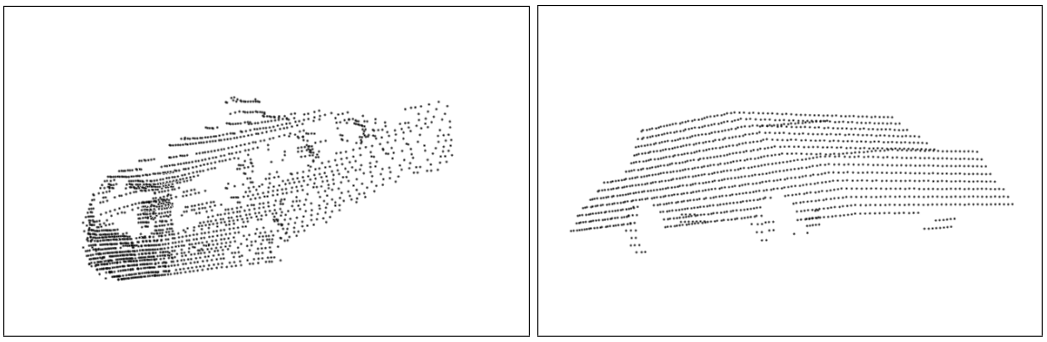


Figure 4.14: A comparison of the real data generated by Velodyne (left) and the virtual scan data generated from the 3D model of a car in google's 3D Warehouse (right).

the cluster of points generated from the 3D model of a car in the 3D Warehouse in comparison to that generated from a real car by a Velodyne laser scanner. It can be clearly seen that the point cloud of the synthetic object is quite regular and dense at all height levels as compared to that of the real object scanned by real laser scanner. Moreover, it does not model the transparent surfaces of the object such as the glass of the windows or windscreen. Thus this model is too complete and dense to be considered for realistic environments.

Based on this experience, we decided to generate the training datasets directly from the Velodyne data. In order to do so, we used the scans obtained from the different scenarios containing the moving objects of interest and hand-labeled those objects. As the performance of the classification module depends on the offline learning of AdaBoost classifiers therefore we ensured a variety of data for each class by taking the scans from diverse outdoor scenarios ranging from highways to urban traffic scenes. We used our method for detection of moving objects as the preprocessing step for filtering the raw data to label the objects. An advantage of using this method is that the extracted object samples are closely related to the objects that the intelligent vehicle is expected to come across in the real outdoor scenarios.

We selected the sets of scans from the dataset described in section 4.4.1 and passed those to our moving object detection module. The selected scans contained dynamic objects including, but not limited to, the pedestrians, bikes, cars and buses and also some static objects of interest such as trees, light poles, street signs etc. which were often wrongly detected by our detection module as dynamic. The results of detection were displayed as the clusters of different colors on our GUI, as described in section 4.4.2. We used the camera images provided with the dataset as

the ground truth to visually identify the moving objects from the clusters as shown in Fig. 4.15. Thus each cluster was manually assigned a label according to the object class. Taking the consecutive laser scans to extract and label the objects provided us with different views of the objects as they evolve with time thus giving a variety for each class. Additionally, it helps in labeling other objects than those appearing in the camera image by identifying them based on their positions in the previous frames. As a result, the object clusters such as the yellow cluster in Fig. 4.15 is identified as a pedestrian despite the fact that it does not appear in the camera image. All other clusters which were visible in the camera images but did not belong to any of the four classes were labeled as *background* to be used as negative training examples later.

We selected 10 different sequences of data including 3 sequences each from urban environments and road environments and 4 sequences from the pedestrian zones to ensure the diversity of the data. We selected the scenes with relatively larger number of objects to obtain a reasonably large dataset. We extracted 50 scans from each scene for labeling. The selected scenes totalled to 3550 exemplars of the four object classes of interest with a large number of other objects.

In the next step, we formed four training sets corresponding to these classes. In order to do so, we first extracted the features from each object cluster (as explained in section 3.6.3). We saved the feature vector of each object along with its class label in a common database. Then, for defining a training dataset for a specific class (e.g. car), we took the exemplars of that class as the positive examples and all other classes (e.g. pedestrians, bikes, bus, trees, fences, poles and unknown objects) as the negative examples and formed a new training set. In this way, we defined four different training sets corresponding to the four classes. As the number of positive examples for *bus* class were too few in the selected scans, therefore we augmented the corresponding training set by using the examples of bus and truck provided in the Velodyne dataset of (Lai and Fox, 2010). The details about each training set are given in Table 4.5.

To get an impression about the visual appearance of this training data, Fig. 4.16 shows some of the extracted examples for the *car* class. The complete training set contains a total of 3723 examples which are split into 1231 positive and 2492 negative examples. As illustrated in this figure, the training set contains the positive examples observed from a variety of viewing directions as well as distances. Fig-

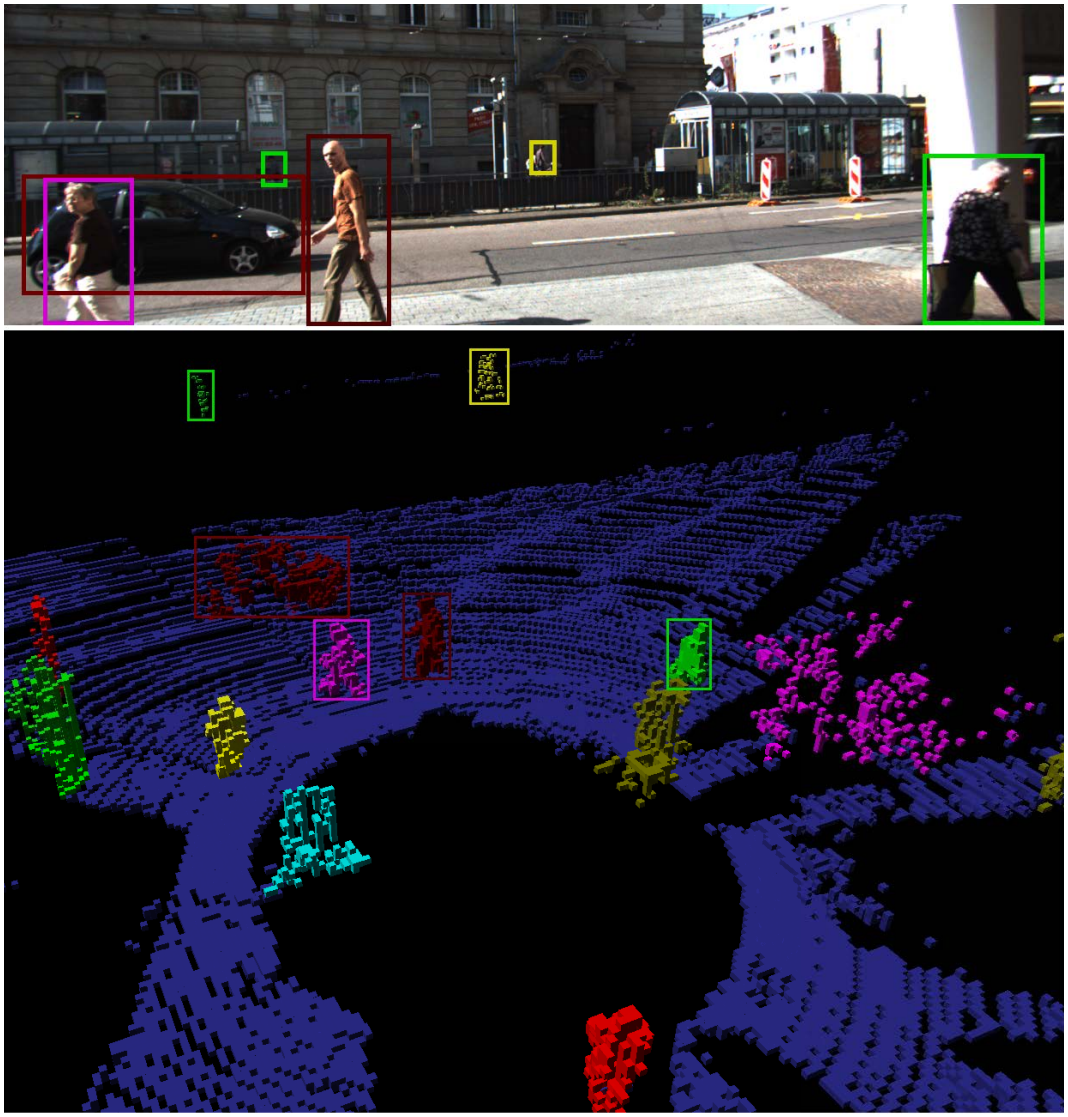


Figure 4.15: An illustration of the data labeling process. Results of the dynamic object detection step (bottom) are compared with the image of the scene (top) and appropriate class labels are assigned to the clusters of voxels. The rectangular boxes represent the detected objects matched with the camera image. In addition to the objects appearing in the camera image, there are some other moving objects which can be identified based on their positions in the previous frames. For instance, the yellow object cluster on the left of the screen is not visible in the camera image but it can still be labeled as a pedestrian as we have already observed it in the previous frames. Similarly, the orange cluster of voxels behind the sensor is identified and labeled as a bicycle.

Table 4.5: Training datasets for different classes of moving objects

Object Class	Total Examples	Positive Examples	Negative Examples
Pedestrian	3114	1038	2076
Bicycle	2715	895	1820
Car	3723	1231	2492
Bus	1176	386	790

ure 4.17 is another illustration to show some of the positive examples for different classes of interest.

Classifier Learning and Classification Results

After defining the training datasets for each class, the next step is to learn the classifiers by using AdaBoost training procedure. In order to do so, we randomly divided the complete set of labeled training examples for each class into the training and validation sets according to a ratio of 2 : 1. For each of the four classes, the training set was used to learn a strong classifier using AdaBoost whereas the test set was used for the evaluations.

An important parameter of the AdaBoost algorithm is the number of weak classifiers T used to form the final strong classifier. In order to find the optimal value for T, we performed multiple experiments and found that a value around 100 gives good results. In this way, we trained four classifiers: $pedestrianClassifier$, $bikeClassifier$, $carClassifier$ and $busClassifier$. For each of these classifiers, the results from the test sets to measure the performance of the object classification are given in Table 4.6.

Table 4.6: Classification results for the individual binary classifiers

Object Class	Total Objects	Correctly Classified	Wrongly Classified
Pedestrian	342	336	6
Bicycle	295	286	9
Car	406	392	14
Bus	127	124	3

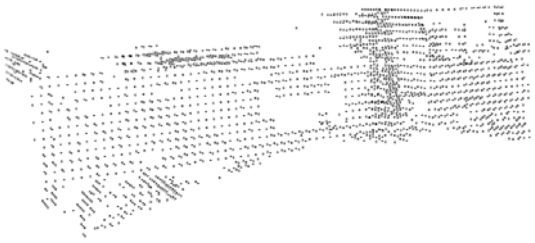
Once we have trained the four classifiers offline, the next step is to use them in our perception system for classifying the detected objects. For this purpose, we use the sequential multi-class classification (as described in section 3.6.5) and arrange the classifiers according to their individual estimated error with the classifiers with

Figure 4.16: Some examples of the hand-labeled data used for training the *car* classifier. First four rows contain the positive examples while the last two rows contain the negative examples.

(a) Different examples of pedestrian.

(b) Vans.

(c) Bicycles.



(d) A truck.

(e) Motor bikes.

(f) Buses.

Figure 4.17: Some examples of the hand-labeled data used for training different classifiers.

Table 4.7: Estimated error from the training data for the individual binary classifiers

Object Class	Estimated Error
Pedestrian	1.9
Bicycle	3.0
Car	3.6
Bus	2.3

lower error arranged before the ones with higher error. Table 4.7 gives the estimated error computed from the training data for each of the individual binary classifiers. The error rates vary between 1.9% and 3.6%. The *pedestrian* classifier yields the lowest error rate. Therefore we arrange it as the first classifier in the list followed by the *bus*, *bike* and *car* classifiers respectively.

In order to verify our algorithm for classification of moving objects, we performed the quantitative tests. As there is no ground truth information available, we compared the results of classification with the datasets manually. Table 4.8 illustrates the results that we obtained in three different types of environments. The first column gives the type of environment used for testing. The second column shows the number of instances of the objects present throughout the duration of execution. Each appearance of the object is considered as a separate instance i.e if an object appears in multiple scans, then each time the object appears, it is counted as a new object. This number is calculated manually using the images corresponding to each sequence of data as well as the estimations from the previous positions of the objects. The third and fourth columns give the results of classification including the number of times the objects are correctly or wrongly classified respectively. The results shows that most of the times the objects are classified correctly in all scenarios. We observed that most of the wrong classifications result from the objects detected at relatively larger distances from the sensor. The possible reason is that the data corresponding to these objects is not dense enough to be able to classify them correctly. However, the performance of our system for classification still proves to be satisfactory despite this problem.

Table 4.8: Classification of moving objects: Quantitative results

Type of Dataset	Total Objects	Correctly Classified	Wrongly Classified
Highway	195	161	34
Urban	532	463	69
Pedestrian Zone	518	457	61

4.4.4 Multiple Object Tracking

After detection and classification of moving objects, we perform the multiple object tracking to estimate the position of these objects with time. For this purpose we use a sliding window of 10 frames and the objects detected within this sliding window are stored in the trellis structure of the Viterbi algorithm. At each time step, when we receive a new set of observations, we perform the data association with the existing tracks in the trellis. For the creation and deletion of a track, we use a waiting time of 3 scans. This ensures that a track is not confirmed until it is observed for at least three scans and it is considered as a false alarm otherwise. Similarly, if a track does not appear in a scan, it is not deleted immediately rather it is continued with its predicted position used as the actual position. However, if the track does not appear for 3 consecutive scans, than it is deleted from the track list.

In order to test our system for DATMO in 3D environments, we used the datasets recorded in different scenarios. As there is no ground truth information available for tracking, we provide the qualitative results here. Each tracked object is represented by a minimum oriented bounding box corresponding to the cluster of voxels contained in that object. The colors of the boxes and the voxels representing an object are chosen according to the class of that object. Cars are shown in green, bikes in yellow, buses in orange and pedestrians in magenta. In addition to the bounding box, the tracked object is also represented by a *track ID* which is formed by combining the object class and the track number e.g. *car_3* or *bike_2*.

