Introduction to Autonomous Navigation and Mapping for Unmanned Ground Vehicles

Stéphanie Aravecchia

▶ **To cite this version:**

Stéphanie Aravecchia. Introduction to Autonomous Navigation and Mapping for Unmanned Ground Vehicles. Georgia Tech-Lorraine; CNRS. 2024. hal-04585273

HAL Id: hal-04585273
https://hal.science/hal-04585273

Submitted on 23 May 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Public Domain

Introduction to Autonomous Navigation and Mapping for Unmanned Ground Vehicles

Stéphanie Aravecchia[1]

[1]IRL 2958 Georgia Tech - CNRS, 2 rue Marconi, 57070 Metz, France

May 2024

Abstract

This technical report addresses the fundamental challenges associated with Navigation and Mapping for Unmanned Ground Vehicles (UGVs), with a particular focus on wheeled robots. Aimed at students beginning robotics research projects, this document provides an overview of the fundamentals of UGV navigation and mapping. Notably, it introduces perception, localization, mapping, and autonomous navigation, with an emphasis on the associated challenges.

This document is based on [Aravecchia, 2023].

1 Mobile Robots

This section is designed to provide an overview of fundamental concepts in mobile robotics.

Typically, in the domain of mobile robots, we categorize their abilities into low-level and high-level functions. This distinction is shown in Figure 1, adapted from [Siegwart et al., 2011]. This figure illustrates the core concepts of mobile robots. Read from bottom to top, the diagram outlines low-level to high-level abilities.

The lower-level abilities are perception and motion control. They capture the essence of the robot's interaction with the real world. On the perception side, the robot acquires information from its surroundings, whereas on the control

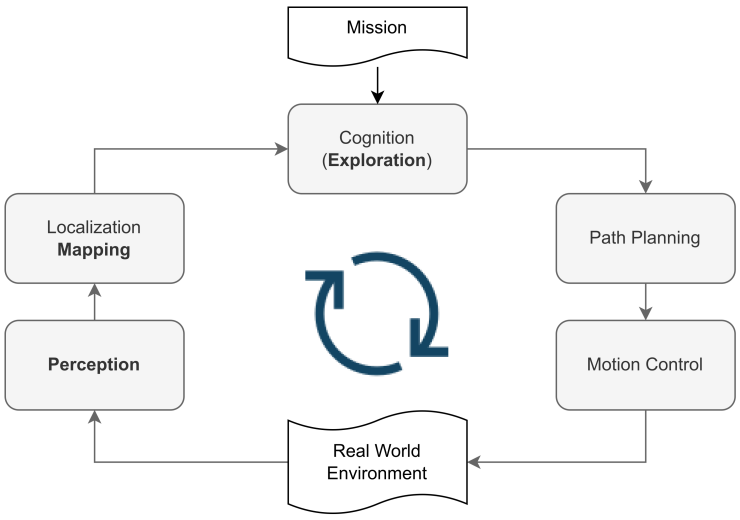

Figure 1: Key concepts of mobile robots (adapted from [Siegwart et al., 2011]). The concepts used in this work are highlighted in bold in the diagram.

side, the robot actions translate into real-world consequences. For instance, by rotating its wheels, the robot moves within its environment. Advancing up the hierarchy in the diagram, we encounter higher-level abilities, such as localization, map building and path planning. Localization, directly linked to perception, represents the robot's ability to determine its own position. Directly connected to perception and localization is map building, a key feature of this work that we will explore in detail later. Similarly, path planning is a high level ability, typically associated to motion control in the context of mobile robots. It represents the robot's ability to navigate robustly through its surroundings. Reaching the pinnacle of the diagram, cognition tasks come into play. These tasks involve more abstract functions, such as exploration. All these concepts find comprehensive explanation in [Siegwart et al., 2011].

It is important to note that the field of mobile robots is vast, encompassing various types such as aerial robots, underwater robots, and more. Focusing on ground robots, specifically Unmanned Ground Vehicles (UGVs), there still exists a wide spectrum of locomotion mechanisms, ranging from legged robots to tracked robots, including wheeled robots. The choice of locomotion impacts the right side of the diagram in Figure 1. Motion control depends directly on the robot locomotion. Indeed, control laws, commands send to the robot's actuators, heavily depend on the robot locomotion. Even tasks at a higher level,

such as path planning, rely on this specific design aspect. For instance, planning a path involving stairs may be achievable for a legged robot, but unrealistic for a one with wheels.

In what follows, we will outline the fundamental concepts introduced here.

1.1 Perception

As introduced before, robot perception is a core concept in mobile robots. Every mobile robot is equipped with sensors, and robot sensing is the essence of robot-environment interaction. By taking measurements using various sensors, the robot acquires knowledge about its environment. Often, the sensors are classified between proprioceptive and exteroceptive sensors. While proprioceptive sensors measure values internally to the robot, exteroceptive sensors measure information from the robot's environment. A thorough review of robot sensors is provided in [Siegwart et al., 2011]. In what follows, we simply present the common sensors for UGVs, and specifically highlight those necessary to understand this research.

1.1.1 Proprioceptive sensors

Wheeled robots are generally equipped with at least two proprioceptive sensors: one or more wheel encoders and one Inertial Measurement Unit (IMU). A wheel encoder measures the position or the speed of the wheel. In robotics, it is often an optical incremental encoder. It is a device that contains a mechanical light chopper that produces a certain number of pulses for each shaft revolution. By counting the number of pulses, and integrating it in time, the sensor provides an estimate of the position of the robot. An IMU is a device that contains different sensors: a gyroscope, an accelerometer, and often a magnetometer. They provide a measure of the robot's orientation and inclination. While the measurements are integrated in time, the sensor provides an estimate of the robot position and orientation.

