A Novel Occupancy Mapping Framework for Risk-Aware Path Planning in Unstructured Environments

Johann Laconte, Abderrahim Kasmi, François Pomerleau, Roland Chapuis, Laurent Malaterre, Christophe Debain, Romuald Aufrère

sensors

MDPI

Article

A Novel Occupancy Mapping Framework for Risk-Aware Path Planning in Unstructured Environments

Johann Laconte [1,*], Abderrahim Kasmi [2], François Pomerleau [3], Roland Chapuis [1], Laurent Malaterre [1], Christophe Debain [4] and Romuald Aufrère [1]

[1] Institut Pascal, CNRS, Clermont Auvergne INP, Université Clermont Auvergne, F-63000 Clermont-Ferrand, France; roland.chapuis@uca.fr (R.C.); laurent.malaterre@uca.fr (L.M.); romuald.aufrere@uca.fr (R.A.)
[2] Sherpa Engineering, R&D Department, 333 Avenue Georges Clemenceau, 92000 Nanterre, France; abderrahim.kasmi@etu.uca.fr
[3] Norlab, Université Laval, Québec, QC G1V 0A6, Canada; francois.pomerleau@ift.ulaval.ca
[4] INRAE, UR TSCF, Université Clermont Auvergne, F-63178 Aubière, France; christophe.debain@inrae.fr
* Correspondence: johann.laconte@uca.fr

Abstract: In the context of autonomous robots, one of the most important tasks is to prevent potential damage to the robot during navigation. For this purpose, it is often assumed that one must deal with known probabilistic obstacles, then compute the probability of collision with each obstacle. However, in complex scenarios or unstructured environments, it might be difficult to detect such obstacles. In these cases, a metric map is used, where each position stores the information of occupancy. The most common type of metric map is the Bayesian occupancy map. However, this type of map is not well suited for computing risk assessments for continuous paths due to its discrete nature. Hence, we introduce a novel type of map called the Lambda Field, which is specially designed for risk assessment. We first propose a way to compute such a map and the expectation of a generic risk over a path. Then, we demonstrate the benefits of our generic formulation with a use case defining the risk as the expected collision force over a path. Using this risk definition and the Lambda Field, we show that our framework is capable of doing classical path planning while having a physical-based metric. Furthermore, the Lambda Field gives a natural way to deal with unstructured environments, such as tall grass. Where standard environment representations would always generate trajectories going around such obstacles, our framework allows the robot to go through the grass while being aware of the risk taken.

Keywords: risk assessment; path planning; risk modeling; occupancy grid; safe navigation; field robotics

Citation: Laconte, J.; Kasmi, A.; Pomerleau, F.; Chapuis, R.; Malaterre, L.; Debain, C.; Aufrère, R. A Novel Occupancy Mapping Framework for Risk-Aware Path Planning in Unstructured Environments. *Sensors* **2021**, *21*, 7562. https://doi.org/10.3390/s21227562

Academic Editor: Ramon Barber

Received: 27 September 2021
Accepted: 12 November 2021
Published: 14 November 2021

1. Introduction

Nowadays, autonomous robots are more and more visible in our lives. They start to prove themselves useful in a very broad spectrum of applications, from autonomous driving to supporting humans in dangerous jobs such as mining or search and rescue missions. One common aspect of every robot's tasks is the notion of safety; before taking any action, the robots have to assess the associated risk of the action.

To assess such a risk, robots need a way to represent and store the surrounding environment. In structured and controlled environments, such as warehouses, the easiest solution is to provide the robot with a map of the environment, as well as the positions of every obstacle, robot, and operator. Storing such entities leads to the construction of semantic maps, where each obstacle is stored as an object (e.g., a wall, operator, or robot). Under this representation, the robot has to keep track of every moving obstacle while avoiding collisions with the environment. However, such a representation of the environment is not always available or easy to build from raw data in all situations. For example, it is impossible to perfectly describe the underlying environment of a snowy forest or a crowded park. There are indeed many unstructured obstacles in the first

case that are not easily storable in such semantic maps, while in the second one, a lot of dynamic obstacles hinder the construction of a precise map. Clustering raw data from Light Detection And Ranging (lidar) measurements, as done by Fulgenzi et al. [1], for example, might not be possible for the aforementioned scenarios.

When such high-level environmental representation is not available or possible, a lower-level map is constructed, which called a metric map. Instead of storing features, the metric map tessellates the environment into cells, where each one stores the information of occupancy. This kind of map has been heavily studied and used since the beginning of robotics. They were introduced by Elfes [2], who proposed the concept of occupancy grids. Each cell stores the probability that the underlying environment is occupied and, hence, not traversable for the robot. This type of map is easy to construct and can be used to perform a great variety of tasks, such as Simultaneous Localization and Mapping (SLAM) or path planning. However, as previously demonstrated by Heiden et al. [3], a problem quickly arises when the robot wants to assess the probability of collision for a given path. Indeed, we are tempted to assess the probability of collision as the joint probability that every cell is free of obstacles. As an example, Figure 1 shows a robot crossing an environment where the probability of occupancy is 0.1 for each cell. Depending on the tessellation size, the probability of collision can be 0.19 or 0.34 for a tessellation half as small as the first one. This behavior comes from the fact that the correlation between the occupancy of two positions of the environment is not null. Nevertheless, it is impossible to accurately estimate this correlation. The same problem arises when dealing with occupancy grids stored in quad-trees [4]. Indeed, the robot could decide to cross a large high-probability cell instead of ten small low-probability ones.

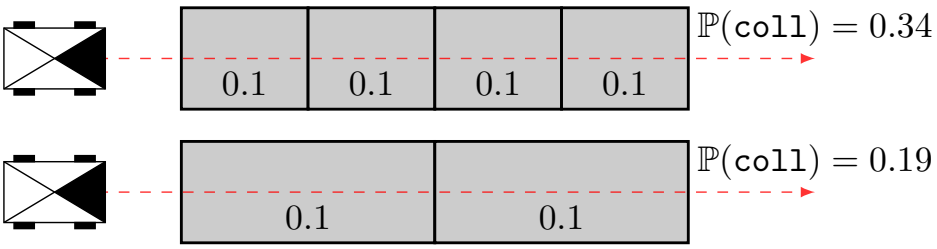


Figure 1. Example of collision assessment in an occupancy grid. The robots (black boxes with their front represented as a filled triangle) want to cross an environment by following the dashed red line. The collision probability is uniform for the whole environment (0.1). The discretization size greatly influences the probability of collision, with the bottom scenario yielding a safer path even though the underlying environment is the same.

Furthermore, the probability of collision is not well suited to describing the risk in complex situations. For example, crossing a part of the environment at 5 km/h is not as risky as crossing the same environment at 70 km/h. The damages caused by a potential collision are far more consequential at a higher speed.

Under these considerations, we introduce the concept of the Lambda Field. The Lambda Field is a representation of an environment that allows the computation of the probability of collision while being independent of the tessellation size. It also provides a natural way to assess more complex risks than the probability of collision. Our framework consists of a novel occupancy-mapping technique, as well as a formulation for assessing risks on it, thus yielding a direct way for path-planning algorithms to work on these maps. Our key contributions are:

- A novel type of map called the Lambda Field, which is specially designed to allow generic risk assessments and is better fitted for unstructured environments;
- A mathematical formulation of risk assessment over a path with an application to path planning in tall grass;
- A theoretical and experimental evaluation of the Bayesian occupancy grid, showing that such a framework can over-converge in the case of unstructured and sparse obstacles.

In this paper, we provide a revised and extended version of our previous work [5]. We give an extended theory that takes into account the mass of the obstacles, allowing the robot to move in unstructured environments. Moreover, we improve the theory of Heiden et al. [3] to take into account the robot size and prove that the Lambda Field is a generalization of their theory. We also extend our results with tests in real-world conditions, in both structured and unstructured environments, showing that Lambda Fields better map unstructured environments and allow behaviors that are impossible using classical occupancy grids.

2. Related Work

In order to perform path planning, the first step is to construct a representation of the environment. In the context of unstructured or complex environments, a semantic map is impossible to create and a lower-level representation, called a metric map, has to be used. This kind of map tessellates the environment into cells, where each one stores the information of occupancy. The idea of tessellating the sensed environment was originally proposed by Elfes [2]. Later, Coué et al. [6] enhanced this idea by adding a Bayesian layer, which better handles uncertainty and noisy readings, increasing the robustness of the map. Many variations of the Bayesian occupancy filter have been developed over the years, mainly adding dynamic obstacles in the grid. Saval-Calvo et al. [7] wrote a review of the different Bayesian Occupancy Filter frameworks, presenting a taxonomy of the methods. O'Callaghan and Ramos [8] proposed a way to store the occupancy map without discretization by using Gaussian processes. This method keeps the dependence between cells in the grid, which was not the case in the original occupancy grid from Elfes [2]. In an attempt to reduce the complexity of the Gaussian process, Kim and Kim [9] used overlapping local Gaussian processes. Regardless of this amelioration, the time computational complexity is still an issue for these kinds of methods, whereas the standard occupancy grids, as well as our method, do not suffer from such problems. Ramos and Ott [10] developed an analog method using Hilbert maps (HMs), overcoming the computational complexity of the Gaussian process. In order to take into account the uncertainty in the different parameters, the method was extended to Bayesian Hilbert maps by Senanayake and Ramos [11]. Lately, HMs have been generalized to dynamic environments by Guizilini et al. [12], allowing real-time occupancy predictions. These methods still need to tune parameters that have great consequences on the quality of the resulting maps. An alternative algorithm for occupancy grids was presented in Agha-mohammadi et al. [13] by storing richer data in a map, taking into account the estimation of the variance for each cell. Under these considerations, our framework and the Bayesian occupancy grid are very alike, as the environment is tessellated into cells. The Lambda Field also stores a confidence interval over each cell in the same fashion as Agha-mohammadi et al. [13].

Once a representation of the environment is available, the robot can start to assess risk in its map. The risk was first defined by the likelihood of not colliding with anything, as suggested by Fraichard [14]. In the context of autonomous driving, the Time to Collision introduced by Lee [15] is widely used. This metric measures the time at which the robot will collide with a specific obstacle given the current path. The Time to Collision is useful in accident mitigation systems, but is not well fitted for long-term planning. It is mainly useful for mitigating the speed of a vehicle in traffic. It has been demonstrated by Laugier et al. [16] that the Time to Collision lacks context, and is hence not the best solution for every situation. Furthermore, this kind of metric is used in the context of known dynamic obstacles and is not easily transposable for path-planning algorithms. Given these factors, the risk has to be defined in another fashion in the context of path planning in occupancy grids.

The risk is also very dependent on the application. For example, Vaillant et al. [17] and Caborni et al. [18] tackled the problem of path planning for neurosurgery. For this application, the risk depends essentially on which zones of the brain the tool goes through. It is very different from standard path-planning risks, as in this context, we are sure to collide

with a part of the brain for every possible path. In that sense, the path-planning framework should be able to tackle different types of risk, which is the case of the Lambda Field.

Majumdar and Pavone [19] addressed the issue of how a robot should quantify risk and what constitutes a 'good' risk metric. They came to the conclusion that a risk measure is said to be coherent if it satisfies axioms, showing that otherwise, the risk metric can have undesired behaviors. However, the physical meaning is neglected, which leads to non-intuitive definitions for risk metrics and difficult parameter settings for path planning. In our work, we thereupon define the risk as an understandable quantity, which is, for our application, the expected force of collision on a given path. Our metric also respects the axioms defined by Majumdar and Pavone [19], leading to a risk proven to behave as desired, as well as having a physical unit.

In the context of occupancy grids, a risk map is widely used to deal with the risk. For path planning, the occupancy grid is replaced with a risk map, where each cell stores the risk at this position. The higher the risk, the more the robot should avoid this place while planning its trajectory. Tsiotras and Bakolas [20] used wavelets to store the environment and a risk map at different scales; for this application, the risk was defined as the probability of occupancy. Then, the path-planning method was to find the path minimizing the overall sum of the risk of each traversed cell. Hence, the total risk lacks physical meaning. Our framework differs from the previous one, as the total risk of a path has the same unit as the risk metric used. In another context, De Filippis et al. [21] used a risk map to control the altitude of an unmanned aerial vehicle. The risk was defined as the probability of flying in unsafe conditions for a given point on the map. In the same fashion, Primatesta et al. [22] defined the risk as the hourly probability of lethal incidents for each position of the unmanned aerial vehicle. The risk was also set to the maximum for obstacles and no-fly zones. Joachim et al. [23] used a risk map to prevent a robot from going too close to dangerous obstacles, such as pedestrians or other cars, allowing the robot to safely navigate in narrow spaces, such as parking spots. Pereira et al. [24] also used a risk map to find the best path for underwater vehicles, where the risk was set to the probability that the position was occupied by an obstacle. Although all these methods demonstrated good results, they all assumed that the risk was only a function of the position, omitting the robot configuration. As said before, the robot configuration can greatly change the risk. For instance, going to a position at high speed is often more dangerous than going at a low speed. Feyzabadi and Carpin [25] defined a risk function that depends on the position as well as the robot action. As a result, the robot could choose to go to a position only if its speed was low enough. Our framework uses the same idea while giving a physical meaning to the cost of the overall path. Therefore, the probability of collision is a metric that is too simple to perfectly describe the risk. As said by Eggert [26], in the case of ADAS systems, we would rather want to assess the expected damage done to the vehicle than the probability of collision. We then propose a framework allowing the computation of a generic risk that can be defined depending on the application.

Using its representation of the environment and a risk function, the robot can start planning. Many of the popular methods use a binary representation of the environment, meaning that any point in the environment is either free or occupied. A review of such algorithms can be found in Tsardoulias et al. [27]. The most common way to convert the Bayesian grid into a binary grid is to apply a user-defined threshold, as done by Yang et al. [28]. However, applying a threshold to the environment might lead to discarding some obstacles, commanding the robot to plan entry into potentially occupied zones. A review of algorithms of path planning in occupancy grids was done by Čikeš et al. [29]. The different algorithms presented in this article all aim to minimize the cost function of the path, which is the sum of the cost of each traversed cell. The cost of a path has no physical meaning; thus, determining if the path is truly safe might become a difficult task.