In the following, we illustrate the results of multiple object tracking in a city scenario with many cars and a bus. Figure 4.18a shows the scenario at the start of the sequence. There is a single moving object behind the ego vehicle which is correctly detected and tracked as *car_1* at $t = 0.5$ s as shown in Fig. 4.18b. Figure 4.19 shows the results of tracking at $t = 13.0$ s and 15.8 s where the track for *car_1* continues along with two new cars tracked as *car_3* and *car_4*. It can be observed that *car_4* is not identified at $t = 13.0$ s due to being far from the sensor and generating too few voxels to be clustered. However, it is correctly classified at $t = 15.8$ s even though it is only partially visible from behind. Figure 4.20 and 4.21 show the results of tracking later in time when the vehicle reaches an intersection and stops. As it can be seen, our method for tracking correctly tracks many cars crossing in the intersection as well as those passing by (e.g. *car_7* and *car_8* in Fig. 4.21) and coming from behind (e.g. *car_12* in Fig. 4.21b). Moreover, the previously tracked cars *car_1*

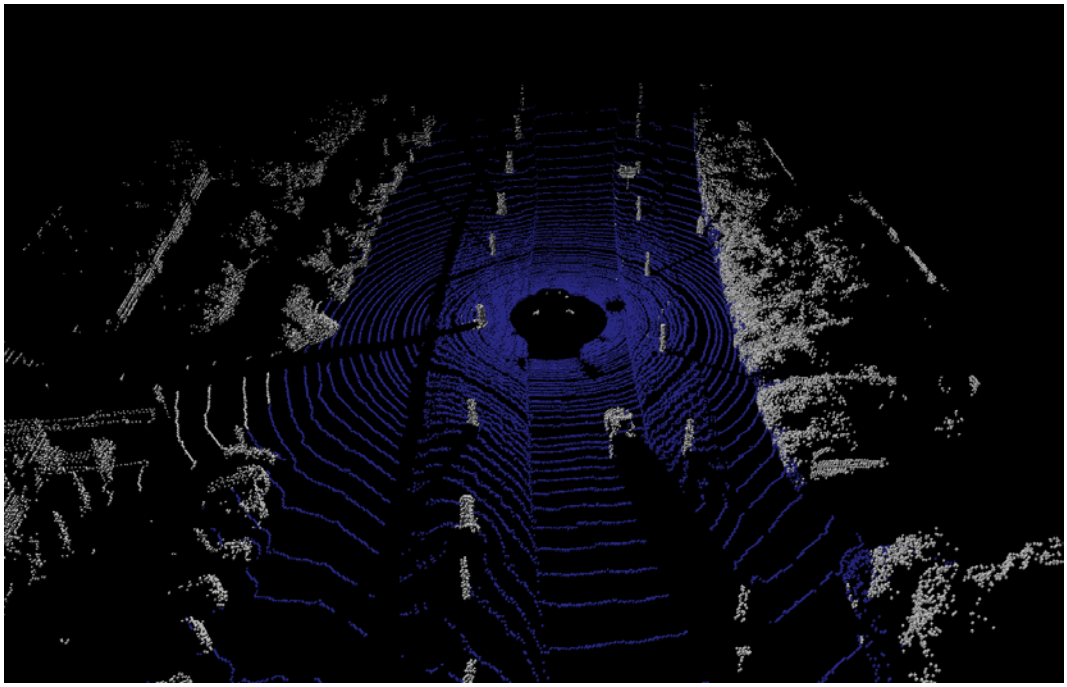
and *car_4* also stopped at the intersection along the ego-vehicle and therefore they are not tracked any more (until and unless they start moving again) as shown in Fig. 4.21 onwards. In addition to the many cars which are tracked throughout the sequence, a bus is also detected, classified and tracked successfully as shown in Fig. 4.22.

Continuing with the qualitative results, Fig. 4.23 to 4.27 are an illustration of the experiments conducted for evaluating multiple object tracking in an urban scenario with many pedestrians and bicyclists around the vehicle. Figure 4.23b shows the result of tracking at $t = 0.3$ s. There are three objects, two bicycles and a pedestrian, detected by the system displayed in yellow and magenta colors respectively. Figure 4.24a and 4.24b are the results from the same scenario after 1 second and 2.4 seconds respectively where another object (*bike_4*) is detected in addition to the three objects previously tracked.

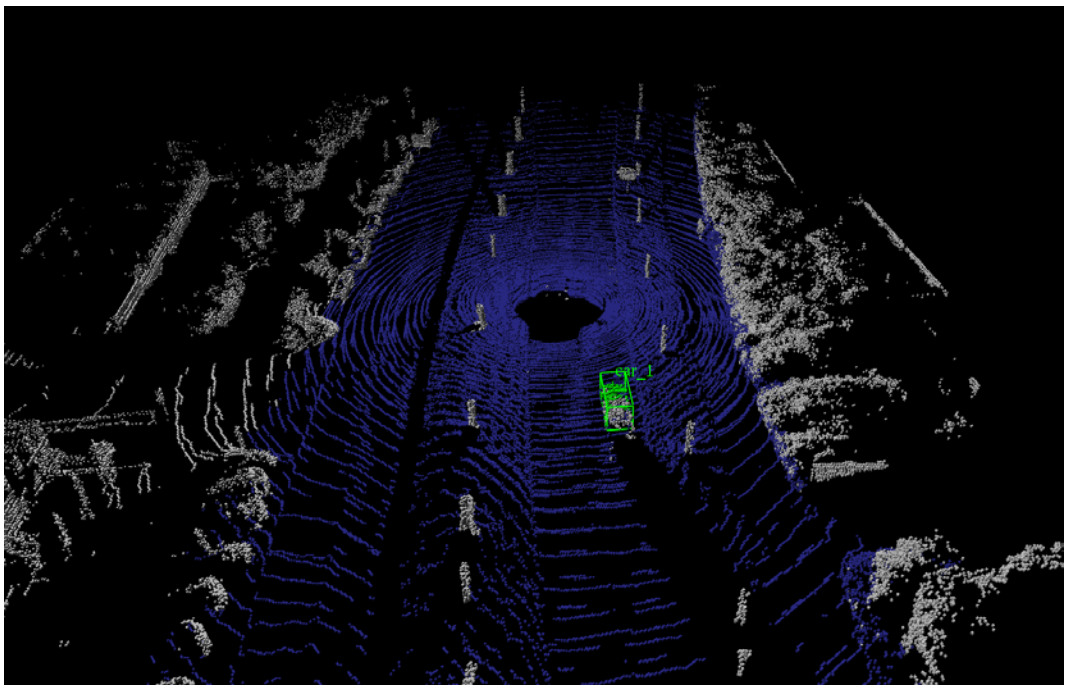
Figure 4.25 shows the same scenario after approximately 7 and 10 seconds with a relatively larger number of objects around the vehicle. Here, most of the objects are correctly classified and tracked, however, there are some false alarms too such as *pedestrian_4* in Fig. 4.25a. This false alarm is suppressed by our tracker soon after. Additionally, track for *bike_5* was lost due to an occlusion and missed detection for longer than 3 scans and therefore a new track (*bike_6*) is generated for the same object (Fig. 4.25b). Despite these issues owing to the limitations of the sensor and the detection module as well as occlusion of the objects, we observe that the cases of false and lost tracks are very few and most of the objects are tracked successfully. Figure 4.26 and 4.27 show the tracking results continued until the end of the sequence.

It is evident from these illustrations that some objects which were closer to the ego-vehicle were successfully tracked over a long period of time. For example, the tracks for *pedestrian_1* and *bike_3* were initialized as early as $t = 0.3$ s and they continued till the end of the sequence. Most of the other objects are also tracked for a considerable span of time thus showing the efficacy of our system for multiple object tracking.

As mentioned earlier, we did not have the ground truth information to evaluate our system for multiple object tracking quantitatively. In order to get an idea about the performance of the system, we used a method to manually identify the tracks and compare them against the results of our tracking algorithm. For this purpose, we

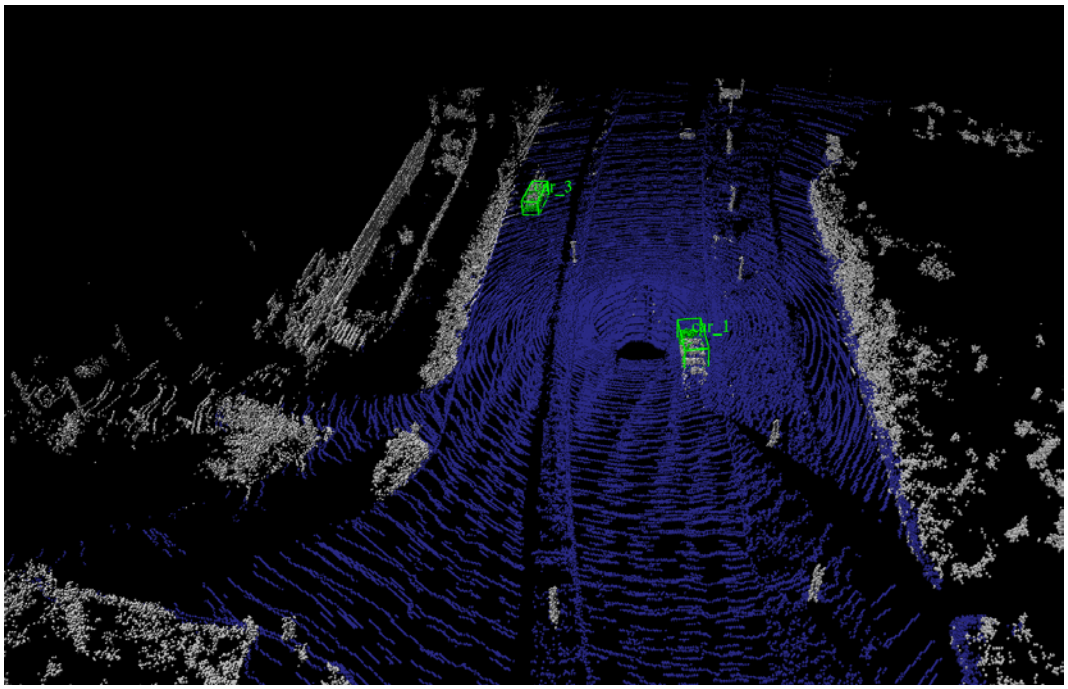


(a) A city road scenario at beginning of the sequence, $t = 0$ s.

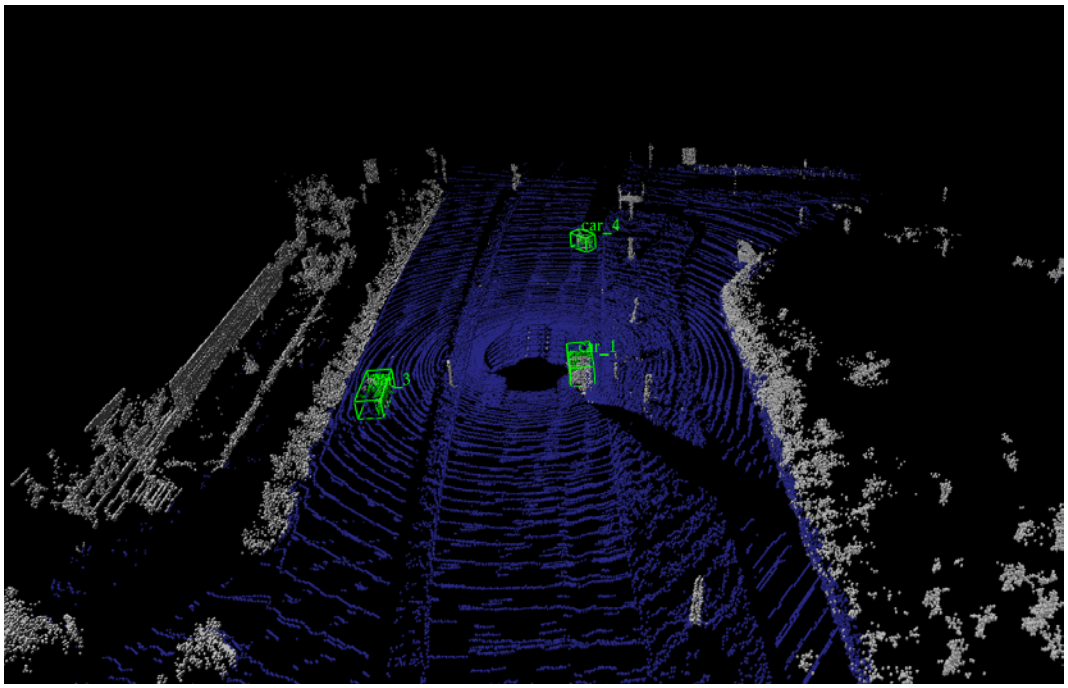


(b) Tracking results at $t = 0.5$ s.

Figure 4.18: Tracking results at $t = 0.5$ s. There is only one moving object which is correctly detected and classified as a car. A track is initialized and confirmed for this car, marked as *car_1*.