1.1.2 Exteroceptive sensors

Exteroceptive sensors are generally split in two categories: passive and active sensors. While passive sensors measure energy coming from the environment, active sensors emit their own energy and measure the response.

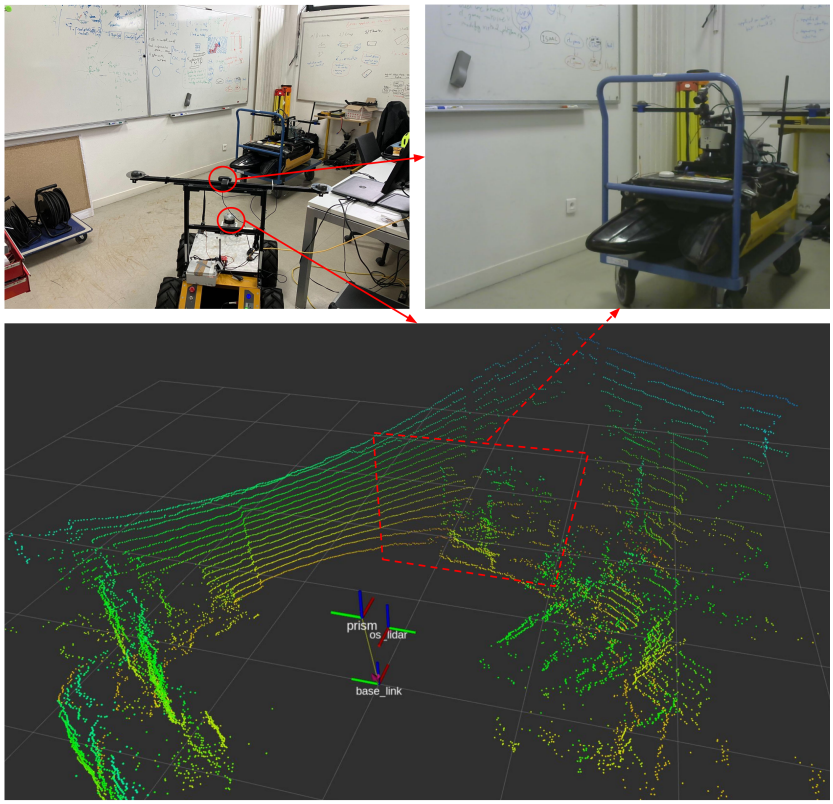

Figure 2: Top-left: external view of the scene. Top-right: image from the camera. Bottom: point-cloud from the Lidar. Top-right and bottom correspond to what the robot "sees". The dashed trapezoid provides an idea of the correspondence between camera and point-cloud. The color in the point-cloud simply encodes the height, for visualization purpose.

In the outdoor application we consider in this work, a common passive exteroceptive sensor used for localization is a GPS (Global Positioning System). The GPS is a constellation of satellites that continuously transmit their location around Earth, in a synchronized way. The GPS receiver reads the transmission from all the visible satellites, and computes its distance to each satellite through the arrival time difference in the signals. By combining information on distance and location of several satellites, it can infer its own position, commonly within a few meters accuracy. To improve the accuracy of the system, the GPS can be improved to RTK-GPS (RTK, Real Time Kinematics). An RTK-GPS system is composed of two receivers: a fixed one, and a mobile one. While the fixed re-

ceiver does not move, it computes corrections on the signals it receives from the satellites' constellation. It then sends these corrections to the mobile receiver, that integrates them when inferring its own position. Such a system typically reaches a centimeter accuracy.

Apart from GPS, mobile robots are typically equipped with other exteroceptive sensors directly linked to their task. The most common ones are likely cameras and Lidars. Although common, these two sensors are fundamentally distinct. Firstly, cameras are passive sensors, while Lidars are active sensors. Additionally, their data outputs diverge significantly.

Traditional cameras are sensors that record the light that is reflected on their environment and output an image. This image is a 2D-projection of the environment: a grid of pixels. In traditional cameras, each pixel corresponds to an individual photodetector, that measures the reflected light. This measure is then encoded into a color model, typically RGB (Red Green Blue). A displayed image is the addition of the light beams of those three fundamental colors. Most computer vision applications use RGB images, and RGB cameras are very common sensors in robotics application. One of their main drawback is that they do not provide any information on the distance to the object. All the robot's surroundings are simply projected on a plane. The functioning of camera sensors and their output is thoroughly detailed in [Corke et al., 2011].

Lidars, for LIght Detection And Ranging, are drastically different. Firstly, as they are active sensors, Lidars generate energy, in this case light, to get a measurement. A Lidar is actually a system emitting light from a rapidly firing rotating laser. This light travels, and eventually hits an object. The reflected light energy then returns to the sensor where it is measured. The sensor measures two quantities. The first is the time it takes for the emitted light to come back: the time of flight. This time is used to calculate the distance, or range. The second is the waveform of the pulse of light that is returned to the sensor. From this waveform, the sensor calculates the intensity of the returned energy. Secondly, the output of a Lidar is fundamentally different from the output of a camera. A 3D-Lidar emits laser rays in known directions. From the direction and the range, the sensor calculates the XYZ coordinates of the returned point expressed in the center of the Lidar frame, along with an intensity value. From the returned points, the Lidar outputs a point-cloud: an unordered collection of points. Each point in the point-cloud contains XYZ coordinates and the intensity value. Figure 3 provides an example, illustrating that only returned points are added to the point-cloud. A description of the technology and its

associated challenges can be found in [Center and Services, 2012], whereas a thoroughly detailed study of Lidars and point-clouds is provided by [Vosselman, 2010]. This research deals with mapping with a 3D-Lidar.