Another method proposed by Fulgenzi et al. [1] is to cluster the occupancy grid, leaving the unclustered space as free or occluded. The risk assessment is then reduced to evaluating the risk for probabilistic known obstacles. However, such clustering can be very difficult to compute in unstructured environments. We thus need a way to evaluate the cost

of a path in occupancy grids while taking into account the probability of occupancy. Using Rapidly Exploring Random Trees, Fulgenzi et al. [30] and Fulgenzi et al. [31] defined the cost of a path as the joint probability of not having a collision in each node. Their framework assumes that traveling between nodes is risk-free; if we do not make this assumption, we fall back on the initial problem of computing a cost over a continuous path. To overcome the problem of computation of the risk over a continuous path, several methods have been proposed. Rummelhard et al. [32] defined the risk in a Bayesian occupancy grid as the maximum probability of collision over the cells. Nevertheless, there is no natural way to include a more complicated risk in the framework, and these metrics can show unintended behaviors in complicated scenarios. Indeed, traversing one high-probability cell has the same risk as traversing ten cells of the same probability for the second metric. Dhawale et al. [33] chose to represent the environment as a Gaussian process and represented the obstacles using a threshold over the Gaussians. Doing this dissociates free space and occupied space, falling back on the methods presented in Tsardoulias et al. [27]. Gerkey and Konolige [34] computed the cost of a path by summing the probability of occupancy of the cells that the path crosses. This sum is then injected into a global cost function, taking into account other constraints, such as the speed or the distance to the objective, where each constraint has a user-defined coefficient. Francis et al. [35] used the same idea for path planning in Hilbert maps, which were introduced by Ramos and Ott [10]. The drawback of these methods is that the cost lacks physical meaning, as they sum probabilities. Since this sum does not have any physical unit, its associated coefficient does not have one either, making its tuning non-intuitive for the user. Finally, Heiden et al. [3] used the concept of the product integral to compute the probability of collision over a path. It leads to a probability of collision, but this method has no physical meaning. We show in this article that our framework can be seen as the generalization of their framework.

3. Theoretical Framework

We present in this section the theoretical framework for assessing a generic risk over a path in Lambda Fields. First, we justify the use of the mathematical tools by showing how they naturally arise while dealing with continuous environments. Then, we address the construction of the Lambda Fields in Section 3.1, as well as a way to compute confidence intervals over the field in Section 3.2. Indeed, the more the cells are measured, the more confident the robot should be to move. Next, we present in Section 3.3 a framework capable of assessing a generic risk over a path in a Lambda Field. We then extend this framework and design a risk function allowing the robot to navigate in unstructured environments, such as tall grass, in Section 3.4. Finally, we improve the framework of Heiden et al. [3] in Section 3.5 to take into account the size of the robot and show that, under our improvement, it can be seen as a special case of our framework.

The key concept of the Lambda Field is its ability to assess the probability of collision inside a subset of the environment (e.g., the path of the robot), leading to the computation of a generic risk that can be adjusted depending on the scenario. To better understand the reasons for the following framework, we will first demonstrate its construction. We assume that the probability of encountering a collision for a path of area Δa is $\lambda_i \Delta a$, where $\lambda_i \in \mathbb{R}_{\geq 0}$ is the rate of the event 'collision' and $\Delta a \to 0$ such that $\lambda_i \Delta a \leq 1$. The larger the intensity λ_i is, the more likely it is that a collision will occur. In a macroscopic approach, the intensity λ_i corresponds to the expected number of collisions in a cell of area $1\,\mathrm{m}^2$ and can, therefore, vary from 0 (i.e., the cell will never create a collision) to $+\infty$ (i.e., the cell will create an infinite number of collisions during the traversal).

The probability of crossing N surfaces of areas Δa with a rate λ_i without collision is

$$\prod_{i=0}^{N-1} (1 - \lambda_i \Delta a). \tag{1}$$

Taking the limit of the path area $\Delta a \to 0$ leads to the computation of the Volterra type I product integral. For a path crossing a total area of A where each subregion of area Δa has a rate $\lambda(a)$, a being the total area crossed from the beginning, we have

$$\lim_{\Delta a \to 0} \prod_{i=0}^{A/\Delta a} (1 - \lambda(i\Delta a)\Delta a) = \exp\left(- \int_0^A \lambda(a)\, \mathrm{d}a \right). \tag{2}$$

A proof of Equation (2) can be found in [36]. The probability of encountering no collisions over a path is then the probability that no event 'collision' happens in a heterogeneous Poisson point distribution of rate $\lambda(a)$. Taking the limit of a binomial distribution indeed leads to a Poisson point process distribution. Hence, the natural way of dealing with collisions in a continuous manner is to use a Poisson point process distribution. This process counts the number of events that have happened given a certain area, depending on the mathematical space. In our case, we want to count the number of the 'collision' events that could occur given a path (i.e., a subset of $\mathbb{R}^2$). We point out that the theory is here presented for 2D paths, but the extension in $\mathbb{R}^3$ is trivial, as the only change is the tessellation of the map being in 3D instead of 2D. For a positive scalar field $\lambda(\boldsymbol{x})$, with $\boldsymbol{x} \in \mathbb{R}^2$, the probability of encountering at least one collision in a path $\mathcal{P} \subset \mathbb{R}^2$ is

$$\mathbb{P}\big(\texttt{coll}|\mathcal{P}\big) = 1 - \exp\left(- \int_{\mathcal{P}} \lambda(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} \right). \tag{3}$$

Nonetheless, it is impossible to both compute and store the field $\lambda(\boldsymbol{x})$, as it has an infinite number of degrees of freedom. Hence, we tessellate our field into cells of a fixed size in a fashion similar to that of Bayesian occupancy grids. Throughout this paper, we will assume that we are dealing with a tessellated field where each cell has an area $\Delta a \in \mathbb{R}_{>0}$. By tessellating the field, the probability of collision is given by

$$\mathbb{P}\big(\texttt{coll}|\mathcal{P}\big) = 1 - \exp\big(-\Lambda(\mathcal{C})\big) \quad \text{with } \Lambda(\mathcal{C}) = \Delta a \sum_{c_i \in \mathcal{C}} \lambda_i, \tag{4}$$

for a path $\mathcal{P}$ crossing the cells $\mathcal{C} = \{c_i\}_{0:N-1} = \{c_0, \ldots, c_{N-1}\}$, where each cell c_i has an area of Δa and an associated lambda λ_i, which is the intensity of the cell. The lambda can be seen as a measure of the density of the cell: The higher the lambda is, the more likely it is that a collision will happen in this cell.

Using this representation, we hereby see that the probability of collision is not dependent on the size of the cells. It is indeed the same to compute the probability of collision for crossing two cells of area $\Delta a/2$ or one cell of area Δa for a constant λ.

3.1. Computation of the Field

As we established a new approach to representing the occupancy of an environment, we need to develop a way to dynamically compute the lambdas. We assume that the robot is equipped with a lidar sensor, which gives a list of cells crossed by beams without collision and another list of cells where the beams collided. Using this sensor model, we construct the Lambda Field in the following manner. We want to find the combination of $\lambda = \{\lambda_i\}_{0:M_C-1}$ for a map tessellated into M_C cells that maximizes the expectation of the K beams that the lidar has shot since the beginning. In addition, each lidar beam has an associated error region $\mathcal{E}_k$ of area e_k centered on the measurement, meaning that the actual obstacle is in $\mathcal{E}_k$. Figure 2 shows an example of such a lidar beam error region.

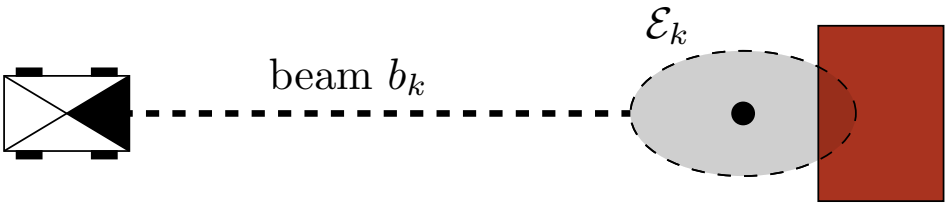


Figure 2. The robot measures an obstacle using a lidar sensor. The obstacle (in red) is in the area $\mathcal{E}_k$ (in gray) centered on the measurement (black dot).

Therefore, each lidar collision gives a region where an obstacle is. This kind of sensor simplification is common and was used, for example, by [37]. At this stage, we assume that every lidar measurement possesses the same error region area e. The case where each beam has a different error region is covered in Appendix A, which can be useful for radar measurements or lidars with substantial beam divergence. For each lidar beam b_k, the beam crossed the cells $c_m \in \mathcal{M}_k$ without collision and hit an obstacle contained in the cells $c_h \in \mathcal{E}_k$. The log-likelihood of the beam b_k is

$$\mathcal{L}(b_k|\lambda) = \ln\Big[\exp\big(-\Lambda(\mathcal{M}_k)\big)\Big(1-\exp\big(-\Lambda(\mathcal{E}_k)\big)\Big)\Big]. \tag{5}$$

The log-likelihood of K lidar beams is then

$$\begin{aligned}\mathcal{L}(\{b_k\}_{0:K-1}|\lambda) &= \sum_{k=0}^{K-1}\mathcal{L}(b_k|\lambda)\\ &= \sum_{k=0}^{K-1}\Big[-\Lambda(\mathcal{M}_k)+\ln\Big(1-\exp\big(-\Lambda(\mathcal{E}_k)\big)\Big)\Big].\end{aligned} \tag{6}$$

We want to maximize this quantity and, hence, nullify its derivative, as the function is concave. In order to find a closed form, we approximate the derivative with the assumption that the variation of lambda inside the error region of the lidar is small enough to be negligible. Thus, for each $\lambda_i \in \mathcal{E}_k$, we have

$$\Delta a \sum_{c_h \in \mathcal{E}_k} \lambda_h \approx e\lambda_i. \tag{7}$$

Using this approximation, the derivative is

$$\frac{\partial\mathcal{L}(\{b_k\}_{0:K-1}|\lambda)}{\partial\lambda_i} \approx -m_i\cdot\Delta a + h_i\frac{\Delta a}{\exp(e\lambda_i)-1}, \tag{8}$$

where m_i is the number of times that the cell c_i has been counted as 'miss' (i.e., was outside the error region), and h_i is the number of times the cell c_i has been counted as 'hit' (i.e., was in the error region of the sensor). We finally find the zero of the derivative, leading to

$$\lambda_i = \frac{1}{e}\ln\left(1+\frac{h_i}{m_i}\right). \tag{9}$$

This closed form allows a low computational complexity of the Lambda Field. We also see that the formula is independent of the size of the cells, which is the main limitation of the current representation that we were aiming at resolving.

We are then able to construct the Lambda Field using Equation (9).

3.2. Confidence Intervals

In the same way as [13], we define the notion of confidence over the values in the Lambda Field. Indeed, the more the cells are measured, the greater the confidence in the robot's movements should be. For each cell c_i, we seek the bounds λ_L and λ_U such that

$$\begin{aligned} &\mathbb{P}(\lambda_L \leq \lambda_i \leq \lambda_U) \geq 95\,\% \\ \Leftrightarrow &\mathbb{P}(\lambda_L \leq \frac{1}{e}\ln\left(1+\frac{h_i}{m_i}\right) \leq \lambda_U) \geq 95\,\%. \end{aligned} \tag{10}$$

To compute those bounds, we introduce the notion of false positives and false negatives: Every cell measurement j has a probability p_j^h of rightfully reading 'hit' and a probability p_j^m of rightfully reading 'miss'. The probabilities p_j^h and p_j^m have to be experimentally computed and can vary according to a great number of parameters; for example, the probability p_j^h is lower in the event of heavy rain or snow.

Using the relation $h_i = M - m_i$, where M is the number of times the cell has been measured, we can rewrite the above equation as

$$\mathbb{P}(K_L \leq h_i \leq K_U) \geq 95\,\%, \tag{11}$$

such that

$$\begin{aligned} \lambda_L &= \frac{1}{e}\ln\left(\frac{K_L}{M-K_L}+1\right), \\ \lambda_U &= \frac{1}{e}\ln\left(\frac{K_U}{M-K_U}+1\right). \end{aligned} \tag{12}$$

The quantity h_i can be seen as a sum of M Bernoulli distributions, such that

$$h_i = \sum_{j=0}^{h_i-1} \bar{h}_j + \sum_{j=0}^{m_i-1} \left(1-\bar{m}_j\right), \tag{13}$$

where $\bar{h}_j$ and $\bar{m}_j$ are Bernoulli variables equal to 1 if the reading was right and 0 otherwise. The quantity $\sum_j \left(1-\bar{m}_j\right)$ is hence the number of times the sensor wrongfully reads 'hit' instead of 'miss'.

The distribution of h_i is not binomial, but a Poisson binomial distribution with poor behaviors in terms of computation. Since the Poisson binomial distribution satisfies the Lyapunov central limit theorem, we can approximate its distribution with a Gaussian distribution of same mean and variance:

$$\begin{aligned} \mu &= \sum_{j=0}^{h_i-1} p_j^h + \sum_{j=0}^{m_i-1} 1-p_j^m \quad \text{and} \\ \sigma^2 &= \sum_{j=0}^{h_i-1} p_j^h(1-p_j^h) + \sum_{j=0}^{m_i-1} p_j^m(1-p_j^m). \end{aligned} \tag{14}$$

We can then have the bounds at 95%, for example, with

$$\begin{aligned} K_L &\approx \max(\mu - 1.96\sigma, 0), \\ K_U &\approx \min(\mu + 1.96\sigma, M). \end{aligned} \tag{15}$$

The bounds λ_L and λ_U are then retrieved from K_L and K_U using Equation (12).