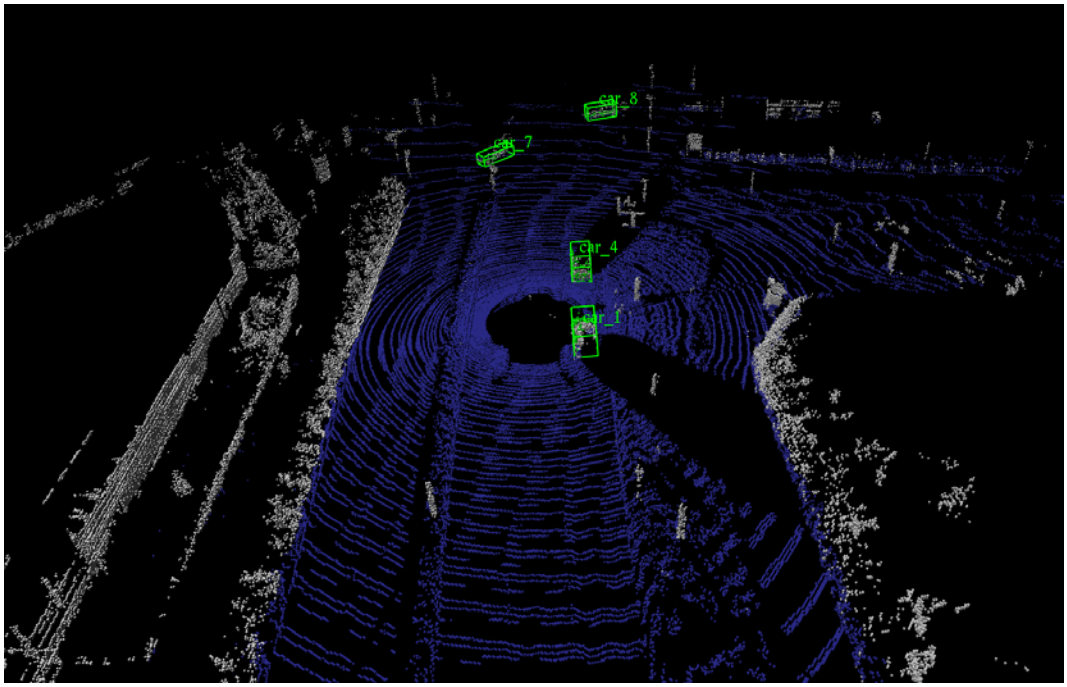


(a) Tracking results at $t = 13.0$ s.

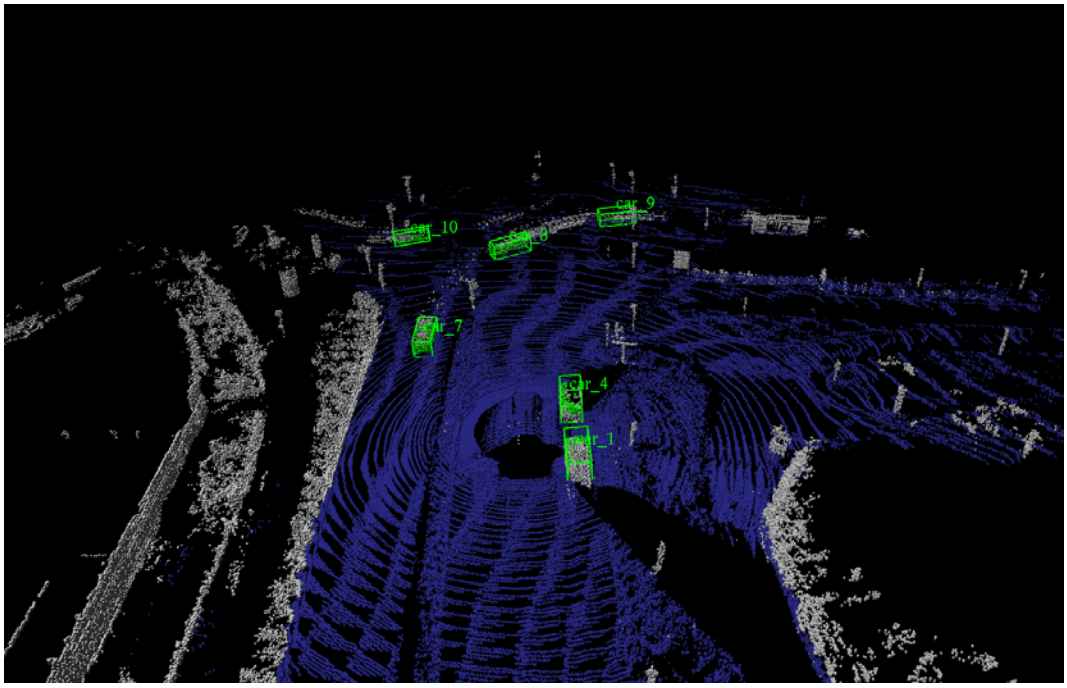


(b) Tracking results at $t = 15.8$ s.

Figure 4.19: Tracking results continued from Fig. 4.18 at $t = 13.0$ s and $t = 15.8$ s. *Car_4* is not identified at $t = 13.0$ s as it is far from the sensor and generates small number of voxels but correctly identified at $t = 15.8$ s despite being visible partially from behind.

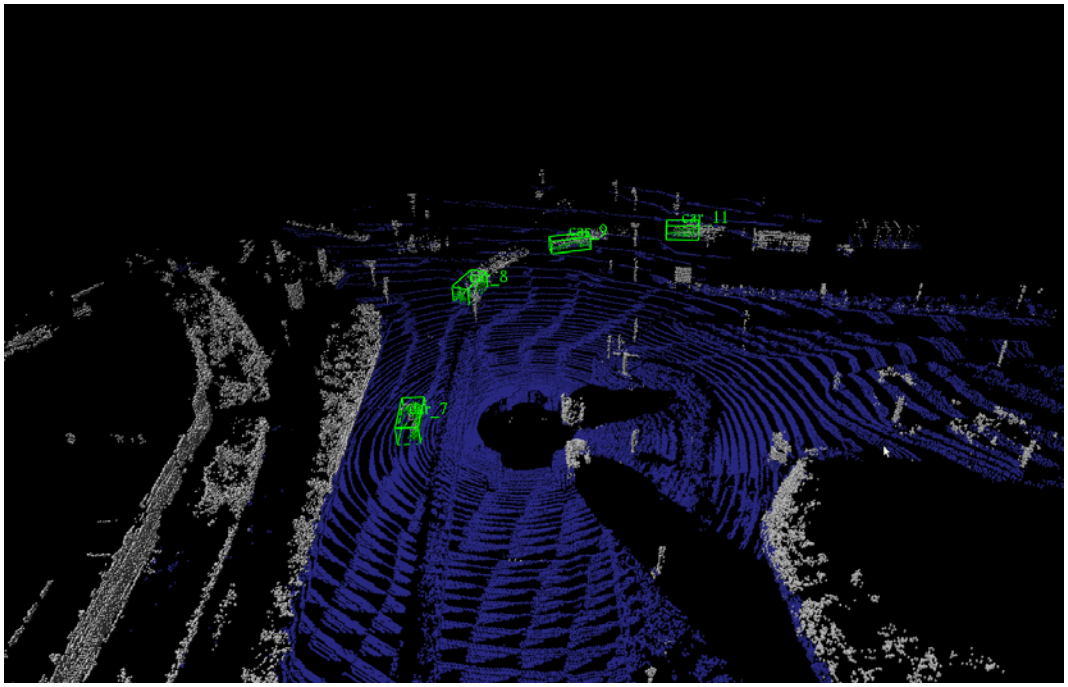


(a) Tracking results at $t = 20.4$ s.

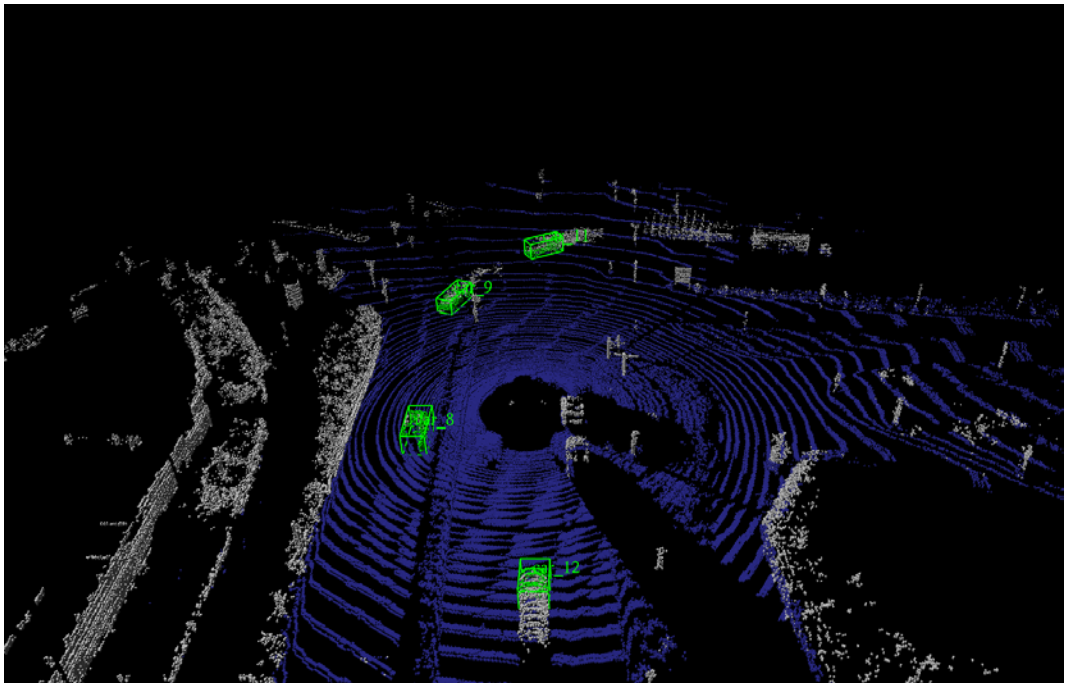


(b) Tracking results at $t = 23.0$ s.

Figure 4.20: Tracking results continued from Fig. 4.18 at $t = 20.4$ s and $t = 23.0$ s. The vehicle has reached at an intersection with many cars crossing in front of it.

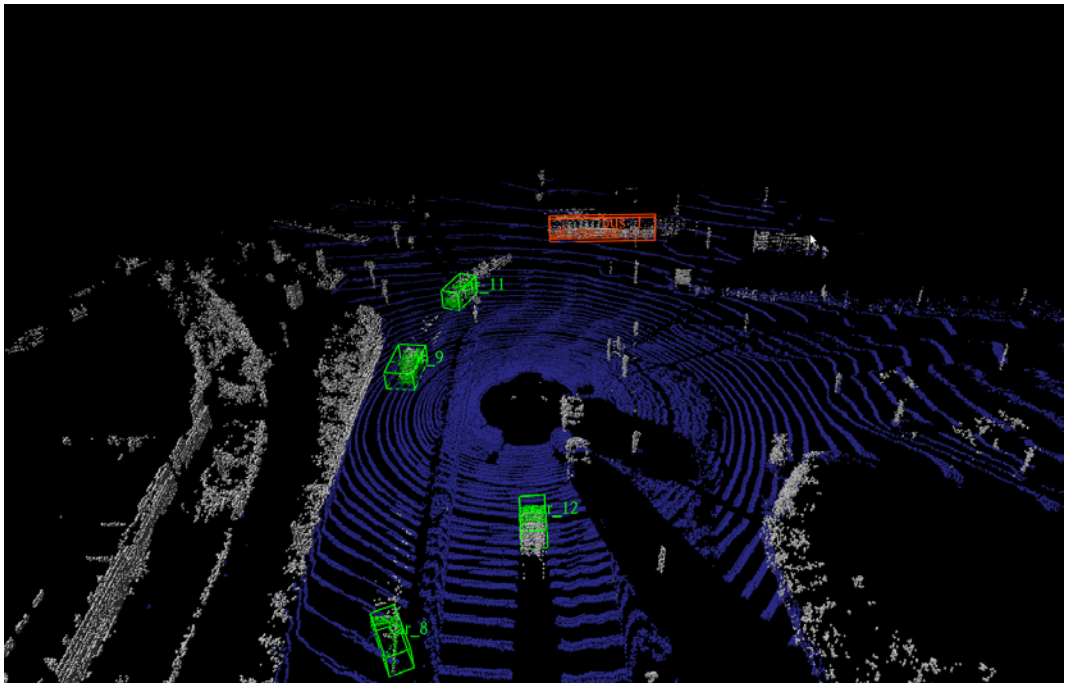


(a) Tracking results at $t = 24.8$ s.



(b) Tracking results at $t = 29.4$ s.

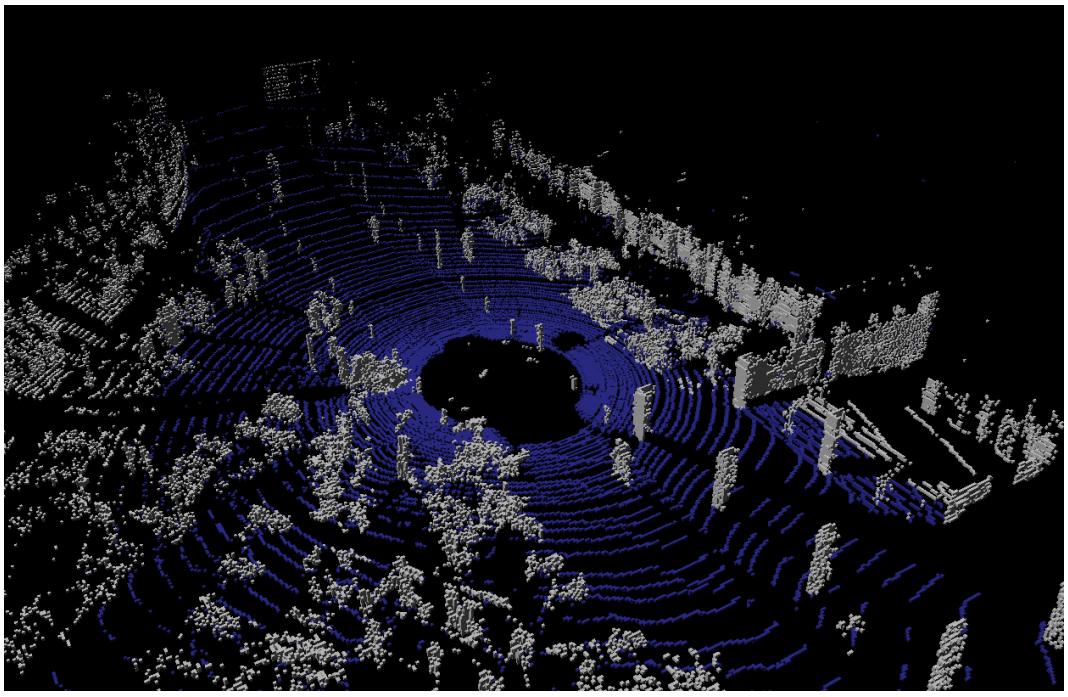
Figure 4.21: Tracking results continued from Fig. 4.18 at $t = 24.8$ s and $t = 29.4$ s. The cars which stopped (*car_1* and *car_4*) are not tracked any more until they start moving again.



