UNIVERSITÀ DI PARMA

UNIVERSITÀ DEGLI STUDI DI PARMA

Dottorato di Ricerca in Tecnologie dell'Informazione

XXXI Ciclo

Design and development of stereoscopic obstacle detection systems for UAV

Coordinatore:

Chiar.mo Prof. Marco Locatelli

Tutor:

Chiar.mo Prof. Massimo Bertozzi

Dottorando: *Francesco Valenti*

Anni 2015/2018

Ai miei genitori, Alberto e Barbara

Abstract

Recently a lot of attention has been given to enabling autonomous capabilities on a broad range of vehicles. Among them Unmanned Aerial Vehicles (UAVs), and in particular small quadrotors, must comply to payload-related constraints which poses difficult challenges on the choice of the sensors used to perceive the world.

This thesis introduces an obstacle detection and mapping system designed for an UAV based on stereoscopic vision. A small consumer drone, equipped with cheap and lightweight stereo cameras, is used as development robotic platform for this system, for directly demonstrating its applicability.

The proposed algorithm is designed to exploit visual information collected from several pair of cameras in order to detect obstacles around the aircraft. Each pair of cameras collect stereoscopic images which are used to recover depth information about the surroundings. Depth quantities incoming from each stereo camera are then fused together and used to build a single representation of the world. This environment description, completely tri-dimensional, is suited for defining any type of environment, because no assumptions have been made on its geometry. The final result is provided as an occupancy grid map, hence conveys information about each portion of the space being occupied or free. Such representation can be used as such, for visualization purposes in an exploration mission, or as main input for subsequent modules in a completely autonomous system.

Inverse sensor models for both pinhole and spherical camera are presented and applied during the occupancy grid mapping procedure.

A ground truth building approach is presented and used to asses the quality of the obstacles detection and mapping algorithm. The proposed ground truth generation, consisting in exploiting high-accuracy data to build a supposedly true representation of the environment, is tested on a freely available dataset.

Results are obtained by comparison between the proposed obstacle detection system and the constructed ground truth.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Autonomous or semi-autonomous robots have been widely used in industry for quite a long time. In such an environment they are used to perform tasks that are usually hard, repetitive or difficult to carry out for humans. Robotics systems of several types have been increasingly employed during the last decades, ranging from static semi-autonomous platforms, performing repetitive tasks in highly controlled scenarios, to moving robots capable of carrying out complex tasks while interacting with other agents.

Robots and robotic systems are usually designed with the aim of substituting human beings in performing repetitive or dangerous tasks or tasks otherwise considered beyond human possibilities. The concept of robots and artificial servants has its origins in the ancient world, but the modern concept began to develop along with the Industrial Revolution. The use of complex mechanics and the introduction of electricity made it possible to power machines with small compact motors. Towards the end of the 20th century the miniaturization of electronic components, along with the introduction of microprocessors, provided the means for embedding such robots with technological devices allowing them to be controlled, either remotely or not.

First widespread uses of modern robots were in factories as industrial automated systems, static machines capable of carrying out repetitive tasks in highly controlled scenarios. Before, human workers were employed to perform such tasks, for example

within the manufacturing process. Starting from the beginning of the XX Century the working paradigm of the production line has been used to maximize productivity and reduce assembly times. However, working personnel soon began to experience both mental and physics problems. Such issues pushed scientists and researchers in the direction of substituting humans with artificial systems, to have them instead of people perform repetitive tasks. Nowadays industrial robots working in assembly lines represent the most applied robotics technology worldwide, also known as industrial automation, while humans have to supervise their operations or partially cooperate white them in certain cases. This class of applications is facilitated in the industrial field because of the relatively controlled environment in which they have to operate. Usually, such autonomous robots are isolated from other agents, whether they are other robots or humans. Furthermore, the environment in which they operate is mostly static and does not require the ability to react to sudden and unexpected stimuli. More important, since the task they have to carry out is always the same, they can be programmed to perform a particular set of actions repetitively.

A slightly different class of autonomous robots used in industry is the Automated Guided Vehicle (AGV), which is usually responsible for handling packages inside storage facilities. This automatic vehicle usually has to follow predetermined paths, and can generally rely on a given description of the environment in which it has to operate. Such information, commonly referred to as a map, is used to extract the path that the robot has to follow in order to reach a desired position inside the facility while avoiding static obstacles. In some cases, a single AGV can operate in a specific area, and any human-robot interaction is strictly forbidden. In these cases, there is no need for handling dynamic obstacles that therefore are not taken into account while building the map. There are cases, however, where multiple AGVs and occasionally some people have to operate inside the same area simultaneously. This may be needed for different reasons, such as productivity increase or necessity for human intervention in performing complex tasks. In this scenario, a robotic vehicle must be able to perceive the environment, detect other agents and adjusting its precomputed path accordingly in order to avoid collisions. The dynamic nature of this semi-controlled environment increased the level of complexity required for this technology if compared to the

previous class of applications. The robotic system, in this case, has to be equipped with sensors perceiving its surroundings and needs additional computational power in order to process and extract information from collected data.
Nowadays robots are widely used in our everyday life. We use autonomous or semi-autonomous systems to have various tasks performed, from cleaning our houses to parking our cars. These systems are continuously improved, and industry is more than ever interested in cutting-edge technologies making robots and autonomous systems capable of doing more and more complex tasks.
These autonomous systems belong to a different class of robotics applications if compared to the previous ones. They are designed to perform their tasks neither in a repetitive way nor in a structured or static scenario. They have to operate in an environment that is not known apriori, thus they have to perceive their surrounding, understand what is happening in the local neighborhood and consequently adapt their behavior.
Obstacle detection is thus a crucial requirement for completely autonomous systems, such as self-driving cars or drones capable of flying autonomously. In the most general case of an autonomous robot carrying out a given task it is not so easy to define what is an obstacle precisely.
For a self-driving car, for instance, it is common to consider as obstacles non-traversable shapes, but such definition is not so clear. In fact, a tri-dimensional shape could be considered as traversable depending on factors such as the type of autonomous vehicle, the kind of terrain, or the task that has to be carried out, just to name a few. For an autonomous drone, instead, it is quite easy to define as obstacle pretty much anything tangible, since it represents a threat to the aircraft. Also, rules and regulations have to be taken into consideration where they are available for a specific task. Only recently robotics applications that require an obstacle detection module had started to be commercialized, while before they were confined to research prototypes from both universities and industries.
For example, Tesla Autopilot[1], an advanced driver-assistance system (ADAS) from

[1]Tesla Autopilot was marketed as Enhanced Autopilot during 2016

Tesla[2], enables everyday vehicles to perform various tasks autonomously. Among them, there is adaptive cruise control, self-parking, and lane departure warning. Tesla vehicles are however already equipped with various sensors, and Tesla announced that a future update would enable full self-driving capabilities. As today, self-driving cars require human presence with full awareness at the driving seat for security reasons. In the future, however, the possibility of having intelligent vehicles that does not require any possible human intervention will provide several benefits to our everyday life.

In the past decades also unmanned aerial vehicles (UAV), commonly known as

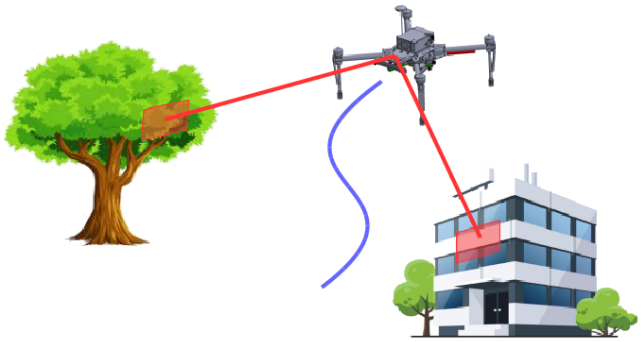


Figure 1.1: Obstacle detection and avoidance during flight.

drones, has been increasingly popular. Accepted definition for UAV states that it a "powered, aerial vehicle that does not carry a human operator, uses aerodynamic forces to provide vehicle lift, can fly autonomously or be piloted remotely, can be expendable or recoverable, and can carry a lethal or nonlethal payload" [3]. UAVs have been widely used for various applications, including inspection, mapping and surveying, rescuing, entertainment and so on. Any unmanned aerial vehicle is an aircraft which can fly without a human pilot aboard, requiring either a remotely piloting person or an autonomous flying system. For certain types of applications, self-flying drones are not currently employed, which means that a person remotely piloting the

[2]Tesla, Inc. (formerly Tesla Motors) is an American multinational corporation that specializes in electric vehicles based in Palo Alto, California.

[3]Unmanned Aerial Vehicle, TheFreeDictionary.com, retrieved on the 8th of January, 2015.

aircraft is needed. Sometimes visual feedback is provided to the pilot and optionally to other assistants using cameras mounted aboard. There are some cases, however, where a stable connection cannot be ensured due to various factors, such as long distances or the presence of barriers potentially obstructing the signal transmission. Moreover, piloting an aircraft becomes quite tricky in complex areas, such as GPS-denied environments. A robotic system designed to show a certain level of autonomy can overcome these problems, being capable of performing some tasks without the need for human intervention. For example, in cases where a human pilot would not be able to fly the aircraft, the autonomous system could take over and fly the aircraft, preserving the safety of both the drone itself and other people.

For example, in case the remote control signal is lost, the drone must be able to return to its takeoff position to be recovered by its pilot. In order to do that, a simple non-autonomous drone usually adopts the azimuth method, flying straight starting from its current position to the takeoff area. This method presents many drawbacks, for example, if obstacles, such as trees or buildings, are present along the flying path. Otherwise, a certain level of autonomy is required to detect and avoid any obstacles the aircraft may encounter during its journey. This is just an example of a task to which an autonomous or semi-autonomous drone can be applied, but there are lots of cases where autonomous or semi-autonomous drones are applied to perform different tasks, ranging from personal entertainment to humanitarian or industrial applications. These drones can be used without human intervention only in specific areas or under certain conditions, such that possible malfunctions would not expose anyone at risk. In order to build an autonomous drone, the aircraft must be equipped with sensors of different nature and purpose, as well as processing units capable of interpreting perceived data and making decisions based on them. Necessary sensors comprise barometers, gyroscopes, magnetometers, GPS receivers and so on, which collect data needed for retrieving an estimation of the aircraft pose. Furthermore, perception sensors like lasers, sonars, radars or cameras can be used to harvest information regarding the surrounding of the drone. These two kinds of sensors can be used in different combinations, mainly dependent on the specific application or use case. If the aircraft have to be able to navigate in an unknown scenario autonomously, both kind of sensors must

be used. Moreover, for the most challenging situations, such as navigation in GPS-denied areas, perception sensors have to be employed to aid in providing localization information. While the sensor suite devoted to the pose estimation is well established and comprises the previously mentioned set of devices on almost the majority of drones, different approaches have been adopted for the choice of perception sensors. Previously, researchers mainly focused on the use of laser or radar sensors, which have the advantage of directly measuring depth to the nearest obstacle, which is something of absolute importance for an autonomous flying robot. Nevertheless, with the rapid development of computer vision, which utilize cheaper and more flexible visual sensors, computer-vision based methods have shown significant advantages in the field of UAV navigation. Besides, the recent availability of high computational power made it feasible to extract information from images, which is something computationally expensive due to the high dimensionality of the data. Self-flying robots capable of carrying out complex tasks in challenging scenarios need tools to collect information regarding their surroundings and build an adequate representation of the world around them. They will thus be able to take decisions in order to reach their goals without putting at danger themselves or other agents. However, in order to fly safely, a UAV must be able to detect and avoid obstacles encountered along its path. An obstacle detection system is thus mandatory for any drone that has to navigate autonomously. The detection of obstacles consists in estimating the position of the dangerous entities that may appear in the vicinity of the robotic system. Various definitions of "dangerous entities" may give rise to a different interpretation of obstacle detection systems, which may or may not be used for different applications or use cases. If the robotic system to be considered consists of a drone, almost anything which is tangible must be treated as a dangerous entity.

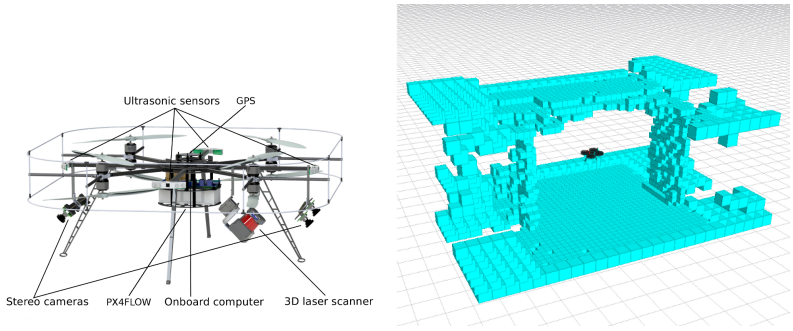
Recent research advancements in obstacle detection and environment representation are briefly summarized in Chapter 2. The UAV robotic platform, as well as equipped sensors and computing units, are presented in Chapter 3. Pinhole and spherical stereoscopic camera models are presented in Chapter 4 since will be used in subsequent parts of the thesis. Chapter 5 presents the obstacle detection algorithm, leaving the description of the mapping procedure for Chapter 6. A pair of possible environment

representations is presented in Chapter 7, along with their advantages and disadvantages. Chapter 8 describes a method for building a ground truth, then a constructed true state of the environment is used to asses the quality of the proposed system with respect to a set of evaluation metrics. Conclusions and future works are discussed in Chapter 9.

Chapter 2

State of the Art

Recently, there has been much work in enabling autonomous navigation for UAVs. A drone capable of independently taking decisions and carrying out complex tasks in an unknown scenario has to be able to gain information regarding the environment in which is going to operate. Such awareness consists essentially in detecting potential harmful objects that can represent obstacles along its path. They have to be avoided during the fly, in order not to cause damage either to them or to the aircraft itself.
Several sensors can be used to detect obstacles, and each of them is more or less suited to detect a specific type of object, depending on various features of the objects such as texture, shape or reflectivity, to name a few.
A collection of different sensors is then attached to an aircraft platform to detect obstacles in a wide variety of environments. These sensors will provide raw data that have to be processed in order to extract information regarding obstacles, such as their presence, size, shape, position, an so on, depending on the specific application. The term obstacles can thus be used to refer to the collection of all these information regarding a single object instance.
Perceived obstacles have to be subsequently stored in a representation of the world. This description must be efficient in term of both memory occupation and computational load required to retrieve information from it. Lots of different representations have been designed to address various requirements. They mainly depend on which



(a) (b)

Figure 2.1: A drone equipped with a rotating laser scanner as main sensor (a) builds a tri-dimensional map (b) by aggregating consecutive scans. Images from *"Obstacle detection and navigation planning for autonomous micro aerial vehicles"*, Nieuwenhuisen et al., ICUAS, 2014

type of information they must convey, how big the portion of the world to be represented must be and which level of detail is required for the description.

2.0.1 Non-vision

Range detection sensors can be used as central sensing units to detect obstacles or, in general, to collect information regarding the surroundings. This class of sensors, including ultrasonic, radars and laser scanners, usually exploit the time-of-flight as a direct way of measuring distances.
Probably the cheapest and most compact usable device is the ultrasonic range sensor. It emits a sonic beam and measures the time necessary for receiving it back. Sonars, or ultrasonic sensors, have been widely used for obstacle detection in robotics applications.
Sonar sensors were mainly applied to robots capable of planar movements, and multiple devices arranged in rings are usually employed to recover for the lacking of angular resolution, as in [1].
More recently, many obstacles detection systems based on ultrasonic sensors were

designed for unmanned aerial vehicles, as presented in [2], [3] and [4]
In [5] the authors introduced an autonomous drone capable of flying through indoor environments without the need for human intervention. Only sonar sensors were used as sensing devices to overcome intrinsic difficulties arising in a typical indoor environment, like ambient infrared radiation or poor lighting conditions.
An obstacle detection system for UAV based on low-cost sensors, like ultrasonic and infrared range finders, was presented in [6].
Sonars have the main advantage of being cheap, compact and lightweight. In contrast, they suffer from poor angular resolution and a limited range of detection. Laser scanners represent a valid alternative as an active depth sensor since they can provide precise measurements and consistent angular resolution even at great distances. Gronka et al. [7] presented a drone equipped with a 2D laser scanner. In this case, the drone was capable of localizing itself while exploring an unknown environment by exploiting precise measurements. This paradigm is known as SLAM, which stands for simultaneous localization and mapping. The main disadvantage of this approach is that environment perception is only planar, and thus obstacle avoidance is limited to the navigation plane.
Ramasamy et al. [8] presented a laser scanner-based obstacle detection and avoidance system for unmanned aerial vehicles. It is interesting to note that their development platform was a fixed-wing aircraft, which resulted in a particular choice for the sensor mounting position and actuation. The laser scanner is mounted in the frontal part of the aircraft and actuated such that it always covers a specific field of view around the flying trajectory.
In [9] Nieuwenhuisen et al. used a 2D laser scanner as the primary sensor mounted on a continuously rotating servo actuator to achieve 3D sensing. This approach allowed for building a 3D description of the environment without the need for using a multi-layer 3D laser scanner, which would significantly impact the payload. Their perception setup also comprised a pair of fish-eye stereo cameras for visual odometry and current pose estimation, as depicted in Figure 2.1a.
Laser scanners provide nearly noise-free measurements, but cameras usually outperform them concerning compactness and expensiveness. Vision-based obstacle detec-

tion and navigation has been a widely explored topic in the last years, mainly because cameras represent an optimal compromise between the richness of information, the field of view and the measurement precision.