3.3. Generic Framework for Risk Assessment

As mentioned before, the motivation for the Lambda Field is its ability to compute path integrals and, hence, a risk along a path. This risk can be defined depending on the application and is independent of the following framework, meaning that it can be

interchanged without any modification of the theory. For a path $\mathcal{P} \subset \mathbb{R}^2$ crossing the cells $\mathcal{C} = \{c_i\}_{0:N-1}$ in order, the probability density function (p.d.f) over the Lambda Field is

$$f(a) = \exp\Big(n\Lambda\big(\{c_n\}\big) - \Lambda\Big(\{c_j\}_{0:n-1}\Big)\Big) \cdot \lambda_n \exp(-a\lambda_n), \tag{16}$$

where $n = \lfloor a/\Delta a \rfloor$, $\lfloor \cdot \rfloor$ is the standard floor function and $\Lambda(\cdot)$ is defined in Equation (4). The variable a denotes the area the robot has crossed. Note that $\{c_n\}$ is a singleton, i.e., $\Lambda(\{c_n\}) = \Delta a \lambda_n$, whereas $\Lambda(\{c_j\}_{0:n-1}) = \Delta a \sum_{i=0}^{n-1} \lambda_i$ sums n elements (and equals zero if $n = 0$).

One can note that the conversion of the area a into the curvilinear abscissa is trivial, the latter being more convenient for path-planning applications. For a robot of width W that has crossed an area a, its curvilinear abscissa s equals a/W. The length of the robot is not considered, as we assume that the body of the robot only spans cells that have already been crossed. As such, only the front of the robot of width W discovers new cells that are potentially risky.

Furthermore, Equation (16) can be easily proved, as integrating $f(a)$ over a path $\mathcal{P}$ crossing the cells $\mathcal{C} = \{c_i\}_{0:N-1}$ in order gives the probability of encountering at least one collision:

$$\mathbb{P}\big(\texttt{coll}|\mathcal{P}\big) = \int_0^{N\Delta a} f(a)\,\mathrm{d}a = 1 - \exp\big(-\Lambda(\mathcal{C})\big). \tag{17}$$

We can then define the expectation of a risk function $r(\cdot)$ over the path as

$$\mathbb{E}[r(A)] = \int_0^{N\Delta a} f(a) r(a)\,\mathrm{d}a. \tag{18}$$

The random variable A denotes the crossed area at which the first 'collision' event occurs. If the cells are small, we can assume that the function $r(\cdot)$ is constant inside each cell. Using this assumption, we simplify the above equation to

$$\mathbb{E}[r(A)] = \sum_{i=0}^{N-1} K_i r(\Delta a i) \quad \text{with } K_i = \exp\Big(-\Lambda\big(\{c_j\}_{0:i-1}\big)\Big)\Big[1 - \exp\big(-\Lambda(\{c_i\})\big)\Big], \tag{19}$$

for a path $\mathcal{P}$ going through the cells $\{c_i\}_{0:N-1}$.

The risk function $r(\cdot)$ is generic and can take into account the state of the robot, as well as the state of the world. One can notice that the special case $r(\cdot) = 1$ leads to the probability of collision given by Equation (4). Furthermore, the probability density $f(a)$ only looks at the risk generated by the first collision occurring on the path. Therefore, it is assumed that the robot stops after any collision and does not continue its course. This assumption can be lifted if necessary, as shown in the next section.

For our applications, we chose to model the risk as the force of collision (i.e., loss of momentum) if the collision occurs at the area a. It is indeed a good quantification of the damage induced by the collision and is a better metric of the risk than the probability of collision, as shown by Eggert [26]. First, we present as an example a way to assess this risk by assuming that every obstacle has an infinite mass. Indeed, this assumption holds for most scenarios where the robot's mass is negligible compared to the obstacles' masses (e.g., a tree or a wall). We then lift this assumption in Section 3.4, where each obstacle now has a probabilistic mass, allowing the robot to evolve in unstructured environments.

Assuming the obstacle that the robot collides with has an infinite mass, the force of collision is computed as

$$r(a) = m_R \cdot v_R^n(a), \tag{20}$$

where m_R is the mass of the robot, and $v_R^n(a)$ is its velocity towards the obstacle at the area a. As shown in Figure 3, the velocity towards the obstacle of normalized normal $\boldsymbol{n}$ is

$$\begin{aligned} v_R^n &= \left| \boldsymbol{n}^T \boldsymbol{v_R} \right| \\ &= \left| v_R \cdot \cos(\theta) \right| \qquad \text{for } ||\boldsymbol{n}|| = 1 \end{aligned} \tag{21}$$

where $\cdot^T$ stands for the usual vector transpose, $v_R = ||\boldsymbol{v_R}||$ the robot velocity, and θ the angle between the robot heading and the obstacle's normal. The angle of collision is interesting to take into account for numerous scenarios, such as an autonomous vehicle driving over a cliff. Because of skidding, the vehicle may find itself in a configuration where it has no choice but to collide with the safety railing. The best choice will intuitively be to minimize the collision and, hence, collide with the railing with a high incidence angle.

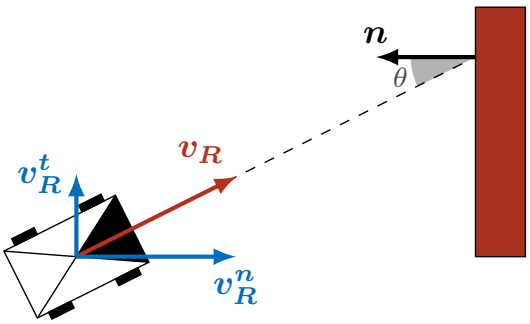


Figure 3. A robot of speed $v_R = ||\boldsymbol{v_R}||$ collides with an obstacle of normal $\boldsymbol{n}$ with an angle θ. The speed of the robot can be decomposed into the tangential component $\boldsymbol{v_R^t}$ and the normal component $\boldsymbol{v_R^n}$. Only the latter influences the collision with the obstacle.

This risk metric assumes that every obstacle the robot might encounter has an infinite mass. We also assume perfect inelastic collision, as most deployed vehicles are designed to absorb collisions as much as possible. This means that if the robot collides with an obstacle, the resulting collision would lead the robot to stop (i.e., losing a momentum of $m_R \cdot v_R^n$). Depending on the application, other metrics can be developed. We present in the next section the development of a more complicated metric allowing the robot to navigate through unstructured obstacles such as tall grass by lifting the approximation that all obstacles have infinite masses.

3.4. Taking into Account the Mass of the Obstacles

In the context of autonomous navigation, the robot might have to go through objects that look like obstacles from the point of view of the lidar, but are in fact harmless for the robot. An ideal example of this scenario is where the robot has to go through tall grass to reach its goal. Since the lidar returns very close measurements of the grass around the robot, the robot would be unable to move. However, with images provided by a camera, an algorithm could clearly detect that the obstacles are only tall grass; hence, the robot should proceed and reach its goal.

As the risk metric developed in the previous section assumes that every obstacle has an infinite mass, it is unable to deal with such scenarios. Thus, this assumption is lifted, and each obstacle is assumed to have a probabilistic mass. We thereby estimate the class of the obstacles in each cell and infer the associated probabilistic mass distribution. This can be done with a camera and deep learning segmentation [38] or radar classification [39]. In addition to the Lambda Field, we store a map of the probability distribution function of the mass distribution for each cell, which is provided by one of the above-cited methods. Furthermore, as collisions with low-mass obstacles do not pose a threat to the robot, the risk metric is defined as the force of collision with obstacles that will stop the robot, therefore discarding threat-less collisions.

Figure 4 shows examples of probability distribution functions for several obstacle classes. The main use of a probabilistic formulation for the masses is to deal with the uncertainty of the labels. Indeed, the grass can easily hide a high-density obstacle, such as a rock. Moreover, the mass of the vegetation is very variable, and the robot can expect a harmless collision as much as a harmful collision while going through these kinds of obstacles. In the case where no label is available for a cell, the worst case is taken into account, meaning that the mass of the cell is set to infinity.

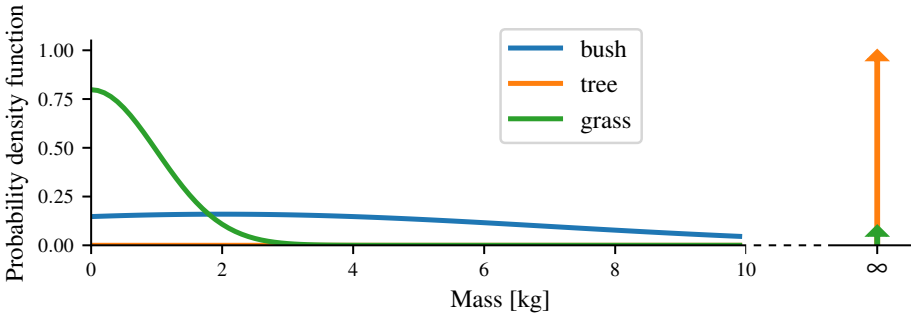


Figure 4. Examples of probability density functions of several labels. The arrow represents the Dirac delta function. The mass of the grass is very likely to be close to zero, but there is a chance that a high-mass obstacle is hiding in it (e.g., a rock). The mass of a bush is very uncertain, as it may be more or less dense. In contrast, the mass of a tree is always very high.

We chose to discretize the probability density function into a sum of Dirac impulsions $\delta(\cdot)$. The mass p.d.f $f_i^m(\cdot)$ of the cell c_i is then

$$f_i^m(m) = \sum_{k=0}^{\infty} \alpha_{ik} \cdot \delta(m - k\Delta m), \tag{22}$$

with Δm being the discretization step and α_{ik} being the probability that $m \in [k\Delta m, (k+1)\Delta m]$. In addition, only a finite number of α_{ik} are not null in order to store the p.d.f.

A problem quickly arises from Equation (19) if we want to take into account the mass of the obstacles. The equation only looks at the first collision, as it assumes that any collision would lead the robot to stop its course. For very light obstacles, such as grass, this assumption falls apart. Hence, we need to add a term to the equation to allow the robot to continue its course after a collision. To do so, we need to understand the meaning of the lambdas. For an area Δa where the Lambda Field is constant with a value λ, the expected number of 'collision' events is $\Delta a \lambda$. Using the probability p_{si}, the probability of the robot being stopped because of the collision at the cell c_i, we want that each collision has the probability p_{si} of being harmful for the robot. Hence, we use our probability of traversal p_{si} as a new measure over the field. Given the harmful probability p_{si} for the cell c_i, the intensity function $\Lambda(\mathcal{C})$ becomes

$$\Lambda_m(\mathcal{C}) = \Delta a \sum_{c_i \in \mathcal{C}} \lambda_i p_{si}. \tag{23}$$

Using this newly defined intensity measure, only hazardous collisions are investigated, treating collisions that do not stop the robot as harmless.

Assuming that the robot can go through obstacles if their mass is below a certain threshold $m_{\mathtt{max}}$, the probability p_{si} is then

$$\begin{aligned} p_{si} &= \mathbb{P}(m_i > m_{\mathtt{max}}) \\ &= 1 - \int_0^{m_{\mathtt{max}}} f_i^m(m)\, \mathrm{d}m, \end{aligned} \tag{24}$$

where $f_i^m(\cdot)$ is the p.d.f of the mass of the cell c_i. In addition, since we do not assume anymore that the obstacles have infinite masses, the risk $r(\cdot)$ (i.e., the loss of momentum at the impact) becomes

$$\begin{aligned} r_m(\Delta ai, m) &= m_R \left(v_R^n(\Delta ai) - \frac{m_R v_R^n(\Delta ai)}{m_R + m} \right) \\ &= m_R \frac{m \cdot v_R^n(\Delta ai)}{m_R + m}, \end{aligned} \tag{25}$$

where m is the mass of the ith cell. Since the mass of each obstacle is probabilistic, we need to sum over all the possible masses to find the expected force of collision over a path, leading to (the proof is detailed in Appendix B):

$$\begin{aligned} \mathbb{E}[r_m(A, M)] &= \sum_{i=0}^{N-1} K_i \int_0^\infty f_i^m(m) r_m(\Delta ai, m) \, \mathrm{d}m \\ &= \sum_{i=0}^{N-1} K_i \sum_{k=0}^{\infty} \alpha_{ik} r_m(\Delta ai, k\Delta m). \end{aligned} \tag{26}$$

where M is the random variable corresponding to the mass of the cell in which the collision happened, and K_i is computed in the same way as in Equation (19), but using $\Lambda_m(\cdot)$. One can note that we can rewrite the above equation in the form

$$\begin{aligned} \mathbb{E}[r_m(A, M)] &= \sum_{i=0}^{N-1} K_i \, r'(\Delta ai) \\ \text{with} \quad r'(\Delta ai) &= \sum_{k=0}^{\infty} \alpha_{ik} r_m(\Delta ai, k\Delta m), \end{aligned} \tag{27}$$

hence going back to the known expectation formula of Equation (19). We omitted the parameters α_{ik} in the parameters of $r'(\cdot)$, as they are directly retrievable from the crossed area Δai. In addition, notice that setting $\mathbb{P}(m_i = \infty) = 1$ for all of the cells leads, as expected, to the same risk as when using Equation (20).

One can note that there is no direct way of taking into account the mass of the obstacles in the Bayesian occupancy grids. Indeed, the Bayesian occupancy grid only stores the information of occupancy for a given cell instead of more abstract information, that is, the intensity of the 'collision' event in the case of the Lambda Field. As such, the Lambda Field possesses an extra layer of assessment where, using the risk function, the framework quantifies the risk associated with the event. This layer allows one to take into account numerous types of information, such as the mass of the obstacles, as done in this section. Furthermore, the Bayesian occupancy grid suffers from its dependence on the tessellation size and is, thus, not suited to inferring generic risks on a path, as stated in the introduction.

In the following, we analyze a method for assessing the probability of collision in Bayesian occupancy grids that does not depend on the tessellation size [3]. We show that, with our improvement to take into account the size of the robot, their method can be seen as a special case of our framework.