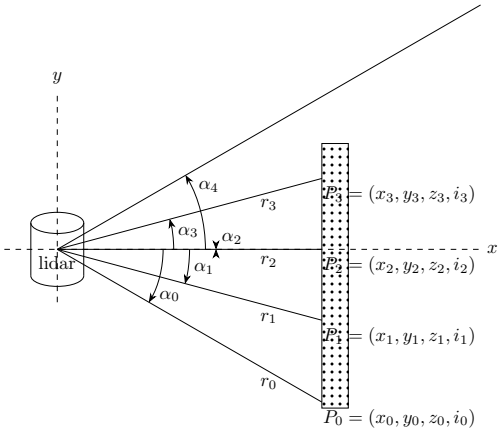

Figure 3: This figure illustrates 3D-Lidar. The light rays are emitted from the Lidar, in known directions (angles α). When the ray hits an obstacle, the distance of the returned point is computed (distances r), along with the intensity of the signal i. A point P (x, y, z, i) is added to the point-cloud. Rays that does not return are not in the point-cloud. In this example, although five rays are emitted, the point-cloud contains only four points.

Finally, we would like to emphasize how the data from a camera and a Lidar are different. We do so in Figure 2. It shows an external view of the scene, showing the Husky and its sensors, along with the image from the camera, and the point-cloud from the Lidar. The image and the point-cloud are what the robot "sees" with these sensors.

1.2 Localization

Another core ability of mobile robots, as introduced earlier, lies in localization. Any image or point-cloud described previously becomes useless if the observation cannot be linked with its position in the environment. While we previously explained that certain sensors provide estimations of position and/or orientation, solving localization is a standalone research subject. In the upcoming content, the term "pose" simply means position and orientation.

Commonly, the localization is the problem of estimating the robot's pose in an external reference frame, from sensor data, using a map of the environment

[Fox et al., 2006]. Alternatively, the problem can be reformulated to estimate at the same time the robot's pose and the map, making it a SLAM problem (Simultaneous Localization and Mapping). In this section, we briefly introduce the challenges linked with localization, and its problem formulation.

Here, we simply introduce the challenges associated with localization. For readers interested in an in-depth study of localization, [Fox et al., 2006] would be the authoritative reference book.

1.2.1 Frames

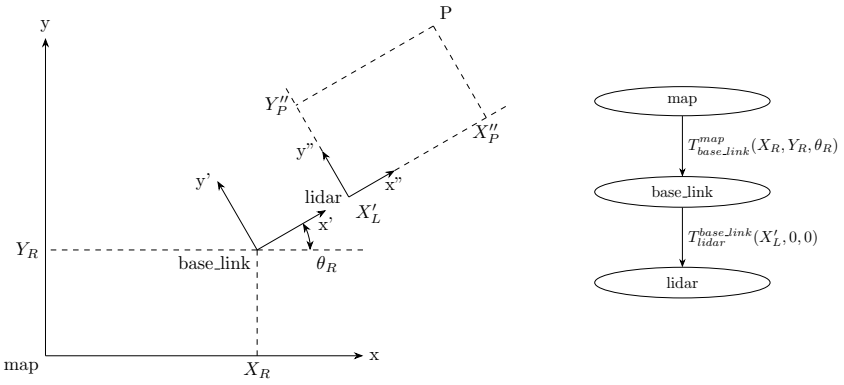

Figure 4: Illustration of the concept of frames in a 2D-example. On the left: different coordinate systems, or frames. The external reference frame is called *map*, the robot frame is called *base_link*, and the Lidar frame is called *lidar*. On the right, the tree structure storing the transformations between frames.

When considering localization, the primary concept to grasp is linked to frames. In a robotic system, several frames exist, each defining a different coordinate system. Solving the localization consists in estimating the robot pose, in a reference frame, from observations that may be in different frames. When the problem is formulated in 3D, the pose corresponds to the six degrees of freedom of the system: its 3D Cartesian coordinates, and its three elemental rotations, in the reference frame.

We illustrate this concept in Figure 4, with a 2D example. This figure shows different coordinate systems. First, on the bottom left, the *map* frame represents the global coordinate system. Second, the *base_link* frame corresponds to the current robot pose. In the *map* frame, this pose is the vector (X_R, Y_R, θ_R). To this vector corresponds the transformation between *base_link* and *map*. Next,

the *lidar* frame corresponds to the coordinate system in which the point-cloud from the Lidar is expressed. In the *base_link* frame, the lidar pose is the vector (X'_L, Y'_L, θ'_L). In this particular case, $Y'_L = 0$ and $\theta'_L = 0$. To this vector corresponds the transformation between *lidar* and *base_link*. Finally, the coordinates of the point P1 in the point-cloud are (X''_P, Y''_P) in the *lidar* frame.

In robotics applications, it is convenient to keep track of the relative position of a frame with respect to another one in a tree structure. Indeed, it is likely that *lidar* remains fixed with respect to *base_link*, and that *base_link* will move with respect to *map*. By chaining the transformations between frames, we can, for instance, easily recover the position of the point P in the *map* frame.