2.0.2 Monocular vision

Obstacle avoidance using monocular vision is desirable since micro aerial vehicles must usually comply with strict payload limitations. Many vision based system detect obstacles using optical flow-based techniques [10, 11, 12, 13, 14, 15, 16].
An intrinsic problem with optical flow-based methods is the detection of frontal objects, which is usually poorly performed. Mori et al. [17] presented a monocular obstacle detection system designed to tackle that problem specifically. Their algorithm used SURF feature [18] matches in combination with a template matching approach to detect changes in the size of for frontal objects, thus allowing for distance estimation and obstacle detection.
In [19] Watanabe et al. proposed a monocular camera-based system for obstacle avoidance. This work presented an autonomous UAV capable of autonomously navigating using a single front-looking camera. Detected obstacles were tracked, and their 3D position was estimated utilizing an Extended Kalman Filter (EKF).
A similar approach was proposed by [20], where features extraction and tracking was performed on consecutive monocular images. An inverse depth model was estimated for each tracked feature to recover for the actual 3D position. Persistent features were then accumulated on a terrain map used for obstacle avoidance and path planning purposes.
The system proposed in [21] used a monocular setup to perform obstacle detection and avoidance. In this case, an optical-flow extraction method was used as a basis for depth estimation, thus leading to obstacle detection. Disadvantages of this approach comprise arising difficulties when dealing with multiple obstacles, as well as the impossibility of detecting low-textured objects or the degradation of performances under particular weather conditions.
In [22] a UAV with a single camera mounted onboard was used to obtain obstacle avoidance capabilities. Authors inspiration came from perspective laws which state

that objects in the field of view are getting bigger as they are closer to the sensor. They thus designed the system to search for obstacles size changing during the flight, and mark objects as dangerous obstacles based on their visual appearance among several sequential images.
In [23] the authors proposed an autonomous obstacle avoidance system for a low-cost quadrotor, based on monocular vision. Their system was designed to carry out a simultaneous localization and mapping (SLAM) task during flight, allowing for the drone to estimate its pose as well as gaining information regarding the environment at the same time. The vicinity of the reconstructed map, which is usually quite sparse, was processed in order to obtain a more dense map suitable for obstacle avoidance.
In [24] Weiss et al. presented a drone equipped with a single monocular down-looking camera, capable of performing simultaneous localization and mapping, using a Visual SLAM [25] algorithm from Klein and Murray. Such an approach proved to be very performing since the mapping, and the feature tracking threads were split up, allowing them to run at different frequencies. Furthermore, they showed that their system succeeded in building a tri-dimensional map from the 3D point cloud collected by the SLAM algorithm. A path planning algorithm exploited such map in order to avoid obstacles in both indoor and outdoor scenarios.
Moreover, if the perception is limited to the area below the aircraft in order to maximize the performances of localization and mapping tasks, no obstacles on the navigation plane can be detected. Another disadvantage of a SLAM approach is usually the massive amount of memory required to store the map.

2.0.3 Stereoscopic vision

Besides monocular approaches, much work has been done on stereo-vision based application. The main advantage of using a stereo rig to perceive the world is that distances can be measured explicitly, without the need for making further assumptions.
In [26] an obstacle detection and tracking system for UAV application have been presented. The authors addressed the disparity inconsistency issue, due to texture lacking in image areas, taking into account for monocular visual cues. In particular,

a segmentation stage allowed to discard uncertain disparities from within untextured areas.

Heng et al. [27] presented an obstacle avoidance and mapping system for an autonomous aerial vehicle based on stereoscopic vision. In this approach a quadrotor was equipped with two stereo cameras, pointing both forward and downward, used to perceive obstacles and localize the robot during flight. The perceived depth measurements were projected in spherical space, mimicking a multi-beam range sensor and later used to build a 3D global obstacle map of the environment.

In [28] Fraundorfer et al. described an autonomous micro aerial vehicle capable of navigating in unknown scenarios. They exploited depth information collected by a front-looking stereo camera to build a 3D occupancy grid map incrementally, used to perform obstacle avoidance by planning safe trajectories.

Barry el al. [29] designed an obstacle detection and avoidance system for a fixed-

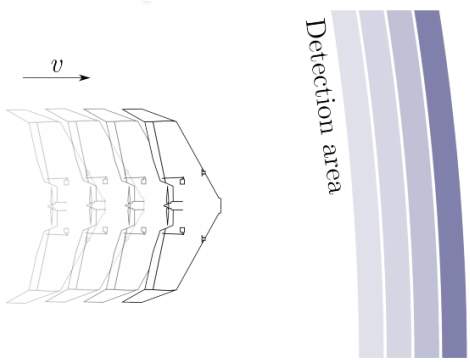
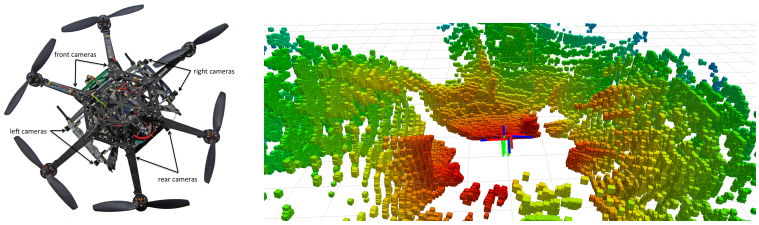


Figure 2.2: Pushbroom stereo detection area. Image from *"Pushbroom stereo for high-speed navigation in cluttered environments"*, Barry et al., ICRA, 2015

wing micro aerial vehicle. In this case, since the movement of the aircraft is highly constrained, they developed a novel stereo vision algorithm that searches for obstacles only at a certain distance, as depicted in Figure 2.2. The system tracks detected objects by integrating the aircraft odometry. This highly efficient solution can, however, handle only static obstacles for a small period, since odometry integration suffers from drifting.

A drone capable of autonomous navigation in both indoor and outdoor scenario is presented in [30]. The authors equipped an aircraft with a unified perception system comprising a stereoscopic camera, an IMU and other processing units to perform obstacle detection, localization and 3D mapping of the environment. They designed the system such that given the desired destination, the drone would be able to reach it while avoiding harmful obstacles along the path.
An obstacle avoidance system based on stereo vision was presented in [31]. Such an algorithm was designed to analyze disparities and build more straightforward representations, called UV-maps [32], which are later used for detecting and avoiding obstacles.
Another solution for reactive obstacle detection based on an embedded stereo vision system was presented in [33]. The authors decided to use an embedded stereo vision system to satisfy power consumption-related constraints. Their platform was indeed capable of performing the entire computation needed on-board with low power consumption, without requiring off-board control from a ground station.
An obstacle avoidance system designed for a flapping wing MAV and based on stereo vision was presented in [34]. The proposed system exploited information collected from a front-looking stereo camera to continuously monitor the area in front of the drone, called *Droplet*. The drone would have performed an evasive maneuver if any obstacle had entered inside such area.
In [35] the authors proposed an obstacle detection system based on a stereo vision for a rotorcraft unmanned aerial vehicle. In this case, the aircraft was capable of avoiding obstacles incoming from the front by analyzing only specific regions of the disparity map.
In [36] and [37] the authors presented an autonomous UAV, equipped with a stereo camera looking forward as their main obstacle sensing sensor. Problems arising from a limited field of view on a time-frame basis perception were solved by accumulating measurements over time in an allocentric map. Such a perception setup can be used in static environments, while dynamically changing scenarios require an all-around view sensing suite.
For example in [38] an omnidirectional stereo camera is used to collect range mea-

surements on a time-frame basis. Captured images are warped into panoramic frames, and a stereo matching algorithm is applied to exploit left to right disparities and perform depth estimation.
Gohl et al. proposed a different approach in [39], where a set of four narrow-lenses stereo cameras are used to collect depth information simultaneously and achieve omnidirectional sensing by fusing gathered data on a subsequent step, as illustrated in Figure 2.3. The authors efficiently used the omnidirectional local spherical representation to perform obstacle avoidance tasks.



(a) (b)

Figure 2.3: Omnidirectional sensing setup. A drone equipped with four stereo cameras (a) can build an all-around view spherical representation (b) of its surroundings. Images from *"Omnidirectional visual obstacle detection using embedded FPGA"*, Gohl et al., IROS, 2015

2.1 Environment representation and mapping

Suitable environment description is essential for autonomous vehicles since it is usually the basis for subsequent high-level modules. Depending on the final system requirements, the chosen representation should have enough describing power to capture all the relevant aspects of the world.
One of the first environment representation proposed for mobile robots applications was the vector field histogram [40]. According to this method, collected range data is accumulated in two-dimensional Cartesian histogram grid, which is therefore used as

a world model. Subsequent stages are performed to extract information depending on particular requirements for the specific application.
The occupancy grid mapping algorithm [41, 42, 43] is a widely used method for estimating a map from noisy measurements. It consists of a probabilistic framework for fusing multiple sensor readings into a single map, provided that the robot pose is known with respect to the map at all times. Figure 2.4 shows a bi-dimensional occupancy grid map, encoding the occupancy likelihood of each cell.
This approach has been used in many robotics systems, such as [44, 45, 46, 47].

Figure 2.4: Occupancy grid map of a large exhibit space. Image from *"Probabilistic robotics"*, Thrun et al., MIT press, 2005

Occupancy grid maps belong to the family of metric maps, which capture the geometric properties of the environment. Another metric-based mapping algorithm was proposed in [48], in which a robot equipped with multiple sensors was given the task of modeling the environment as it was perceived during exploration while trying to localize itself. This approach relied on sets of polyhedra to describe the geometry of the environment.
Opposite of metric maps, topological-based approaches represents environments using lists of objects or places, also known as nodes. Such maps describe spatial relations between nodes in the map using arcs, which usually contains also information on how

to navigate from one node to another. Topological maps have been used in many works, such as [49, 50, 51]. For this work, only metric-based maps and related environment representations will be taken into consideration.
The critical difference between UAVs and planar robots, such as road vehicles, is that they can move in three dimensions. This makes classical 2D environment representation undersized for aerial vehicles applications. A simple extension of classical 2D models such these to their corresponding 3D version is not always feasible due to computational and memory-related performance constraints.
For this reason, in literature, some representations are neither completely bi-dimensional nor fully tri-dimensional. Such is the case, for example, of the Digital Elevation Model (DEM), which maps the environment by subdividing it into several sectors and providing an elevation measurement for each of them. Such a representation implicitly consider each sector of the map as neither completely flat nor exactly tri-dimensional. The simplest variant considers each cell as starting from the same ground plane, storing only an elevation value. There are however more complex variants were also the starting elevation of each sector is estimated.
Such representation can be effectively used only in specific applications, depending on the domain and the requirements of the system. For example, it has been successfully applied to a perception system designed for an intelligent vehicle in [52] and [53]. In this case, the ground surface is considered as the reference region and the elevation of the various map sectors is used to represent the environment. A subsequent classification step has been applied to the cells to distinguish between obstacles and free space, as shown in Figure 2.5.
Triebel et al. proposed Multi-Level Surface maps, an evolution of the DEM representation. This kind of representation allows for describing multiple vertical structures since each cell can contain more than a single surface patch, as depicted in Figure 2.6b. Each of them is designed to store a height, modeled by a Gaussian distribution, and a depth, which is the difference of the height of the surface and the lowest measurement that is considered to belong to that patch. This representation, which is almost tri-dimensional, is useful to represent scenarios with complex geometries as shown in Figure 2.6a.

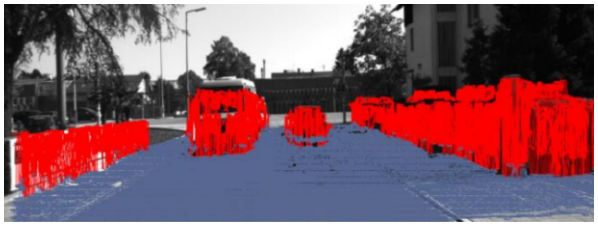
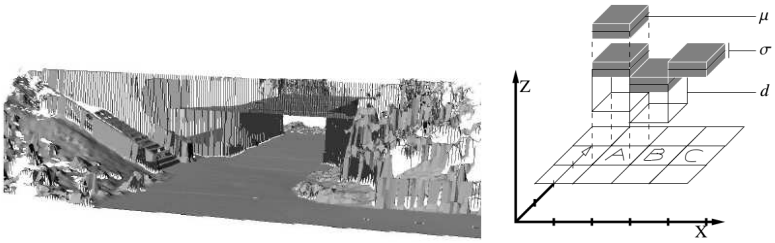


Figure 2.5: Digital Elevation Model map example. Image from *"Processing Dense Stereo Data Using Elevation Maps: Road Surface, Traffic Isle, and Obstacle Detection", Oniga et al., IEEE Transactions on Vehicular Technology, 2010*

A 3D world representation based on a hierarchical subdivision of the space can over-



(a) (b)

Figure 2.6: Multi-level surface map. A complex scenario can be described effectively using multiple surface patches. Images from *"Multi-level surface maps for outdoor terrain mapping and loop closing"*, Triebel et al., Intelligent Robots and Systems, 2006

come the memory consumption problem by not necessarily describing all areas of the environment. Such an organization usually depends on how the sensors perceive different characteristics of the environment. A well know approach based on this idea is OctoMap [55] a probabilistic framework for environment representation which is based on octrees[1]. The authors propose an extension of occupancy grid maps hierarchically structured, dynamically storing in memory only cells observed as occupied.

[1]For more information about octrees, please refer to [56].

Such a hierarchical structure leads to much more compact environment representations, at the expense of higher computation load in accessing map values. Figure 2.7 depicts an example of such representation.

Fu et al. [57] used an OctoMap implementation for storing 3D information collected

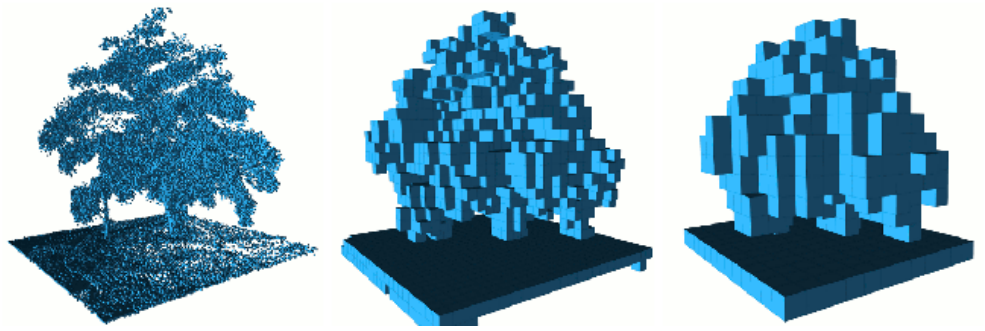


Figure 2.7: Octomap representation of a tree, as seen at different level of details. Image from *"OctoMap: an efficient probabilistic 3D mapping framework based on octrees"*, Hornung et al., Autonomous Robots, 2013

from a stereo-camera mounted on a UAV navigating in a cluttered scenario.

In [27] a modified version of the octree-based 3D map was used to represent the area explored by an unmanned micro aerial vehicle.

Specific UAVs applications require long-distance navigation capabilities. In such cases, it is unfeasible to store the entire map in memory and besides it would be useless to remember the position of an obstacle far from the current drone position. For such reason, it would be better to adopt map caching strategies, which would allow storing in memory only a given subregion of interest of the entire map.

In [36] the authors presented an autonomous UAV, equipped with a stereo camera looking forward as their main sensor. Problems arising from a limited field of view were solved by accumulating measurements over multiple frames in an allocentric map. They used a tiled octree-based 3D occupancy grid map with dynamic tile caching. The nearest four tiles to the drone position make up a map that is continuously updated as incoming measurements are processed. Remaining tiles are stored on disk, and dynamic caching is responsible for loading and storing tiles as the UAV moves inside

the grid.

In [58] Dryanovski et al. present a Multi-Volume Occupancy Grid which consists of a 2D grid made of squared evenly spaced cells. Each cell contains two sets of volumes for storing both positive and negative volumes. Each volume represent a tri-dimensional area and contains occupancy information which is updated given perceived measurements.

In [9] incoming depth measurements are stored in a hybrid local multi-resolution map, where cells contain both occupancy information and perceived distances, shown in Figure 2.1b. Individual grid cells are stored in ring buffers, and their size increases with distance from the drone center. Multiple ring buffers are interlaced to achieve three-dimensional environment description. This representation is fixed in size and egocentric, enabling local planning capabilities.

In this thesis, a system comprising several stereoscopic cameras, having different intrinsic properties, is used to perceive the surroundings of the drone. A local three-dimensional occupancy grid is used as a simple but effective way of representing the environment. The locality of the grid allows to keep it compact and of reduced memory footprint. On the other hand, it describes a region of the space which is of high interest for the aircraft, making it suitable for local navigation.

Chapter 3

System Setup

For this project a DJI Matrice 100 MAV [1] was used. This consumer drone is a development quadrotor from DJI and an official SDK [2] is released with the aircraft. Such programming framework can be used to communicate with the onboard processing unit, also known as the flight control system, through serial communication. It is thus possible to receive sensory information from onboard pre-installed sensors, such as GNSS receiver, accelerometer, gyroscope, and magnetometer, and send commands for the drone to follow. The aircraft is quite big, with a diagonal length of 65 cm and four 33.02 cm diameter propellers. The maximum load weight for takeoff is 3.6 Kg. A DJI Matrice 100 aircraft is depicted in Figure 3.1.

This drone is intended for development and is then designed to be equipped with other sensors, processing units, and communication devices. This characteristic was exploited to build a potentially autonomous system. A custom processing board, named SuperDrone board and realized by Ambarella [3], was installed onboard. The board was designed to be the only computational unit needed to process incoming data, to perform the necessary computation to extract relevant information, to plan safe trajectories to reach the goal and finally to generate and send control signals to the

[1] Shenzhen, China, https://www.dji.com/matrice100

[2] https://developer.dji.com/onboard-sdk/

[3] Ambarella, Inc. is a semiconductor design company from Santa Clara, California (US), specialized on low-power, high-definition video compression and image processing products

Figure 3.1: DJI Matrice 100.

drone flight controller. The board was connected to the Matrice 100 processing unit through a serial port, in order to send commands and receive data. The only purposes of this processing unit are to collect data incoming from pre-installed sensors, forward them to other processing units and control the drone motors according to eventually received commands. The aircraft system was equipped with various sensors and other devices, as shown in Fig. 3.2.

The sensing suite comprises four stereo cameras and one sonar sensor. Cameras

(a) (b)

Figure 3.2: DJI Matrice 100 equipped with the SuperDrone board, perception sensors and communication devices. Top (a) and bottom (b) views.

characteristics, such as sensor resolution or lenses, as well as their mounting positions were chosen according to their purpose in the project. The idea was to build an autonomous flying system, capable of maneuvering and adapt even in the most challenging conditions. These comprise both outdoor and indoor unknown and unstructured environments.

A forward-looking stereo camera equipped with narrow lenses was mounted in front

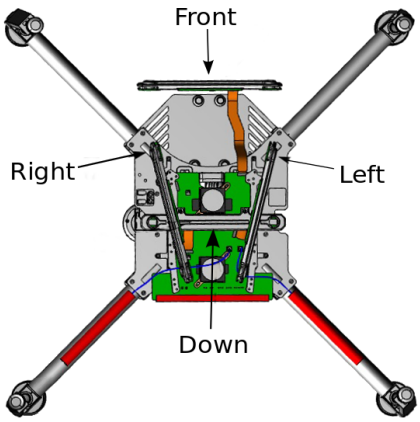


Figure 3.3: Bottom view of the final platform, with highlighted stereo cameras mounting positions.

of the UAV. Its task is to analyze the portion of the space head of the drone, searching for potentially dangerous obstacles as well as other visual cues. Since these cameras are equipped with narrow-lenses are suited for long distance perception. This design choice was made under the assumption that the drone could fly faster when moving forward.