3.5. Comparison and Improvement of the Reachability Metric

In this section, we analyze and adapt the concept of *reachability* defined in [3]. These authors' work was indeed the first to address the problem of risk assessment in occupancy grids, which is shown in Figure 1. We first investigate the different metrics proposed in the article and then show that, with our improvement to consider the size of the robot, our framework can be seen as a generalization of their method. They propose the use of the concept of the product integral, which is the product counterpart of the standard integration. A summary of the product integration can be found in [36]. They introduced

the probability of occupancy $p_o(\cdot)$ (defined in their article as $m(\cdot)$) as the density of the cell. At first, they defined the reachability R_t for a path from the time $t = 0$ to T as a product integral, computed as

$$R_t = \prod_0^T (1 - p_o(x(t)))^{\mathrm{d}t}, \tag{28}$$

where $x(t)$ is the robot position at the time t and $p_o(x(t))$ is the probability that the position $x(t)$ is occupied. The higher the reachability, the safer the corresponding path is. However, they argue that it would be better to consider the distance traveled through a cell instead of the time. It is indeed better, as the first metric leads to a counter-intuitive reachability: for a robot crossing at a speed v a straight path of length l, where all cells have the probability p_o of being occupied, the reachability is

$$\begin{aligned} R_t &= \prod_0^{l/v} (1 - p_o)^{\mathrm{d}t} \\ &= \lim_{\Delta t \to 0} \prod_{i=0}^{l/v/\Delta t} (1 - p_o)^{\Delta t}. \end{aligned} \tag{29}$$

Using the fact that $(1 - p_o)^{\Delta t} = \exp\big(\ln(1 - p_o)\Delta t\big)$ and the Riemann definition of the integral, the expression can be simplified to

$$\begin{aligned} R_t &= \exp\left(\int_0^{l/v} \ln(1 - p_o)\, \mathrm{d}t\right) \\ &= (1 - p_o)^{l/v}. \end{aligned} \tag{30}$$

The reachability from the first metric R_t is then higher when the speed is high, meaning that it would be safer to cross the path at a higher speed. Indeed, the trajectory is parametrized on the time of traversal; as such, the faster the robot is going, the smaller the number of position samples to evaluate will be. A robot of infinite speed would thus consider all paths safe, as the integration would not carry any sample points, whereas a very slow robot would lead to consideration of more sample points and, therefore, lower the reachability, as shown in Equation (30).

Their second reachability metric R_L does not possess such a behavior, as they parametrized the integral over the traveled distance $L(t, t + \mathrm{d}t)$ between two instants, leading to

$$\begin{aligned} R_L &= \prod_0^T (1 - p_o(x(t)))^{L(t, t+\mathrm{d}t)} \\ &= \prod_0^T (1 - p_o(x(t)))^{|\dot{x}(t)|\, \mathrm{d}t}. \end{aligned} \tag{31}$$

Since the traveled distance $d(t)$ equals $\int_0^t |\dot{x}(t)|\, \mathrm{d}t$, we have $\mathrm{d}d(t) = |\dot{x}(t)|\, \mathrm{d}t$, and Equation (31) can be simplified to

$$\begin{aligned} R_L &= \prod_0^D (1 - p_o(x_d(d)))^{\mathrm{d}d} \\ &= (1 - p_o)^D \quad \text{in case of homogeneous field,} \end{aligned} \tag{32}$$

where $D = \int_0^T |\dot{x}(t)|\, \mathrm{d}t$ is the total distance crossed by the robot and $x_d(\cdot)$ is the position of the robot as a function of the traveled distance. Using Equation (32), the probability of collision does not depend on the tessellation size or the speed of the vehicle. The main drawback is that there is no natural reason to use the concept of product integrals in Bayesian occupancy grids, as it is here merely a tool to make the probability constant.

Furthermore, the robot is considered to be reduced to a point. The well-known solution to this problem is to inflate the obstacles, at the cost of assuming that the robot is round. As the Lambda Field takes into account the size of the robot, we propose an improvement of their theory to take into account the robot's width W. Instead of only integrating over the robot line path, we also integrate over the entire width of the robot (i.e., the size of the front of the robot) for each position $x_d(d)$. Under this consideration, the reachability equation becomes

$$R_L = \prod_{0}^{D} \prod_{-W/2}^{W/2} \left(1 - p_o(x(d,w))\right)^{\mathrm{d}w\,\mathrm{d}d}, \tag{33}$$

where $x(d,w)$ is a point of the robot parametrized as the distance that the robot has traveled d and the distance from the center of the robot head in its width direction w. We can see that for the special case $W = 1$ and $p_o(x(d,w))$ constant for $w \in [-W/2, W/2]$, we fall back on Equation (32). Assuming that the robot fully crosses the cells it encounters, we can develop a more convenient formulation for calculations. If the robot fully crosses the N cells $\mathcal{C} = \{c_i\}_{0:N-1}$ of size $\mathrm{S} \times \mathrm{S}\,\mathrm{m}^2$ and probability of occupancy p_{oi}, Equation (33) can be rearranged to give

$$\begin{aligned} R_L &= \prod_{i=0}^{N-1} \prod_{x=0}^{S} \prod_{y=0}^{S} (1 - p_{oi})^{\mathrm{d}x\,\mathrm{d}y} \\ &= \prod_{i=0}^{N-1} (1 - p_{oi})^{\Delta a}, \end{aligned} \tag{34}$$

where $\Delta a = S^2$ is the area of each cell.

From there, the improvement in Equation (34) of the theory of Heiden et al. [3] can be linked to the theory of the Lambda Field. Indeed, for a path crossing the cells $\mathcal{C} = \{c_i\}_{0:N-1}$ in a Lambda Field, the probability of not colliding during the traversal is computed as

$$\begin{aligned} 1 - \mathbb{P}(\texttt{coll}) &= \exp\left(-\Delta a \sum_{c_i \in \mathcal{C}} \lambda_i\right) \\ &= \prod_{i=0}^{N-1} \exp(-\lambda_i)^{\Delta a} \\ &= \prod_{i=0}^{N-1} (1 - p_{oi})^{\Delta a} \text{ with } p_{oi} = 1 - \exp(-\lambda_i). \end{aligned} \tag{35}$$

Hence, the probability of occupancy p_{oi} of a cell in [3] is the probability of colliding in the cell c_i of area $1\,\mathrm{m}^2$ in a Lambda Field. Therefore, with our improvement given by Equation (34), the theory of Heiden et al. [3] is then a special case of our framework, where the risk function $r(\cdot)$ is set to 1 and the area of the cells is assumed to be equal to 1. Compared to [3], we propose a more meaningful approach, where the theory provides a way to assess risk that is not restricted to be the probability of collision.

4. Validation

4.1. Setup

We implemented our framework in a robot equipped with an LMS151 lidar and a camera, as shown in Figure 5. Since the robot has four-wheel steering, it was not impacted much by slipping and skidding, and the odometry was sufficient to estimate the robot displacements.

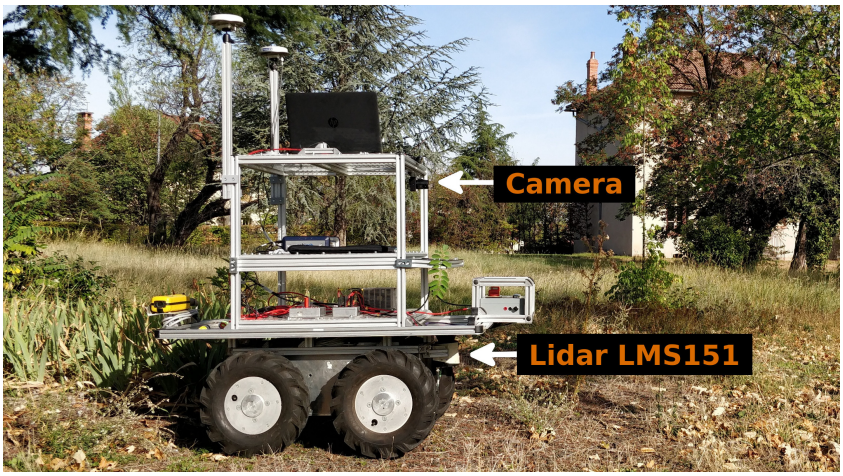


Figure 5. Robot used in the experimentations. It is equipped with a Sick LMS151 lidar and a camera.

For every new lidar scan, the displacement between the current and previous position was estimated, and the map was updated. The Lambda Field was estimated at each iteration using Equation (9). Moreover, the map was centered on the robot. We chose not to rotate the map, but to rotate the robot instead to nullify the errors coming from the rotation. Indeed, a straight wall was quickly distorted after a few rotations because of the tessellation of the map. We also had to keep a global offset of the map, as otherwise, small displacements were not taken into account. Without this offset, the map was not precisely updated, as any displacement below half the cell size was discarded. The mass of the robot was also set to $m_R = 50\,\text{kg}$. For safety purposes, the maximum speed of the robot was set to $0.5\,\text{m}\,\text{s}^{-1}$, and the maximum acceleration to $0.05\,\text{m}\,\text{s}^{-2}$. The parameters used for the confidence intervals of the mapping were $p^m = 0.9999$ and $p^h = 0.99$ for all cell measurements, where the cells had a size of $0.1 \times 0.1\,\text{m}$. The value of p^m was intentionally very high, as it was nearly impossible for a lidar beam to go through obstacles. The normals of the obstacles were estimated using the method developed in [40]. At each lidar scan, the normals of the points were estimated by using their nearest neighbors, and the normal of each underlying cell c_i was updated as follows:

$$\bar{\theta} = \begin{cases} \arctan(\bar{S}/\bar{C}) & \text{if } \bar{C} \geq 0 \\ \arctan(\bar{S}/\bar{C}) + \pi & \text{otherwise,} \end{cases} \tag{36}$$

with

$$\bar{C} = \sum_{k=0}^{N_i-1} \cos(\theta_k) \quad \text{and} \quad \bar{S} = \sum_{k=0}^{N_i-1} \sin(\theta_k), \tag{37}$$

where N_i is the number of normal measurements θ_k for the cell c_i.

4.2. Comparison with the Bayesian Occupancy Grid

In order to demonstrate the discrepancies between the Bayesian occupancy grid and the Lambda Fields, we theoretically investigated the key differences between the two frameworks, and then, in real-world experiments, showed their consequences on the quality of the maps.

First, we investigated the convergence of the occupancy of a single cell. In the context of unstructured environments, it is very common to have cells that are only partially occupied. This can come from either very thin objects, such as tall grass or crops, or from obstacles that do not reflect the laser beam well, such as a dense bush. In both cases, the cell will be measured as both 'hit' and 'miss', as the beam can cross or hit the obstacles in the cell. Using the theory presented in [41] to construct the Bayesian occupancy grid, we used the log odds representation of occupancy. Assuming that a cell is measured N times and is

filled at a ratio of $r \in [0,1]$ (1 is completely filled, 0 is completely empty), the cell will be measured 'hit' rN times and 'miss' $(1-r)N$ times. The Bayesian occupancy grid estimates the occupancy probability of the cell as

$$\mathbb{P}(\texttt{occ}_b) = 1 - \frac{1}{1+\exp\Big(rNl_o + (1-r)Nl_f\Big)}, \tag{38}$$

where l_o is the log odds representation of the probability that the cell is occupied given a 'hit' measurement, whereas l_f is the log odds representation that the cell is occupied given a 'miss' measurement (i.e., it informs that the cell is free). These quantities are computed as

$$\begin{aligned} l_o &= \ln\left(\frac{\mathbb{P}\big(\texttt{occ}|z=\texttt{hit}\big)}{1-\mathbb{P}\big(\texttt{occ}|z=\texttt{hit}\big)}\right), \text{ and} \\ l_f &= \ln\left(\frac{\mathbb{P}\big(\texttt{occ}|z=\texttt{miss}\big)}{1-\mathbb{P}\big(\texttt{occ}|z=\texttt{miss}\big)}\right), \end{aligned} \tag{39}$$

with z being the measurement of the sensor that is either 'hit' or 'miss'. Substituting the definition of the log odds representation with its expression, we have

$$\begin{aligned} \mathbb{P}(\texttt{occ}_b) &= 1 - \frac{1}{1+\left[\left(\frac{\mathbb{P}\big(\texttt{occ}|z=\texttt{hit}\big)}{1-\mathbb{P}\big(\texttt{occ}|z=\texttt{hit}\big)}\right)^r\left(\frac{\mathbb{P}\big(\texttt{occ}|z=\texttt{miss}\big)}{1-\mathbb{P}\big(\texttt{occ}|z=\texttt{miss}\big)}\right)^{1-r}\right]^N} \\ &\triangleq 1 - \frac{1}{1+\Big[O_o^r \cdot O_f^{1-r}\Big]^N}, \end{aligned} \tag{40}$$

where $O_o, O_f \in \mathbb{R}_{\geq 0}$ are defined as the odds of, respectively, $\mathbb{P}\big(\texttt{occ}|z=\texttt{hit}\big)$ and $\mathbb{P}\big(\texttt{occ}|z=\texttt{miss}\big)$. Taking the limit $N \to \infty$, we have

$$\lim_{N\to\infty} \mathbb{P}(\texttt{occ}_b) = \begin{cases} 1 & \text{if } O_o^r \cdot O_f^{1-r} < 1 \\ 0.5 & \text{if } O_o^r \cdot O_f^{1-r} = 1 \\ 0 & \text{if } O_o^r \cdot O_f^{1-r} > 1. \end{cases} \tag{41}$$

Therefore, we see that the Bayesian occupancy grid will always converge to an extremum (apart from the special case where $O_o^r \cdot O_f^{1-r} = 1$, meaning that the measurements do not provide information about the occupancy). On the contrary, the Lambda Field does not converge to an extremum. Indeed, putting the estimation of lambda given by Equation (9) into the probability of collision of Equation (3), we have

$$\begin{aligned} \mathbb{P}(\texttt{occ}_\lambda) &= 1 - \exp\left(-\Delta a \cdot \frac{1}{e}\ln\left(1+\frac{Nr}{N(1-r)}\right)\right) \\ &= 1 - \left(1+\frac{r}{1-r}\right)^{-\frac{\Delta a}{e}}. \end{aligned} \tag{42}$$

In the case where the lidar error region e is equal to the area of the cells Δa, meaning that we are sure that the collision comes from this cell, the equation simplifies to

$$\begin{aligned} \mathbb{P}(\texttt{occ}_\lambda) &= 1 - \frac{1}{1+\frac{r}{1-r}} \\ &= r, \end{aligned} \tag{43}$$

meaning that the probability of collision is purely the ratio of occupancy of the cell and does not depend on the number of measurements N. Figure 6 shows the convergence of the Bayesian occupancy grids and the Lambda Field. In order to ease the reading, the amount of information for a 'hit' measurement is the same as for a 'miss' measurement of the cell, meaning that $\mathbb{P}(\texttt{occ}|z = \texttt{hit}) = 1 - \mathbb{P}(\texttt{occ}|z = \texttt{miss})$ and thus $l_o = -l_f$.