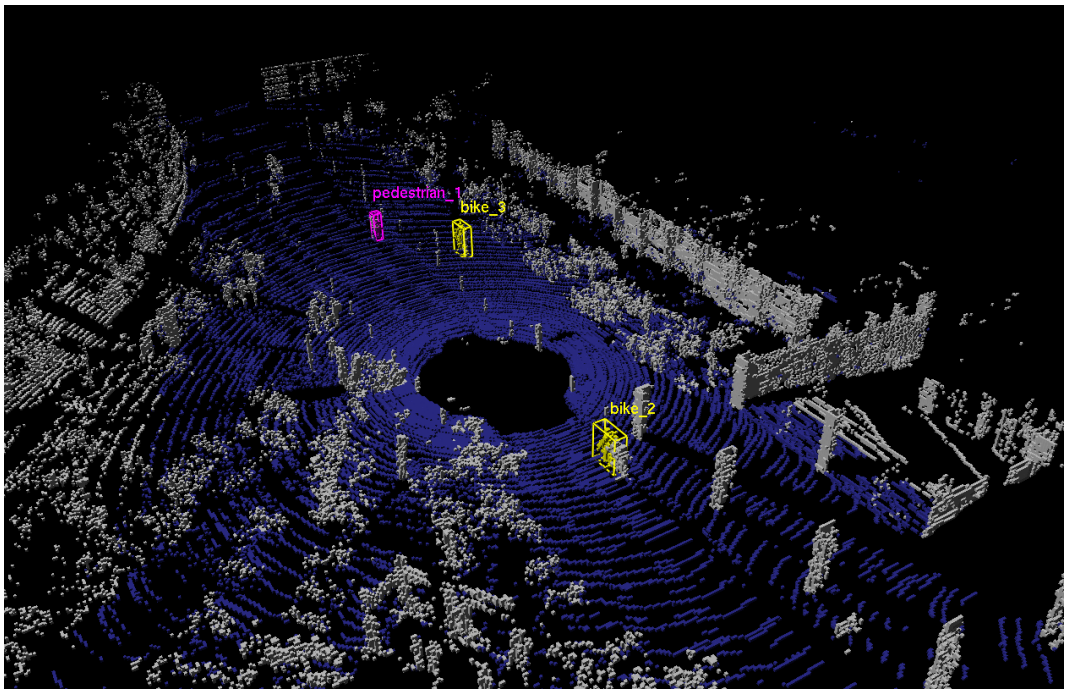
(a) Tracking results at $t = 32.4$ s.

(b) Tracking results at $t = 38.4$ s.

Figure 4.22: Tracking results continued from Fig. 4.18 at $t = 32.4$ s and $t = 38.4$ s. In addition to many cars, a bus is also detected and tracked.

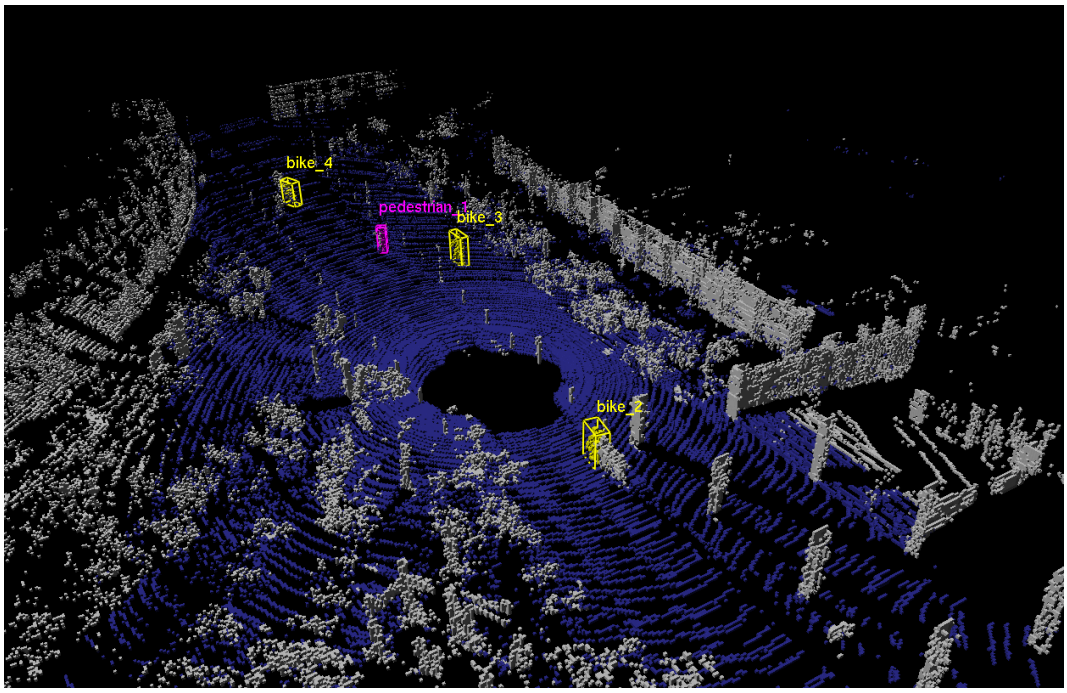


(a) Scenario at $t = 0$ s.

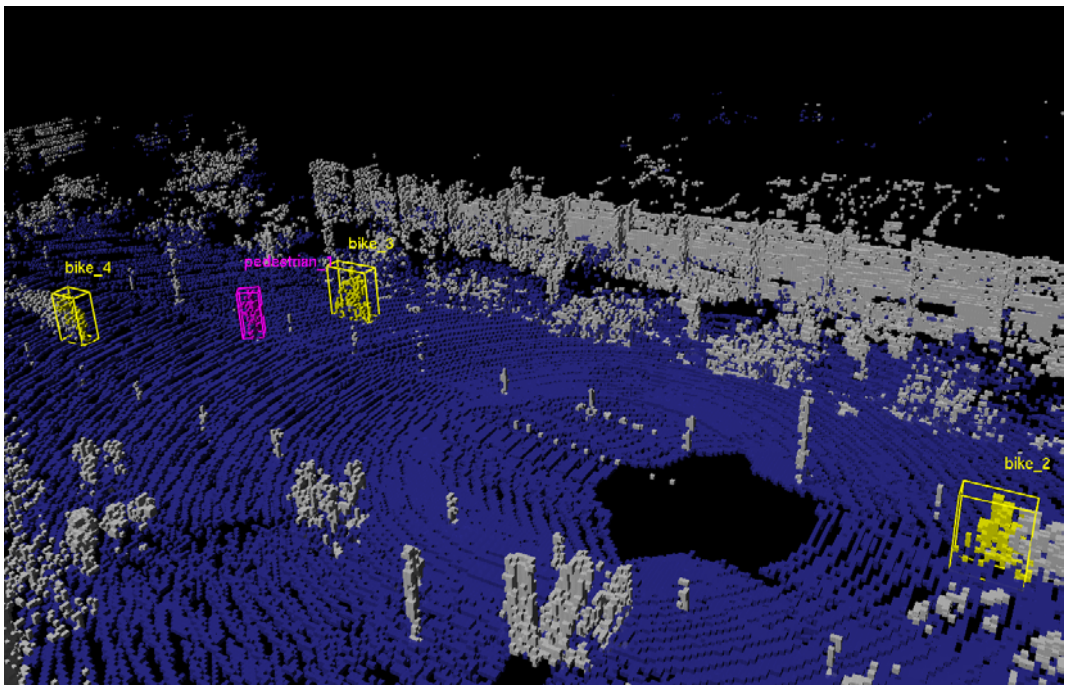


(b) Tracking results at $t = 0.6$ s.

Figure 4.23: Tracking results at $t = 0.6$ s. There are three moving objects correctly detected and classified as two bikes and a pedestrian.

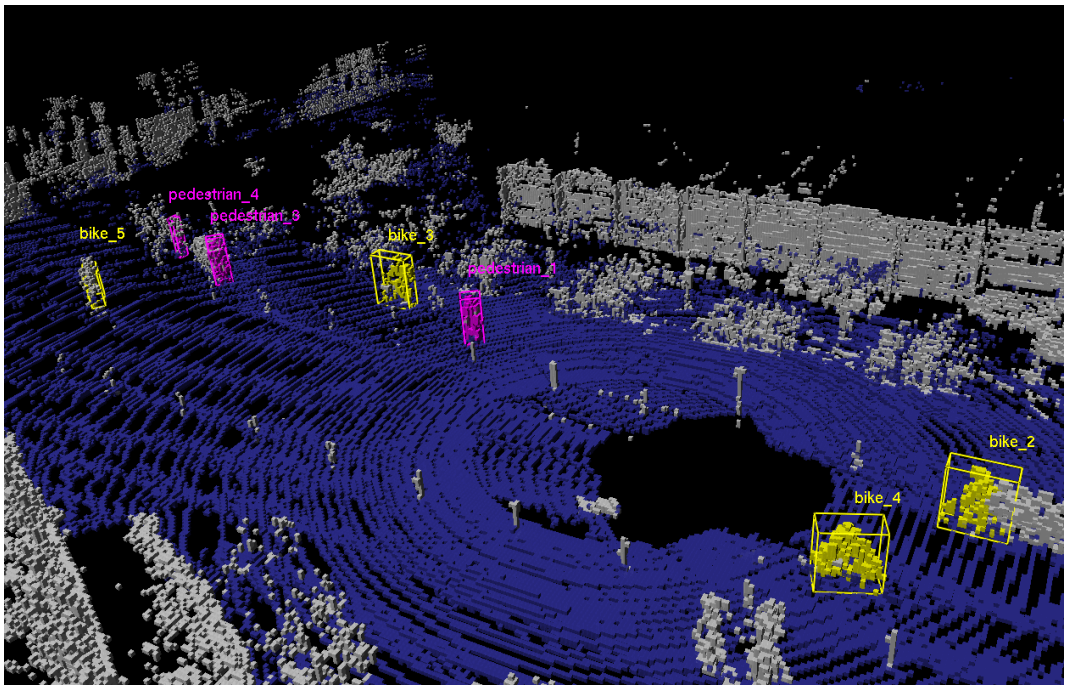


(a) Tracking results at $t = 1.0$ s.

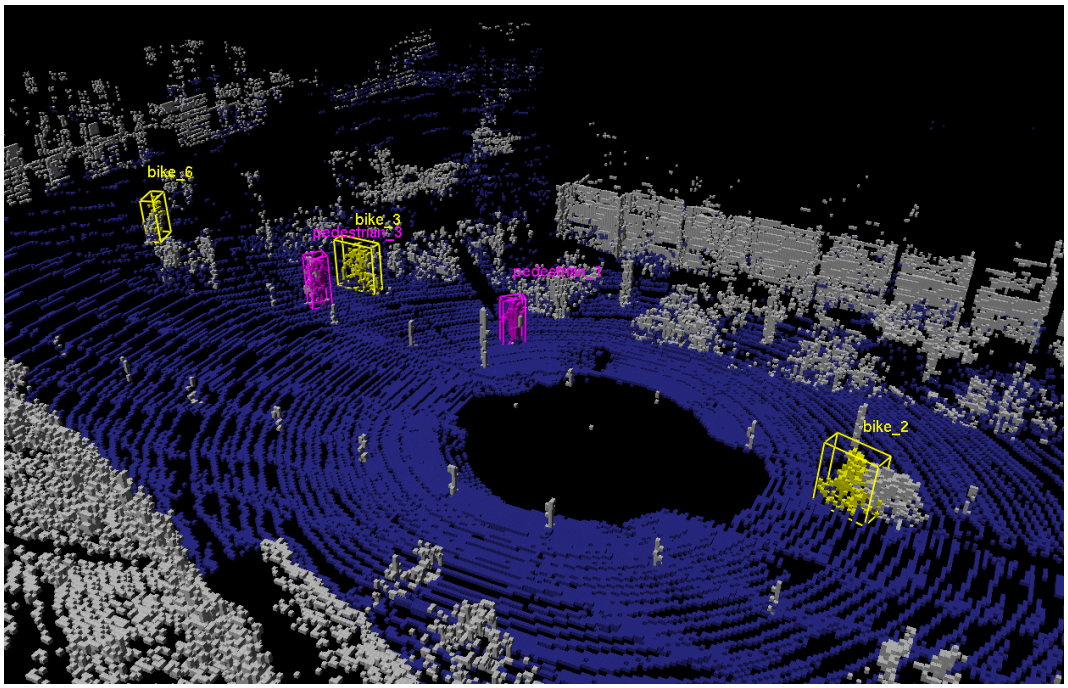


(b) Tracking results at $t = 2.4$ s.

Figure 4.24: Tracking results continued from Fig. 4.23 at $t = 1.0$ s and $t = 2.4$ s. Another object (*bike_4*) is detected and tracked in addition to the previously tracked objects.