Two stereo cameras are mounted laterally and slightly rotated backward. They were intended to provide sensing data from both sides of the aircraft, as well as the rear side. They were equipped with fish-eye lenses in order to achieve nearly omnidirectional

Camera	HFov(deg)	Focal Lenght(mm)	Base Line(cm)	Resolution(px)
Front	91	1.8	15	1920x1080
Left	175	FishEye	15	1280x720
Right	175	FishEye	15	1280x720
Down	105	2.05	15	1280x720

Table 3.1: Specifications of the equipped visual perception setup.

sensing. This type of lenses causes fewer pixels to cover the same area if compared to narrow-lens sensors. This increases the field of view at the expense of a precision loss that increases more steadily with respect to the distance. Thus these two lateral stereo cameras are suitable for short-range wide perception. A down-looking stereoscopic camera, mounted below the aircraft frame, completes the vision sensing suite. This camera always points approximately towards the region immediately below the drone. It is used for localization purposes mainly in indoor scenarios, hence needs to perceive at a middle range with reasonable accuracy. Such a choice is motivated by the assumption that for indoor application the UAV would never fly higher than approximately 2 to 10 meters above the ground. Instead, during outdoor missions the GNSS signal would be more than adequate to provide accurate localization information. A detailed specification for each stereo camera is provided in Table 3.1 while their mounting positions are better depicted in Figure 3.3.

All eight cameras mount rolling shutter sensors and are forced to capture frames simultaneously through hardware synchronization. Other communication devices comprise Wi-Fi antennas and a single Ethernet interface. Wi-Fi connection is used to send onboard computation results to ground stations for visualization purposes and to receive mission instructions.

Chapter 4

Stereoscopic camera models

As outlined in Chapter 3, the aircraft was equipped with two different kinds of stereoscopic sensors. The front-looking camera was equipped with narrow lenses and is thus suited for long-range obstacle detection. Both lateral cameras, equipped with wide-angle lenses, are meant to provide short-range perception, covering a large area around the drone. As a result, it is necessary to rely on different projective models to explain the behavior of the two types of cameras. In the first case, a pinhole model is used while a spherical camera model is better suited for the second one. A detailed description of both models will be given in the remainder of this chapter, including equations and expected theoretical errors involved in the stereoscopic re-projection process. For both camera models, the re-projective equations are used to compute the 3D position of a point, denoted as c, given its (u, v) coordinates and its disparity d, along with others camera calibration parameters that depend on the considered model. The resulting 3D point is expressed with respect to the right virtual[1] camera reference system.

[1]Virtual means that we are referring to the camera that should have been used to record the de-distorted and rectified right image.

4.1 Pinhole

The process of obtaining a point cloud from a disparity map is referred to as stereoscopic re-projection. This procedure involves using equations which differs with respect to the camera model used. The following set of equations has to be used when the incoming disparity map has been obtained from stereoscopic images de-distorted according to the pinhole camera model. A sample pair of such images is depicted in Figure 5.2. This is the case of the front-looking stereo pair, which characteristics make it suitable for the adoption of the pinhole model. Equation 4.1 is then used to compute the actual 3D position of the pixel given its (u, v) coordinates and its disparity d.

$$\begin{cases} c_x = (u - u_0) \cdot \frac{B}{d} \\ c_y = (v - v_0) \cdot \frac{B}{d} \\ c_z = f \cdot \frac{B}{d} \end{cases} \tag{4.1}$$

Where B is the baseline, i.e. the relative distance between right and left virtual camera reference systems and f denotes the focal length of the de-distorted and rectified right camera. Remaining terms u_0 and v_0 refer to the position of the optical center, expressed in pixels. As expected the disparity value is inversely proportional to the depth of a given point. For a pinhole camera model, the expected depth uncertainty can be derived by differentiating the expressions in Equation 4.1 with respect to d, leading to Equation 4.2.

$$\begin{aligned} \Delta c_x &= (u - u_0) \cdot \tfrac{B}{d^2} \cdot \delta d \\ \Delta c_y &= (v - v_0) \cdot \tfrac{B}{d^2} \cdot \delta d \\ \Delta c_z &= \tfrac{B \cdot f}{d^2} \cdot \delta d \end{aligned} \tag{4.2}$$

Where Δc_x, Δc_y and Δc_z denote the lateral, vertical and longitudinal expected uncertainties in distance measurements respectively, while δd represent the expected error involved in the disparity matching algorithm. Equation 4.2 states that the uncertainty of distance measurement is inversely proportional to the disparity squared d^2, which is nothing but the squared distance from the measured point, up to a multiplicative factor. This means that the uncertainty in position increases dramatically with the

distance of the measurements. In Figure 4.1 the uncertainty in computing the position is depicted only for the central row of the image.

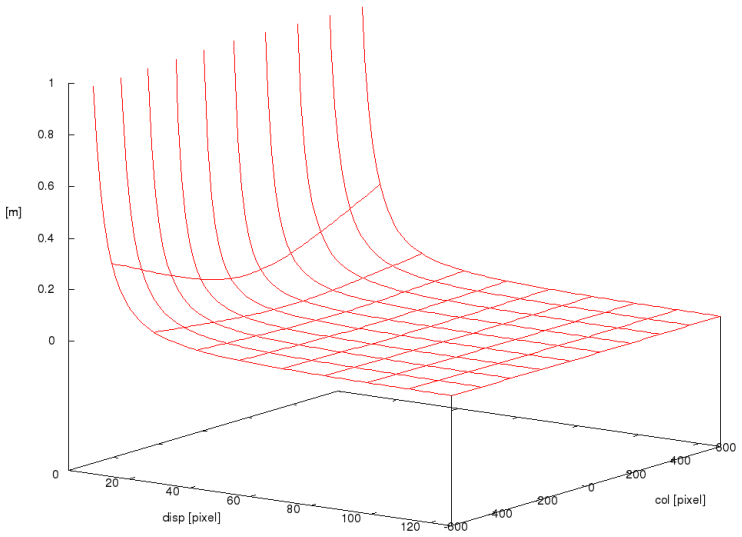


Figure 4.1: Expected depth uncertainty for a pinhole camera model with $B = 0.1$, $u_0 = 600$, $v_0 = 600$ and $f = 250$.

4.2 Spherical

In case a spherical model has been employed for de-distorting the stereoscopic images, Equation 4.3 has to be used to compute the position of a point corresponding to a pixel (u, v) having a disparity d. This is the case of lateral stereo cameras. They were in fact equipped with distortion lenses which characteristics can be better described through a spherical model.

$$\begin{cases} c_x = k \cdot \sin((u - u_0) \cdot \delta\theta_u) \\ c_y = k \cdot \cos((u - u_0) \cdot \delta\theta_u) \cdot \sin((v - v_0 \cdot \delta\theta_v)) \\ c_z = k \cdot \cos((u - u_0) \cdot \delta\theta_u) \cdot \cos((v - v_0 \cdot \delta\theta_v)) \end{cases} \tag{4.3}$$

Where k is given by Equation 4.4.

$$k = \frac{B}{\sin(\delta\theta_u \cdot d)} \cdot \cos(\delta\theta_u \cdot (u - u_0 + d)) \tag{4.4}$$

Remaining terms $\delta\theta_u$ and $\delta\theta_v$ consist respectively in the width and height of each pixel, expressed in radians. This quantity is a direct consequence of the spherical de-distortion applied to the input image. This procedure is such that the resulting image is warped on a sphere, and each pixel is thus remapped on a sector of this sphere. In this case, it is not immediate to notice how the depth of a given point depends on its disparity.
It is useful to notice that disparity d only appear in term k, in which $\cos(\cdot)$ term can be easily expanded yielding Equation 4.5.

$$k = \frac{B}{\sin(\delta\theta_u - d)} \cdot \cos(\delta\theta_u \cdot (u - u_0) + \delta\theta_u \cdot d)) \tag{4.5}$$

Then using the trigonometric addition formula for the cosine, as well as performing an appropriate regrouping of sine and cosine terms that are related to disparity values, Equation 4.6 is obtained.

$$k = B \cdot \frac{\cos(\delta\theta_u \cdot (u - u_0))}{\tan(\delta\theta_u \cdot d)} - \sin(\delta\theta_u \cdot (u - u_0)) \tag{4.6}$$

It is now quite easy to verify that the disparity is again inversely proportional to the depth of the corresponding point. The only difference is that now the inverse proportionality is not linear, as in the pinhole case, but scaled by a tangent function.
For a spherical camera model, the expected uncertainty of a perceived 3D point can be derived by differentiating the expressions in Equation 4.3 with respect to d, leading to Equation 4.7.

$$\begin{aligned}
\Delta c_x &= B \cdot \sin((u-u_0)\cdot\delta\theta_u)\cdot\cos((u-u_0)\cdot\delta\theta_u)\cdot\delta\theta_u\cdot(1+\tfrac{1}{\tan^2(d\cdot\delta\theta_u)})\cdot\delta d \\
\Delta c_y &= B \cdot \cos^2((u-u_0)\cdot\delta\theta_u)\cdot\sin((v-v_0)\cdot\delta\theta_v)\cdot\delta\theta_u\cdot(1+\tfrac{1}{\tan^2(d\cdot\delta\theta_u)})\cdot\delta d \\
\Delta c_z &= B \cdot \cos^2((u-u_0)\cdot\delta\theta_u)\cdot\cos((v-v_0)\cdot\delta\theta_v)\cdot\delta\theta_u\cdot(1+\tfrac{1}{\tan^2(d\cdot\delta\theta_u)})\cdot\delta d
\end{aligned} \tag{4.7}$$

Where Δc_x, Δc_y and Δc_z represent the expected uncertainties along lateral, vertical and longitudinal distances respectively. As usual u and v are used to describe the position of a given pixel in column and row notation, while u_0 and v_0 refer to the optical center. This formulation highlights how the uncertainty in perceiving the position of a 3D point is affected by a given error δd in the disparity matching computation. As expected, the lesser the value of the computed disparity, the higher the error in retrieving the 3D position of a point. By considering only the central row in an image, the expected depth error is depicted in Figure 4.2.

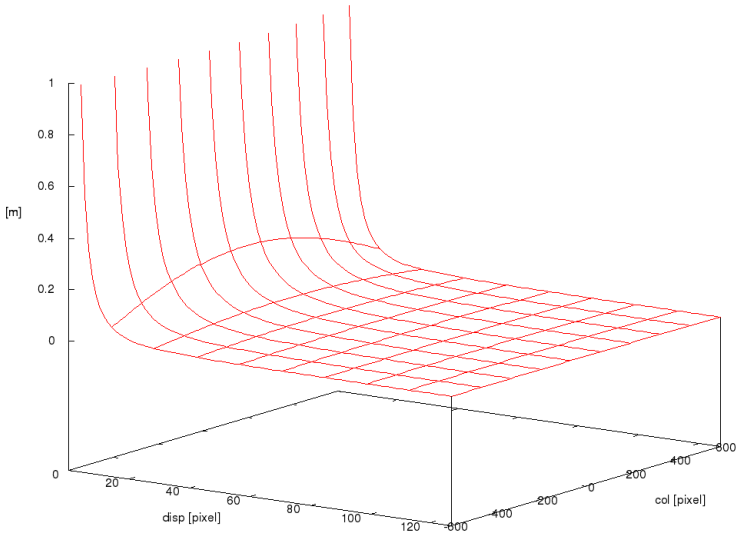


Figure 4.2: Expected depth uncertainty for a spherical camera model with $B = 0.1$, $u_0 = 600$, $v_0 = 600$, $\delta\theta_u = 0.002$ and $\delta\theta_v = 0.002$.

Chapter 5

Obstacles Detection

The obstacle detection algorithm is responsible for collecting information about the surroundings of the aircraft.
This module should provide detailed data regarding the state of the environment such that the drone would be able to fly without dangerous consequences. Before describing the obstacle detection algorithm, it would be useful to give a definition of obstacle that will better explain many of the subsequent developing choices.
It is entirely reasonable to suppose that a flying drone should consider any physical object as an obstacle. Thus anything substantial is something that the aircraft should be able to detect while flying. Each detected obstacle should then be represented as a collection of data such as its shape, its size, and its position. This kind of representation should provide subsequent path planning modules enough information to avoid dangerous obstacles while flying.
As a direct consequence of the previously given definition of obstacles follows that they cannot be detected from their appearance or their functional characteristics, but only by the fact that they are physical entities. The best detection sensor would be a contact sensing device, but for obvious reasons, it has no applicability in this project. It is then evident that a depth sensor has to be used as perception tool. Such sensing devices include laser range finders, radar and sonar sensors, among many others. In this project, a stereoscopic pair of cameras has been used. They are characterized

by low weight if compared to laser rangefinders. This is a fundamental feature when dealing with UAVs since an increased takeoff weight would result in reduced hovering time. The choice of stereo cameras is also motivated by their low price, making them employable in producing a consumer drone.
The designed obstacle detection algorithm then exploits information embedded in one or more pair of stereoscopic images to detect obstacles and build a sensible representation of the surrounding of the drone.
An overview of the proposed obstacle detection algorithm is shown in Figure 5.1. A detailed description of each step will be given in the following sections.

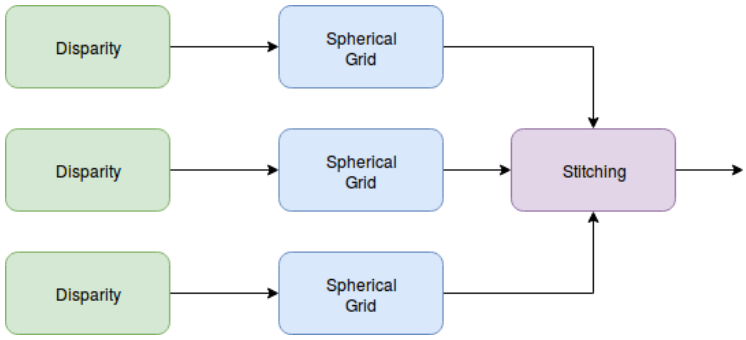


Figure 5.1: Flow chart of the obstacle detection algorithm.

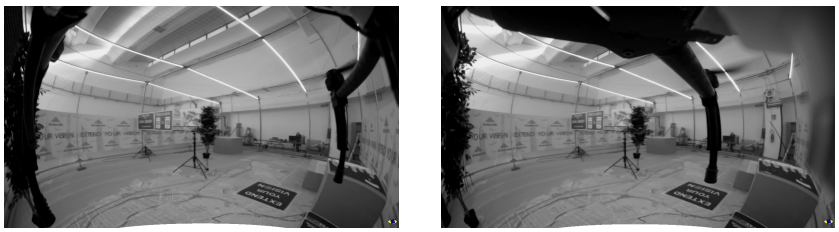
5.1 Stereo matching

A stereo camera acquires two frames simultaneously and exploits disparities between images to recover depth information. Such information was lost during the image formation process, which is a projective transformation that maps points in 3D to their corresponding 2D counterpart. It is essential to specify that more than a single 3D point could have generated each resulting 2D point. Moreover, the converse is not true, meaning that each 3D point has only one possible projection on the image plane. Incoming stereoscopic images are de-distorted and rectified according to the most

(a) (b)

Figure 5.2: Left and right de-distorted and rectified pinhole stereoscopic images.

appropriate camera model. This is such that for each point in one image its correspondent point in the other one lies on the same image row. This will restrict the number of pixels the disparity matching algorithm has to process. Corresponding pixels are guaranteed to be in the same row of both images. By exploiting this constraint, it is possible to greatly reduce the computational time required for the matching algorithm to execute.
A sample pair of stereoscopic images both pinhole and spherical are shown in Figure 5.2 and 5.3 respectively. Disparities from left and right frames are computed using



(a) (b)

Figure 5.3: Left and right de-distorted and rectified spherical stereoscopic images.

Semi-Global Matching algorithm [59], also referred to as SGM. This approach provides dense disparity maps if compared to classical correlation based stereo estimation algorithms, meaning that nearly all pixels have been assigned a disparity value.
A sample result also referred to as a disparity map, is depicted in Figure 5.4. It is essential to specify that the disparity map is aligned with the right image.

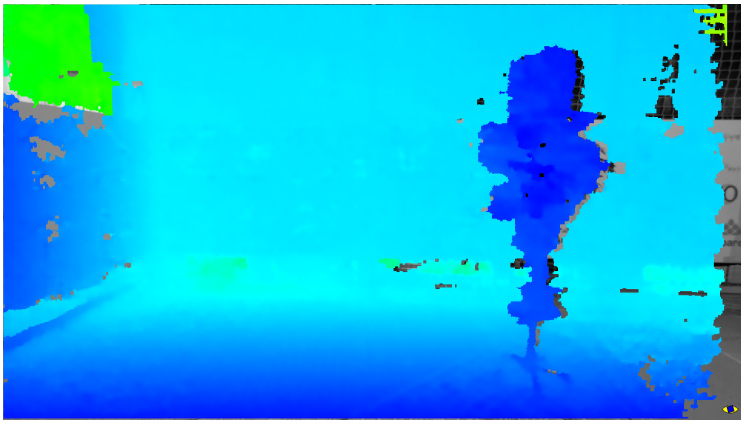


Figure 5.4: Disparity map aligned with the right stereoscopic image.

In this case, both input images were down-sampled by a factor of two, before applying the disparity estimation algorithm. This is done to increase the local smoothness of the result at the expense of doubling the expected estimation error with increasing distances.

A disparity matching algorithm has thus been used to recover 3D information regarding the world, by comparison of two bi-dimensional representation of it. The resulting map hence contains a disparity value for each pixel of the right de-distorted and rectified image.

A map like this will be considered as the primary input for the obstacle detection algorithm. As mentioned in Section 3 the aircraft has been equipped with three stereoscopic cameras, each of which will produce a disparity map. Each of these maps will be independently analyzed by the following step, leaving the data fusion for a subsequent part of the algorithm.

5.2 Preprocessing steps

Before actually exploiting the depth information stored in a disparity map, some preprocessing steps are performed. They mainly consist of removing the disparity pixels that, for different reasons, cannot be found in both the left and the right image.

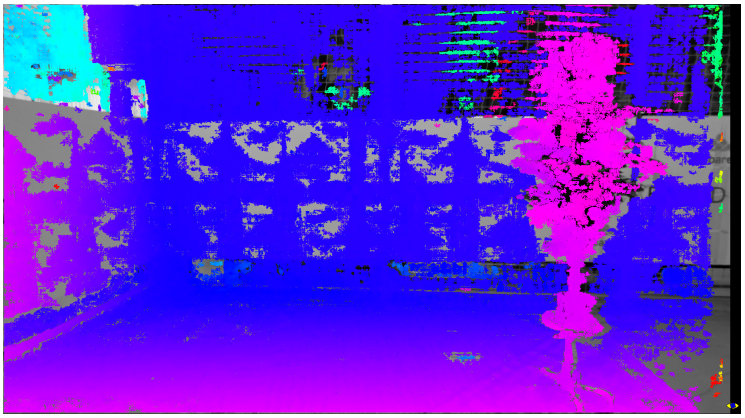


Figure 5.5: Example of false positives in a disparity map.

This is the case, for example, of pixels that are on the boundary to the field of view of the camera. These pixels are not present in both images, and the matching algorithm cannot succeed in finding their correct depth. In most cases, these pixels are assigned an invalid disparity value, but sometimes the matching algorithm finds an incorrect match and estimates a wrong disparity for those pixels.