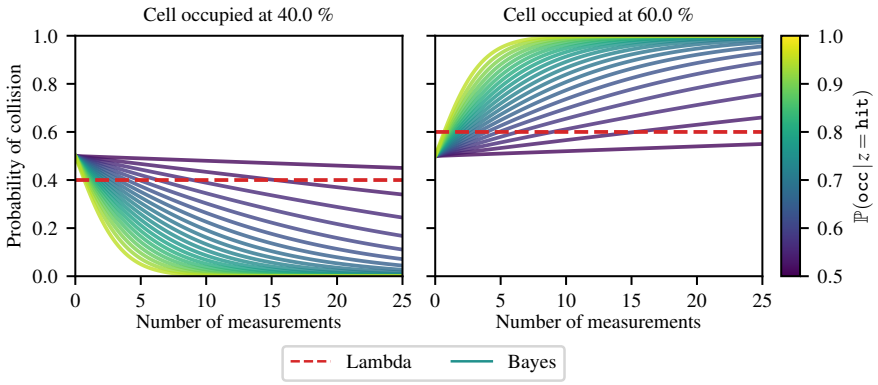


Figure 6. Convergence of the collision probability for a given cell occupied at 40% and 60%. The Bayesian occupancy grid will quickly converge to a probability of occupancy of either 0 or 1, whereas our framework stays at the true occupancy value of the cell.

The behavior still remains the same without this assumption, as shown by Equation (41). The Lambda Field estimation stays at the true occupancy value of the cell, whereas the occupancy probability of the Bayesian occupancy grid converges to either 0 or 1 depending on the occupancy. As expected, the higher the confidence of the sensor's measurement, the faster the probability converges. This behavior can be hazardous in an unstructured environment, as a cell filled at 49% will converge to a probability of occupancy of 0 after a few seconds.

We also consider the case where an obstacle is wrongly undetected. This situation can happen when measuring sparse or unstructured obstacles. For instance, a wire fence can either stop or let through the laser beams depending on the position of the robot. Thus, we look at the speed at which the Bayesian occupancy grid and the Lambda Field can recover the true state of an obstacle wrongly labeled as free. For a single cell measured m times as 'miss', we assume that m is large enough such that the Bayesian occupancy grids converge to the probability of occupancy $\mathbb{P}(\texttt{occ}) = 0$. Then, assuming that after m measurements, the robot starts to measure the obstacle as 'hit', we can approximate the probability of occupancy around the number of 'hit' measurements $h \approx 0$ using first-order Taylor expansion, as

$$\begin{aligned} \mathbb{P}(\texttt{occ}_\texttt{b}) &= 1 - \frac{1}{1 + \exp\left(ml_f + hl_o\right)} \\ &\approx \frac{l_o \exp\left(ml_f\right)}{\left(\exp\left(ml_f\right) + 1\right)^2} \cdot h \\ &\approx \frac{l_o}{2\left(\cosh(ml_f) + 1\right)} \cdot h, \end{aligned} \tag{44}$$

meaning that the recovery rate of the Bayesian occupancy grid vanishes exponentially as a function of the number of times m the cell has been wrongfully measured. In the case of the Lambda Field, we have

$$\begin{aligned}\mathbb{P}(\mathtt{occ}_\lambda) &= 1 - \exp\left(-\frac{\Delta a}{e}\ln\left(1+\frac{h}{m}\right)\right)\\ &= 1 - \frac{1}{(1+h/m)^{\Delta a/e}}\\ &\approx \frac{\Delta a/e}{m}\cdot h,\end{aligned} \tag{45}$$

indicating that the recovery rate vanishes linearly as a function of the number of times m the cell has been wrongfully measured. Figure 7 shows the convergence curves towards the true state of the cell (i.e., 100% filled) for different values of confidence of the sensor measurements. The cell was measured 50 times as 'miss' before (e.g., the robot stayed still during 2 s for a 25 Hz lidar such as the Sick LMS151). Then, the robot changed position, allowing the lidar beams to hit the obstacle in the cell, leading to 'hit' measurements in the cell. As expected, the higher the confidence on the sensor (i.e., smaller error region for the Lambda Field and higher probability for the Bayesian occupancy grid), the faster the recovery speed is. However, the Lambda Field allows better recovery at the beginning by growing faster, whereas the Bayesian occupancy grid prefers to quickly converge to the 'occupied' state after 50 'hit' measurements. The Lambda Field does, however, take more time to converge toward a full occupancy of the cell, as it still takes into account the previous wrong 'miss' measurements. Indeed, as shown in Figure 6, the Bayesian occupancy grid only converges to zero or one. Therefore, as soon as the 'hit' measurements become predominant over the wrong 'miss' measurements, the framework quickly converges to 1. Both frameworks can have their convergence speed shortened by applying a threshold on the probability of occupancy and the lambda (i.e., cannot go above or below certain values). However, this enhancement does not modify the previous analysis, as it only bounds the vanishing of the recovery rate.

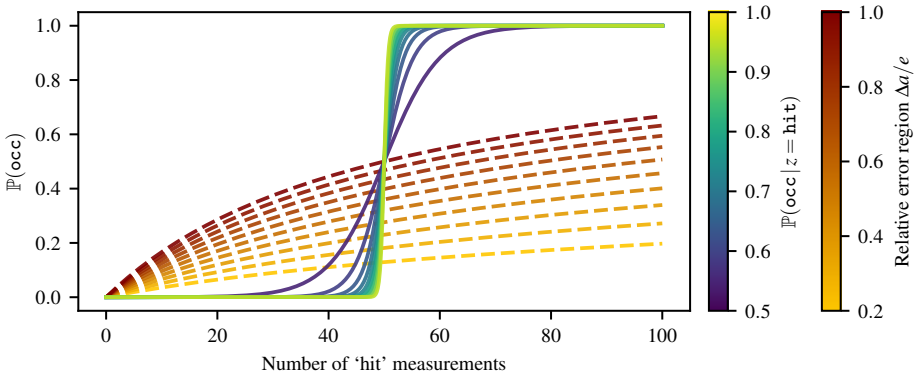
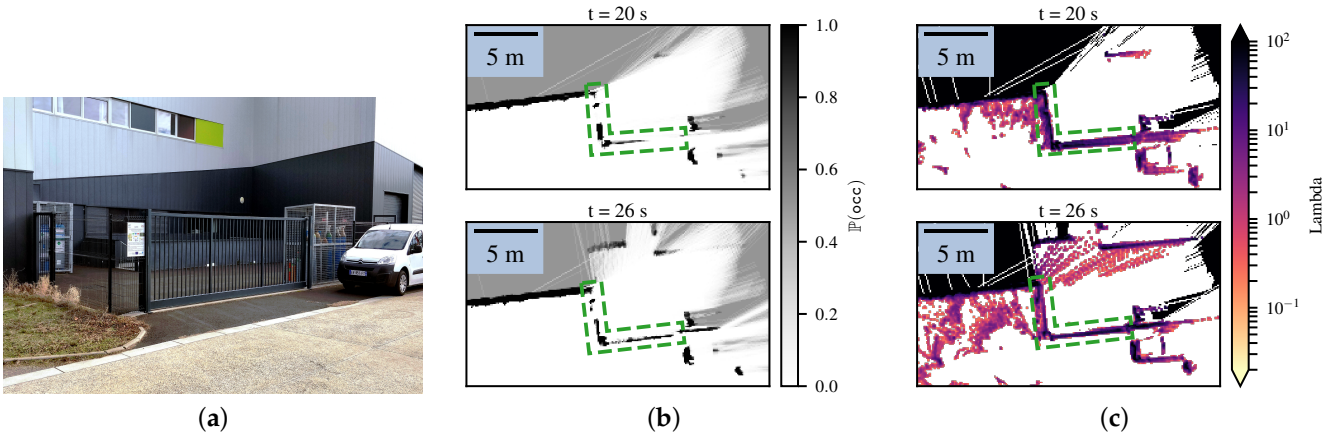


Figure 7. Evolution of the collision probability of a fully occupied cell that has been wrongly measured as empty 50 times before. As theoretically shown, the Lambda Field manages to recover more quickly from the wrong estimation, whereas the Bayesian occupancy grid converges faster to the true state of the cell after 50 'hit' measurements.

In order to demonstrate these considerations in real-world conditions, we mapped a small zone consisting of a black wire fence where lidar beams could easily get through, as shown in Figure 8. At first, the position of the robot led the laser beams to cross the fence without collision, yielding both the Bayesian occupancy grid and the Lambda Field to converge to a false state. After a few seconds, the robot turned in front of the fence, leading more lasers beams to actually collide with the obstacle. In this configuration, the two aforementioned differences between the Lambda Field and Bayesian occupancy grid were involved. On the one hand, the laser beams still had a chance to go through the fence, leading cells that were not completely filled to wrongly converge for the Bayesian occupancy grid. If the lasers collided with the cell less than 50% of the time, the Bayesian

occupancy grid would wrongly converge to 0. On the other hand, because of the wrong 'miss' measurements at the beginning, the Bayesian occupancy grid would struggle more to recover the true state. In order to measure the quality of the map, we manually labeled the fence at each iteration and used patches of 4×4 cells running along the fence. The inspected zone is shown in Figure 8 in dashed green.



(**a**) (**b**) (**c**)

Figure 8. Mapping of a wire fence where, depending on the robot pose, the laser beams can either collide or go through the fence (outlined in dashed green in the maps). (**a**) Picture of the mapped environment. (**b**) Bayesian occupancy grid of the wire fence. (**c**) Lambda Field of the wire fence. One can note that the Bayesian occupancy grid did not have the time to converge at $t = 20\,\mathrm{s}$ because of the wrong 'miss' measurements at the beginning of the experiment.

Using patches instead of directly analyzing the cells removes the noise due to the manual labeling and the noisy odometry of the robot. Using these patches, we evaluated the recall of the detection, computed as the sum of the probabilities of not colliding in each patch p_k over the number P of patches (i.e., the proportion of wrongly detected patches) as

$$\begin{aligned} \mathrm{Recall}_b &= \frac{1}{P} \sum_{k=0}^{P-1} \prod_{c_i \in p_k} \left(1 - \mathbb{P}(c_i = \mathtt{occ})\right), \\ \mathrm{Recall}_\lambda &= \frac{1}{P} \sum_{k=0}^{P-1} \exp\left(-\Delta a \sum_{c_i \in p_k} \lambda_i\right). \end{aligned} \tag{46}$$

In addition, the recall can be seen as the mean of the probabilities of not colliding in the patches. Thus, we also computed the associated standard deviation of the patches. Figure 9 depicts the recall for the Lambda Field and Bayesian grid, as well as the distribution at two sigmas (approximately 95%) of the probability of not colliding in the patches.

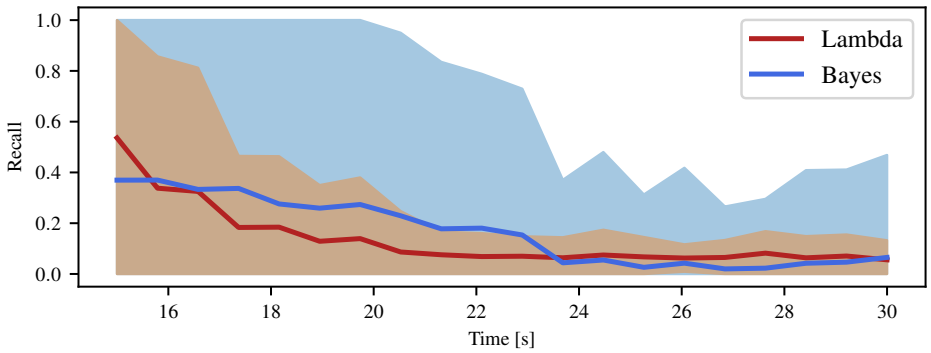
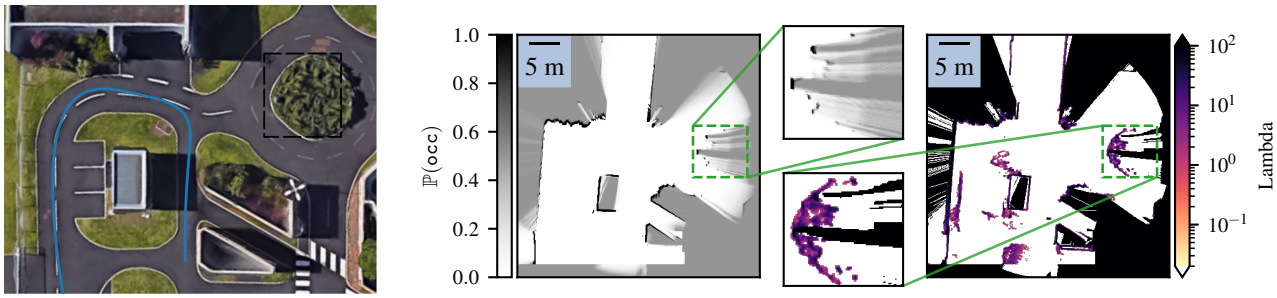


Figure 9. Mapping error of the wire fence for the Bayesian occupancy grid and the Lambda Field. The error is defined as the ratio of free space over the whole space that represents the obstacle. As expected, the Lambda Field converges more quickly to a low error, whereas the Bayesian occupancy grid needs more time to assess its occupancy.