1.2.2 Problem formulation

If a sensor could offer a highly accurate estimation of the robot pose, localization would not be such a challenging task. However, even if such a perfect sensor were to exist, a new challenge would emerge: how could we then be certain of the robot's position with respect to its environment with absolute certainty ? Moreover, what if we introduced another complication, like the presence of humans moving in the robot's surroundings ?

In mobile robotics applications, the fundamental reality is that nothing is known with absolute precision. Every measurement, from any sensor, carries some noise. Additionally, every action the robot takes introduces its own noise. When a command is transmitted to the robot's actuators, the actual movement of the robot always slightly deviates from the anticipated trajectory. This can be attributed to sensor noise, such as in the wheel encoders, but more broadly, it comes from the challenge of precisely modeling interactions with the environment. This, once again, contributes to the noise inherent in the robotics system.

For these reasons, probabilistic algorithms are the fundamental algorithms in mobile robotics, as expansively detailed in [Fox et al., 2006]. The key idea of such algorithms is to represent information by probability distributions over a whole space of possible hypothesis. In the case of localization, the initial belief (the initial robot's pose estimate) is represented by a probability density function over the space of all locations. Every new sensor measurement (observation) and every new movement of the robot (action) update the previous belief. Generally, with enough actions and observations, the probability mass is moved to a single location, and the robot's localization is known with a good confidence.

This problem formulation is a state estimation, and this description depicts a Bayes Filter. The two more widely used localization algorithms in mobile robotics derive from this algorithm: the Kalman filter and the particle filter, which we simply mention here. Once more, our focus here is on introducing the localization challenges, and we encourage readers to consult [Fox et al., 2006] for comprehensive explanations.

2 Mapping

2.1 Fundamentals of mapping

A map is a representation of the environment. The selection of a specific map representation depends on various factors. First and foremost, if the map is intended for robot navigation, it must be computed online on the robot, which introduces a significant computational complexity limitation. Conversely, if the map is computed offline, this limitation is no longer a concern. Secondly, the map's intended purpose and the type of information it needs to store are essential considerations. Lastly, does the map exclusively stores static objects? Different map representations are available based on these considerations. We will now describe a few examples, focusing solely on static maps.

Firstly, a continuous map is an exact decomposition of the environment. It compiles a list of all objects along with their location. Typically, to manage computational costs, the objects are selected and abstracted. For instance, line segments are extracted from sensor data, and their coordinates are stored, as presented by [Gonzalez et al., 1994]. These types of maps are commonly employed in indoors applications. Secondly, a discrete map is a discretization of the environment. These maps typically assume that geometry is the most salient feature of the environment. Such maps are prevalent in various mobile robotics applications. We employ them in this research, and we will delve into their details in the following section. Thirdly, topological maps are graphs that define nodes and the connectivity between nodes. GPS navigation relies on this map type, where a node signifies an intersection, and the connections between nodes represent feasible trajectories. These maps also find widespread application in robotics, due to their capacity to enable fast search algorithms within them. More recently, research explored semantic information incorporation into the map representations. For instance, in a topological map, each node is assigned a semantic label like "grass" or "asphalt", as shown in [Kostavelis and Gasteratos,

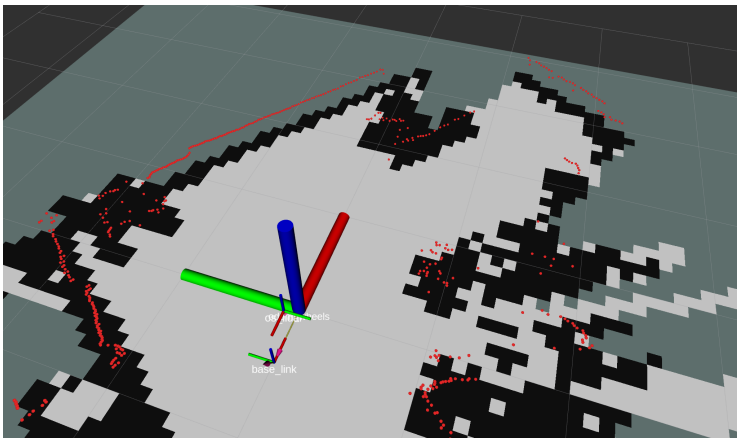

Figure 5: Example of a 2D occupancy grid map, mapping the same place depicted in Figure 2. Black: pixel corresponding to occupied space, light-grey: free space, dark-grey: unknown space.

2015].

In the upcoming paragraphs, we will elaborate on a specific type of discrete map known as the occupancy grid. We will then showcase different challenges related to mapping.

2.2 Occupancy grids

As previously mentioned, achieving optimal navigation planning stands as one of the most prevalent objectives for building a map. In other words, the intention behind the map is to plan the robot trajectory, from its current position, to a goal. In such a case, the map is directly linked to how the trajectory will be planned. The most prevalent map in this case is the occupancy grid, a discrete map. The occupancy grid is a 2D-map, an image, representing the plane the robot will evolve in. Every cell in this map, each pixel in the image, encodes the necessary information to compute a safe navigation plan. The simplest form of occupancy grid contains ternary information: each pixel is either free, occupied or unknown. A slightly more advanced form of occupancy grid encodes the occupancy likelihood of each pixel. When the robot is equipped with a Lidar, every point-cloud provides information. Basically, we know that every return point corresponds to occupied space, and the ray between the robot and the return point corresponds to free space. Behind the return point, we do not have any information. Building the occupancy grid consists in accumulating

the robot's observations in time.