This may be due to a particular scenario configuration. For example, if a pixel in the right image is not present in the left image, but there are candidates with a similar neighborhood in the proximity, it is possible that a wrong disparity value is estimated for that pixel. This is the case where a false positive has been introduced in the disparity map, as depicted in Figure 5.5.

In such example, wrong disparity values have been assigned to a large number of pixels in the top-right corner of the image.

This is because the matching algorithm has found correspondences for pixels in the left image, despite them being outside of the left camera field of view. The repetitive pattern of a net has been probably responsible for the wrong match since the neighborhood of a pixel in the right image is likely to be similar to more than one pixel in the left image.

This kind of problem is usually addressed by removing the right-most part of the disparity map since it usually contains invalid values or false positives. The area of

Figure 5.6: Example of false positives removal in a disparity map, using $rcrop = 64$.

the right image that does not overlap with the left image depends on the depth of the pixels, which is not known apriori. Therefore an arbitrarily chosen number of columns are discarded in this process, according to Equation 5.1.

$$d_{i,j} = \begin{cases} d_{i,j} & \text{if } (width - j) <= rcrop \\ d_{invalid} & \text{otherwise} \end{cases} \tag{5.1}$$

Here i, j represent rows and columns respectively and $width$ is equal to the total number of columns. A sample disparity map obtained by applying the described preprocessing step is depicted in Figure 5.6.

A similar kind of false positives is due to occlusions in only one of the two images. If an object is close to the stereoscopic camera, such that only one of the two images is affected by its presence, occlusion issues could arise. In such a situation a certain number of pixels, just as before, cannot be observed in both images and their depth cannot be recovered. This can sometimes lead to false positives or incorrect disparities retrieval during the stereo matching process. An example of this circumstance is shown in Figure 5.7.

In this case, erroneous disparities are removed using a masking approach. A binary map is used to specify, for each pixel, if a disparity for that pixel can be considered valid or not. Such a mask is computed by taking into account for both the position

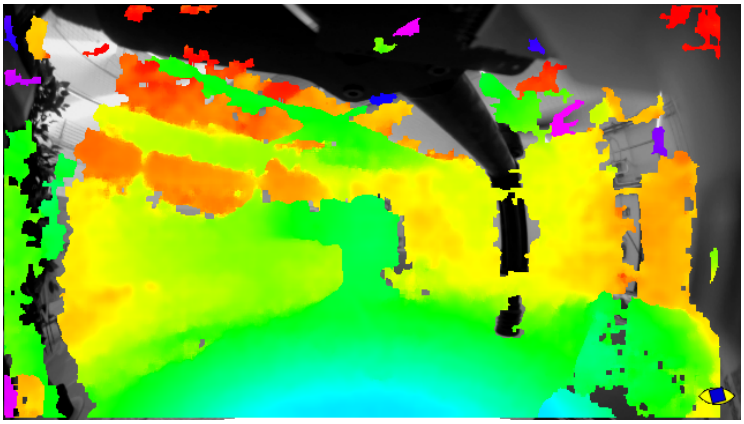


Figure 5.7: Example of false positives due to occlusions in a disparity map.

of the camera and the shape of the aircraft and all of its components. In particular, parts of the aircraft that are a source of occlusion for one or both the cameras, are considered.
For this project, a set of rectangular-shaped regions of interest (ROIs) have been defined for each stereo camera, in order to filter out specific parts of the image. A pixel is thus considered as valid if it does not fall in any of those previously defined ROIs. A sample result of this step is depicted in Figure 5.8.
Both lateral cameras are equipped with wide-angle lenses, which should make them capable of simultaneously perceive a big portion of the environment. However, in this case, their field of view is substantially limited by occlusions.
The disparity map, refined and filtered, is used as the primary input for the next step of the obstacle detection algorithm.

5.3 Spherical grid accumulation

This stage aims at storing 3D information contained in a disparity map in a 2D spherical grid. Such grid has its origin coincident with the drone body reference frame. Efficiency reasons led to this choice, as will be better described later. This step aims at reducing the amount of data that subsequent steps will have to process while preserv-

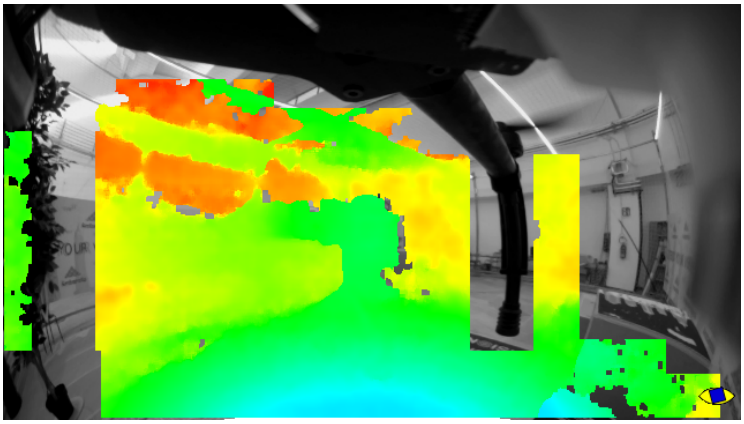


Figure 5.8: Example of false positives removal in a disparity map, using a binary mask.

ing information that is sensible enough to detect obstacles accurately. This is because the incoming disparity maps usually contain many points, and to process all of them would be almost infeasible.

As stated before, the input of this algorithm is a disparity map, which contains 3D information regarding the right image pixels. The actual 3D position of a pixel can be calculated utilizing an equation that depends on the chosen camera model. Further details on this procedure are given in Chapter 4. The entire disparity map is thus transformed in a corresponding 3D point cloud.

The next step involves the accumulation of the obtained point cloud into a spherical grid. This grid is organized such that it covers a given portion of the space around the drone. The area that is taken into account is defined by the extrinsic properties of the camera, i.e., its mounting position and orientation. The discretization of the grid is only angular, resulting in a two-dimensional grid. In particular azimuthal ϕ and polar θ angles are taken into account.

Grid coverage is then chosen by defining a starting and an ending point for the azimuthal angle (ϕ_{min} and ϕ_{max}) as well as a starting and an ending point for the zenith angle (θ_{min} and θ_{max}). Grid resolution is then defined by angular steps on both angles ($\delta\theta$ and $\delta\phi$). An example of such grid is depicted in Figure 5.10.

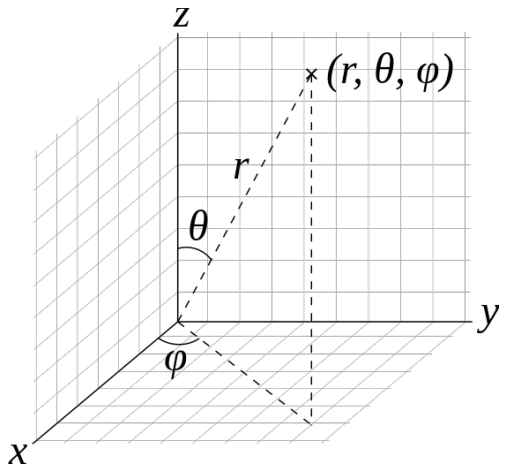


Figure 5.9: Graphical illustration of the spherical reference system.

This configuration will end up in a spherical grid covering an area such that almost the entire perceived point cloud will be projected onto the grid surface. Each cell will then represent a solid angle in the 3D world, and is meant to store a depth value.
In order to perform the point cloud projection, it is necessary to process the previously computed point cloud, changing its representation from cartesian to spherical. Each point, defined by a vector (x, y, z) will be converted into a corresponding (ρ, θ, ϕ) vector, using non-linear Equation 5.2.

$$\begin{aligned} \rho &= \sqrt{x^2 + y^2 + z^2} \\ \theta &= acos(z/\rho) \\ \phi &= atan2(y, x) \end{aligned} \tag{5.2}$$

In this notation, ρ represent the depth of the point with respect to the body of the drone, while (θ, ϕ) are respectively the polar and the azimuthal angles of the vector, as depicted in Figure 5.9.
Then each point, expressed in spherical notation, will be projected into a single grid cell, depending on its (θ, ϕ) coordinates. A linear mapping on (θ, ϕ) is used to retrieve, for each spherical point, in which cell it is projected. Therefore only ρ is stored in the destination cell. At the end of this process, each cell (i, j) will contain a set D of

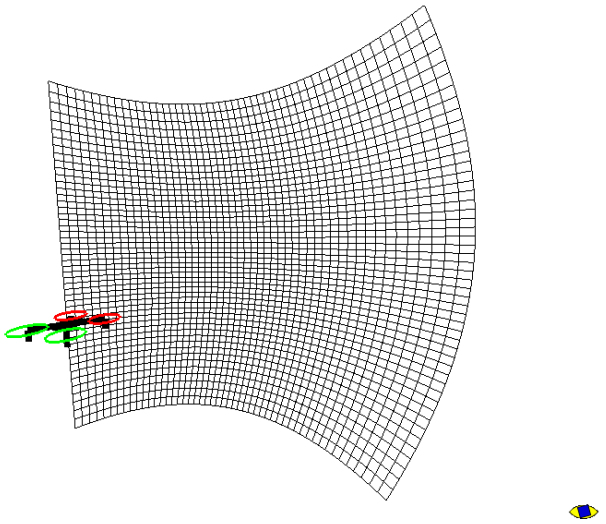


Figure 5.10: Spherical grid with $\phi_{min} = -50°, \phi_{max} = 50°, \theta_{min} = 40°, \theta_{max} = 140°$, $\delta\theta = 2°$ and $\delta\phi = 2°$.

depth measurements.

$$D_{i,j} \in \{d_0, d_1, \ldots, d_{n-1}\} \tag{5.3}$$

For each cell, an overall depth measurement is then estimated from D. This is initially done by computing average and standard deviation statistics over set D. Finally, implicitly considering each cell as independent, the most dangerous depth measure is estimated using Equation 5.4.

$$depth = \begin{cases} \mu_D - 3\sigma_D, & \text{if } \sigma_D < \sigma_{th} \\ \min_{d_i \in D} d_i, & \text{otherwise} \end{cases} \tag{5.4}$$

The idea is to retrieve a robust minimum estimate from the average depth value when measurements inside a single cell are reliable. Otherwise, the minimum depth value is taken as a raw estimate. This is usually the case when measurements from multiple surfaces are projected onto the same cell.

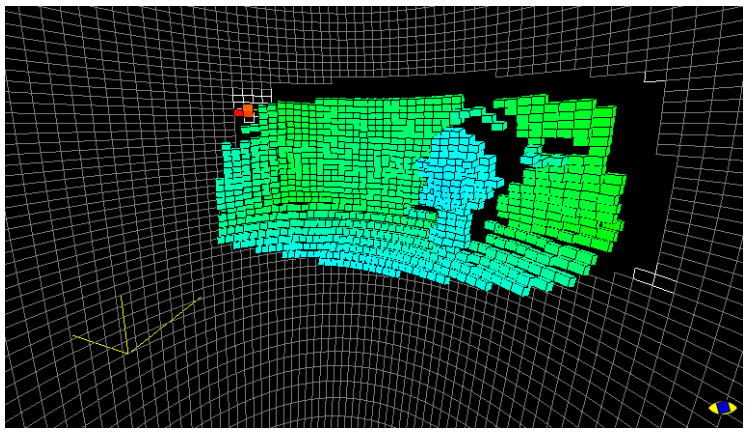


Figure 5.11: Final result of the spherical grid accumulation step.

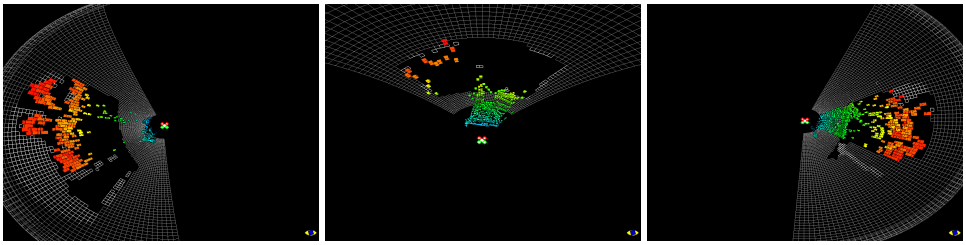
An example of this representation is depicted in Figure 5.11. This is the spherical grid obtained accumulating depth measurements incoming from the disparity map depicted in Figure 5.4. In this representation different colors are used to represent various levels of depth.

It is worth mentioning that this representation does not depend on the camera model used for the interpretation of the input disparity. Thus whether the input stereo camera is the front-looking one or the lateral ones, this output datum can be interpreted in the same manner. The next step is responsible for merging multiple spherical grids, incoming from different stereoscopic pairs.

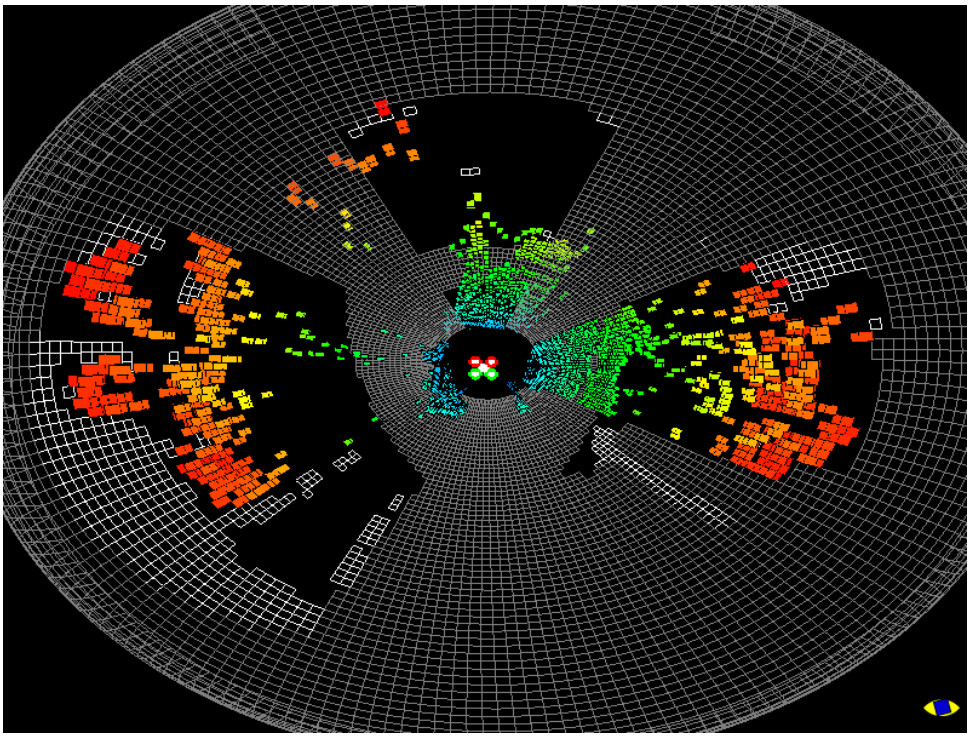
5.4 Spherical grids stitching

A stitching procedure aims at fusing single spherical grids, in order to build a single homogeneous representation of the surroundings of the drone. Overlapping regions, as well as non-visible areas, are taken into account in this process. Since every input grid is described in the same reference system, it is quite easy to compare them. Besides, they are configured to have identical angular steps, and if their boundaries are chosen carefully, their stitching can be further optimized. In case multiple cells

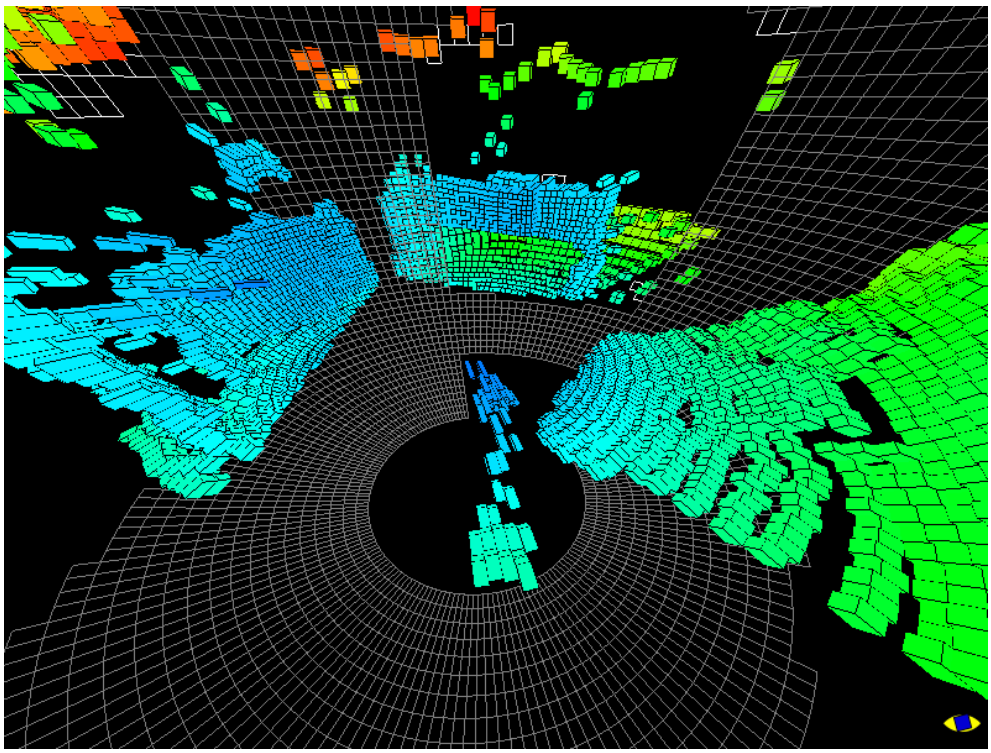
are overlapping, the nearest measurement is taken into account, to ensure that the depth of the most dangerous obstacle is not underestimated. The result of this stage is a bigger spherical grid, covering the entire area around the drone. For each angular sector of this grid, depth values are stored when available.
This stage is better illustrated in Figure 5.12. Starting from the top separate spherical grids are shown, incoming respectively from the left, front, and right camera. Then a resulting stitched grid is depicted. Single spherical grids, as well as the stitched result, are shown from above to highlight the purpose of the stitching. In contrast, a perspective view of the final result is shown in Figure 5.12e. Since lateral cameras have a bigger field of view, they cover a wider area if compared to the front camera.
The following step will exploit measurements embedded in this spherical grid to build a different representation of the environment. This result will be considered an intermediate outcome for the remainder of this text, but it would be appropriate to consider it even as a final result. The information stored in this grid is ready to be processed by other systems. For example, this spherical representation could be used to design an automatic safety system.
In this case, a hypothetical pilot, whether human or not, would be prevented from flying the UAV into some obstacle. This grid would provide the system with information regarding the closest obstacles for each possible direction, something similar to a tri-dimensional all-round view. The system would then only have to check if the chosen flying direction does not go through an obstacle, hence it is safe, or not.
In this text, however, we will consider the stitched spherical grid as an intermediate step for subsequent obstacle detection and mapping refinement.