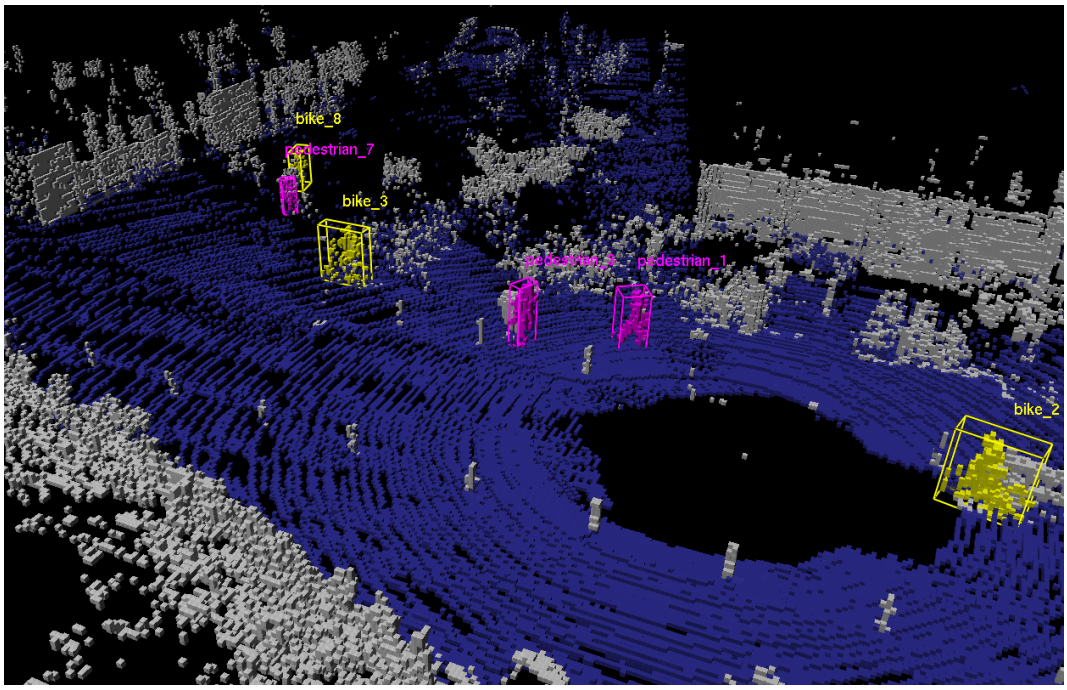


(a) Tracking results at $t = 7.2$ s.

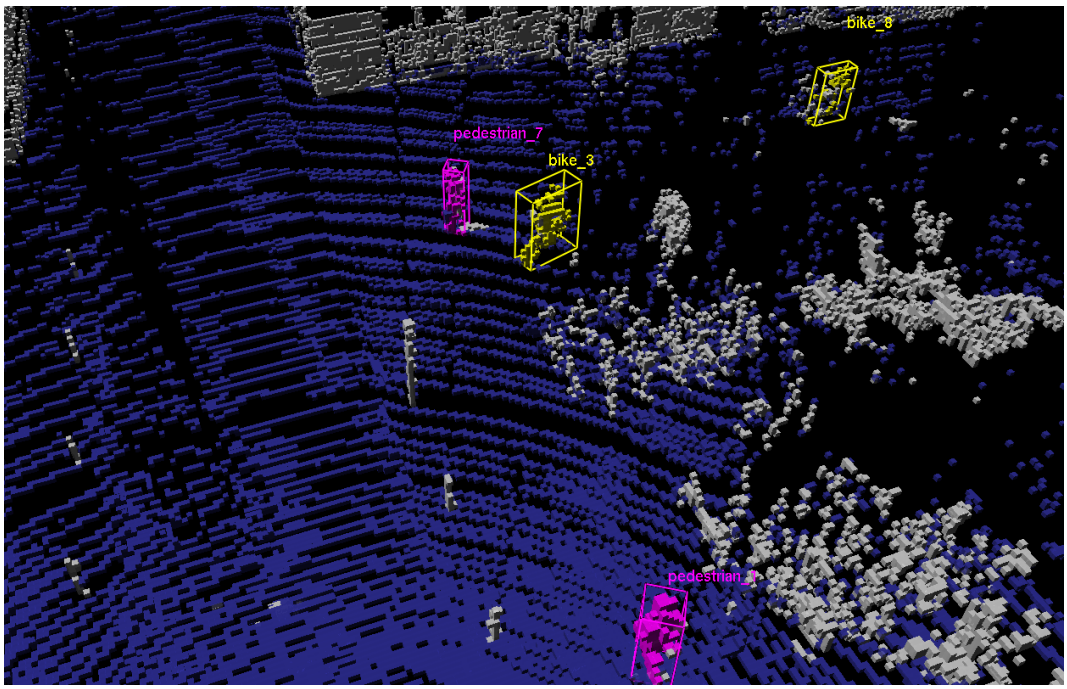


(b) Tracking results at $t = 10.2$ s.

Figure 4.25: Tracking results continued from Fig. 4.23 at $t = 7.2$ s and $t = 10.2$ s. This shows the case of a false alarm generated as *pedestrian_4* (Fig. 4.25a) and a new track (*bike_6*) initialized for an existing object (*bike_5*) due to a sequence of missed detections (Fig. 4.25b).

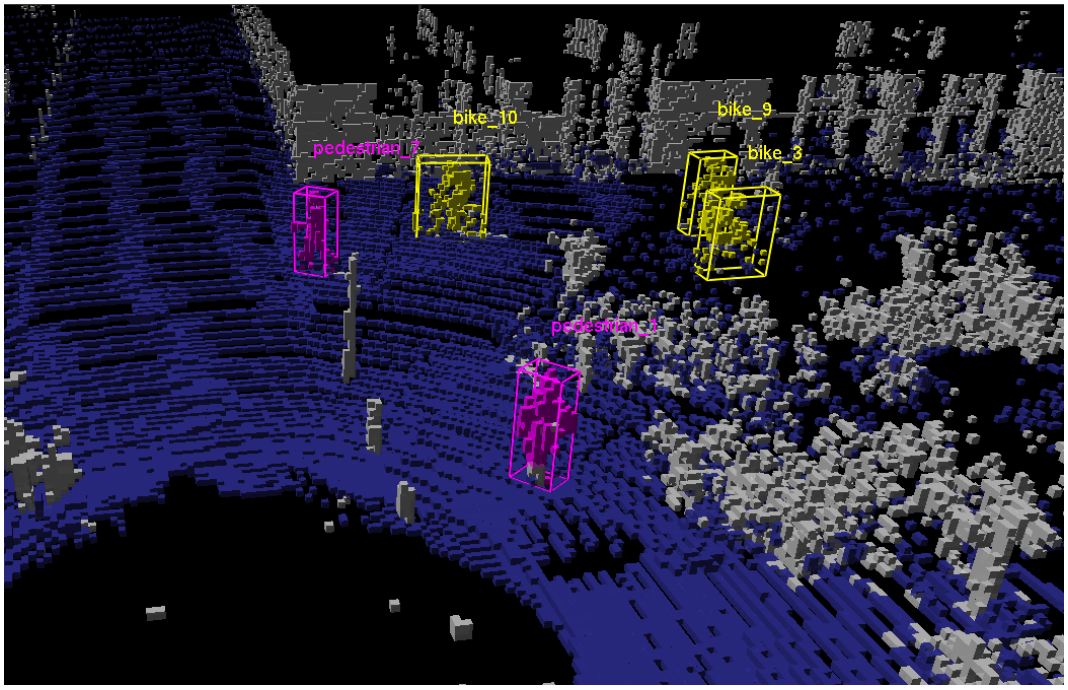


(a) Tracking results at $t = 13.4$ s.

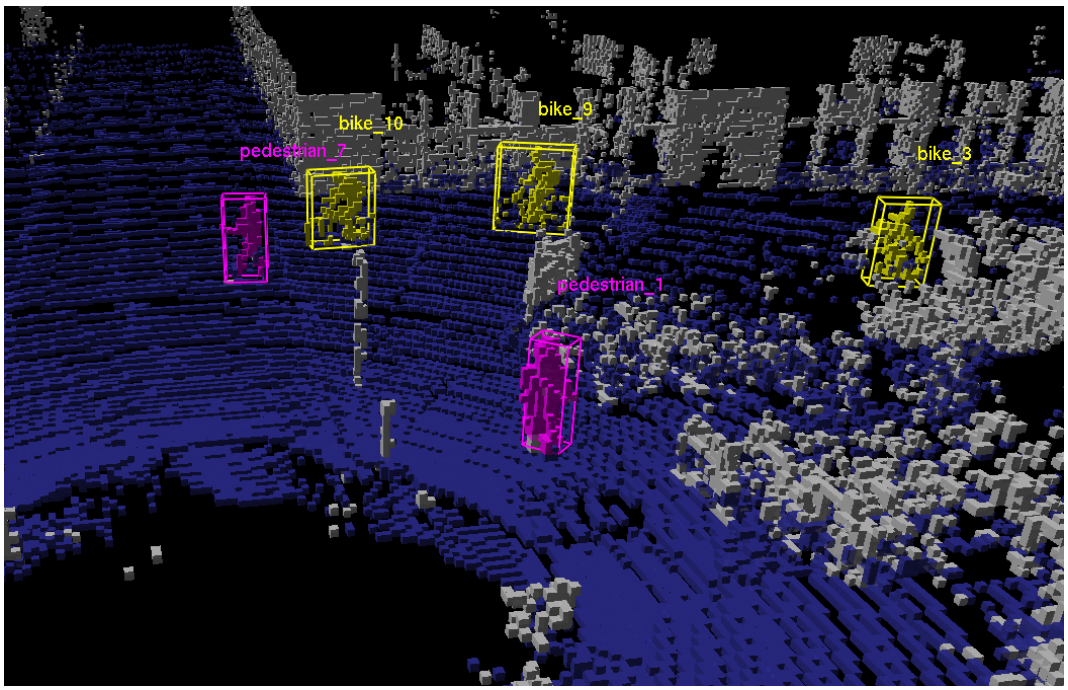


(b) Tracking results at 16.8 s.

Figure 4.26: Tracking results continued from Fig. 4.23 at $t = 13.4$ s and $t = 16.8$ s.



(a) Tracking results at $t = 20.0$ s.



(b) Tracking results at 21.8 s.

Figure 4.27: Tracking results continued from Fig. 4.23 at $t = 20.0$ s and $t = 21.8$ s.

Table 4.9: Multiple object tracking: Quantitative results

Type of Dataset	Total Objects	Non-detections	False Alarms	Total Tracks
Highway	17	162	53	58
Urban	54	341	132	74
Pedestrian Zone	67	284	156	93

used the sequences of data captured from the three types of environment: highway, urban and pedestrian zone.

Table 4.9 illustrates the quantitative results we obtained for tracking of moving objects in three different scenarios listed in the first column. The second column gives the number of distinct objects present in the environment throughout the sequence of data. This number was calculated manually using the images corresponding to each sequence of data as well as the estimations from the previous positions of the objects. The third column gives the number of frames in which an object was not detected but still tracked by our algorithm. It corresponds to the number of times a case of *non-detection* occurred which was correctly handled by our method of moving object tracking. These non-detections could be caused by the failure of the sensor, detection module or classification module. The fourth column gives the number of false alarms detected by our algorithm. These false alarms are again generated by the detection or classification module where a non-dynamic object is wrongly identified as a dynamic object. In our tracking method, a false alarm which does not appear sufficient number of times is not confirmed as a track therefore it is not identified as a dynamic object. The last column gives the total number of tracks generated during the complete execution of the system for the sequence of data.

The results illustrate that most of the dynamic objects are correctly tracked by our method along with handling a large number of non-detections and false alarms generated during the previous steps. We observe that the total number of the tracks is greater than the total number of objects present in the environment. This phenomenon can be explained by the fact that the objects often leave and re-enter the range of the sensor thus giving rise to the creation of new tracks. These results clearly show the importance of a reliable tracking system which can handle the failure of the sensor as well as detection and classification systems for the reliable perception of the dynamic environment.

Table 4.10: Processing times for different components

Component	Processing Time
Mapping	5.35 s
Ground Segmentation	1.87 s
ICP	3.21 s
Moving Object Detection	0.11 s
Classification	0.17 s
Tracking	0.13 s

Our system for perception was tested on a computer with a processor of 2.1GHz running a Linux operating system. We used several sequences of data from urban, highway and pedestrian zone scenarios. The system generally gave good results also in difficult scenarios with many objects in the surrounding. Average processing times per frame for each component of the system are given in Table 4.10. As it can be seen that the average processing time for generating the simple octree-based 3D occupancy grid map is 5.35 seconds. Whereas, the time required for ground segmentation and ICP localization is 1.87 seconds and 3.21 seconds respectively. In comparison to that the average computation times for detection (with clustering), classification and tracking are 0.11, 0.17 and 0.13 seconds respectively. This shows that the average time required for DATMO in each frame is around 0.41 seconds which is, though not real time in the context of a sensor with scanning frequency of 10Hz, but still very promising. Moreover, optimizing and parallelizing the code with the use of advanced and efficient hardware can improve the processing speed considerably.

4.5 Conclusion

In this chapter we detailed the implementation of our system for perception in outdoor environments and presented the experimental results obtained by it. In order to test our system, we used different publicly available datasets obtained from the experimental vehicles equipped with 3D Velodyne laser scanner, driving in different environments. For testing our method for SLAM with ground segmentation, we used two complex datasets and obtained good results. For each new scan, the first step was to convert it into octree-based occupancy grid representation. Then we used our method for ground segmentation to differentiate between the voxels

corresponding to ground and non-ground objects. An important observation in this context was the huge amount of data corresponding to the ground. After ground segmentation, we used the non-ground voxels for ICP based localization. We notice that the data used for ICP at this step is a lot less than the original amount of data acquired at each step. We experimented with different parameters and selected the appropriate ones based on the results obtained.

For evaluating our method for detection, classification and tracking of moving objects, we used several datasets gathered from different environments including urban, highway and pedestrian zones and obtained good results even in complex scenarios. The amount of noise generated by the sensor was a big challenge. However, our method for detection and density-based clustering of dynamic objects performed well in terms of high detection rate and low false alarms. Most of the false alarms which still existed were handled by our supervised classification module. An additional contribution of our work that we detailed in this chapter is the labeled dataset consisting of different dynamic objects that we generated for the supervised training of classifiers. We trained four classifiers corresponding to each object class of interest namely; *bus*, *car*, *bike* and *pedestrian*. The classifiers were trained offline using simple features and AdaBoost. The trained classifiers were then used online in our system for the classification of the detected moving objects. At the end, we presented some qualitative results for detection, classification and tracking of moving objects from different scenarios.

The results obtained from application of the proposed technique on real data collected from different scenarios show a reliable environment representation with good segmentation between the non-ground objects and ground. The segmentation is accurate enough to be used for clustering and classification of the static objects which was out of scope of this work. Moreover, the moving objects are clearly and efficiently segmented from the static environment despite a considerable amount of noise. The spurious elements are further minimized by use of classification and tracking. This results in a robust and reliable representation of the dynamic outdoor environment in 3D.