Figure 5 provides an example of an occupancy grid map. This figure showcases some mapping features described previously. Firstly, this map is an image, projected on a plane. The dark-grey areas on the top corners are outside the map. Secondly, the figure shows the current projection of the 3D-Lidar on a horizontal plane (red dots). Thirdly, on the map, we can see the occupied space where those points are projected on the ground (black), the unknown space behind (middle-grey), and the free space between the robot and the points (light-grey). Secondly, this figure shows the results of the ray casting operation. Some previous returned Lidar points have marked the space empty only between the return point and the robot, leaving unknown space around the rays (middle-right of the figure). Finally, although this is not obvious in the figure, it also displays the different frames involved in the mapping. The map is built in the frame represented by the thick axes. The small axes in the figure are the lidar frame and the base_link frame (the current robot pose in the map).

2.3 3D-grids

A 3D-grid is a generalization in 3D of the 2D-grid we introduced earlier. Such a map is the discretization of the space into voxels, each voxel containing some information. That information can be:

1. an occupancy map, where the voxels have 3 states: free, occupied or unknown;
2. a probability map where each cell contains its probability of occupancy;
3. or something different, where the voxels contain data such as semantic information.

Here, we focus on the prevailing map representation for outdoor robotics: probabilistic 3D-grids. The appearance of this map is directly derived from the resolution, that is, the size of an edge of a voxel. Ideally, the lower the resolution, the more detailed the map. Figure 6 provides an example of a 3d-grid map. This map is a representation of the lab shown in Figure 2. Due to the small vertical field of view of the Ouster-16 3D-Lidar, only a slice of the lab is actually mapped. Also, because of occlusions, the back side of the other robot (top right corner) is not observed, and therefore, not mapped. In this example, the resolution is set to 5cm-side voxels. The resolution should be

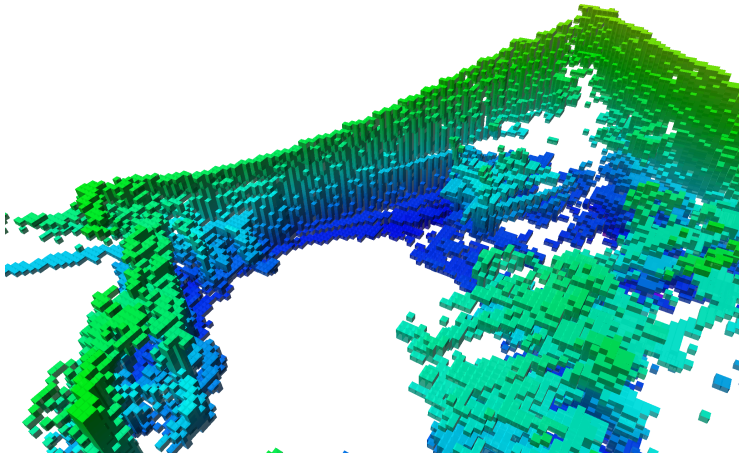

Figure 6: Example of a 3D-grid map, mapping the same place depicted in 2. Each cube is a 5cm side voxel, representing only the occupied space. The color encodes only the height, to help the visualization.

chosen carefully to be consistent at the same time with the type of objects we want to map in, and the level of noise when building the map, particularly in the robot localization. As an example, if the position of the robot is known with an $0.2m^2$ uncertainty, building a 3D-grid with a $0.05m$ resolution is not suited and will lead to a highly noisy map.

2.4 Challenges with mapping

The core of the mapping challenge comes from the fact that the robot is moving. The robot's observations are always in the sensor frame. The very frame attached to the sensor. As shown in Section 1.2, that frame is easily transformed into the robot frame, because the pose of the sensor with respect to the robot is known, even though some uncertainty remains. The challenge comes from transforming the robot's frame into the map frame, the only fixed frame in time. This challenge is the localization introduced previously. Every robot's observation, in the robot's frame, updates the map. Every error in the estimate of the robot's pose in the map leads to errors in the map's updates. The uncertainty in the robot pose produces uncertainty in the map, as shown in [Chahine et al., 2021].

An additional issue is linked to the sensor itself. A Lidar is not exempt from measurement noise. Furthermore, the noise level tends to increase in natural