As theoretically expected, the Lambda Field recovered more quickly from the wrong 'free' state of the wire fence. At $t = 20\,\text{s}$, the Lambda Field converged to the state where the whole fence was considered as an obstacle, whereas the Bayesian occupancy grid was still converging and had more than 20% of the fence that was considered as free. At $t \approx 23\,\text{s}$, the robot changed position such that parts of the fence were not visible to the lidar sensor. As such, the Lambda Field started to converge toward a lower lambda that was mainly seen on the left vertical wall. Although the recall was lower for the Bayesian occupancy grid, one can still see holes in the fence at $t = 26\,\text{s}$, while the Lambda Field had the whole fence mapped. Indeed, the distribution at 95% of the probabilities of the patches for the Lambda Field was always contained in the one of the Bayesian occupancy grid. This means that although the recall was lower, the Bayesian occupancy grid had patches that were poorly represented compared to the Lambda Field. One can also note that the car on the right of the map was better represented in the Lambda Field than in the Bayesian occupancy grid. The low-lambda obstacles on the left of the Lambda Field correspond to tall grass and unstructured, sparse vegetation.

Next, we show the effectiveness of our framework in mapping large environments. To do so, we implemented a simple robot follower scenario in an urban-like environment. The robot had to follow a pedestrian while keeping the risk of the chosen path below $5\,\text{kg}\,\text{m}\,\text{s}^{-1}$. While following the pedestrian, the robot created a Lambda Field as well as a Bayesian occupancy grid of the environment, as shown in Figure 10. The maps were globally alike, except for the unstructured obstacles, which were, in this case, the bushes in the roundabout, as well as tall grass around the pavement. As the Bayesian occupancy grid needed to converge to either the 'occupied' or 'free' state, a lot of information about the occupancy of the roundabout was discarded. Furthermore, the robot was not able to see the entire obstacle at first, leading the frameworks to wrongly converge. As shown in the previous section, the Lambda Field recovered faster in these situations, leading to a more precise map. As such, most of the information was preserved in the Lambda Field, and the global shape of the roundabout was more easily recognizable. The other disparities between the two maps also came from unstructured obstacles, which were small trees and tall grass.

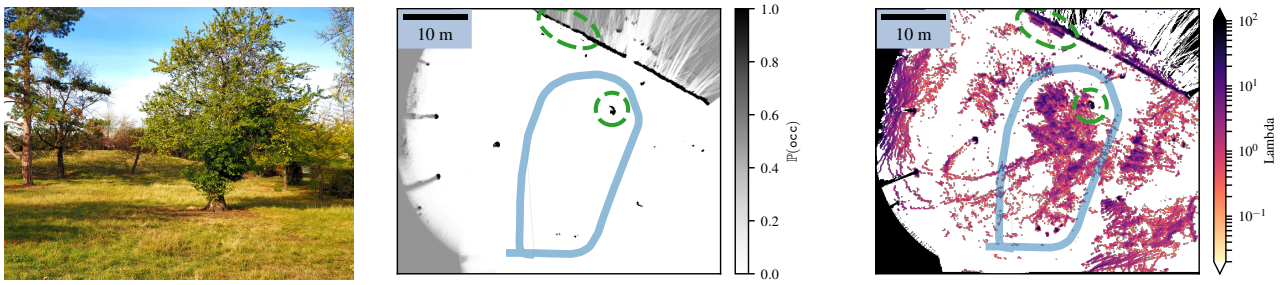


(**a**) (**b**)

Figure 10. (**a**) Aerial view of the mapped environment, with the robot path in blue and the roundabout in dashed black. (**b**) *Left:* Bayesian occupancy grid; *Right:* Lambda Field. The Lambda Field is better suited for storing the occupancy of unstructured obstacles where the Bayesian occupancy grid may over-converge, especially for the roundabout.

Finally, we also mapped an unstructured environment, where the resulting maps are shown in Figure 11 and the environment is depicted in Figure 11a. The environment consisted of several trees with a lot of tall grass disrupting lidar measurements. The robot went around the tree in the center of the picture while navigating in the grass. However, due to the grass and the wind, the lidar returned many measurements corresponding to the grass. Whereas the Bayesian occupancy grid only kept the hedge and the main trees, the Lambda Field kept more information, such as the wooden benches on the top of the map or the tall grass. This behavior was caused by the necessity of the Bayesian occupancy grid to always converge to an extremum, leading it to discard a lot of information that

could be critical. For instance, in the case of agricultural robotics, keeping the crops on the map is essential for not rolling over them.



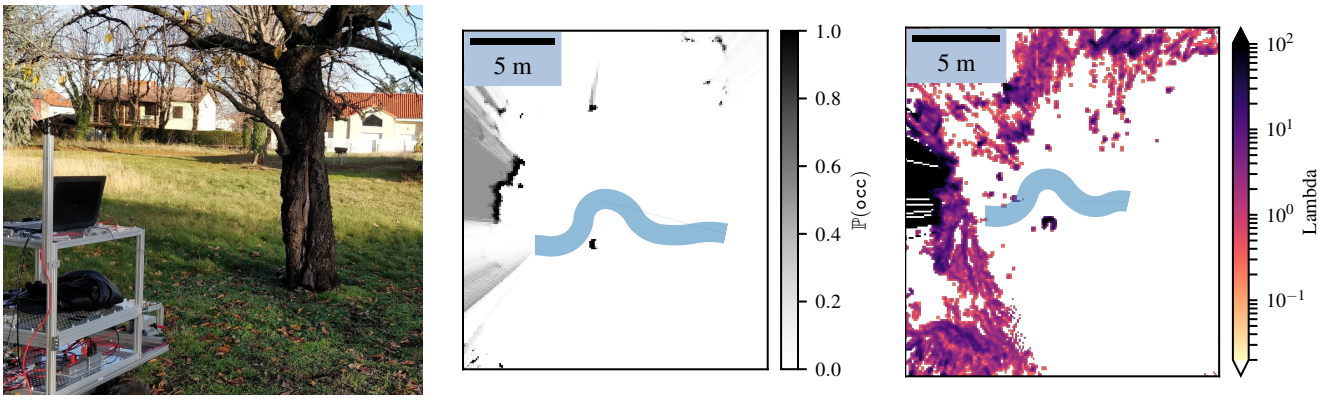
(**a**) (**b**) (**c**)

Figure 11. Mapping of an unstructured zone. (**a**) Picture of the environment. (**b**) Bayesian occupancy grid of the environment. (**c**) Lambda Field of the environment. The robot, with its path in light blue, went around the nearest tree (circled in green) before going back to its initial position. Because of the tall grass, the Lambda Field stored a lot of obstacles during the traversal, while the Bayesian occupancy grid discarded the vast majority of them. The hedge and the tree trunk (circled in green) are still visible in both maps, as they are the only structured obstacles, whereas more unstructured obstacles, such as benches on the top of the map (circled in green) or bushes, were discarded by the Bayesian occupancy grid.

4.3. Basic Path Planning

Here, we demonstrate that our framework can be used to perform classical path planning. As shown in Figure 12a, the robot had to go around a tree to reach the goal that was set behind. To do so, we implemented the path-planning algorithm of [34]. Every 3 s, we sampled feasible commands for the robot, that is, a velocity and steering angle, and chose the best one. The best path (i.e., command applied for 3 s) was the one that stayed below a risk threshold and led the robot as close as possible to the target, which was, in this case, behind the tree. The chosen path also required an upper risk (i.e., the risk computed using the upper bound of the lambdas) below a certain risk threshold. For each feasible command, the N cells crossing the path induced by the command applied for 8 s were extracted and the risks were computed. Estimating the risk of a longer time than the one applied by the command avoided the robot choosing paths that led to a dead end. Indeed, if the risk was computed only for the time of the command, the applied command might have led the robot to be right in front of a wall in a configuration where it was impossible to escape. In the case where no command met the criteria, the robot stopped. This could happen when the robot was at high speed; because of the limited deceleration, all of the high-speed commands led to a risk higher than the maximum that was allowed. Then, the robot completely stopped before continuing its course, as it could now sample low-speed commands.

We also implemented the reachability metric of [3] with our improvement for handling the robot size. As converting the Lambda Field into an occupancy grid using Equation (35) would lead to computing the risk $r(\cdot) = 1$ and using our theory, we used the Bayesian occupancy grid directly computed from lidar measurements. Using the same method for path planning, we sampled paths and chose the one that led the closest to the goal, where a path was considered safe if its reachability R_L was above a certain threshold $1 - \epsilon$, with $\epsilon \in (0, 1)$. The threshold ϵ was set to 0.1 during our experiments. For each command applied, 300 samples were evaluated.



(**a**) (**b**) (**c**)

Figure 12. The robot had to avoid a tree that was on its path. (**a**) Picture of the environment. (**b**) Bayesian occupancy grid with the path that the robot took in light blue. (**c**) Lambda Field with the path the robot took in light blue.

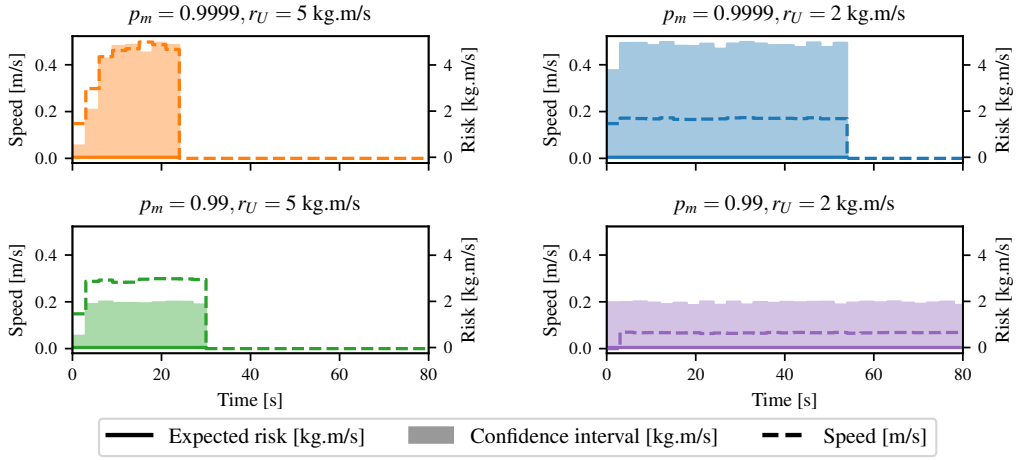


Figure 13. Speed (dashed line) and risk (solid line with a shaded area for its confidence interval) of the chosen paths of the robot going around a tree for different configurations. The robot was able to navigate at a higher speed when it was confident about the measurements and had a higher upper risk limit.

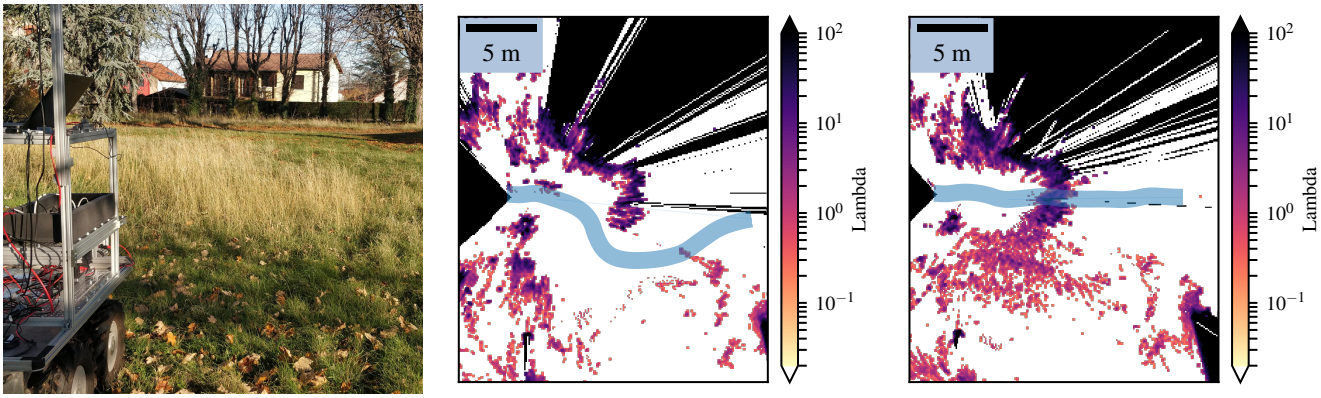
Using both algorithms in the same environment allowed the comparison of the behaviors of the robot in a simple case. Figure 12 shows the results of the path planning for the two environment representations. For the Lambda Field, the maximum allowed expected risk was set to $0\,\mathrm{kg\,m\,s^{-1}}$, meaning that the robot must remain clear of any collisions. We see that the paths were much alike, and the robot effectively avoided the obstacles in both cases. It can be seen for the Lambda Field that the robot path crossed some cells where the lambda was not null, which would lead to a collision. However, as the Lambda Field was computed in real time, the lidar measured collisions in this cell after the robot crossed it. The lidar beams could indeed go through the grass or returned a collision depending on the position of the robot.

The same experiment was conducted using different parameters for the Lambda Field. Figure 13 shows the resulting speed of the robot for different configurations of parameters. While the robot had to expect no risk on its path, it was first allowed to have an upper risk at $5\,\mathrm{kg\,m\,s^{-1}}$, meaning that we were sure at 95% that any unexpected collision had an expected risk below $5\,\mathrm{kg\,m\,s^{-1}}$. The robot quickly reached its maximal speed with full acceleration while keeping the upper risk below the threshold. Under the same configuration, the robot had to reach the goal while keeping the upper risk below $2\,\mathrm{kg\,m\,s^{-1}}$. As the upper risk was smaller, the robot had to reduce its speed. The robot had the same type of reaction if its confidence in the lidar sensor decreased. In the third

experiment, the probability of a correct 'miss' measurement p^m was decreased from 0.9999 to 0.99. The direct implication was that the confidence interval broadened, forcing the robot again to decrease its speed. Then, the robot had a poor confidence in the lidar, as well as a small upper risk, leading to a very slow traversal speed. Thereupon, the Lambda Field allowed classical path planning in the same fashion as in [3]. Our framework also regulated the speed of the robot to cope with the allowed risk level.