Chapter 5

Conclusion and Perspectives

5.1 Conclusion

This thesis addressed the problem of vehicle perception in the context of outdoor dynamic environments. Environment perception plays a key role in an intelligent system since any error in the perceived information can drastically affect the performance and security of the whole system. The aim of this work was to provide an intelligent vehicle with a robust and reliable perception of the surroundings built upon the recently developed sensor technology. A survey of the literature revealed that many of the recent works in this context have relied upon the fusion of information from multiple sensors to achieve this goal. However, we posit that with the recent advancement in sensor technology, a single 3D laser scanner is sufficient to be used as an external sensor for environment perception. Based on this hypothesis, we presented a perception framework addressing the core tasks of the problem including simultaneous localization and mapping (SLAM) with detection and tracking of moving objects (DATMO). The proposed framework is tested through various off-line datasets acquired by the 3D laser scanner mounted on the vehicles driven in different outdoor scenarios to demonstrate its applicability and performance in real environments.

In chapter 2, after reviewing the existing solutions in the literature for simultaneous localization and mapping, we proposed an algorithm for SLAM in 3D. Our method was built on an optimized octree-based occupancy grid mapping framework. We preferred the grid-based environment representation due to its compactness and efficiency as well as the possibility to detect dynamic objects. As in occupancy

grid mapping, the free areas of the environment are explicitly modeled therefore the dynamic objects can be detected by observing the occupancy of the previously free areas. While using a 3D laser scanner, an important observation is that a large fraction of the data corresponds to the ground. Therefore, ground segmentation was an important step in our approach before the subsequent processing of the information. We used a cell-based method with variance and mean of the heights as the main features for identifying the ground voxels. After ground segmentation, we performed the incremental ICP based scan matching by using the odometry for the initial estimate of transition. The performance of ICP is improved by two methods. First, we use the octree occupancy grid for scan matching which is a sub-sampled version of the original point cloud. Further, after segmenting the ground voxels, the remaining voxels (which serve as the point cloud for scan matching) are again reduced by a large factor. Second, we use a fast approximation of the point correspondences in ICP algorithm by using kd-tree for approximate nearest neighbor search.

Based on the results of chapter 2, we presented a framework for detection, classification and tracking of moving objects in chapter 3 which is the most significant contribution of this work. Our method for motion-based detection of dynamic objects reliably identifies the possible moving objects without a prior knowledge of the objects. However, due to the sensor uncertainty and huge amount of data generated by a 3D laser scanner, the amount of noise and thus the wrongly detected dynamic objects is also huge. Thus we perform the supervised classification of the detected objects to differentiate the real objects. In comparison to the other approaches for object classification, our method does not require the pre-defined routines to extract high-level features. Instead, we extract the single-valued features from each layer and use the AdaBoost algorithm to boost these simple features to strong classifiers for object identification. Moreover, using a supervised learning method results in assigning the semantic labels to the objects corresponding to the pre-defined classes of road users with known dynamics. As the number of moving object classes appearing on the roads are quite limited therefore the supervised approach proves feasible in this context. The classification of dynamic objects further helps in the process of tracking for estimating the positions of these objects with time. For multiple object data association, we used Viterbi algorithm which overcomes the problem of abrupt changes in motion (for example in case of pedestrians) as well as temporary occlusions by taking advantage of dynamic programming.

The proposed framework was evaluated on the datasets obtained from different outdoor scenarios using a 3D laser scanner as described in chapter 4. The results obtained from these evaluations show a reliable performance in different circumstances. The ground segmentation and scan matching was shown to yield a good performance with a considerable improvement in the localization in comparison to the vehicle odometry. The dynamic object detection, classification and tracking was shown to work reliably even in complex urban scenarios.

In summary, this thesis has addressed a relatively new problem of 3D perception in outdoor dynamic environments and we have demonstrated that we can reliably perform simultaneous localization and mapping with detection, classification and tracking of moving objects from a ground vehicle equipped with a single 3D laser scanner. Despite its implementation with ground vehicles in this work, the proposed approach can also be used in other areas and we hope that the obtained results will serve as potential basis for a wide range of robotic applications.

5.2 Perspectives

As mentioned before, perception with 3D laser scanners is a new area of research and it has several open questions that need to be addressed. This thesis has focused on some of those questions, however, there are many possible extensions for enhancing the results and taking the framework further.

One potential extension concerns the simultaneous localization and mapping approach: We used odometry as the initial estimate for transition while registering the new scan with existing map, however, using a heuristic method for calculating the initial transition estimate in ICP can lead to a more robust and reliable localization in the situations where the odometry is too erroneous to be used. Further, the results of ground segmentation provide a good segmentation of the static objects as illustrated in Fig. 5.1. These segmented objects can be clustered and classified into known categories such as trees, bush, walls, light poles, traffic signs etc. Moreover, these clustered objects can be used for object based scan alignment for an improved localization of the vehicle.

Another future line of research can focus on improving the performance of detection, classification and tracking of moving objects algorithm by integrating the road border detection. The results of this step can be used as a prior information about

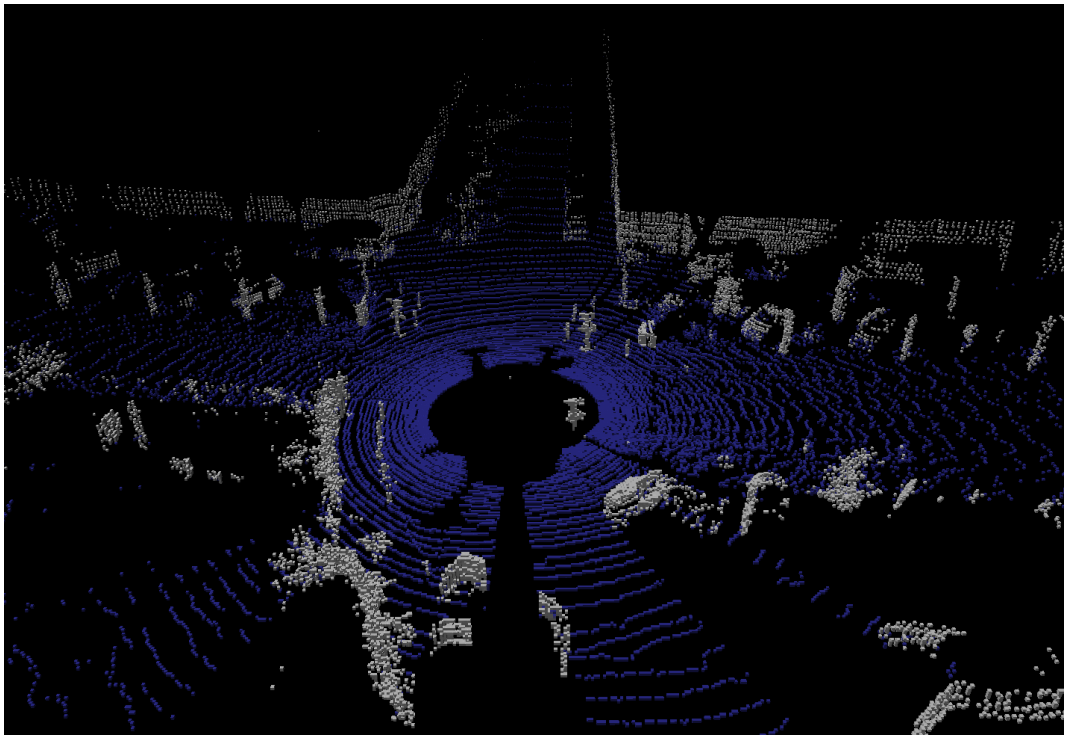


Figure 5.1: An illustration of object segmentation based on ground identification. Different static objects such as light poles, sign boards, traffic signs, façades of the buildings and parked cars are separated from each other. A proximity based clustering of the occupied voxels can result in the individual clusters corresponding to these objects which can be used for classification as well as object based localization.

different regions of the environment which will constrain the appearance and motion estimation of the dynamic objects. The possible constraints can be such as the dynamic objects should only be detected on or within a threshold of the road borders (*appearance constraint*) or the cars can only move along the direction of the road (*motion constraint*). Using these constraints, the number of dynamic object hypotheses can be reduced significantly thus improving the quality and efficiency of the tracking algorithm. An alternative or complementary option to road border detection can be the use of the navigation maps. The performance of tracking algorithm can also be improved by modeling the interactions between the dynamic objects.

On a separate note, the current implementation of the proposed framework is not real-time with a frame rate of upto 10 Hz therefore a possible line of future work can address the problem of attaining a real-time implementation. The possible solutions include the use of general-purpose graphical processing units (GPGPU) as well as the multi-core CPUs.

References

Adams Martin, Zhang Sen, and Xie Lihua. Particle filter based outdoor robot localization using natural features extracted from laser scanners. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 2, pages 1493–1498. IEEE, 2004.

Amanatides John, Woo Andrew, *et al.* A fast voxel traversal algorithm for ray tracing. In *Proceedings of EUROGRAPHICS*, volume 87, pages 3–10, 1987.

Anderson Brian D. and Moore John B. *Optimal filtering*. Prentic-Hall, Englewood Cliffs, New Jersey, 1979.

Anguelov D., Taskarf B., Chatalbashev V., Koller D., Gupta D., Heitz G., and Ng A. Discriminative learning of markov random fields for segmentation of 3d scan data. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 169–176 vol. 2, 2005. doi: 10.1109/CVPR.2005.133.

Arras Kai Oliver, Mozos O Martınez, and Burgard Wolfram. Using boosted features for the detection of people in 2d range data. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3402–3407. IEEE, 2007.

Arulampalam M Sanjeev, Maskell Simon, Gordon Neil, and Clapp Tim. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Signal Processing, IEEE Transactions on*, 50(2):174–188, 2002.

Arun K.S., Huang T. S., and Blostein S. D. Least-squares fitting of two 3-d point sets. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-9(5): 698–700, 1987.

Arya S and Mount D. Ann: library for approximate nearest neighbor searching. In *2nd CGC Workshop on Computational Geometry*, 1997.

Azim Asma and Aycard Olivier. Multiple pedestrian tracking using viterbi data association. In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pages 706–711. IEEE, 2010.

Azim Asma and Aycard Olivier. Detection, classification and tracking of moving objects in a 3d environment. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 802–807. IEEE, 2012.

B. Douillard J. Underwood N. Melkumyan S. Singh S. Vasudevan C. Brunner A. Quardros. Hybrid elevation maps: 3d surface models for segmentation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1532–1538, 2010.

Baig Qadeer. *Multisensor Data Fusion for Detection and Tracking of Moving Objects From a Dynamic Autonomous Vehicle*. PhD thesis, PhD thesis, University of Grenoble, 2012.

Bailey Tim. *Mobile robot localisation and mapping in extensive outdoor environments*. PhD thesis, The University of Sydney, 2002.

Bar-Shalom Y, Li X Rong, and Kirubarajan T. Estimation with applications to tracking and navigation, 2001.

Bar-Shalom Yaakov and Fortmann Thomas E. *Tracking and data association*. Academic Press, 1988.

Baum L. E. and Petrie T. Statistical inference for probabilistic functions of finite state markov chains. *The Annals of Mathematical Statistics*, Vol. 37:1554–1563, 1966.

Bertozzi Massimo, Broggi Alberto, and Fascioli Alessandra. Vision-based intelligent vehicles: State of the art and perspectives. *Robotics and Autonomous systems*, 32(1):1–16, 2000.

Besl Paul J and McKay Neil D. Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics, 1992.

Blackman S and Popoli R. Design and analysis of modern tracking systems. *Artech House, Norwood, MA*, 1999.

Blackman Samuel S. Multiple hypothesis tracking for multiple target tracking. *Aerospace and Electronic Systems Magazine, IEEE*, 19(1):5–18, 2004.

Borrmann Dorit, Elseberg Jan, Lingemann Kai, Nüchter Andreas, and Hertzberg Joachim. Globally consistent 3d mapping with scan matching. *Robotics and Autonomous Systems*, 56(2):130–142, 2008.

Boser Bernhard E, Guyon Isabelle M, and Vapnik Vladimir N. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

Botsch Mario, Wiratanaya Andreas, and Kobbelt Leif. Efficient high quality rendering of point sampled geometry. In *Proceedings of the 13th Eurographics workshop on Rendering*, pages 53–64. Eurographics Association, 2002.

Brechtel Sebastian, Gindele Tobias, and Dillmann Rüdiger. Recursive importance sampling for efficient grid-based occupancy filtering in dynamic environments. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3932–3938. IEEE, 2010.

Brenner C, Dold C, and Ripperda N. Coarse orientation of terrestrial laser scans in urban environments. *ISPRS journal of photogrammetry and remote sensing*, 63(1): 4–18, 2008.

Brown James Anthony and Capson David W. A framework for 3d model-based visual tracking using a gpu-accelerated particle filter. *Visualization and Computer Graphics, IEEE Transactions on*, 18(1):68–80, 2012.

Burgard Wolfram, Cremers Armin B, Fox Dieter, Hähnel Dirk, Lakemeyer Gerhard, Schulz Dirk, Steiner Walter, and Thrun Sebastian. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114:3–55, 2000.

Burlet Julien, Vu Trung Dung, Aycard Olivier, *et al.* Grid-based localization and online mapping with moving object detection and tracking. Technical report, INRIA-UJF, 2007.

Chavez-Garcia R Omar, Burlet Julien, Vu Trung-Dung, and Aycard Olivier. Frontal object perception using radar and mono-vision. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 159–164. IEEE, 2012.

Choset Howie and Nagatani Keiji. Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization. *Robotics and Automation, IEEE Transactions on*, 17(2):125–137, 2001.

Clodic A. Fleury S. Alami R. Herrb M. & Chatila R. Supervision and interaction: Analysis from an autonomous tour-guide robot deployment. In *International Conference on Advanced Robotics, Seattle, USA*, 2005.

Cole David M and Newman Paul M. Using laser range data for 3d slam in outdoor environments. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1556–1563. IEEE, 2006.

Csorba Michael. *Simultaneous localisation and map building*. PhD thesis, University of Oxford, 1997.