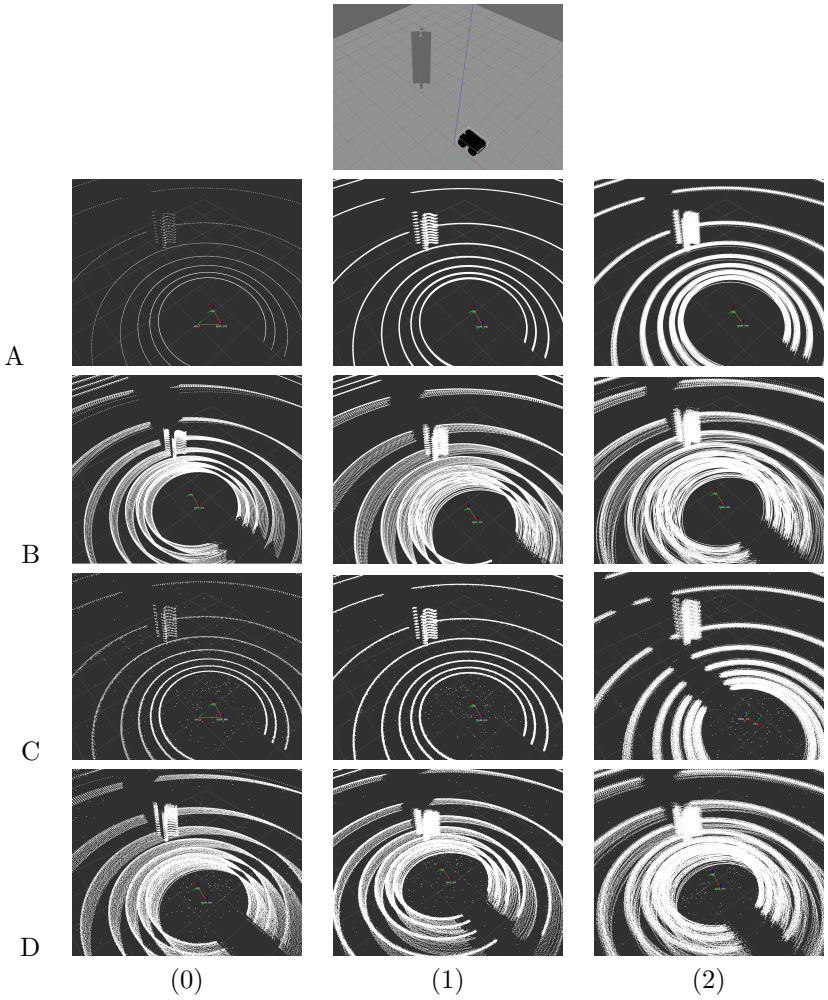

Figure 7: Illustration of the effect of noise on the map, on a toy-case. The noise is applied either on the localization, on the sensor, or on both. Top: the simulation: a husky-robot facing a cross-extruded shape. All the other figures display the point-clouds accumulated for five seconds in a given map frame. The frame is either the perfect localization from the simulation (0), or a noisy localization, where we apply a Gaussian noise to (0). We display two level of noise: (1) and (2). Rows A, B: the Lidar sensor is perfect. Rows C, D: the Lidar sensor is noisy. Rows A, C: the robot is not moving. Rows B, D: the robot is moving. If we focus on the cross-extruded shape, we can see that the higher the noise in the localization and / or on the sensor, the blurrier the shape.

environments, due to several factors. For instance, from a laser's perspective, trees can behave as semi-transparent structures, thereby inducing errors in the measurements. The reason is that when laser-rays, light cones in practice, reach a small object, like a branch, often only a portion of the energy is reflected. This causes multiple echoes, which are a common source of inaccurate distance readings. For instance, when measuring distance to a small branch, depending on the amount of vegetation, the intensity of each echo may vary. The distance reading, associated to the highest intensity echo, may be the small branch, another branch behind, or some larger structure behind the tree, for example. A comprehensive analysis of this phenomenon is provided by [Vosselman, 2010]. Two other causes of errors in laser measurements, that are particularly abundant in natural environments, are linked to the distance to the objects and the incidence angle: the error increases with each of them [Laconte et al., 2019].

Figure 7 illustrates the effect of noise on the map. Rows A and B show the effect of noise only in the localization. Row C depicts the effect of noise on the Lidar only on the map. Finally, row D exemplifies the effect of a combination of noise of the lidar and noise on the localization. Looking at column (0), where the localization is perfect, we can see that the addition of noise in the sensor blurs the cross extruded shape, but much less than the addition of noise in the localization (column (2)). The examples in Figure 7 are not maps, but simply accumulations of point-clouds in time. Regarding maps, especially occupancy grids, it is worth noting that every point-cloud contributes to the map. Indeed, every return point in the point-cloud, with its noise, is transformed into the map frame, with the noise in the transformation, and used to update the map. In that case, the occupancy likelihood of the pixel containing the returned point is increased, the occupancy likelihood of the pixels traversed by the laser ray are decreased.

2.5 Purpose of mapping

Often, robot mapping is intrinsically linked to the autonomous navigation of the robot. Planning a trajectory within an occupancy grid consists in computing a trajectory from the robot's current pose to the goal, while ensuring it stays in free space and maintains a safe distance from obstacles. This computation relies on the values associated with the pixels of the occupancy grid, that is, the occupancy likelihoods. Optimal planning in a 2D-plane may be considered a solved research problem, and several algorithms exist to compute optimal plans.

It is worth noting that we are specifically discussing here global planning, which involves going from point A to point B. The distance from A to B can be large, and in addition to the global plan, the robot actually needs to follow what is called a local plan, reactive for example to moving obstacles, and to be controlled on this local plan. Planning will be briefly presented later in this manuscript.

However, there are cases where the purpose of the map is not the planning. For instance, it may be the need to build a representation of a scene enabling its monitoring, such as in [Chahine et al., 2022].

3 Autonomous Navigation

Autonomous navigation is a high-level task in mobile robotics. It consists of enabling for the robot to move autonomously and robustly in its environment. Autonomous navigation is divided in three components: global planning, local planning and control. Exploration is a higher level task that consists in choosing the next goal. Often, global planning is part of the exploration task, as we will see right after.

First, we focus on the three navigation tasks.