(a) (b) (c)



(d)



(e)

Figure 5.12: Spherical grid stitching step. Spherical grids from left (a), front (b) and right (c) cameras are stitched together in a single homogeneous grid, depicted from above (d) and in third person view (e).

Chapter 6

Obstacles Mapping

Separate obstacles parts were detected during the previous steps, leading to a representation of the surroundings of the drone. In this representation, each grid sector is considered independently and contains the depth of the closest object. Thus consists of a bi-dimensional description of the world.
This stage, however, aims at building a 3D representation of the world, taking into account for subsequent measurements, as well as their intrinsic uncertainty. Considering each spherical cell as a collection of measurements, they will be back-projected into a 3D probabilistic representation of the world. A detailed description of this back-projection procedure will be given in the remainder of this section.

6.1 Occupancy grid mapping

In order to obtain both an efficient and theoretically valid solution for the obstacles mapping problem, an occupancy grid mapping algorithm [60] has been employed. Such an approach aims at estimating the posterior probability distribution over a map, given a series of measurements. In this case, the idea is to estimate the occupancy state of each cell, given a set of measurements and the location on the map where each of them was collected. For tractability reasons, each cell is treated independently during the estimation process, which leads to the problem being decomposed into multiple

simpler to solve problems, as stated in Equation 6.1.

$$p(m|z_{1:t}, x_{1:t}) = \prod_i p(m_i|z_{1:t}, x_{1:t}) \tag{6.1}$$

Where m denotes the map, and m_i represents a single cell. The set of measurements collected up to time t is denoted as $z_{1:t}$ and $x_{1:t}$ represents the locations on the map where each of them was collected. Following the derivation in [61] the occupancy probability is updated, in log odds fashion for better numerical stability, as in Equation 6.2.

$$l_{t,i} = l_{t-1,i} + \log \frac{p(m_i|z_t)}{1 - p(m_i|z_t)} - l_0 \tag{6.2}$$

This formulation describes how a new measurement z_t affects the state of a cell, updating its prior $l_{t-1,i}$ into the posterior $l_{t,i}$. The constant l_0 is usually referred to an initial belief, before any measurement. Assuming a starting condition of maximum uncertainty, i.e., $p(m) = 0.5$, this term is discarded from the equation. Given the definition of log odds, it is quite straightforward to recover the actual occupancy probability for a cell, using Equation 6.3.

$$p(m_i|z_{1:t}, x_{1:t}) = 1 - \frac{1}{1 + \exp\{l_{t,i}\}} \tag{6.3}$$

By observing Equation 6.2, it is quite obvious to identify the central term in the summation as the core of this formula. The term at numerator $p(m_i|z_t)$ is referred to as the inverse sensor model since it estimates the occupancy probability of a cell m_i given that a particular measurement z_t has been collected. The inverse sensor model is thus a function that has to be designed to modeling how a given sensor perceives the world. The design of a function modeling the perception of a stereoscopic camera will be detailed in the following section.

6.2 Inverse sensor model

Inverse sensor models provide a probability distribution over a set of possible world states, given a perceived measurement. Intuitively, they can be used to estimate the

occupancy probability given that a specific measurement has been collected. Considering the detection process of a stereoscopic sensor, it is reasonable to assume that if an obstacle has been perceived at a given distance, hence no obstacles are present between the observer and the perceived obstacle, and nothing can be said about what lies behind it. A sensor model is a function that embeds these concepts, by simultaneously taking into account how much uncertainty affects the perceived quantities.
In this section, a stereoscopic inverse sensor model will be presented in general terms, while the influence of the perception uncertainty will be discussed subsequently. Employed inverse sensor model, inspired to [36], should provide an occupancy probability for each cell m_i given a depth measurement z_t. In this case, measurements are obtained during steps explained in Section 5.3. They consist of a collection of depth values, each one of them stored in a spherical cell. For each cell, its angular coordinates (θ, ϕ) represent a ray exiting from the center of the drone and passing through the point defined by the vector (ρ, θ, ϕ) where ρ is the distance stored in that cell. The depth contained in each cell will not be subsequently referred to as ρ but more generally as a measurement z.
Defined sensor model will provide occupancy probabilities for each map cell traversed by that ray. According to the previously defined sensor behavior, the occupancy should be maximum at the point where the measurement falls, while decreasing before and after that point in different manners. Hence Equation 6.4 is used as inverse sensor model.

$$p(m_i|z_t) = \begin{cases} p_{free} + (a + p_{unknown} - p_{free})e^{-\frac{1}{2}(\frac{m_i - z_t}{\Delta z})^2}, & \text{if } 0 < m_i \leq z_t \\ p_{unknown} + ae^{-\frac{1}{2}(\frac{m_i - z_t}{c})^2}, & m_i > z_t \end{cases} \tag{6.4}$$

Where p_{free} and $p_{unknown}$ are constant values. The former represents the occupancy probability of a free and cell and is the value to which the sensor model asymptotically tends for cells lying before the measurement. The latter defines the occupancy probability of an unobserved cell and is the value returned by the sensor model for cells standing beyond the measurement. Term a, as stated in Equation 6.5 controls the height of the peak, which is always located where the measurement falls and is inversely proportional to the depth of the measurements.

$$a = k(1 - p_{unknown})e^{-\Delta z} \tag{6.5}$$

Where $0 \leq k \leq 1$ is a constant weighting factor, which defines how important the measurement is. Finally, term c is responsible for making both Gaussian tails symmetric around the peak, as shown in Equation 6.6.

$$c = \sqrt{\frac{\ln(1 + \frac{a}{a+1-2p_{free}})}{\ln 2}}\Delta z \tag{6.6}$$

In this project, two different kinds of stereoscopic sensors have been used. The front-looking stereo pair mounts narrow-lenses, and thus a pinhole camera model is used to describe its projective behavior. Besides, lateral stereo pairs were equipped with wide-angle lenses, and a spherical camera model is better suited at modeling its behavior. Hence different formulations of Δz have to be used in the equations as mentioned above. As described in the following sections using a pinhole or spherical camera model changes the expected perception uncertainty, and both cases will be further detailed.

6.2.1 Pinhole

The previously described model contains a term, Δz, which refers to the expected disparity re-projection error along the longitudinal axis z. As outlined in Section 4.1, the expected uncertainty of this computation depends not only on the disparity d but also on the pixel coordinates (u, v). However, by observing Figure 4.1 it can be seen how the main contribution comes from the disparity term. For this reason, the expected uncertainty can be approximated to a formulation that only depends on the disparity value. Furthermore, if we consider only the central row and column in the image, Equation 4.2 collapses on the term which only depends on z, yielding to Equation 6.7.

$$\Delta z = \frac{B \cdot f}{d^2} \cdot \delta d \tag{6.7}$$

This formulation is not directly usable because in each spherical cell only depth information is stored. Considering Equation 4.1 and taking into account the formulation of c_z, it can be easily obtained Equation 6.8.

$$d = \frac{z^2}{B \cdot f} \tag{6.8}$$

Where z has been used instead of c_z in the original equation. Substituting this equation in the expression d of Equation 6.7 leads to Equation 6.9.

$$\Delta z = \frac{z^2}{B \cdot f} \cdot \delta d \tag{6.9}$$

Where δd represent the expected disparity error involved in the stereo matching algorithm and is thus a constant value. This represents an approximated estimation of the uncertainty of a depth measurement given its depth z and camera parameters B and f. Obtained formulation of the uncertainty can now be used in Equation 6.4, leading to an inverse sensor model suited at modeling the perceiving behavior of a stereoscopic pinhole camera. Resulting occupancy probabilities for different map locations, given some distinct measurements, are depicted in Figure 6.1.

6.2.2 Spherical

In case a spherical model has been used, a different formulation for the stereo re-projection error must be considered. Following what discussed in Section 4.2, the expected uncertainty depends on the disparity error by means of Equation 4.7. Thus a precise estimation of the re-projection error should take into account for both disparity d and pixel position (u, v). For optimization purposes, however, the dependence on (u, v) has been discarded, leading to a formulation of the error as a function of d. Thus only the central row and the central column have been considered. Under this assumption ($u = u_0$ and $v = v_0$) Equation 6.10 approximates the expected depth uncertainty.

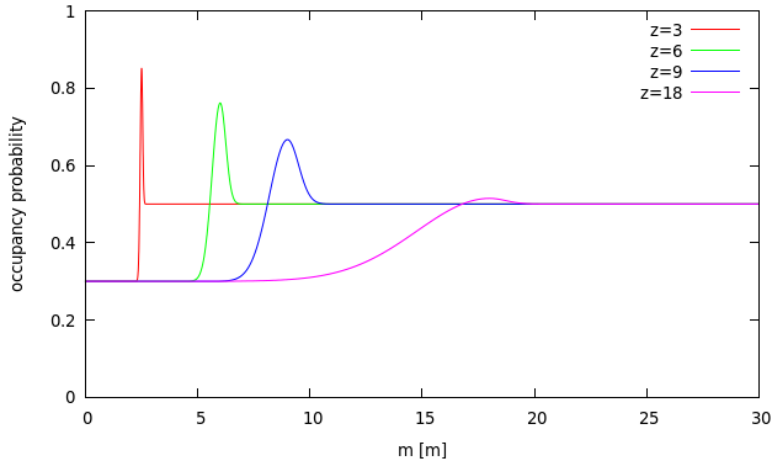


Figure 6.1: Pinhole camera inverse sensor model computed for different measurements with $B = 0.1$, $f = 500$, $k = 0.75$, $p_{unknown} = 0.5$ and $p_{free} = 0.3$.

$$\Delta z = B \cdot \delta\theta_u \cdot (1 + \frac{1}{\tan^2(d \cdot \delta\theta_u)}) \cdot \delta d \tag{6.10}$$

For motivations similar to those described before, this expression cannot contain a disparity term d. Thus recalling the last term of Equation 4.3 an approximated formulation of d with respect to z can be obtained, as stated in Equation 6.11.

$$d|_{(u=u_0)\wedge(v=v_0)} = \arctan(\frac{B}{z}) \cdot \frac{1}{\delta\theta_u} \tag{6.11}$$

Obtained formulation of the uncertainty can now be used in Equation 6.4. Resulting sensor model can be used for describing the perceiving behavior of a stereoscopic spherical camera. Figure 6.2 depicts this model reporting occupancy probabilities for different map locations, given a set of possible measurements.

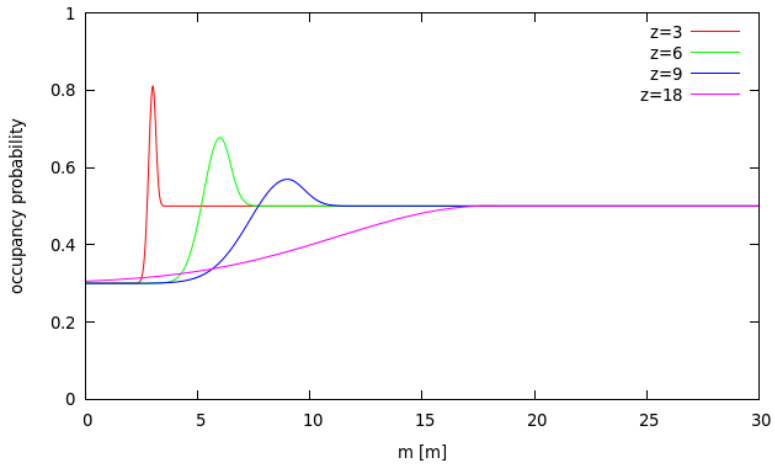


Figure 6.2: Spherical camera inverse sensor model computed for different measurements with $B = 0.1$, $\delta\theta_u = 2.5/1200$, $k = 0.75$, $p_{unknown} = 0.5$ and $p_{free} = 0.3$.

Chapter 7

Environment Representation

Obstacle detection and mapping provide occupancy information regarding a given portion of the world. The resulting map needs to be described efficiently, depending on the requirements of the application. In this section, the chosen environment representation will be described in detail. The nature of this project requires the map to describe the world in three dimensions, and a cartesian notation has been chosen. Both map boundaries and resolution are thus defined with respect to cartesian axes x, y, and z. In this notation, x, y and z refer to vectors pointing respectively forward, to the left and upward. A set of configuration values comprising minimum, maximum and resolution measures along each axis then define a grid map.
Two approaches have been explored, which mainly differ on the reference frame chosen to describe the map content:

- Global map: its content is always described with respect to an inertial reference frame, independently of the current drone position;

- Local map: its content is always described with respect to the drone reference frame.

Further details about each method will be detailed during the following sections. Figure depicts some reference frames involved in the subsequent explanations.

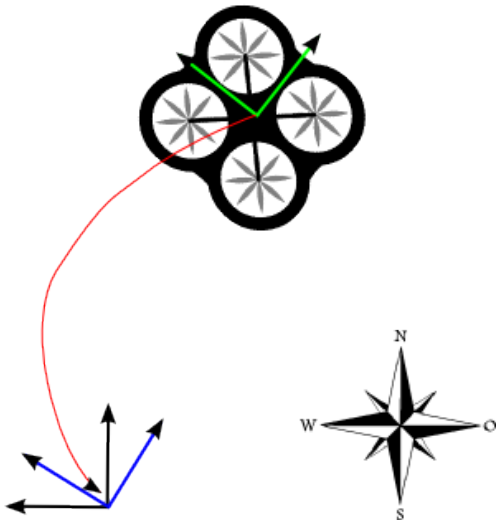


Figure 7.1: Obstacle map reference systems: drone body frame (green), initial drone pose (blue) and north-aligned inertial reference frame (black).

7.1 Global map

A map whose content is described with respect to an inertial frame is known as allocentric, or global. This is because its content is always described with respect to a reference frame remain fixed, independently on the movement of the exploring robotic platform. As reference system it is usually chosen the UAV starting position and orientation, also referred to as the inertial frame. Alternatively, it can be used a reference frame that is north-oriented, preserving the global property of the map. During exploration, the drone moves inside the map collecting measurements that are natively described with respect to the drone reference system. Those measurements cannot be directly used to update the map, because they are referred to a different reference system. They have to be transformed such that they are described with respect to the map reference system. This consists in a roto-translation of each measurement z, as stated in Equation 7.1.

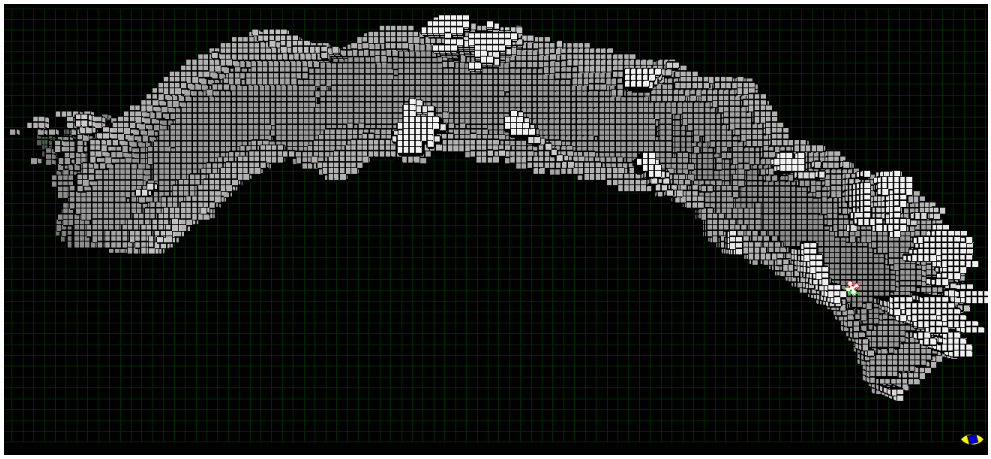


Figure 7.2: Global map configured with $x_{range} = 100\,\text{m}$, $y_{range} = 40\,\text{m}$, $z_{range} = 10\,\text{m}$ and $x_{step} = 0.5\,\text{m}$, $y_{step} = 0.5\,\text{m}$, $z_{step} = 0.5\,\text{m}$.

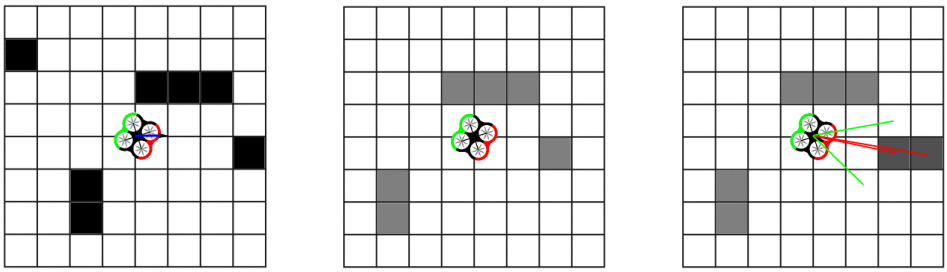
$$^{M}z = {}^{M}T_{B} \cdot {}^{B}z \tag{7.1}$$

Where ${}^{M}T_{B}$ is a 4x4 matrix representing the drone body pose with respect to the map reference system. Such transformation is depicted in Figure 7.1 with a red arrow. For this reason, both the position and the orientation of the drone with respect to the map are required. In this project, this information is obtained exploiting both navigation sensors, such as accelerometers or gyroscopes, and visual stimuli. A sensor fusion module is used to obtain the drone pose with respect to the map.
Once the measurements are described in a global reference frame, the occupancy grid mapping algorithm from Chapter 6 can be applied to build a map.
An example of a map built during an exploration flight is represented in Figure 7.2. In this case, the map reference frame is chosen to be coincident with the drone starting position and orientation. The map boundaries, as well as the cell resolution, are chosen with respect to the requirements of the navigation plan. Each map cell stores only occupation information, which can be represented with reasonable precision using 2-bytes. This means that the map depicted in Figure 7.2 requires at least 625 KB to be stored.

Advantages of this approach are the easiness of implementation and the possibility of defining apriori the area that has to be mapped during exploration. This is chosen when configuring the map limits. Using an allocentric map presents some drawbacks. For example, if a long distance flight is expected, then a large map is required to cover the entire area of exploration. In any case, depending on the chosen flight path, some areas in the map could be explored or not. For instance, if a long straight flying path is chosen, the majority of the map would not be explored, leading to a significant waste of memory. Furthermore, the drone cannot leave the explorable area defined by the map contours, which has to be chosen apriori, resulting in a limited navigation range. Another disadvantage is that the actual drone pose has to be precisely defined for each collected set of measurements. These disadvantages led to the adoption of a locally defined, or egocentric, occupancy grid map.