Darms Michael, Rybski Paul, and Urmson Chris. Classification and tracking of dynamic objects with multiple sensors for autonomous driving in urban environments. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 1197–1202. IEEE, 2008.

Davey Samuel J., Rutten Mark G., and Cheung Brian. A comparison of detection performance for several track-before-detect algorithms. *EURASIP Journal on Advances in Signal Process*, 2008, January 2008. ISSN 1110-8657. doi: 10.1155/2008/428036. URL `http://dx.doi.org/10.1155/2008/428036`.

del Blanco Carlos R, Jaureguizar Fernando, and García Narciso. An advanced bayesian model for the visual tracking of multiple interacting objects. *EURASIP Journal on Advances in Signal Processing*, 2011(1):1–13, 2011.

Dietmayer Klaus CJ, Sparbert Jan, and Streller Daniel. Model based object classification and object tracking in traffic scenes from range images. *Proceedings of IV*, pages 2–1, 2001.

Dissanayake MWM Gamini, Newman Paul, Clark Steven, Durrant-Whyte Hugh F, and Csorba Michael. A solution to the simultaneous localization and map building (slam) problem. *Robotics and Automation, IEEE Transactions on*, 17(3):229–241, 2001.

Douillard Bertrand, Underwood J, Kuntz N, Vlaskine V, Quadros A, Morton P, and Frenkel A. On the segmentation of 3d lidar point clouds. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2798–2805. IEEE, 2011.

Durrant-Whyte Hugh F, Rao BYS, and Hu H. Toward a fully decentralized architecture for multi-sensor data fusion. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 1331–1336. IEEE, 1990.

Eggert David W, Lorusso Adele, and Fisher Robert B. Estimating 3-d rigid body transformations: a comparison of four major algorithms. *Machine Vision and Applications*, 9(5-6):272–290, 1997.

Elfes Alberto. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

Elfes Alberto. Multi-source spatial data fusion using bayesian reasoning. *Data fusion in robotics and machine intelligence*, -:137–163, 1992.

Eliazar Austin and Parr Ronald. Dp-slam: Fast, robust simultaneous localization and mapping without predetermined landmarks. In *International Joint Conference on Artificial Intelligence*, volume 18, pages 1135–1142. LAWRENCE ERLBAUM ASSOCIATES LTD, 2003.

Elseberg Jan, Borrmann Dorit, and Nüchter Andreas. Efficient processing of large 3d point clouds. In *Information, Communication and Automation Technologies (ICAT), 2011 XXIII International Symposium on*, pages 1–7. IEEE, 2011.

Ess Andreas, Schindler Konrad, Leibe Bastian, and Van Gool Luc. Object detection and tracking for autonomous navigation in dynamic environments. *The International Journal of Robotics Research*, 29(14):1707–1725, 2010.

Ester Martin, Kriegel Hans-Peter, Sander Jörg, and Xu Xiaowei. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231, 1996.

Fairfield Nathaniel, Kantor George, and Wettergreen David. Real-time slam with octree evidence grids for exploration in underwater tunnels. *Journal of Field Robotics*, 24(1-2):03–21, 2007.

Fairfield Nathaniel, Wettergreen David, and Kantor George. Segmented slam in three-dimensional environments. *Journal of Field Robotics*, 27(1):85–103, 2010.

Fardi Basel, Dousa Jaroslav, Wanielik Gerd, Elias Bjorn, and Barke Alexander. Obstacle detection and pedestrian recognition using a 3d pmd camera. In *Intelligent Vehicles Symposium, 2006 IEEE*, pages 225–230. IEEE, 2006.

Fayad Fadi and Cherfaoui Veronique. Tracking objects using a laser scanner in driving situation based on modeling target shape. In *Intelligent Vehicles Symposium, 2007 IEEE*, pages 44–49. IEEE, 2007.

Forney Jr G David. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

Fortmann Thomas E and Baron Sheldon. Problems in multi-target sonar tracking. In *Decision and Control including the 17th Symposium on Adaptive Processes, 1978 IEEE Conference on*, volume 17, pages 1182–1188. IEEE, 1978.

Fournier Jonathan, Ricard Benoit, and Laurendeau Denis. Mapping and exploration of complex environments using persistent 3d model. In *Computer and Robot Vision, 2007. CRV'07. Fourth Canadian Conference on*, pages 403–410. IEEE, 2007.

Frank Oliver, Nieto Juan, Guivant Jose, and Scheding Steve. Multiple target tracking using sequential monte carlo methods and statistical data association. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 3, pages 2718–2723. IEEE, 2003.

Freund Yoav and Schapire Robert E. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37. Springer, 1995.

Gad Ahmed and Farooq Mohammad. Viterbi-based data association techniques for target tracking. In *Proceedings of SPIE*, volume 5096, pages 37–46, 2003.

Gambino Fabio, Ulivi Giovanni, and Vendittelli Marilena. The transferable belief model in ultrasonic map building. In *Fuzzy Systems, 1997., Proceedings of the Sixth IEEE International Conference on*, volume 1, pages 601–608. IEEE, 1997.

Garulli Andrea, Giannitrapani Antonio, Rossi Andrea, and Vicino Antonio. Mobile robot slam for line-based environment representation. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 2041–2046. IEEE, 2005.

Geiger Andreas, Lenz Philip, and Urtasun Raquel. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and PatternRecognition (CVPR)*, 2012.

Gordon Neil J, Salmond David J, and Smith Adrian FM. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)*, volume 140, pages 107–113. IET, 1993.

Grisettiyz G, Stachniss Cyrill, and Burgard Wolfram. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2432–2437. IEEE, 2005.

Gutmann J-S and Konolige Kurt. Incremental mapping of large cyclic environments. In *Computational Intelligence in Robotics and Automation, 1999. CIRA'99. Proceedings. 1999 IEEE International Symposium on*, pages 318–325. IEEE, 1999.

Hahnel Dirk, Schulz Dirk, and Burgard Wolfram. Map building with mobile robots in populated environments. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pages 496–501. IEEE, 2002.

Hall David L and Garga Amulya K. Pitfalls in data fusion (and how to avoid them). In *Proceedings of the Second International Conference on Information Fusion (FusionâĂŹ99)*, volume 1, pages 429–436, 1999.

Harris Chris and Stephens Mike. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

Hellaker Jan. Prometheus: Strategy. *ITS technology collection on CD-ROM: SAE's essential resource for ITS vehicle applications, 1998*, 1990.

Himmelsbach M. Hundelshausen F.V. ; Wuensche H. Fast segmentation of 3d point clouds for ground vehicles. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 560–565, 2010.

Himmelsbach Michael, Mueller Andre, Luettel Thorsten, and Wuensche Hans-Joachim. LIDAR-based 3D Object Perception. In *Proceedings of 1st International Workshop on Cognition for Technical Systems*, Munich, October 2008.

Horn Berthold KP. Closed-form solution of absolute orientation using unit quaternions. *JOSA A*, 4(4):629–642, 1987.

Horn Berthold KP, Hilden Hugh M, and Negahdaripour Shahriar. Closed-form solution of absolute orientation using orthonormal matrices. *JOSA A*, 5(7):1127–1135, 1988.

Huang Hui-Min, Pavek Kerry, Albus James, and Messina Elena. Autonomy levels for unmanned systems (alfus) framework: an update. In *Proceedings of the 2005 SPIE Defense and Security Symposium*, pages 439–448, 2005.

Jensen Björn, Tomatis Nicola, Mayor Laetitia, Drygajlo Andrzej, and Siegwart Roland. Robots meet humans-interaction in public spaces. *Industrial Electronics, IEEE Transactions on*, 52(6):1530–1546, 2005.

Kalman Rudolph Emil *et al.* A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

Kammerl Julius, Blodow Nico, Rusu Radu Bogdan, Gedikli Suat, Beetz Michael, and Steinbach Eckehard. Real-time compression of point cloud streams. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 778–785. IEEE, 2012.

Katz Roman, Douillard Bertrand, Nieto Juan, and Nebot Eduardo. A self-supervised architecture for moving obstacles classification. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 155–160. IEEE, 2008.

Khan Zia, Balch Tucker, and Dellaert Frank. Mcmc-based particle filtering for tracking a variable number of interacting targets. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(11):1805–1819, 2005.

King Bradford J, Malisiewicz Tomasz, Stewart Charles V, and Radke Richard J. Registration of multiple range scans as a location recognition problem: Hypothesis generation, refinement and verification. In *3-D Digital Imaging and Modeling, 2005. 3DIM 2005. Fifth International Conference on*, pages 180–187. IEEE, 2005.

Kluge Boris, Kohler Christian, and Prassler Erwin. Fast and robust tracking of multiple moving objects with a laser range finder. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 1683–1688. IEEE, 2001.

Koller Jost and Ulmke Martin. Multi hypothesis tracking of ground moving targets. *GI Jahrestagung (2)*, 68:307–311, 2005.

Konolige Kurt. Improved occupancy grids for map building. *Autonomous Robots*, 4 (4):351–367, 1997.

Krishna K Madhava and Kalra Prem K. When does a robot perceive a dynamic object? *Journal of Robotic Systems*, 19(2):73–90, 2002.

Labayrade Raphaël, Royere Cyril, Gruyer Dominique, and Aubert Didier. Cooperative fusion for multi-obstacles detection with use of stereovision and laser scanner. *Auton. Robots*, 19(2):117–140, September 2005.

Lai Kevin and Fox Dieter. Object recognition in 3d point clouds using web data and domain adaptation. *The International Journal of Robotics Research*, 29(8):1019–1037, 2010.

Laine Samuli and Karras Tero. Efficient sparse voxel octrees. *Visualization and Computer Graphics, IEEE Transactions on*, 17(8):1048–1059, 2011.

Lam Joseph, Kusevic Kresimir, Mrstik R, Harrap P, and Greenspan Michael. Urban scene extraction from mobile ground based lidar data. In *Proc. 3DPVT*, volume 2010, pages 478–486, 2011.

Lau Boris, Arras Kai O, and Burgard Wolfram. Multi-model hypothesis group tracking and group size estimation. *International Journal of Social Robotics*, 2(1): 19–30, 2010.

Lee Sang-Mook, Im Jeong Joon, Lee Bo-Hee, Leonessa A., and Kurdila A. A real-time grid map generation and object classification for ground-based 3d lidar data using image analysis techniques. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 2253 –2256, sept. 2010.

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

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

Leonard John J and Durrant-Whyte Hugh F. Simultaneous map building and localization for an autonomous mobile robot. In *Intelligent Robots and Systems' 91.'Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on*, pages 1442–1447. Ieee, 1991b.

Lindstrom M and Eklundh J-O. Detecting and tracking moving objects from a mobile platform using a laser range scanner. In *Intelligent Robots and Systems, 2001.*

Proceedings. 2001 IEEE/RSJ International Conference on, volume 3, pages 1364–1369. IEEE, 2001.

Lowe David G. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

Lu Feng and Milios Evangelos. Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic Systems*, 18(3):249–275, 1997.

Lundquist C Schon T. Road geometry estimation and vehicle tracking using a single track model. In *IEEE Intelligent vehicles symposium (IV)*, 2008.

MacKay David JC. Introduction to monte carlo methods. In *Learning in graphical models*, pages 175–204. Springer, 1998.

Meagher Donald. Geometric modeling using octree encoding. *Computer graphics and image processing*, 19(2):129–147, 1982.

Mendes Abel, Bento Luis Conde, and Nunes Urbano. Multi-target detection and tracking with a laser scanner. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 796–801. IEEE, 2004.

Montemerlo Michael, Thrun Sebastian, and Siciliano Bruno. *FastSLAM: A scalable method for the simultaneous localization and mapping problem in robotics*, volume 27. Springer Verlag, 2007.

Montemerlo Michael, Becker Jan, Bhat Suhrid, Dahlkamp Hendrik, Dolgov Dmitri, Ettinger Scott, Haehnel Dirk, Hilden Tim, Hoffmann Gabe, Huhnke Burkhard, Johnston Doug, Klumpp Stefan, Langer Dirk, Levandowski Anthony, Levinson Jesse, Marcil Julien, Orenstein David, Paefgen Johannes, Penny Isaac, Petrovskaya Anna, Pflueger Mike, Stanek Ganymed, Stavens David, Vogt Antone, and Thrun Sebastian. Junior: The stanford entry in the urban challenge. *J. Field Robot.*, 25(9):569–597, September 2008.

Moosmann Frank and Fraichard Thierry. Motion estimation from range images in dynamic outdoor scenes. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 142–147. IEEE, 2010.

Moosmann Frank and Sauerland Miro. Unsupervised discovery of object classes in 3d outdoor scenarios. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1038–1044. IEEE, 2011.

Moosmann Frank and Stiller Christoph. Velodyne slam. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 393–398. IEEE, 2011.

Moravec Hans P. Robot spatial perceptionby stereoscopic vision and 3d evidence grids. Technical report, The Robotics Institute, Carnegie Mellon University, Pittsburgh (PA), 1996.

Morris Daniel, Hoffman Regis, and McLean Steve. Ladar-based vehicle detection and tracking in cluttered environments. Technical report, DTIC Document, 2008.

Moutarlier Philippe and Chatila Raja. An experimental system for incremental environment modelling by an autonomous mobile robot. In *Experimental Robotics I*, pages 327–346. Springer, 1990.

Murphy Kevin. Bayesian map learning in dynamic environments. *Advances in Neural Information Processing Systems (NIPS)*, 12:1015–1021, 1999.

Nashashibi F. and Bargeton A. Laser-based vehicles tracking and classification using occlusion reasoning and confidence estimation. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 847 –852, june 2008.

Nedevschi Sergiu, Tiberiu Marita, Danescu Radu, Oniga Florin, and Bota Silviu. On-board stereo sensor for intersection driving assistance architecture and specification. In *Intelligent Computer Communication and Processing, 2009. ICCP 2009. IEEE 5th International Conference on*, pages 409–416. IEEE, 2009.

Needham Joseph. *Science and Civilisation in China: Volume 2, History of Scientific Thought*. Cambridge University Press, 1991.