3.1 Global Planning

Global planning consists in planning a safe trajectory from the current position of the robot, A, to its target position, B. Planning such a trajectory requires several components. First, we need to define where the robot can or cannot go in the map. If we take the example of a 2D-occupancy grid map, we can assume the robot can navigate safely in free space. Two questions would be: can it navigate safely close to obstacles ? Can it navigate safely in unknown space ? Answering those questions require the computation of an intermediary component called the cost-map. This cost-map is a 2D-grid where each cell is associated to a cost. Planning from A to B would then consist in finding the lower cost trajectory in the set of feasible ones from A to B. A typical cost-map consists in simply expending the obstacles in the occupancy grid with a Gaussian Kernel. Cells close to obstacles would then have a higher cost than those far from obstacles.

More advanced cost-maps would take other factors than simply the distance to obstacles. For example, the cost of a cell may be linked to the slope of the terrain it is associated to. Similarly, each cell could be assigned a traversability

score depending on the very nature of the terrain. For instance, asphalt is easily traversable, whereas rocks are not traversable. In the middle, short grass could be traversable, although less easily than asphalt, and tall grass may be traversable, but with some risk, and the cost should reflect the level of risk of planning through each cell. Global planning is finally an optimization to seek for the cheaper trajectory from A to B. Several algorithms allow to compute this best trajectory, among which the more broadly used are Djikstra, A*, or RRT*. We recommend the interested reader to look for [Siegwart et al., 2011] that provide a comprehensive description of the main planning algorithms.

3.2 Local planning and control

Although local plan and control are out of the scope of this work, we provide to the reader high-level information about what they are in the context of mobile robots, and again invite the reader to look for [Siegwart et al., 2011], a reference book in mobile robots.

Local planning enables the local adaptation of the robot to the global plan. The more obvious reason is moving obstacle avoidance. By definition, it is not possible to plan a path from A to B avoiding moving obstacles. This reactive behavior is what local planning deals with. Local planning is reactive planning to keep executing the general trajectory from A to B, while taking into account local changes.

Control is the final step in autonomous navigation. It consists in computing and applying the commands that will drive the robot to follow the local plan.

4 Exploration

4.1 Exploration with frontier points

As introduced before, in an exploration problem, we consider a volume unknown at the beginning. The robot moves into this volume, and at the same time gathers information and reduces the unknown. Crafting an exploration policy consists in defining the rule to choose where the robot should go next. That policy is directly linked to the reason why the robot has to explore the environment. Generally, whether it be only for its navigation, or for other purposes, the map is directly linked to the exploration. Let us assume the map is a 2d-occupancy-grid map introduced before. This map encodes the occupancy likelihood of each

pixel, and therefore contains information about occupied space, empty space, and unknown space. Occupied and empty space together are the known volume, the remaining is the unknown volume. From this frontier between the known and unknown volume [Yamauchi, 1997] introduced the concept of frontier points. A frontier point is a point, within the known volume, with at least an unknown neighbor. From this concept derives the first autonomous exploration policy, also called closest frontier exploration. In closest frontier exploration, the next goal is always the closest from the current robot position in the set of reachable frontier points. This decision-making process, that drives the choice of the next goal to visit, is called the exploration policy. The primary objective of a closest-frontier exploration policy is to rapidly reduce the unknown in the map. Another traditional exploration policy, derived from frontier points, consists in randomly sampling the goal among the frontier points rather than choosing the closest. This policy is commonly referred to as random frontier exploration.

In a typical exploration mission, the sequence of actions involves: setting the goal, moving to the goal, updating the map, setting a new goal, and repeating this cycle. However, this behavior is not necessarily linear. For instance, the map can be updated while the robot moves. Similarly, the next goal is often regularly sampled, without waiting for the robot to reach the current goal. This approach is used because, due to map updates while the robot is moving, a goal that is a few seconds old may no longer be the optimal choice. In traditional policies, the cycle is over when the exploration is considered complete. This determination is often based on specific criteria, such as mission time, or achieving a minimal percentage of the volume being known.

Figure 8 provides an illustration of the concept of frontier points. The current position of the robot is the green square. A, B and C are the main clusters of frontier points (some frontier points are not in A, B or C). Depending on the exploration policy, the robot may pick a goal in A (closest), in B (highest density of unknown points for instance), or either in C (random for instance).

Whereas out of the scope of this work, it should be mentioned that the concept of frontier point exploration is often extended to multi-agents. Indeed, when the objective is to discover the volume as quickly as possible, having robots cooperate to build a common map is an appealing solution, as proposed initially by [Burgard, Wolfram Mark, Moors Dieter, Fox Reid and Sebastian, 2000].

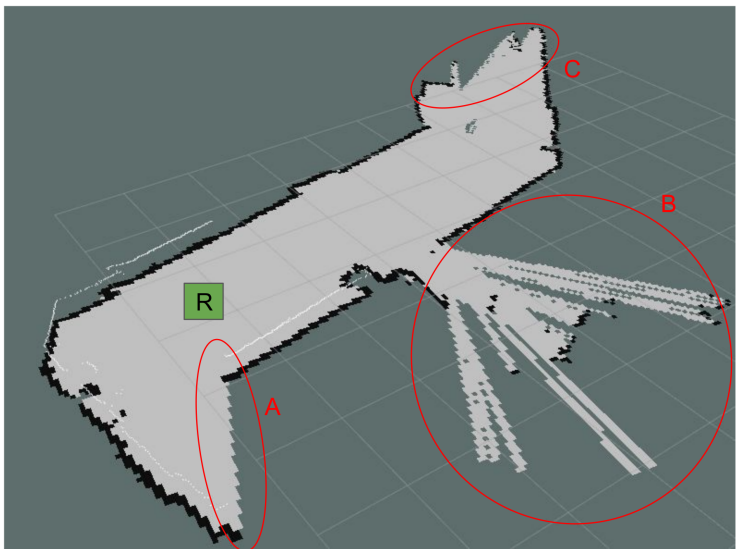

Figure 8: Illustration of the concept of frontier points in an occupancy grid. The current position of the robot is the green square. A, B and C are the main clusters of frontier points