7.2 Local map

This approach is based on keeping the map as egocentric, meaning that the grid content is always described with respect to the current drone position. That means that the center of the map always contains the current drone position. Because of this the occupancy information conveyed by the map always regards a portion of the environment near to the drone itself. The idea is then to represent only a small portion of the environment, but to choose it as the area of highest interest. It is reasonable to assume that the area which has the highest importance for a robot, in an obstacle detection system, is the region in the immediate surroundings. An illustrative example of such representation is depicted in Figure 7.4. A stated before, the current drone position always lies in the central cell of the map, but its orientation is not constrained. The map is only partially egocentric, meaning that the map and the drone reference systems are not entirely coincident. This setup is such that the orientation of the map reference system is fixed, and it can be either coincident with the drone starting orientation or with the North direction. In any case, the drone orientation changes during flight, but this does not affect the map orientation. Thus the drone and the map reference systems are coincident with respect to their position but not with their



(a) (b) (c)

Figure 7.3: Local occupancy grid mapping steps. (a) The previously computed map and the drone movement are taken into account. The map is then (b) predicted and (c) updated with new measurements.

orientation, similarly to what happens with a compass and its needle. In contrast, the position of both the drone and the map are kept coincident, while the drone moves. This means that the egocentric property is maintained by adjusting the map content accordingly to the drone movement, in a step referred to as map prediction. This is only done when the drone exits from a cell to reduce the computational burden at the expense of introducing a minimal parallax error. After the map prediction, the newly collected measurements can be used to enrich the map content, during an update step. These properties are such that the grid is maintained small and fixed in size, as well as computationally efficient to be updated. The grid is however configured to be big enough, ensuring that local trajectory planning is always feasible at the maximum drone speed. The local occupancy grid mapping algorithm thus comprises a prediction and an update step, as depicted in Figure 7.3.

7.2.1 Prediction

The prediction step is responsible for maintaining the egocentric property of the map. It takes as inputs the occupancy grid estimated during the previous step and the current drone position. A prediction step has to be performed only if the current drone position is such that it exits from the central cell of the map. In this case, the entire map content must be transformed according to a translation that recovers from the drone movement. For example in Figure 7.3a, the drone moves towards East such that it exits from its cell and enters the adjacent cell to the right. Assuming that all the obstacles are static, they have to be moved accordingly towards the West by an equal amount. Thus a map such as the one depicted in Figure 7.3b is obtained from the prediction step, where each cell has been translated to the left. The resulting predicted map is then ready to be updated according to the newly collected measurements.

7.2.2 Update

After the prediction step, the current set of measurements have to be used to update the map content. The predicted map contains occupancy information from the surroundings of the drone, described in a reference system that, as stated, is partially local. This means that the drone reference frame is coincident with the center of the map, but they are not necessarily aligned. However, the measurements are described in the drone reference system. Furthermore, the aircraft attitude must be taken into account during this process, since the map described in a leveled reference frame. A level reference system is such that two of its axis are parallel while the third one is orthogonal to the earth surface. All these properties of the map imply that the set of perceived measurements has to be transformed into the map reference system before performing the update step. Given a specific measurement z_i described in the drone reference system, denoted as B, it has to be roto-translated by taking into account the drone current orientation. This can be done with Equation 7.1, but in this case the matrix ${}^{M}T_{B}$ contains only the rotation of the drone with respect to the map and not its position. Once the set of measurements has been transformed, the map can be updated directly. As described in Section 6, the appropriate inverse sensor model

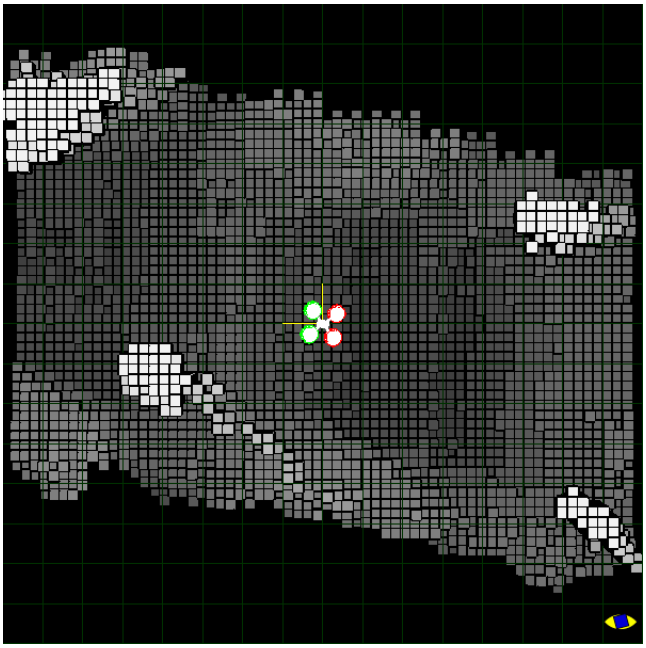


Figure 7.4: Local map configured with $x_{range} = 8\,\text{m}$, $y_{range} = 8\,\text{m}$, $z_{range} = 5\,\text{m}$ and $x_{step} = 0.25\,\text{m}$, $y_{step} = 0.25\,\text{m}$, $z_{step} = 0.25\,\text{m}$.

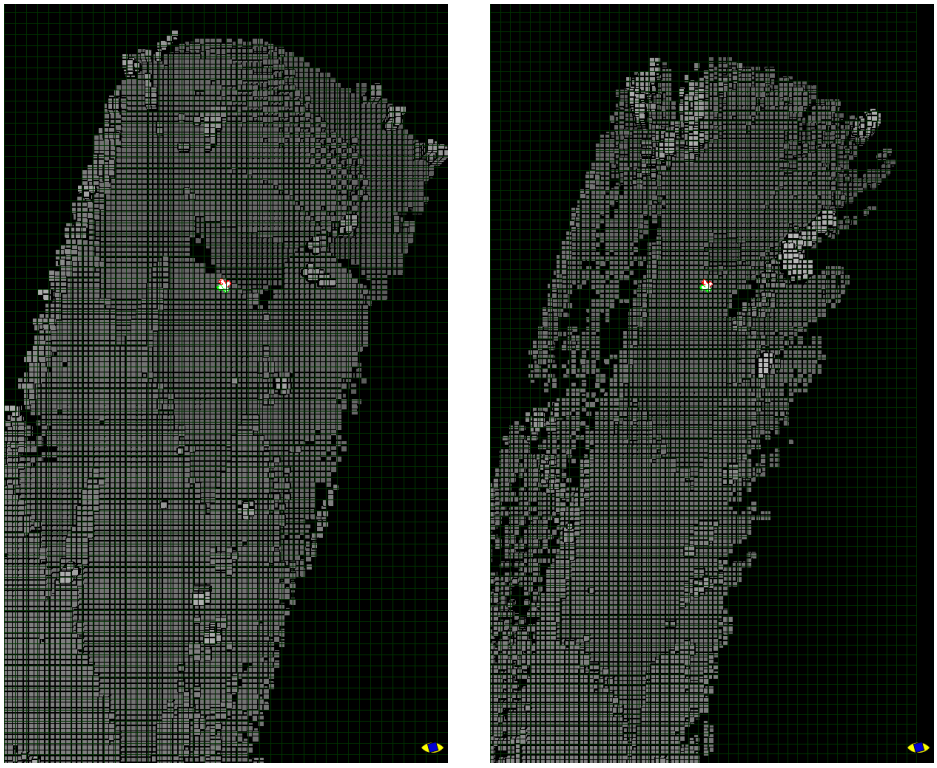
is used to update the map by taking into account for the measurements. In Figure 7.3c, are depicted four cells that are updated based on the measurements. There are two cells, highlighted with green rays, that are perceived as free and their occupancy probability is reduced. The remaining two cells are perceived as occupied but different explanations must be given for each of them. The leftmost cell was previously classified as occupied, and the current measurement confirms this estimation, leading to an increased occupancy probability. In contrast, the other one is considered as unknown apriori, because it just entered into the grid cell during the prediction step. The current measurement, however, states that this cell is occupied and its occupancy probability increases.

Chapter 8

Results

In order to obtain quantitative results a supposedly correct occupancy grid map of an explorable area is needed. If such data is available, the map obtained with the proposed system can be evaluated by simple comparison with the ground truth map. In the early stages of this project, a ground truth was obtained within a simulation framework, consisting of a supposedly correct occupancy grid map. The simulation engine was able to reproduce the perception process of the equipped stereo cameras, which resulted in the acquisition of stereoscopic frames by each of the cameras mounted on-board.
The proposed obstacle detection pipeline was then applied to the acquired pair of images for each camera, and an occupancy grid map was obtained by following the steps depicted in Chapters 5, 6 and 7. This approach has been however dropped out since synthetic images are usually very simplistic and the complex process of image formation is often poorly reproduced.
Another approach, inspired to [62], has been used to tackle the laking of a ground truth map. Such approach aims at building an exact map of a given environment exploiting both precise measurements and manual annotations, which are requirements met by freely available frameworks.
In this case, Kitti object tracking dataset, which is introduced in the next section, has been used effectively. A reliable ground truth has been then constructed for each

sequence of the dataset. Collected ground truth maps have been then compared to the corresponding maps obtained using the proposed system.
The evaluation is carried out by comparing the ground truth map and the obtained map, shown for the same time frame in Figure 8.1.



(a) (b)

Figure 8.1: Map evaluation process. A ground truth map (a) is compared to a map (b) obtained using the proposed method, by computing several evaluation metrics.

8.1 Dataset

The Kitti object tracking dataset [1] provides a set of annotated sequences commonly used for evaluating object detection and tracking systems. In this case, the dataset has been used for a different purpose, confirming its intrinsic utility and extreme value.
This dataset provides, for each frame, a pair of stereoscopic color images and a laser point cloud acquired by a Velodyne HL64-E [2]. Sensors synchronization is such that when the rotating sensing unit of the laser scanner reaches approximately the central position, a triggering signal is sent to the cameras. This will ensure that quantity regarding the same region of the environment are perceived by the camera as simultaneously as possible to the laser scanner.
The static pose of each sensor, as well as the inertial navigation system (INS), is provided with respect to a given camera. Also, the INS frame pose is available at each frame with respect to an inertial navigation frame.
The entire sequence has been annotated with up to eight classes of objects, and both 2D and 3D bounding boxes are available for each object. Annotated objects comprise pedestrians, cycles, motorcycles, cars, vans and so on. Further details on the dataset and how to use it can be found in [63].

8.2 Ground truth building

As stated before, a ground truth building approach has been employed to construct a supposedly accurate representation of a given environment.
In order to do so, exact measurements, as well as handcrafted features per frame, are exploited. The idea is to perform the same steps described in the previous sections, but using different perception sensors and with slight modifications to the proposed pipeline.
First of all, a Velodyne HDL-64E [3] laser scanner is used to collect point clouds from all-around the vehicle, which will later be used to reconstruct the shape of the

[1] http://www.cvlibs.net/datasets/kitti/eval_tracking.php
[2] http://velodynelidar.com/hdl-64e.html
[3] Velodyne HDL-64E is 64-layer rotating laser scanner.

environment. This multi-layer laser scanner has high accuracy and precision, such that its measurements can be considered as ground truth if compared to the information contained in a disparity map. In contrast, data from this laser scanner suffers for high sparsity issue, at least to build a ground truth. This means that a few points are describing the same surface if compared to what is perceived by an average stereo camera. This issue is solved by integrating measurements through multiple frames in a global static map, by exploiting exact localization information.
By defining ${}^{M}T_{B_k}$ the transformation matrix that describes the pose of the robot body with respect to the map in a given frame k, it is possible to describe any set of measurements $z = z_1, \cdots, z_n$ in the map reference system, by means of Equation 7.1. In this case, z_i is a tri-dimensional point as perceived by the laser scanner but represented in the body reference frame. The pose of the robot for each frame ${}^{M}T_{B_k}$ is retrieved using INS data. These measurements, appropriately described in a unique reference system, are ready to be accumulated in a ground truth map.
Before than that, however, it is necessary to remove some measurements from the point cloud. Points belonging to dynamic obstacles cannot be used to build a static map. Thus a filtering step is performed such that only valid points are taken into account.

8.2.1 Point cloud filtering

In order to perform such filtering, annotated 3D bounding boxes are exploited. Since annotations are available only for objects inside the camera field of view, measurements have to be filtered using an additional constraint. In particular, a measurement is considered valid if lies inside the camera field of view but does not belong to any object. Such constraint could have been abandoned if annotations were not limited by the field of view of the camera. This procedure is well illustrated in the subsequent block of pseudo code.

Data: point cloud, objects
Result: filtered point cloud
for *each point in point cloud* **do**
  if *point* $\in$ *camera FOV* & $\nexists$ *obj* $\in$ *objects* | *point* $\in$ *obj* **then**
    filtered point cloud $\leftarrow$ point;
  end
end

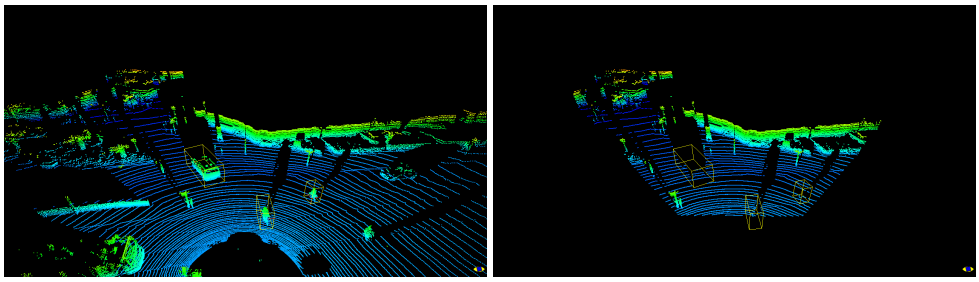
Algorithm 1: Point cloud filtering algorithm

Regarding the first check, i.e., whether the point falls inside the image FOV or not, it is merely performed by projecting the point onto the image surface and checking if its resulting coordinates are within the image boundaries. The second check involves checking whether a 3D point falls inside a three-dimensional bounding box or not, which is better illustrated in Appendix A. A sample result of this filtering process is depicted in Figure 8.2. Points either outside the camera field of view or assumed to be belonging to obstacles, shown with yellow bounding volumes, are discarded during the process. It is interesting to note that only obstacles inside the camera field of view are annotated. This is the main reason why only points falling inside both the laser scanner and the camera field of view are taken into consideration.

Finally using only valid measurements from the entire sequence, a ground truth map is built. The building procedure is then carried out by accumulating valid measurements in a global map. For each frame in a sequence, a cell is considered as occupied if at least one measurement falls into it. This building process is better illustrated in Figure 8.3. The inertial reference system is shown in yellow, at the bottom right corner of each image.

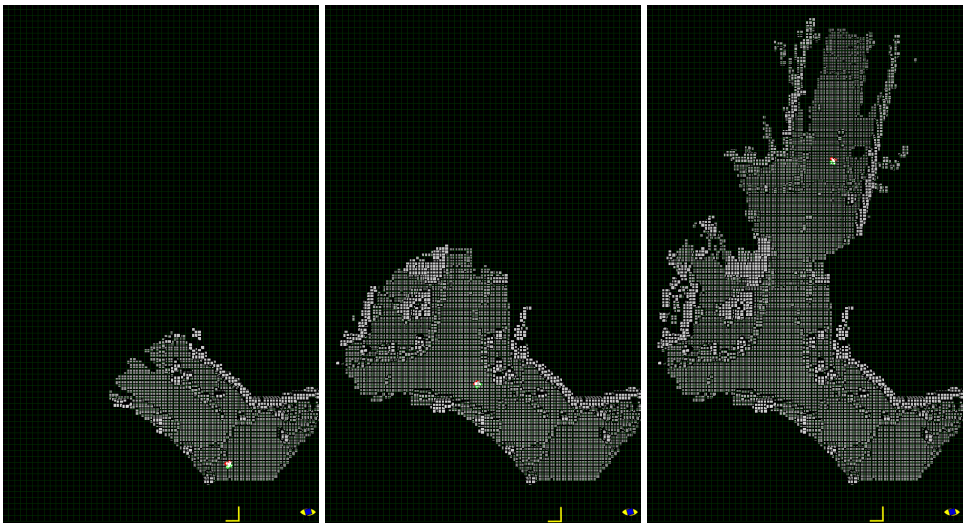
8.3 Evaluation metrics

For each test sequence, some evaluation metrics have been adopted to asses the quality of the proposed system. A brief description of all the evaluation metrics will be given as follows.



(a) (b)

Figure 8.2: Point cloud filtering process. From the original point cloud (a) are removed points outside the camera FOV or inside obstacles annotations and the resulting point cloud (b) is obtained.



(a) (b) (c)

Figure 8.3: Ground truth building process for sequence 0000. The resulting map is shown: after few frames (a); in the middle of the sequence (b); after the final frame (c).

These measured quantities can be mainly divided into two categories: on the one hand there are classification-related metrics, commonly used to evaluate the performances of binary classification tests; on the other hand we have similarity operators, usually exploited in image analysis applications.
The first set of metrics are used to evaluate the system only once a classification step has been performed to separate free and occupied areas of the map. The second set of metrics can be used a priori of a classification procedure, to asses the performance of the map reconstruction system.
For the first kind of evaluation, cells from both the ground truth and the reconstructed map have been classified as either free or occupied using Equation 8.1. Thus the problem of estimating the state of a given cell m_i has been implicitly considered as a binary classification problem.

$$classification_i = \begin{cases} occupied, & \text{if } p(m_i) > p_{th} \\ free, & \text{otherwise} \end{cases} \tag{8.1}$$

Each cell can thus be classified as occupied if its occupancy likelihood $p(m_i)$ is greater than a threshold, which has been chosen as $p_{th} = 0.7$, and free otherwise. It is important to notice that in this case unobserved cells are considered as free.
A system that classifies a cell as occupied or free can be evaluated with a binary classification test. In this case, each cell is considered independently as corresponding to a binary random variable where the actual state is known a priori. The outcome of the test is then compared with the real state, and some statistics are collected to measure how good the system does under different circumstances.
With the aim of facilitating the understanding of the following paragraphs, it is important to specify the meaning of some terms that will be later used, which comprise:

- True Positive (TP) refers to a relevant instance (i.e., an occupied cell) that has been selected (i.e., classified as occupied);

- False Positive (FP) refers to a non-relevant instance (i.e., a free cell) that has been selected (i.e., classified as occupied);

- True Negative (TN) refers to a non-relevant instance (i.e., a free cell) that has not been selected (i.e., classified as free);
- False Negative (FN) refers to a relevant instance (i.e., an occupied cell) that has not been selected (i.e., classified as free);

These quantities will be later used to compute different statistics that are meaningful for assessing the quality of the proposed system, which is here considered as a binary classifier operating several tests simultaneously, one for each cell. Accuracy, precision, recall, specificity, and negative predictive value are the statistics used for this binary classification tests.

Accuracy Accuracy is perhaps the most simple statistics that can be collected in a binary classification test. It is the fraction of all instances that are correctly classified as either occupied or free. Its formulation is then reported in Equation 8.2.