4.4. Going Through Tall Grass

After performing simple path planning, we show that the Lambda Fields allow the robot to navigate in unstructured environments. As shown in Figure 14a, the robot had to reach a goal that was behind tall grass. This kind of environment leads to very noisy maps, which can hinder the robot's displacements if looking at the probability of collision.



(**a**) (**b**) (**c**)

Figure 14. The robot had to reach a goal behind the tall grass. (**a**) Picture of the environment. (**b**) Lambda Field with the path the robot took in blue, where the robot was instructed to take absolutely no risks. (**c**) Lambda Field with the path the robot took in blue, where the robot was allowed to take some risks.

We here show that according to the risk the robot is willing to take, it chooses to either go through the tall grass or tries to find a path around it. This kind of behavior is impossible to have when only looking at the probability of collision. Indeed, the robot is sure to collide with the grass. The probability of collision is high, but the collision caused by the grass is harmless, leading to a very small risk. Using a camera, the robot knows that the obstacles in front of it are tall grass. For any other zone, the mass is set to the worst case (i.e., $\mathbb{P}(m_i = \infty) = 1$), as any other prior may lead to underestimating the risk. We assumed that tall grass has a 95% chance of having a null mass and a 5% chance of having an infinite mass. This probability models the possibility that tall grass can hide very dense obstacles, such as rocks or tree trunks.

Two cases were analyzed: In the first one, the robot had to take no risks, meaning that for every path the robot took, the expectation of the risk had to be zero. Hence, the robot chose to go around the tall grass. In the second case, the robot was allowed to take some risks to reach its goal and went through the tall grass. Figure 14 shows the resulting Lambda Fields for these two different robot configurations. In the first case, since the tall grass had a non-zero probability of having a mass that would lead to a harmful collision, the robot chose to go around the tall grass to reach the goal. Once again, the cells with a lambda higher than zero on the path of the robot were updated after the robot went through them. At the time that the robot crossed these cells, the lambdas were null. Figure 15 shows the speed as well as the risk taken by the robot.

The robot first crossed a zone where the mass was supposed to be low, leading to a very narrow confidence interval. The confidence interval grew quickly as the robot went out of the low-mass zone. The robot also stopped several times during the traversal. Indeed, a lot of grass hindered its movements, as the detection of the grass was very irregular.

Because of the maximum deceleration of the robot, no paths were below the maximum risk allowed. The robot then had no choice but to completely stop to be able to plan with low-speed commands.

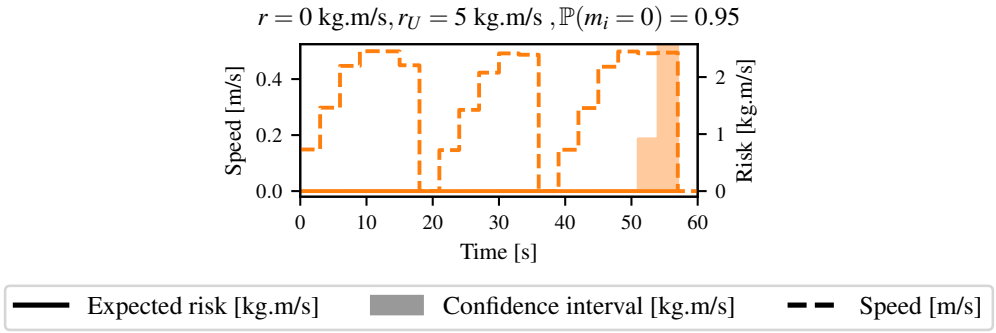


Figure 15. Speed (dashed line) and risk (solid line with a shaded area for its confidence interval) of the chosen paths of the robot when it chose to go around the tall grass. The numerous stops of the robot were due to the very random detection of the tall grass.

In the second case, the robot was allowed to take some risks and chose to go through the tall grass to reach the goal. The differences in the lambdas between the two maps came from the fact that depending on the robot position, the lidar beams could go through the grass or return a collision. Furthermore, there was a lot of wind during the experiments, leading to an accentuation of the noise of the overall map. For this case, different configurations of risk were analyzed. Figure 16 shows the speed as well as the risk taken by the robot. In the first configuration, the robot entered the grass at $t \approx 12$ s. It was allowed to have an expected risk of 0.1 kg m s^{-1} and an upper risk of 5 kg m s^{-1}. The grass had a 5% chance of having an infinite mass. The robot stopped at $t \approx 18$ s as it was about to enter a denser zone, meaning a zone with a higher collision probability. All the high-speed commands led to too high of a risk, and the robot had to completely stop. During the traversal of the tall grass, the speed of the robot was maintained at a low value, as the grass might have been hiding an obstacle. As the robot went out of the grass at $t \approx 36$ s, it increased its speed to its maximum, since a collision was more unlikely to happen. The same experiment was conducted, but this time, the grass zone had a probability of 99% of having a null mass. The robot stopped in the same place as the first time, but increased its speed faster as it was more sure that the collisions were harmless. By doing so, it reached the goal more quickly. The third time, the robot was sure that there were no obstacles in the grass. Hence, it crossed the environment at full speed, since any collisions were harmless, even though the maximum risk allowed was null. A very specific event could appear: Sometimes, the expected risk was outside the confidence interval. This can be seen for $t = 25$ s in the first graph in Figure 16. Computing the risk with lower lambdas could indeed lead to a higher risk in very specific conditions. In our case, the robot computed the risk for a path going out of the area, where the mass of the obstacles was classified as low (i.e., the tall grass) by the camera. By doing so, any collision happening outside this zone would have a higher expected force of collision. It was then considered less risky to collide inside the area of low mass; lowering the lambdas led to a higher chance of colliding outside, as the robot had a lower chance of being stopped inside the zone, leading to a higher risk.

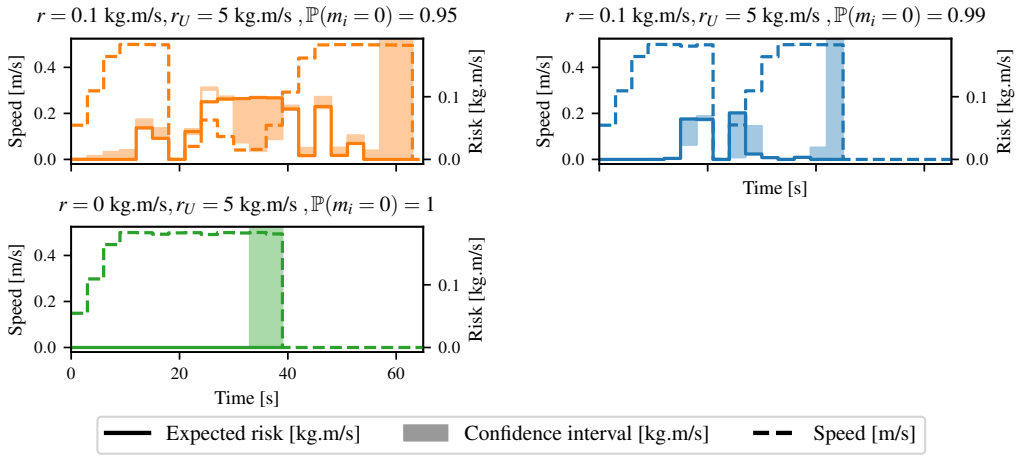


Figure 16. Speed (dashed line) and risk (solid line with a shaded area for its confidence interval) of the chosen paths of the robot when it chose to go through the tall grass for different configurations. The more certain the robot was that there were no obstacles in the grass, the faster it reached its goal.

5. Discussion

The framework was implemented on a standard-grade CPU and ran at more than 10 Hz. Evidently, the smaller the tessellation, the more slowly the framework will operate. If much larger maps or a finer path planner are needed, the whole framework can easily run in parallel on a GPU. Each cell can be updated independently, whereas all of the potential paths can be assessed at the same time. However, we found that for standard applications, a cell size below 10×10 cm is not necessary and does not yield better results, which is mainly due to the sensor's noise, and evaluating 300 commands at each iteration is enough for a smooth navigation.

As mentioned before, the theory of the Lambda Field can be seen as a generalization of the framework of [3]. Under this consideration, it is possible to convert a Lambda Field into a Bayesian occupancy grid and vice versa using Equation (35). In addition to adding meaning to the equations of [3], our framework allows the computation of expectation of a risk. This is possible because the Lambda Field possesses a probability density function. Furthermore, the theory of the Poisson point process was already used by [26] for known obstacles. The Lambda Field can be seen as the transposition of their work for occupancy grids.

One of the major drawbacks of the Lambda Field is the assumption in Equation (7), which is that every cell in the error region of the range sensor carries the same information. Using such an approximation indeed leads to inflation of the obstacles, meaning that some narrow corridors through which the robot could go become impracticable. The modeling of the sensor can also be discussed: For practical reasons, the sensor is assumed to have a deterministic error region where the collision is sure to have happened. Selecting too small of an error region would lead to augmenting the probability of wrong measurements, thus increasing the confidence interval of the lambdas and, hence, decreasing the speed of the robot. Inversely, taking too big of an error region leads the map to inflate the obstacles, hence decreasing the space in which the robot can evolve. However, in the case of lidar sensors, their precision is such that their error region often reduces to a low number or even a single cell, meaning that almost no inflation occurs. This can be seen in Figure 8 or Figure 10, for instance, where the structured obstacles do not appear bigger on the Lambda Field than in the Bayesian occupancy grid. In the case where the range sensor has a bigger error zone, Appendix C gives a way to reduce the inflation by using a standard probabilistic sensor model.

The computation of the confidence intervals can also yield deeper discussions. Indeed, an empty cell close to an obstacle can be considered to have greater chances of failing the reading and returning a 'hit' measurement. Although this problem is already managed by the error zone, taking into account that cells close to the error zone have a greater chance to

be misread can lead to more precise maps. This would lead to a lower upper bound of the lambdas in a large empty zone, thereby increasing the traversal speed (i.e., efficiency) of the robot. As this article only dealt with constant false positive and negative ratios, future works will investigate a deeper use of confidence intervals in harsher environments, such as snowstorms, where many faulty 'hit' measurements coming from the snow can hinder the robot displacements.

So far, only the lidar measurements are used to estimate the lambdas, and a camera is used for the mass estimation. However, in a more cooperative approach, the robot can also assess the hazardousness of the environment while navigating in it. For instance, if the robot goes safely through a zone where the probability of causing a harmful collision p_{si} is not null, the map can be updated accordingly and the probability of harmful collisions can drop. Note that adding such information needs to be done carefully, as informing that a zone is safe for a robot does not mean that another robot with a different mass and mechanical configuration will also safely cross. Furthermore, as the framework creates maps centered on the robot, there is currently no tool for matching temporally distant observations (such as loop closures). In a case where the construction of large-scale maps is necessary, an external Simultaneous Localization and Mapping (SLAM) algorithm, such as the Iterative Closest Point (ICP), can provide a corrected localization.

In addition, some issues can appear while estimating the risk with our framework for global path-planning algorithms. Indeed, it may be harder to understand the metrics. The longer the path, the higher the risk will be. As this behavior seems intuitive, it leads to several questions. For two paths with the same risk but a different length, are we willing to take the two paths with the same confidence? Should we weigh the risk by the length of the path, meaning that we are willing to take more risk for longer paths? We chose to understand the risk as the risk of a given command. Indeed, we believe that as humans, we assess the risk of every step we make without thinking about the length of the path.

Furthermore, the expectation of the risk is not suited to modeling the 'long-tail' of the Gaussian, i.e., the low-probability events that can happen. The faulty measurements of the lidar are handled with the confidence intervals of the lambdas, but other metrics may better estimate the 'long-tail'. The Conditional Value at Risk presented by [19] explicitly measures the risk of the 'long-tail'. It can then be a better indicator of the risk when a low-probability, high-risk situation arises; for example, when a high-mass obstacle hides in the grass.

6. Conclusions

In this article, we presented a novel representation of the occupancy information of the environment, which is called the Lambda Field. We first derived a way to construct the map, as well as confidence intervals over these values. This representation allows the computation of expectations over a path, giving a natural way to assess different types of risks. The Lambda Field is very similar to the Bayesian occupancy grid for mapping, with the only notable mapping difference being that the Lambda Field better stores unstructured obstacles, such as bushes, tall grass, or wire fences. In addition, the Lambda Field provides the computation of a generic risk that depends on the application.

In the case of unmanned ground vehicles, we chose to represent the risk as the force of collision. In contrast to risk metrics defined on Bayesian occupancy grids, our risk possesses a physical meaning. We were able to control the level of risk that the robot could take over its planning, allowing behaviors that are impossible with classical path-planning representations of the environments. The robot was indeed allowed to cross low-mass occupied areas, such as tall grass, as long as the risk level was low enough. Therefore, the Lambda Field provides a framework that regulates the path as well as the speed of the robot, ensuring the robot's safety.

Future works will investigate the use of other metrics than the expectation of the risk while testing the framework in more adverse environments, such as snowstorms or deep woods. Proprioceptive information of the robot will be included to fill the map more accurately. Finally, the framework will be improved to take into account dynamic obstacles

and assess risks in urban-like environments. Using the risk function instead of the standard probability of collision will lead to more informed decisions in case of danger.

Author Contributions: Conceptualization, J.L.; methodology, J.L.; software, J.L. and L.M.; validation, J.L., A.K., and L.M.; formal analysis, J.L.; writing—original draft preparation, J.L. and A.K.; writing—review and editing, R.A., F.P., R.C., and C.D.; visualization, J.L. and F.P.; supervision, R.A., F.P., R.C., and C.D.; funding acquisition, R.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work was sponsored by a public grant overseen by the French National Research Agency as part of the "Investissements d'Avenir" through the IMobS3 Laboratory of Excellence (ANR-10-LABX-0016), the IDEX-ISITE initiative CAP 20-25 (ANR-16-IDEX-0001), and the RobotEx Equipment of Excellence (ANR-10-EQPX-0044).