4.2 Next Best View

As mentioned earlier, the choice of the next goal depends not only on the target destination, but also on the path the robot will take. During the execution of the path, observations continuously update the map. Because of this continuous update, a new goal is typically chosen during the execution of the path. In this context, the robot's actuation is driven by the objective of maximizing perception efficiency, a strategy commonly referred to as *Next-Best-View* (NBV) or *active sensing*

NBV is an active research topic in various contexts. For instance, [Bartolomei et al., 2020] selects NBVs based on their "perceptual informativeness" to minimize localization error. The underlying idea is that when the primary sensor is a camera, avoiding featureless areas increases the performance of visual inertial odometry algorithms.

When NBV and exploration are linked, the exploration policy is the function defining a balance between the need to maximize the efficiency of the perception (the information gain), and some cost (for example, the distance to the candidate), to select the best goal for the exploration task. Often, exploration policies are linked to simultaneous localization and mapping (SLAM). SLAM,

introduced earlier in this chapter (Section 1.2), is an optimization that seeks at the same time to find the map, and the poses, or history of poses, of the robot. When exploration and SLAM are linked, the gain is often formulated to reduce both map and localization uncertainty, as in [Stachniss et al., 2005]. Beyond SLAM, the objective of the NBV selection, within the context of exploration, often revolves around finding the path that maximizes space discovery, as shown in [Bircher et al., 2016] for instance.

References

[Aravecchia, 2023] Aravecchia, S. (2023). *Map Quality Criteria for Autonomous Exploration in Natural Environment.* PhD thesis, Université de Lorraine, France.

[Bartolomei et al., 2020] Bartolomei, L., Teixeira, L., and Chli, M. (2020). Perception-aware Path Planning for UAVs using Semantic Segmentation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5808–5815.

[Bircher et al., 2016] Bircher, A., Kamel, M., Alexis, K., Oleynikova, H., and Siegwart, R. (2016). Receding horizon next-best-view planner for 3D exploration. *Proceedings - IEEE International Conference on Robotics and Automation*, 2016-June:1462–1468.

[Burgard, Wolfram Mark, Moors Dieter, Fox Reid and Sebastian, 2000] Burgard, Wolfram Mark, Moors Dieter, Fox Reid, S. T. and Sebastian (2000). Collaborative Multi-Robot Exploration. In *IEEE international conference on robotics and automation.*

[Center and Services, 2012] Center, N. O. and Services, A. A. N. C. (2012). Lidar 101 : An Introduction to Lidar Technology , Data , and Applications. *NOAA Coastal Services Center*, (November):76.

[Chahine et al., 2022] Chahine, G., Pradalier, C., Chahine, G., Pradalier, C., Alignment, S.-a., and Outdoor, N. (2022). Semantic-aware spatio-temporal Alignment of Natural Outdoor Surveys To cite this version : HAL Id : hal-03738518 Semantic-aware spatio-temporal Alignment of Natural Outdoor Surveys.

[Chahine et al., 2021] Chahine, G., Vaidis, M., Pomerleau, F., and Pradalier, C. (2021). Mapping in unstructured natural environment: a sensor fusion framework for wearable sensor suites. *SN Applied Sciences*, 3(5):1–14.

[Corke et al., 2011] Corke, P. I., Jachimczyk, W., and Pillat, R. (2011). *Robotics, vision and control: fundamental algorithms in MATLAB*. Springer.

[Fox et al., 2006] Fox, D., Thrun, S., and Burgard, W. (2006). *Probabilistic Robotics*. Kybernetes.

[Gonzalez et al., 1994] Gonzalez, J., Ollero, A., and Reina, A. (1994). Map building for a mobile robot equipped with a 2D laser rangefinder. *Proceedings - IEEE International Conference on Robotics and Automation*, (pt 3):1904–1909.

[Kostavelis and Gasteratos, 2015] Kostavelis, I. and Gasteratos, A. (2015). Semantic mapping for mobile robotics tasks: A survey. *Robotics and Autonomous Systems*, 66:86–103.

[Laconte et al., 2019] Laconte, J., Deschênes, S. P., Labussière, M., and Pomerleau, F. (2019). Lidar measurement bias estimation via return waveform modelling in a context of 3D mapping. *Proceedings - IEEE International Conference on Robotics and Automation*, 2019-May:8100–8106.

[Siegwart et al., 2011] Siegwart, R., Nourbakhsh, I. R., and Scaramuzza, D. (2011). *Introduction to autonomous mobile robots*.

[Stachniss et al., 2005] Stachniss, C., Grisetti, G., and Burgard, W. (2005). Information gain-based exploration using rao-blackwellized particle filters. *Robotics: Science and Systems*, 1:65–72.

[Vosselman, 2010] Vosselman, G. (2010). *Airborne and Terrestrial Laser Scanning*. Whittles Publishing.

[Yamauchi, 1997] Yamauchi, B. (1997). Frontier-based approach for autonomous exploration. *Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation, CIRA*, pages 146–151.