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{8.2}$$

Precision Precision, or positive predictive value, involves only instances classified as positives. It is, in fact, the fraction of relevant instances correctly classified. In this particular case are considered as relevant occupied cells in the ground truth map. Thus precision is computed as stated in Equation 8.3.

$$precision = \frac{TP}{TP + FP} \tag{8.3}$$

Precision is an indicator of how good the system is at detecting cells that are occupied.

Recall Recall, also called True Positive Rate, involves only relevant instances, i.e., cells that are considered as occupied by the ground truth. This is the fraction of relevant instances that are correctly classified. This statistics is then computed using Equation 8.4.

$$recall = \frac{TP}{TP + FN} \tag{8.4}$$

The Recall is an indicator of how good the system is at detecting as many cells as possible that are occupied.

Specificity Specificity, also known as True Negative Rate, is computed as the fraction of non-relevant instances correctly classified. Thus specificity measures how many actual negatives are identified as such. For this reason, it only concerns non-relevant instances or cells that are considered free with respect to the ground truth. It is important to remember that also unknown or unobserved cells are regarded as nonrelevant in this case. Specificity is then computed following Equation 8.5.

$$specificity = \frac{TN}{FP + TN} \tag{8.5}$$

Negative predictive value Negative predictive value is measured as the fraction of non selected instances correctly classified. Thus it is computed as stated in Equation 8.6.

$$npv = \frac{TN}{TN + FN} \tag{8.6}$$

The proposed system, however, is also constructing a description of the environment, also referred to as a map. Specifically, it provides an occupancy probability for each cell, which can be used in combination with ground truth to evaluate the system before a classification step. Error and cross-correlation metrics are used to measure how well the system approximates the ground truth performances, regarding likelihood estimation for each cell.

Error In case the map shares both orientation and area of coverage with its corresponding ground truth, a comparison on a cell basis can be performed. Thus an error metric is obtainable by comparing occupancy probabilities directly for each cell. Specifically, using Equation 8.7, the error is computed as the average squared distance between the map to be evaluated $p(m_i)$ and the corresponding ground truth $p(g_i)$. When computing this metric only observed cells are taken into account to

prevent for many unobserved cells to corrupt the evaluation.

$$error = \frac{1}{N}\sum_{i}^{N}\Big(p(m_i) - p(g_i)\Big)^2 \tag{8.7}$$

Cross correlation Cross-correlation is a technique from image analysis used to measure the level of similarity between two images. By considering the map as an image, it is possible to obtain a correlation coefficient expressing how much the evaluated map m is similar to the ground truth g. The cross-correlation coefficient is then computed using Equation 8.8.

$$correlation = \frac{S_{m,g}}{\sigma_m \cdot \sigma_g} \tag{8.8}$$

Where $S_{m,g}$ is computed as a correlation series between the map to be evaluated and the ground truth, as stated by Equation 8.9.

$$S_{m,g} = \sum_{i}^{N}\Big(p(m_i) - \mu_m\Big) \cdot \Big(p(g_i) - \mu_g\Big) \tag{8.9}$$

While terms μ and σ correspond to sample mean and covariance applied to different images regions.

8.4 Results

Using the previously described metrics, some sequences from the chosen dataset have been used to evaluate the proposed system. For each sequence, a frame-by-frame evaluation is provided through a chart, showing the value computed for each metric as a function of the frame in the sequence. In the end, a summary table where only the final result for each sequence is shown. Frame-by-frame results can be used to understand better how environment exploration is carried out.

A low recall characterizes first frames since the majority of the map has not been observed yet, which increases the number of false negatives. As long as the exploration proceeds, more and more cells are observed thus the true negative count lowers

seq #	TP	FP	TN	FN	accuracy	precision	recall	specificity	npv	error	correlation
0000	8476	6341	508263	4920	0.979	0.572	0.633	0.988	0.990	0.089	0.513
0001	28266	16866	4154740	24128	0.990	0.626	0.539	0.996	0.994	0.081	0.493
0002	14267	4220	766715	6798	0.986	0.772	0.667	0.995	0.991	0.081	0.591
0003	17995	5544	843943	12518	0.979	0.764	0.590	0.993	0.985	0.084	0.534
0004	35287	9108	5974393	29212	0.994	0.795	0.547	0.998	0.995	0.076	0.540
0005	32727	12073	2265772	25228	0.984	0.758	0.599	0.995	0.989	0.080	0.553
0006	7344	2540	902593	2723	0.994	0.742	0.730	0.997	0.997	0.074	0.633
0007	51549	22196	15274112	52143	0.995	0.699	0.497	0.999	0.997	0.074	0.521
0008	55472	13000	6488300	43228	0.991	0.810	0.562	0.998	0.993	0.076	0.560
0009	13759	7172	2368631	10438	0.993	0.657	0.569	0.997	0.996	0.086	0.505

Table 8.1: Final results for sequences from the Kitti tracking dataset.

increasing the recall measurement. Similar behavior is shown by the correlation metric, which increases while the map is being explored. This can be seen in results from sequence six, depicted in Figure 8.4g, where the camera stays still for a long time, before starting its movement. In fact, until frame 200 both recall and correlation exhibit constant values, and they start rising after that frame. This is because since that moment the car start moving and the environment as a consequence is explored. On the contrary, quantities like precision and error are approximately constant through the sequence. The reason for this is that these metrics take into account only for selected instances.

It is easy to observe that the average precision among all sequences is quite low. This may be due to the relatively low precision of the disparity, which usually overestimates the size of the objects. For example, a flat surface like a wall is usually perceived as wider than it is. On the contrary, a laser scanner provides very precise measurements, and the deviation of samples from the actual value would be extremely limited. In the above example, the wall would be perceived much thinner if compared to the stereoscopic case. A lot of false positives will then be detected around a flat surface, like a wall or a road pavement. Despite its low precision, the map is consistent with the actual environment that represents, which does not fail in detecting potentially dangerous objects but tends to overestimate their size. This is not necessarily an undesirable behavior, but a purely quantitative metrics do not take into account for such

a qualitative point of view.

In order to understand the innovative contributions of this work, it would be helpful to provide a comparison to other methodologies, previously proposed by the scientific community to solve a similar problem. If we consider the problem of detecting and representing obstacles, or their presence, in a perceived environment, there are other works at the state of the art that approach this task with a variety of sensors, techniques, and representations. Those techniques are usually well described, but a quantitative evaluation of their performances is, to the best of the author knowledge, nearly never provided.

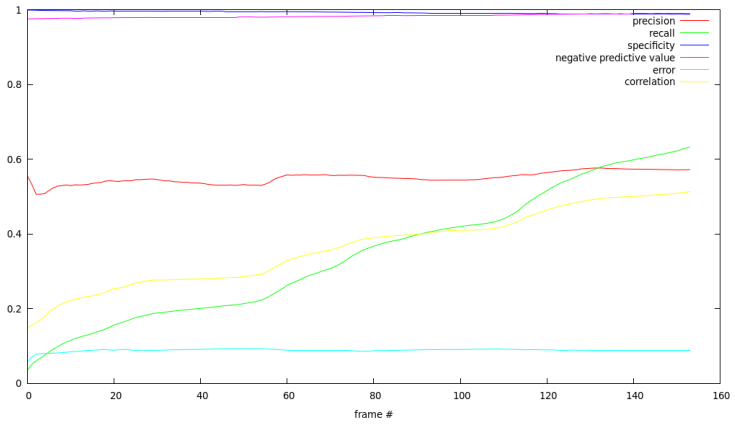
Whenever an evaluation is available, this is usually obtained using methods or metrics not directly applicable in order to make a comparison to this work. For example, there are obstacle detection methods which results are used as input to other higher-level modules for obstacle avoidance. In some such cases, the authors provide an evaluation for the final avoidance capabilities rather than focusing on the obstacle detection system itself. In [6], for example, the proposed system is evaluated through a series of use cases designed to test the obstacle avoidance capabilities specifically.

Other works instead focus their attention mostly on the obstacle map representation, and their evaluation is sometimes purely qualitative. In [36], for example, the authors performed a series of flying tests showing both the obtained map and the corresponding aerial imagery for reference.

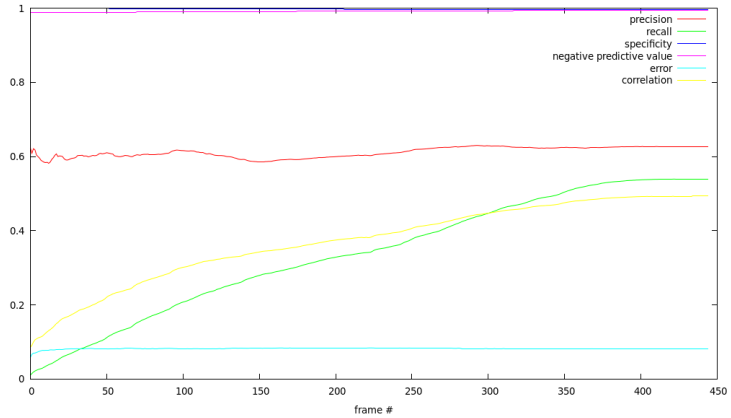
In other cases, the authors provide performance-related evaluations, showing that their representation is better than others under certain points of view, such as processing time or memory footprint requirements. Such an evaluation is provided in [9] where the authors provided the required processing time needed by their planning system, with respect to configuration parameters such as the chosen grid representation and the cell size.

Another case, which is worth mentioning, is reported on [55], where the authors used a dataset comprising both LiDAR point clouds and pose information to evaluate their representation paradigm. They compared the raw sensor measurements to measure the error involved in building a more compact hierarchical representation. In this case, they treated the raw sensor data as ground truth, which is acceptable due to the high

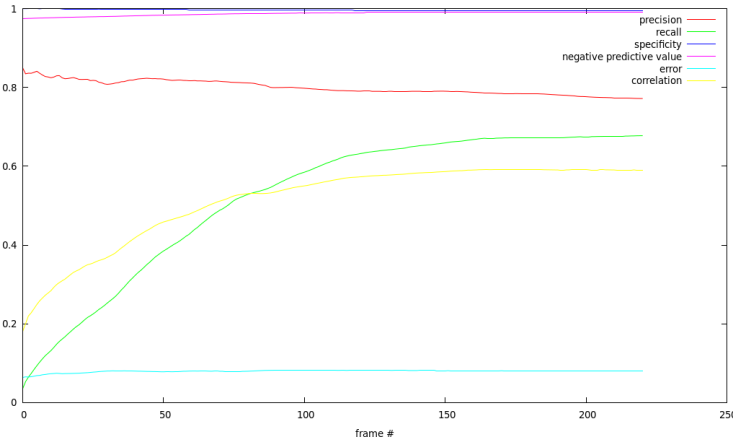
accuracy of LiDAR sensors. Such a method is however not directly applicable to a stereo vision based system because the depth estimated using a matching algorithm is not accurate enough to be treated as ground truth.



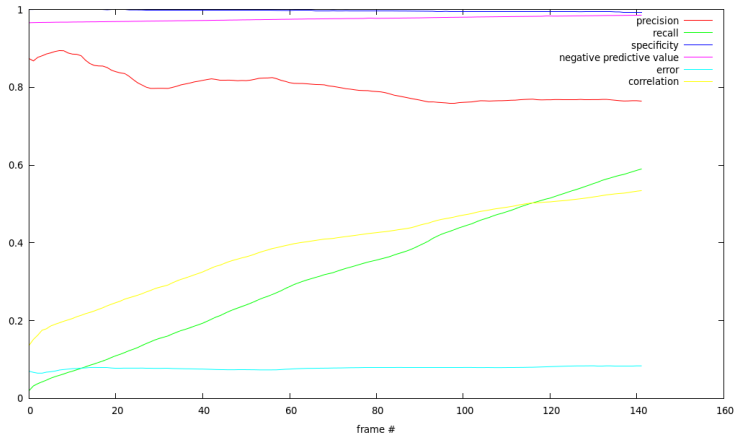
(a) Kitti tracking dataset sequence 0000



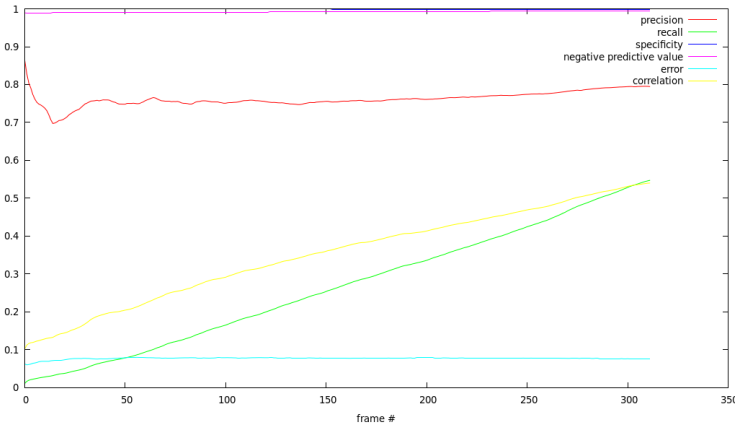
(b) Kitti tracking dataset sequence 0001



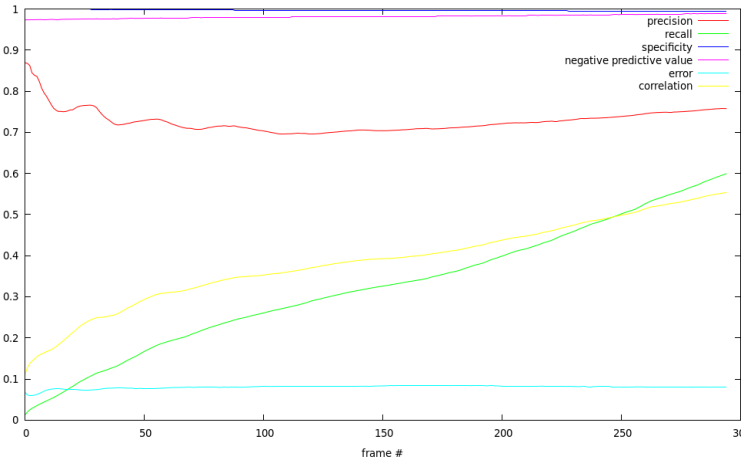
(c) Kitti tracking dataset sequence 0002



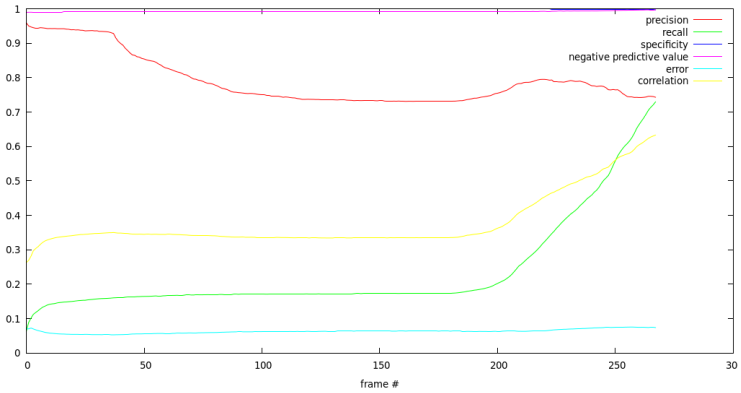
(d) Kitti tracking dataset sequence 0003



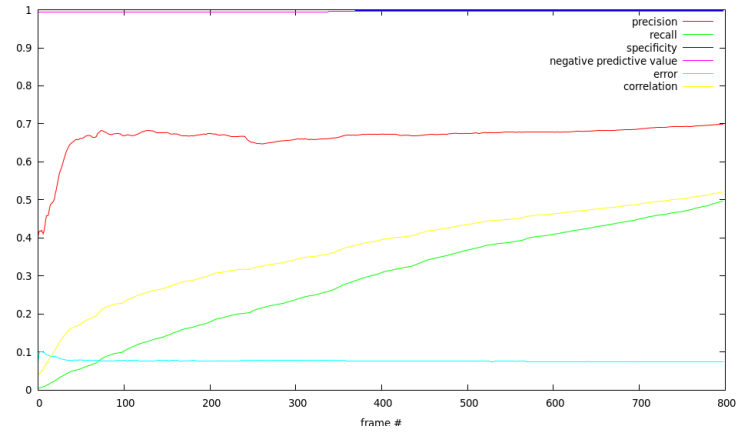
(e) Kitti tracking dataset sequence 0004



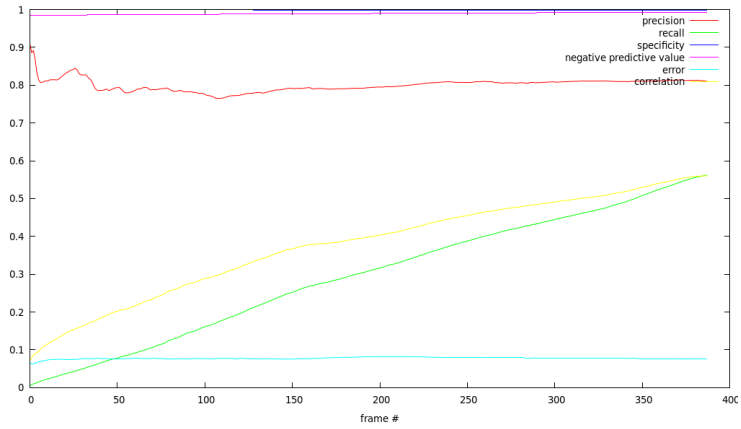
(f) Kitti tracking dataset sequence 0005



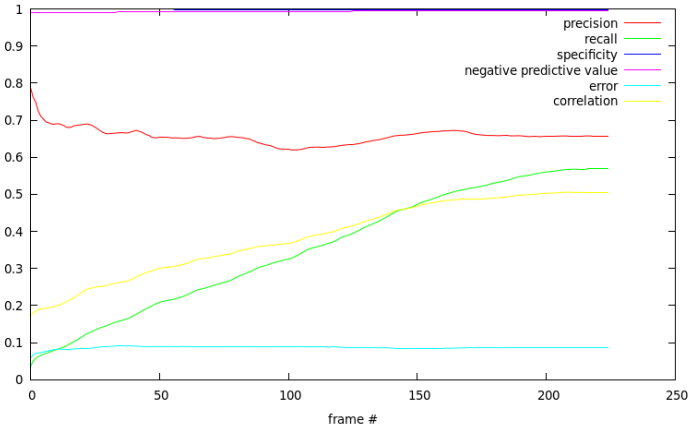
(g) Kitti tracking dataset sequence 0006



(h) Kitti tracking dataset sequence 0007



(i) Kitti tracking dataset sequence 0008



(j) Kitti tracking dataset sequence 0009

Figure 8.4: Frame by-frame results for Kitti tracking dataset sequences

Chapter 9

Conclusions and future works

A system for obstacle detection and mapping for micro aerial vehicles based on stereoscopic vision has been presented.
The algorithm is designed to be modular, since any number of stereoscopic cameras can be mounted on the UAV, provided that their position and orientation with respect to the aircraft is known. It has been proven that even with a single stereoscopic camera the system is capable of building a useful representation of the explored area. In case multiple cameras are needed an optimal mounting position would be of great importance. It has been shown in Section 5.2 that quite a significant portion of the field of view cannot be exploited due to occlusion if a non-optimal mounting position is chosen.
The proposed system is also designed to be efficient. A spherical representation has been used to reduce the dimensionality of the input data for each camera while describing all spherical grids in a shared reference system. This design choice has dramatically increased the efficiency of the subsequent steps. The dimensionality reduction has improved the requirements in term of the needed computational resource while the choice of a share reference system has facilitated the stitching of the spherical grids. Collected measurements are accumulated over multiple frames in a single map, using an occupancy grid mapping algorithm. A pair of inverse sensor models has been introduced, capturing the behavior of the employed sensors. The resulting

map can be described with respect of either a global or a local reference system. If a local description is chosen, the map can be kept smaller and more compact. Under this circumstance only the most essential part of the environment is represented, allowing for a considerable reduction of the memory consumption. This is usually a requirement for embedded applications, which can be used to perform on-board computation on a micro aerial vehicle.