Nguyen Viet, Martinelli Agostino, Tomatis Nicola, and Siegwart Roland. A comparison of line extraction algorithms using 2d laser rangefinder for indoor mobile robotics. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 1929–1934. IEEE, 2005.

Noykov Sv and Roumenin Ch. Occupancy grids building by sonar and mobile robot. *Robotics and autonomous systems*, 55(2):162–175, 2007.

Nüchter A, Lingemann Kai, Hertzberg Joachim, and Surmann Hartmut. 6d slam with approximate data association. In *Advanced Robotics, 2005. ICAR'05. Proceedings., 12th International Conference on*, pages 242–249. IEEE, 2005.

Nüchter Andreas, Lingemann Kai, and Hertzberg Joachim. 6D SLAM with Kurt3D. *Robotics Today, Society of Manufacturing Engineers*, 20:59–63, 2007a.

Nüchter Andreas, Lingemann Kai, Hertzberg Joachim, and Surmann Hartmut. 6D SLAM - 3D Mapping Outdoor Environments. *ournal of Field Robotics (JFR), Special Issue on Quantitative Performance Evaluation of Robotic and Intelligent Systems*, 24: 699–722, August - September 2007b.

Oriolo Giuseppe, Ulivi Giovanni, and Vendittelli Marilena. Fuzzy maps: a new tool for mobile robot perception and planning. *Journal of Robotic Systems*, 14(3): 179–197, 1997.

Pagac Daniel, Nebot Eduardo M, and Durrant-Whyte Hugh. An evidential approach to map-building for autonomous vehicles. *Robotics and Automation, IEEE Transactions on*, 14(4):623–629, 1998.

Pathak Kaustubh, Birk Andreas, Poppinga Jann, and Schwertfeger Sören. 3d forward sensor modeling and application to occupancy grid based sensor fusion. In *Intelligent Robots and Systems (IROS). IEEE/RSJ International Conference on*, pages 2059–2064. IEEE, 2007.

Payeur Pierre, Hébert Patrick, Laurendeau Denis, and Gosselin CM. Probabilistic octree modeling of a 3d dynamic environment. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 2, pages 1289–1296. IEEE, 1997.

Petrovskaya Anna and Thrun Sebastian. Model based vehicle detection and tracking for autonomous urban driving. *Autonomous Robots*, 26(2-3):123–139, 2009.

Petrovskaya Anna V. *Towards dependable robotic perception*. PhD thesis, Stanford University, 2011.

Pfeiffer David and Franke Uwe. Efficient representation of traffic scenes by means of dynamic stixels. In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pages 217–224. IEEE, 2010.

Premebida Cristiano and Nunes Urbano. A multi-target tracking and gmm-classifier for intelligent vehicles. In *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*, pages 313–318. IEEE, 2006.

Premebida Cristiano, Monteiro Gonçalo, Nunes Urbano, and Peixoto Paulo. A lidar and vision-based approach for pedestrian and vehicle detection and tracking. In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 1044–1049. IEEE, 2007.

Pulford GW. Multi-target viterbi data association. In *Information Fusion, 2006 9th International Conference on*, pages 1–8. IEEE, 2006.

Pulli Kari. Multiview registration for large data sets. In *3-D Digital Imaging and Modeling, 1999. Proceedings. Second International Conference on*, pages 160–168. IEEE, 1999.

Reid Donald. An algorithm for tracking multiple targets. *Automatic Control, IEEE Transactions on*, 24(6):843–854, 1979.

Ribo Miguel and Pinz Axel. A comparison of three uncertainty calculi for building sonar-based occupancy grids. *Robotics and Autonomous Systems*, 35(3):201–209, 2001.

Richter E Schubert R Wanielik G. Radar and vision based data fusion-advanced filtering techniques for a multi object vehicle tracking system. In *IEEE Intelligent Vehicles symposium (IV)*, pages 120–125, 2008.

Rivadeneyra Cesar, Miller Isaac, Schoenberg Jonathan R, and Campbell Mark. Probabilistic estimation of multi-level terrain maps. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 1643–1648. IEEE, 2009.

Rong Li X and Bar-Shalom Yaakov. Tracking in clutter with nearest neighbor filters: analysis and performance. *Aerospace and Electronic Systems, IEEE Transactions on*, 32(3):995–1010, 1996.

Rosheim Mark E. *Robot Evolution: The Development of Anthrobotics*. Wiley-IEEE, 1994.

Roth-Tabak Yuval and Jain Ramesh. Building an environment model using depth information. *Computer*, 22(6):85–90, 1989.

Ryoo Michael S and Aggarwal Jake K. Observe-and-explain: A new approach for multiple hypotheses tracking of humans and objects. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

Särkkä Simo, Vehtari Aki, and Lampinen Jouko. Rao-blackwellized particle filter for multiple target tracking. *Information Fusion*, 8(1):2–15, 2007.

Schamm Thomas, Zollner J Marius, Vacek Stefan, and Schroder Joachim. Obstacle detection with a photonic mixing device-camera in autonomous vehicles. *International Journal of Intelligent Systems Technologies and Applications*, 5(3):315–324, 2008.

Schulz Dirk, Burgard Wolfram, Fox Dieter, and Cremers Armin B. Tracking multiple moving targets with a mobile robot using particle filters and statistical data association. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 1665–1670. IEEE, 2001.

Schulz Dirk, Burgard Wolfram, Fox Dieter, and Cremers Armin B. People tracking with mobile robots using sample-based joint probabilistic data association filters. *The International Journal of Robotics Research*, 22(2):99–116, 2003.

Schwarz Brent. Mapping the world in 3d. *Nature Photonics*, 4:429–430, 2010.

Siadat Ali, Kaske Axel, Klausmann Siegfried, Dufaut Michel, and Husson René. An optimized segmentation method for a 2d laser-scanner applied to mobile robot navigation. In *3rd IFAC Symposium on Intelligent Components and Instruments for Control Applications*, pages 153–158. Citeseer, 1997.

Singer Robert A and Stein John J. An optimal tracking filter for processing sensor data of imprecisely determined origin in surveillance systems. In *Decision and Control, 1971 IEEE Conference on*, volume 10, pages 171–175. IEEE, 1971.

Smith Randall, Self Matthew, and Cheeseman Peter. A stochastic map for uncertain spatial relationships. In *Proceedings of the 4th international symposium on Robotics Research*, pages 467–474. MIT Press, 1988.

Smith Randall, Self Matthew, and Cheeseman Peter. Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*, pages 167–193. Springer, 1990.

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

Spinello Luciano, Triebel Rudolph, and Siegwart Roland. Multimodal people detection and tracking in crowded scenes. In *AAAI*, pages 1409–1414, 2008.

Stiene Stefan, Lingemann Kai, Nuchter Andreas, and Hertzberg Joachim. Contour-based object detection in range images. In *3D Data Processing, Visualization, and Transmission, Third International Symposium on*, pages 168–175. IEEE, 2006.

Stiller Christoph, Hipp Johann, Rössig C, and Ewald A. Multisensor obstacle detection and tracking. *Image and vision Computing*, 18(5):389–396, 2000.

Stiller Christoph, Kammel Sören, Pitzer Benjamin, Ziegler Julius, Werling Moritz, Gindele Tobias, and Jagszent Daniel. Team anniewayâĂŹs autonomous system. In *Robot Vision*, pages 248–259. Springer, 2008.

Streller Daniel and Dietmayer Klaus. Object tracking and classification using a multiple hypothesis approach. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 808–812. IEEE, 2004.

Streller Daniel, Dietmayer Klaus, and Sparbert Jan. Object tracking in traffic scenes with multi-hypothesis approach using laser range images. In *8th World Congress on Intelligent Transport Systems*, 2001.

Sun Zehang, Bebis George, and Miller Ronald. On-road vehicle detection: A review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(5):694–711, 2006.

Svensson Lennart, Svensson Daniel, Guerriero Marco, and Willett Peter. Set jpda filter for multitarget tracking. *Signal Processing, IEEE Transactions on*, 59(10):4677–4691, 2011.

Taleghani Sanaz, Aslani Siavash, and Shiry Saeed. Robust moving object detection from a moving video camera using neural network and kalman filter. In *RoboCup 2008: Robot Soccer World Cup XII*, pages 638–648. Springer, 2009.

Teichman Alex, Levinson Jesse, and Thrun Sebastian. Towards 3d object recognition via classification of arbitrary object tracks. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4034–4041. IEEE, 2011.

Thomaidis George, Spinoulas Leonidas, Lytrivis Panagiotis, Ahrholdt Malte, Grubb Grant, and Amditis Angelos. Multiple hypothesis tracking for automated vehicle perception. In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pages 1122–1127. IEEE, 2010.

Thrun S, Burgard W, and Fox D. Probabilistic robotics, ser. intelligent robotics and autonomous agents, 2005.

Thrun Sebastian. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

Thrun Sebastian. Robotic mapping: A survey. *Exploring Artificial Intelligence in the New Millenium*, -:1–35, 2002.

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

Thrun Sebastian, Montemerlo Mike, Dahlkamp Hendrik, Stavens David, Aron Andrei, Diebel James, Fong Philip, Gale John, Halpenny Morgan, Hoffmann Gabriel, *et al.* Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006.

Triebel Rudolph, Pfaff Patrick, and Burgard Wolfram. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2276–2282. IEEE, 2006.

Triebel Rudolph, Shin Jiwon, and Siegwart Roland. Segmentation and unsupervised part-based discovery of repetitive objects. In *Robotics: Science and Systems*, volume 2, Zaragoza, Spain, 2010.

Urmson Chris, Anhalt Joshua, Bagnell Drew, Baker Christopher, Bittner Robert, Clark MN, Dolan John, Duggins Dave, Galatali Tugrul, Geyer Chris, *et al.* Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

Vasudevan Shrihari, Ramos Fabio, Nettleton Eric, and Durrant-Whyte Hugh. Non-stationary dependent gaussian processes for data fusion in large-scale terrain modeling. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1875–1882. IEEE, 2011.

Vihola Matti. Rao-blackwellised particle filtering in random set multitarget tracking. *Aerospace and Electronic Systems, IEEE Transactions on*, 43(2):689–705, 2007.

Viola Paul and Jones Michael. Robust real-time object detection. *International Journal of Computer Vision*, 4:–, 2001.

Vosselman George, Gorte Ben GH, Sithole George, and Rabbani Tahir. Recognising structure in laser scanner point clouds. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 46(8):33–38, 2004.

Vu TD. *Vehicle perception: Localization, mapping with detection, classification and tracking of moving objects*. PhD thesis, PhD thesis, Grenoble Institute of Technology, 2009.

Vu Trung-Dung, Burlet Julien, and Aycard Olivier. Mapping of environment, detection and tracking of moving objects using occupancy grids. In *Intelligent Vehicles Symposium*, pages 684–689, 2008.

Walker Michael W, Shao Lejun, and Volz Richard A. Estimating 3-d location parameters using dual number quaternions. *CVGIP: image understanding*, 54(3):358–367, 1991.

Wang C.-C. *Simultaneous Localization, Mapping and Moving Object Tracking*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 2004.

Wang C.-C. and Thorpe C. Simultaneous localization and mapping with detection and tracking of moving objects. In *IEEE International Conference on Robotics and Automation (ICRA), Washington, DC, USA*, 2002.

Wang Chieh-Chih and Thorpe Charles. A hierarchical object based representation for simultaneous localization and mapping. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 1, pages 412–418. IEEE, 2004.

Wang Chieh-Chih, Thorpe Charles, and Thrun Sebastian. Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 1, pages 842–849. IEEE, 2003.

Wang Chieh-Chih, Thorpe Charles, Thrun Sebastian, Hebert Martial, and Durrant-Whyte Hugh. Simultaneous localization, mapping and moving object tracking. *The International Journal of Robotics Research*, 26(9):889–916, 2007.

Weingarten Jan and Siegwart Roland. 3d slam using planar segments. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3062–3067. IEEE, 2006.

Wender Stefan and Dietmayer Klaus. 3d vehicle detection using a laser scanner and a video camera. *Intelligent Transport Systems, IET*, 2(2):105–112, 2008.

Wojke Nicolai and Haselich M. Moving vehicle detection and tracking in unstructured environments. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3082–3087. IEEE, 2012.

Wurm Kai M, Hornung Armin, Bennewitz Maren, Stachniss Cyrill, and Burgard Wolfram. Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems. In *Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, volume 2, 2010.

Xu Feng, Huang Chenrong, Wu Zhengjun, and Xu Lizhong. Video multi-target tracking based on probabilistic graphical model. *Journal of Electronics (China)*, 28 (4-6):548–557, 2011.

Yguel Manuel, Aycard Olivier, and Laugier Christian. Wavelet occupancy grids: a method for compact map building. In *Field and Service Robotics*, pages 219–230. Springer, 2006.

Yguel Manuel, Tay Christopher, Keat Meng, Braillon Christophe, Laugier Christian, and Aycard Olivier. Dense mapping for range sensors: Efficient algorithms and sparse representations. In *Robotics: Science and Systems*. Cambridge, MIT Press, 2007.

Yguel Manuel, Aycard Olivier, and Laugier Christian. Update policy of dense maps: Efficient algorithms and sparse representation. In *Field and Service Robotics*, pages 23–33. Springer, 2008.

Yilmaz Alper, Javed Omar, and Shah Mubarak. Object tracking: A survey. *Acm Computing Surveys (CSUR)*, 38(4):13, 2006.

Yu Jin-Xia, Cai Zi-Xing, and Duan Zhuo-Hua. Detection and tracking of moving object with a mobile robot using laser scanner. In *Machine Learning and Cybernetics, 2008 International Conference on*, volume 4, pages 1947–1952. IEEE, 2008.

Zhang Sen, Adams M, Tang Fan, and Xie Lihua. Geometrical feature extraction using 2d range scanner. In *Control and Automation, 2003. ICCA'03. Proceedings. 4th International Conference on*, pages 901–905. IEEE, 2003.

Zhao HUIJING, Shao XW, Katabira KYOICHIRO, and Shibasaki R. Joint tracking and classification of moving objects at intersection using a single-row laser range scanner. In *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*, pages 287–294. IEEE, 2006.