A method for building a ground truth map has been presented, which exploits precise measurements and manual annotations. Generated ground truth maps have been used to asses the quality of the proposed system, with respect to some different evaluation metrics.

The system has been used to build a representation of the world that is accurate enough for an autonomous system to be able to navigate in either an outdoor or an indoor scenario safely. Demonstrating experiments have been performed in environments which were not known apriori. The system proved to be able to perceive and avoid various types of obstacles.

In the future, the addition of other lightweight sensors, like sonar or small radars, could be used to improve the performance of the system. Moreover, the adoption of other visual cues could be used to solve problems typically involved in the process of disparity matching, like repetitive patterns or uniformly textured surfaces.

Dynamic obstacles are not handled explicitly in this framework, meaning that an explicit tracking is not performed. Therefore the predicted position for a detected and moving obstacle is not currently available. Obstacles segmentation and tracking should be added in order to handle moving obstacles and avoid them during flight.

An evaluation method has been proposed for quantitatively demonstrating the capabilities of this system. The problem of evaluating a representation of the environment should be further explored, and comparison to other systems should be performed in order to determine the performances of the system better.

Effective and reliable obstacle detection and mapping systems will enable complex robotic applications, resulting in a significant impact from both legal and social points of view on our everyday life. Robotic applications that employ obstacle detection modules to perform simple tasks are already available. Some of these systems are

supposed to work only in specific scenarios or circumstances properly. For example, autonomous flying systems are available for passenger airliners, except for particularly delicate flight phases.
In the automotive field, much interest has been given to such applications during the last decade, leading to a level of autonomy that was unpredictable a few years before. Simple scenarios are now almost entirely handled by artificial systems, using obstacle detection modules to collect information regarding the surroundings. Certain cars are now able to drive at a different level of autonomy inside a highly structured highway scenario. In contrast, a more unpredictable urban environment is not yet entirely handled by autonomous driving cars.
Cruise control systems, which are used to adjust the throttle level to match a desired user-defined speed, are now available in most road vehicles. The addition of an obstacle detection module led to an adaptive version of such a system, which is also capable of maintaining a safe distance from the vehicle ahead. In the future, a more complex system, capable of building a useful representation of the surroundings of a car, will lead to a complete awareness of the world, which could be used to build a completely autonomous car.
Unmanned aerial vehicles were employed from long before than autonomous road vehicles. Initially designed for war-related purposes, they were used mainly to expose human pilots to fewer risks. Self-flying vehicles were used to perform either offending or recognition missions, but in both cases, the vehicles were not supposed to encounter any object during their flight. The relative absence of obstacles lowered the level of complexity required for achieving autonomous flight in these kinds of situations. Nowadays flying robots are also used for other types of applications, such as inspections or entertainment, requiring them to be able to detect and avoid various types of obstacles.
A drone capable of detecting obstacles obstructing its flying path would be able to acquire a much higher level of autonomy, possibly being able to fly without any supervision. For example, a completely autonomous drone could be used for security purposes. The drone would fly through the same way-points at each pass, in order to perform its monitoring task, planning its trajectory by also taking into account for

possibly detected obstacles. These mentioned way-points would be the only required information for the system to achieve its purpose.
Possible applications for an autonomous drone technology would be a delivering package system or even an alternative public transportation service. At first, for safety and regulatory reasons, would be limited to controlled areas or even available only for predetermined routes. Later, public air transport would be available to big cities, for example, to tackle the problem of road traffic and allow people to travel faster. In each case, the development of flying transports would represent a considerable innovation in the public transportation system. This revolution will require the introduction of a new set of rules meant to regulate the air traffic, as well as the definition of routes and policies for occupying the air space.

Appendix A

Point to bounding volume association

Let $p = (x, y, z)$ be the point to be tested and $B = \{p_0, p_1, p_3, p_4\}$ be a set of four points defining a bounding cube, as depicted in Figure A.1. Initially, the three versors defining the bounding cube orientation are retrieved by taking cross products between pairs of delimiting vectors.

$$\begin{aligned} u &= (p_0 - p_3) \times (p_0 - p_4) \\ v &= (p_0 - p_1) \times (p_0 - p_4) \\ w &= (p_0 - p_1) \times (p_0 - p_3) \end{aligned} \tag{A.1}$$

The projection of p and all other delimiting points p_i onto those versors are then computed and, if the projection of p lies inside the cube, p is said to be inside the bounding volume B. Specifically the following equations (Equation A.2) must be satisfied simultaneously for a point to be said inside the bounding volume.

$$\begin{aligned} u \cdot p_0 \leq u \cdot x \leq u \cdot p_1 \\ v \cdot p_0 \leq v \cdot y \leq v \cdot p_3 \\ w \cdot p_0 \leq w \cdot z \leq w \cdot p_4 \end{aligned} \tag{A.2}$$

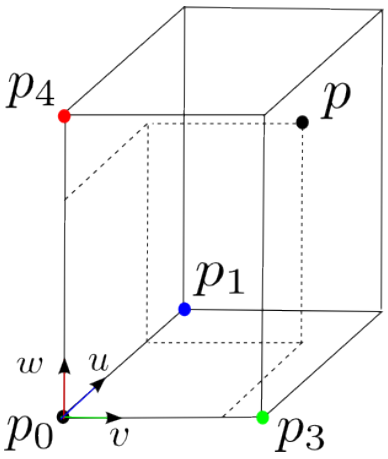


Figure A.1: Test for a 3D point falling inside a bounding volume.

Under the assumption that the points have been arranged properly. This means that p_0 should be nearer to the origin than p_1, p_3 and p_4 along the directions defined by u, v and w respectively. Such conditions is better formalized in Equation A.3.

$$\begin{aligned} u \cdot p_0 &\leq u \cdot p_1 \\ v \cdot p_0 &\leq v \cdot p_3 \\ w \cdot p_0 &\leq w \cdot p_4 \end{aligned} \tag{A.3}$$

Bibliography

[1] S Walter. The sonar ring: Obstacle detection for a mobile robot. In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, volume 4, pages 1574–1579. IEEE, 1987.

[2] Gerard Gibbs, Huamin Jia, and Irfan Madani. Obstacle detection with ultrasonic sensors and signal analysis metrics. *Transportation Research Procedia*, 28:173–182, 2017.

[3] Nils Gageik, Thilo Müller, and Sergio Montenegro. Obstacle detection and collision avoidance using ultrasonic distance sensors for an autonomous quadrocopter. *University of Wurzburg, Aerospace information Technologhy (germany) Wurzburg*, pages 3–23, 2012.

[4] Nischay Gupta, Jaspreet Singh Makkar, and Piyush Pandey. Obstacle detection and collision avoidance using ultrasonic sensors for rc multirotors. In *Signal Processing and Communication (ICSC), 2015 International Conference on*, pages 419–423. IEEE, 2015.

[5] Alireza Nemati, Mohammad Sarim, Mehdi Hashemi, Eric Schnipke, Steve Reidling, William Jeffers, Jesse Meiring, Padmapriya Sampathkumar, and Manish Kumar. Autonomous navigation of uav through gps-denied indoor environment with obstacles. *AIAA SciTech*, pages 5–9, 2015.

[6] Nils Gageik, Paul Benz, and Sergio Montenegro. Obstacle detection and collision avoidance for a uav with complementary low-cost sensors. *IEEE Access*, 3:599–609, 2015.

[7] Slawomir Grzonka, Giorgio Grisetti, and Wolfram Burgard. A fully autonomous indoor quadrotor. *IEEE Transactions on Robotics*, 28(1):90–100, 2012.

[8] Subramanian Ramasamy, Alessandro Gardi, Jing Liu, and Roberto Sabatini. A laser obstacle detection and avoidance system for manned and unmanned aircraft applications. In *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*, pages 526–533. IEEE, 2015.

[9] Matthias Nieuwenhuisen, David Droeschel, Marius Beul, and Sven Behnke. Obstacle detection and navigation planning for autonomous micro aerial vehicles. In *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pages 1040–1047. IEEE, 2014.

[10] Richard JD Moore, Karthik Dantu, Geoffrey L Barrows, and Radhika Nagpal. Autonomous mav guidance with a lightweight omnidirectional vision sensor. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3856–3861. IEEE, 2014.

[11] Andrew M Hyslop and J Sean Humbert. Autonomous navigation in three-dimensional urban environments using wide-field integration of optic flow. *Journal of guidance, control, and dynamics*, 33(1):147–159, 2010.

[12] Stéphane Ross, Narek Melik-Barkhudarov, Kumar Shaurya Shankar, Andreas Wendel, Debadeepta Dey, J Andrew Bagnell, and Martial Hebert. Learning monocular reactive uav control in cluttered natural environments. In *2013 IEEE international conference on robotics and automation*, pages 1765–1772. IEEE, 2013.

[13] Antoine Beyeler, Jean-Christophe Zufferey, and Dario Floreano. Vision-based control of near-obstacle flight. *Autonomous robots*, 27(3):201, 2009.

[14] Jean-Christophe Zufferey, Antoine Beyeler, and Dario Floreano. Optic flow to steer and avoid collisions in 3d. In *Flying Insects and Robots*, pages 73–86. Springer, 2009.

[15] Jean-Christophe Zufferey, Antoine Beyeler, and Dario Floreano. Autonomous flight at low altitude with vision-based collision avoidance and gps-based path following. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, number EPFL-CONF-142989. IEEE, 2010.

[16] Joseph Conroy, Gregory Gremillion, Badri Ranganathan, and J Sean Humbert. Implementation of wide-field integration of optic flow for autonomous quadrotor navigation. *Autonomous robots*, 27(3):189, 2009.

[17] Tomoyuki Mori and Sebastian Scherer. First results in detecting and avoiding frontal obstacles from a monocular camera for micro unmanned aerial vehicles. In *Robotics and automation (icra), 2013 ieee international conference on*, pages 1750–1757. IEEE, 2013.

[18] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[19] Yoko Watanabe, Anthony Calise, and Eric Johnson. Vision-based obstacle avoidance for uavs. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, page 6829, 2007.

[20] Daniel Magree, John G Mooney, and Eric N Johnson. Monocular visual mapping for obstacle avoidance on uavs. In *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 471–479. IEEE, 2013.

[21] Z. Gosiewski, J. Ciesluk, and L. Ambroziak. Vision-based obstacle avoidance for unmanned aerial vehicles. In *2011 4th International Congress on Image and Signal Processing*, volume 4, pages 2020–2025, Oct 2011. doi:10.1109/CISP.2011.6100621.

[22] A. Al-Kaff, Qinggang Meng, D. Martín, A. de la Escalera, and J. M. Armingol. Monocular vision-based obstacle detection/avoidance for unmanned aerial vehicles. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 92–97, June 2016. doi:10.1109/IVS.2016.7535370.

[23] O. Esrafilian and H. D. Taghirad. Autonomous flight and obstacle avoidance of a quadrotor by monocular slam. In *2016 4th International Conference on Robotics and Mechatronics (ICROM)*, pages 240–245, Oct 2016. doi:10.1109/ICRoM.2016.7886853.

[24] Stephan Weiss, Markus Achtelik, Laurent Kneip, Davide Scaramuzza, and Roland Siegwart. Intuitive 3d maps for mav terrain exploration and obstacle avoidance. *Journal of Intelligent & Robotic Systems*, 61(1-4):473–493, 2011.

[25] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

[26] Jeffrey Byrne, Martin Cosgrove, and Raman Mehra. Stereo based obstacle detection for an unmanned air vehicle. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2830–2835. IEEE, 2006.

[27] Lionel Heng, Lorenz Meier, Petri Tanskanen, Friedrich Fraundorfer, and Marc Pollefeys. Autonomous obstacle avoidance and maneuvering on a vision-guided mav using on-board processing. In *Robotics and automation (ICRA), 2011 IEEE international conference on*, pages 2472–2477. IEEE, 2011.

[28] Friedrich Fraundorfer, Lionel Heng, Dominik Honegger, Gim Hee Lee, Lorenz Meier, Petri Tanskanen, and Marc Pollefeys. Vision-based autonomous mapping and exploration using a quadrotor mav. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4557–4564. IEEE, 2012.

[29] Andrew J Barry and Russ Tedrake. Pushbroom stereo for high-speed navigation in cluttered environments. In *Robotics and automation (icra), 2015 ieee international conference on*, pages 3046–3052. IEEE, 2015.

[30] Korbinian Schmid, Teodor Tomic, Felix Ruess, Heiko Hirschmüller, and Michael Suppa. Stereo vision based indoor/outdoor navigation for flying robots. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3955–3962. IEEE, 2013.

[31] Boitumelo Ruf, Sebastian Monka, Matthias Kollmann, and Michael Grinberg. Real-time on-board obstacle avoidance for uavs based on embedded stereo vision. *CoRR*, abs/1807.06271, 2018.

[32] Zhencheng Hu and Keiichi Uchimura. Uv-disparity: an efficient algorithm for stereovision based scene analysis. In *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, pages 48–54. IEEE, 2005.

[33] Helen Oleynikova, Dominik Honegger, and Marc Pollefeys. Reactive avoidance using embedded stereo vision for mav flight. *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 50–56, 2015.

[34] Sjoerd Tijmons, Guido CHE de Croon, Bart DW Remes, Christophe De Wagter, and Max Mulder. Obstacle avoidance strategy using onboard stereo vision on a flapping wing mav. *IEEE Transactions on Robotics*, 33(4):858–874, 2017.

[35] Stefan Hrabar and Gaurav Sukhatme. Vision-based navigation through urban canyons. *Journal of Field Robotics*, 26(5):431–452, 2009.

[36] Lionel Heng, Dominik Honegger, Gim Hee Lee, Lorenz Meier, Petri Tanskanen, Friedrich Fraundorfer, and Marc Pollefeys. Autonomous visual mapping and exploration with a micro aerial vehicle. *Journal of Field Robotics*, 31(4):654–675, 2014.

[37] Stefan Hrabar. 3d path planning and stereo-based obstacle avoidance for rotorcraft uavs. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 807–814. IEEE, 2008.

[38] Hiroshi Koyasu, Jun Miura, and Yoshiaki Shirai. Real-time omnidirectional stereo for obstacle detection and tracking in dynamic environments. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 1, pages 31–36. IEEE, 2001.

[39] Pascal Gohl, Dominik Honegger, Sammy Omari, Markus Achtelik, Marc Pollefeys, and Roland Siegwart. Omnidirectional visual obstacle detection using embedded fpga. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 3938–3943. IEEE, 2015.

[40] Johann Borenstein and Yoram Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE transactions on robotics and automation*, 7(3):278–288, 1991.

[41] Alberto Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal on Robotics and Automation*, 3(3):249–265, 1987.

[42] Hans P Moravec. Sensor fusion in certainty grids for mobile robots. *AI magazine*, 9(2):61, 1988.

[43] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, June 1989. doi:10.1109/2.30720.

[44] Joachim Buhmann, Wolfram Burgard, Armin B Cremers, Dieter Fox, Thomas Hofmann, Frank E Schneider, Jiannis Strikos, and Sebastian Thrun. The mobile robot rhino. *Ai Magazine*, 16(2):31, 1995.

[45] Wolfram Burgard, Armin B Cremers, Dieter Fox, Dirk Hähnel, Gerhard Lakemeyer, Dirk Schulz, Walter Steiner, and Sebastian Thrun. Experiences with an interactive museum tour-guide robot. *Artificial intelligence*, 114(1-2):3–55, 1999.

[46] Sebastian Thrun, Michael Beetz, Maren Bennewitz, Wolfram Burgard, Armin B Cremers, Frank Dellaert, Dieter Fox, Dirk Haehnel, Chuck Rosenberg, Nicholas Roy, et al. Probabilistic algorithms and the interactive museum tour-guide robot

minerva. *The International Journal of Robotics Research*, 19(11):972–999, 2000.

[47] Kohtaro Sabe, Masaki Fukuchi, J-S Gutmann, Takeshi Ohashi, Kenta Kawamoto, and Takayuki Yoshigahara. Obstacle avoidance and path planning for humanoid robots using stereo vision. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 1, pages 592–597. IEEE, 2004.

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

[49] Maja J Mataric. A distributed model for mobile robot environment-learning and navigation. 1990.

[50] Benjamin Kuipers and Yung-Tai Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and autonomous systems*, 8(1-2):47–63, 1991.

[51] Howie M Choset and Joel Burdick. *Sensor based motion planning: The hierarchical generalized Voronoi graph*. PhD thesis, Citeseer, 1996.

[52] F. Oniga and S. Nedevschi. Processing dense stereo data using elevation maps: Road surface, traffic isle, and obstacle detection. *Vehicular Technology, IEEE Transactions on*, 59(3):1172–1182, March 2010. doi:10.1109/TVT.2009.2039718.

[53] Szilárd Mandici and Sergiu Nedevschi. Real-time multi-resolution digital elevation map creation using the sensor model of a stereovision sensor. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pages 608–615. IEEE, 2016.

[54] Rudolph Triebel, Patrick Pfaff, and Wolfram Burgard. 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.

[55] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013.

[56] Octree. Octree — Wikipedia, the free encyclopedia, 2010. [Online; accessed 15-June-2018]. URL: `https://en.wikipedia.org/wiki/Octree`.

[57] Changhong Fu, Adrian Carrio, and Pascual Campoy. Efficient visual odometry and mapping for unmanned aerial vehicle using arm-based stereo vision pre-processing system. In *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*, pages 957–962. IEEE, 2015.

[58] Ivan Dryanovski, William Morris, and Jizhong Xiao. Multi-volume occupancy grids: An efficient probabilistic 3d mapping model for micro aerial vehicles. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1553–1559. IEEE, 2010.

[59] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2008.

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

[61] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[62] Francesco Valenti, Francesca Ghidini, Marco Patander, and Alberto Broggi. A ground truth building approach for evaluation of grid based discretization techniques in automotive scenarios. In *Intelligent Vehicles Symposium (IV), 2016 IEEE*, pages 960–965. IEEE, 2016.

[63] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.