Acknowledgments: We thank Elie Randriamiarintsoa for his help with the implementation of the image segmentation framework. We also thank Simon Vilmin for his advice and the insightful discussions we had about the comparison with related works.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Heterogeneous Error Regions

In the case that the error regions $\mathcal{E}_k$ have a different size for each lidar beam b_k, we need to further approximate the derivative of the log-likelihood. Under the same assumption that the h_i error regions $\mathcal{E}_k$ containing the cell c_i are small, we have

$$\begin{aligned} \frac{\partial \mathcal{L}(X|\lambda)}{\partial \lambda_i} &= -m_c \cdot \Delta a + \sum_{k=0}^{h_i-1} \frac{\Delta a}{\exp(e_k \lambda_i) - 1} \\ &\approx -m_c \cdot \Delta a + \sum_{k=0}^{h_i-1} \frac{\Delta a}{e_k \lambda_i}, \end{aligned} \tag{A1}$$

leading to

$$\lambda_i = \frac{1}{m_i} \sum_{k=0}^{h_i-1} \frac{1}{e_k}. \tag{A2}$$

This approximation over-estimates the lambdas compared to Equation (9). Indeed, in the special case where all the $\mathcal{E}_k$ have the same area e, the computed lambdas from Equation (A2) are

$$\lambda_i = \frac{1}{e} \frac{h_i}{m_i}. \tag{A3}$$

As $\forall x \in \mathbb{R}_{\geq 0}$, $x \geq \ln(1+x)$, we will always over-estimate the lambdas using Equation (A2). This is the desired behavior, as under-estimating the lambdas would lead to under-estimate the risk.

Appendix B. Proof of Equation (26)

We here prove Equation (26). We have two random variables: the area at which the robot collides A and the mass M of the cell where the collision happened, of marginal probability density functions $f(\cdot)$ and $f^m(\cdot)$. Note that we do not have direct access to $f^m(\cdot)$, but only $f_i^m(\cdot) = f^m(\cdot|\Delta ai)$, the probability density function of the mass given the

crossed area, thus given the cell that the robot is currently crossing. Under the assumption that the risk $r(\cdot)$ is constant inside each cell, the expectation of the function $r(A, M)$ is

$$\begin{aligned}
\mathbb{E}\big[r(A,M)\big] &= \int_0^{N\Delta a}\int_0^{\infty} r_m(a,m) f(a) f^m(m|a)\,\mathrm{d}m\,\mathrm{d}a \\
&= \sum_{i=0}^{N-1}\int_{i\Delta a}^{(i+1)\Delta a} f(a)\int_0^{\infty} r_m(a,m) f_i^m(m)\,\mathrm{d}m\,\mathrm{d}a \\
&= \sum_{i=0}^{N-1}\left[\int_{i\Delta a}^{(i+1)\Delta a} f(a)\,\mathrm{d}a\right]\left[\int_0^{\infty} r_m(\Delta a i, m) f_i^m(m)\,\mathrm{d}m\right] \\
&= \sum_{i=0}^{N-1} K_i \int_0^{\infty} r_m(\Delta a i, m_i) f_i^m(m_i)\,\mathrm{d}m \\
&= \sum_{i=0}^{N-1} K_i \sum_{k=0}^{\infty} \alpha_{ik} r_m(\Delta a i, k\Delta_m),
\end{aligned} \tag{A4}$$

for a path $\mathcal{P}$ going through the cells $\{c_i\}_{0:N-1}$, and

$$K_i = \exp\Big(-\Lambda_m(\{c_j\}_{0:i-1})\Big)\Big[1-\exp\big(-\Lambda_m(\{c_i\})\big)\Big]. \tag{A5}$$

Appendix C. Probabilistic Error Region

If the range sensor has a large error zone, the inflation of the obstacles may become problematic. Therefore, we give here a way to estimate the Lambda Field using a probabilistic error region. Let the error region $\mathcal{E} : \mathbb{R} \to \mathcal{P}(\mathbb{R}^2)$ be an application that takes a parameter and returns a subspace of $\mathbb{R}^2$. One example is the application that gives from the radius r the ball centered on the lidar measurement $\boldsymbol{x_l} \in \mathbb{R}^2$: $\{\boldsymbol{x} \in \mathbb{R}^2, |\boldsymbol{x} - \boldsymbol{x_l}| \leq r\}$. Furthermore, let σ be a random variable of probability density function $f_\sigma(\cdot)$ and a Lambda Field $\lambda : \mathbb{R}^2 \to \mathbb{R}_{\geq 0}$. Under these considerations, the expectation of the intensity of the error zone $\mathcal{E}(\sigma)$ is

$$\begin{aligned}
\mathbb{E}\left[\int_{\mathcal{E}(\sigma)} \lambda(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}\right] &= \int_{\mathbb{R}} f_\sigma(s)\int_{\mathcal{E}(s)} \lambda(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}s \\
&= \int_{\mathbb{R}} f_\sigma(s)\int_{\mathbb{R}^2} \lambda(\boldsymbol{x})\cdot \mathbb{1}_{\mathcal{E}(s)}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}s.
\end{aligned} \tag{A6}$$

Under the assumption that the expectation of the intensity of the error zone is finite, we can switch the integration order using Fubini's theorem and find a more convenient form:

$$\begin{aligned}
\mathbb{E}\left[\int_{\mathcal{E}(\sigma)} \lambda(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}\right] &= \int_{\mathbb{R}^2} \lambda(\boldsymbol{x})\int_{\mathbb{R}} f_\sigma(s)\cdot \mathbb{1}_{\mathcal{E}(s)}(\boldsymbol{x})\,\mathrm{d}s\,\mathrm{d}\boldsymbol{x} \\
&= \int_{\mathbb{R}^2} \lambda(\boldsymbol{x})\,\mathbb{P}(\boldsymbol{x}\in\mathcal{E}(\sigma))\,\mathrm{d}\boldsymbol{x},
\end{aligned} \tag{A7}$$

where $\mathbb{1}_X$ is the identity operator, i.e., $\mathbb{1}_X(\boldsymbol{x}) = 1$ if $\boldsymbol{x} \in X$ and 0 otherwise. Using this expectation as the new intensity function $\Lambda(\mathcal{C})$, tessellating the field and putting it back into Equation (6), the lambdas are now estimated using the same counters h_i and m_i, which now represent, respectively, the sum of the probabilities of being in the error zone and the sum of the probabilities of not being in the error zone of each lidar measurement. One can note that in the case where the error region is known, meaning that $\mathbb{P}(x \in \mathcal{E}(\sigma))$ is either equal to zero or one, we fall back on the previously derived equations as h_i and m_i regain their function of counting the number of times the cell has been in the error region or not.

References

1. Fulgenzi, C.; Spalanzani, A.; Laugier, C. Dynamic obstacle avoidance in uncertain environment combining PVOs and occupancy grid. In Proceedings of the IEEE International Conference on Robotics and Automation, Roma, Italy, 10–14 April 2007; pp. 1610–1616.
2. Elfes, A. Using occupancy grids for mobile robot perception and navigation. *Computer* **1989**, *6*, 46–57. [CrossRef]
3. Heiden, E.; Hausman, K.; Sukhatme, G.S. Planning High-speed Safe Trajectories in Confidence-rich Maps. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Vancouver, BC, Canada, 24–28 September 2017; pp. 2880–2886.
4. Kraetzschmar, G.K.; Gassull, G.P.; Uhl, K. Probabilistic quadtrees for variable-resolution mapping of large environments. In Proceedings of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles, Lisbon, Portugal, 5–7 July 2004.
5. Laconte, J.; Debain, C.; Chapuis, R.; Pomerleau, F.; Aufrère, R. Lambda-Field: A Continuous Counterpart of the Bayesian Occupancy Grid for Risk Assessment. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Macau, China, 3–8 November 2019; pp. 167–172.
6. Coué, C.; Pradalier, C.; Laugier, C.; Fraichard, T.; Bessière, P. Bayesian occupancy filtering for multitarget tracking: An automotive application. *Int. J. Robot. Res.* **2006**, *25*, 19–30. [CrossRef]
7. Saval-Calvo, M.; Medina-Valdés, L.; Castillo-Secilla, J.M.; Cuenca-Asensi, S.; Martínez-Álvarez, A.; Villagrá, J. A review of the bayesian occupancy filter. *Sensors* **2017**, *17*, 344. [CrossRef]
8. O'Callaghan, S.T.; Ramos, F.T. Gaussian process occupancy maps. *Int. J. Robot. Res.* **2012**, *31*, 42–62. [CrossRef]
9. Kim, S.; Kim, J. Continuous occupancy maps using overlapping local Gaussian processes. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 4709–4714.
10. Ramos, F.; Ott, L. Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent. *Int. J. Robot. Res.* **2016**, *35*, 1717–1730. [CrossRef]
11. Senanayake, R.; Ramos, F. Bayesian Hilbert Maps for Continuous Occupancy Mapping in Dynamic Environments. In Proceedings of the Conference on Robot Learning, Mountain View, CA, USA, 13–15 November 2017; pp. 458–471.
12. Guizilini, V.; Senanayake, R.; Ramos, F. Dynamic hilbert maps: Real-time occupancy predictions in changing environments. In Proceedings of the IEEE International Conference on Robotics and Automation, Montreal, QC, Canada, 20–24 May 2019; pp. 4091–4097.
13. Agha-mohammadi, A.A.; Heiden, E.; Hausman, K.; Sukhatme, G. Confidence-rich grid mapping. *Int. J. Robot. Res.* **2019**, *38*, 1352–1374 [CrossRef]
14. Fraichard, T. A Short Paper About Motion Safety. In Proceedings of the IEEE International Conference on Robotics and Automation, Roma, Italy, 10–14 April 2007; pp. 1140–1145.
15. Lee, D.N. A theory of visual control of braking based on information about time-to-collision. *Perception* **1976**, *5*, 437–459. [CrossRef] [PubMed]
16. Laugier, C.; Paromtchik, I.E.; Perrollaz, M.; Yong, M.; Yoder, J.D.; Tay, C.; Mekhnacha, K.; Nègre, A. Probabilistic analysis of dynamic scenes and collision risks assessment to improve driving safety. *IEEE Intell. Transp. Syst. Mag.* **2011**, *3*, 4–19. [CrossRef]
17. Vaillant, M.; Davatzikos, C.; Taylor, R.H.; Bryan, R.N. A Path-Planning Algorithm for Image-Guided Neurosurgery. In Proceedings of the CVRMed-MRCAS'97, Grenoble, France, 19–22 March 1997; pp. 467–476.
18. Caborni, C.; Ko, S.Y.; De Momi, E.; Ferrigno, G.; Y Baena, F.R. Risk-based path planning for a steerable flexible probe for neurosurgical intervention. In Proceedings of the IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics, Rome, Italy, 24–27 June 2012; pp. 866–871.
19. Majumdar, A.; Pavone, M. How Should a Robot Assess Risk? Towards an Axiomatic Theory of Risk in Robotics. In *Robotics Research*; Springer International Publishing: Cham, Switzerland, 2020; pp. 75–84.
20. Tsiotras, P.; Bakolas, E. A hierarchical on-line path planning scheme using wavelets. In Proceedings of the 2007 European Control Conference, ECC 2007, Kos, Greece, 2–5 July 2007; pp. 2806–2812.
21. De Filippis, L.; Guglieri, G.; Quagliotti, F. A minimum risk approach for path planning of UAVs. *J. Intell. Robot. Syst. Theory Appl.* **2011**, *61*, 203–219. [CrossRef]
22. Primatesta, S.; Guglieri, G.; Rizzo, A. A Risk-Aware Path Planning Strategy for UAVs in Urban Environments. *J. Intell. Robot. Syst. Theory Appl.* **2019**, *95*, 629–643. [CrossRef]
23. Joachim, S.; Tobias, G.; Daniel, J.; Riidiger, D. Path planning for cognitive vehicles using risk maps. In Proceedings of the IEEE Intelligent Vehicles Symposium, Eindhoven, The Netherlands, 4–6 June 2008; pp. 1119–1124.
24. Pereira, A.A.; Binney, J.; Jones, B.H.; Ragan, M.; Sukhatme, G.S. Toward risk aware mission planning for autonomous underwater vehicles. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 25–30 September 2011; pp. 3147–3153.
25. Feyzabadi, S.; Carpin, S. Risk-aware path planning using hirerachical constrained Markov Decision Processes. In Proceedings of the IEEE International Conference on Automation Science and Engineering, Taipei, Taiwan, 18–22 August 2014; pp. 297–303. [CrossRef]
26. Eggert, J. Predictive risk estimation for intelligent ADAS functions. In Proceedings of the 2014 17th IEEE International Conference on Intelligent Transportation Systems, ITSC 2014, Qingdao, China, 8–11 October 2014; pp. 711–718.
27. Tsardoulias, E.G.; Iliakopoulou, A.; Kargakos, A.; Petrou, L. A Review of Global Path Planning Methods for Occupancy Grid Maps Regardless of Obstacle Density. *J. Intell. Robot. Syst.* **2016**, *84*, 829–858. [CrossRef]

28. Yang, K.; Keat Gan, S.; Sukkarieh, S. A Gaussian process-based RRT planner for the exploration of an unknown and cluttered environment with a UAV. *Adv. Robot.* **2013**, *27*, 431–443. [CrossRef]
29. Čikeš, M.; Dakulovič, M.; Petrovič, I. The path planning algorithms for a mobile robot based on the occupancy grid map of the environment—A comparative study. In Proceedings of the 2011 23rd International Symposium on Information, Communication and Automation Technologies, ICAT 2011, Sarajevo, Bosnia and Herzegovina, 27–29 October 2011.
30. Fulgenzi, C.; Tay, C.; Spalanzani, A.; Laugier, C. Probabilistic navigation in dynamic environment using Rapidly-exploring Random Trees and Gaussian Processes. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France, 22–26 September 2008; pp. 1056–1062.
31. Fulgenzi, C.; Spalanzani, A.; Laugier, C. Probabilistic motion planning among moving obstacles following typical motion patterns. In Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009, St. Louis, MO, USA, 10–15 October 2009; pp. 4027–4033.
32. Rummelhard, L.; Nègre, A.; Perrollaz, M.; Laugier, C. Probabilistic Grid-based Collision Risk Prediction for Driving Application. In *Experimental Robotics: The 14th International Symposium on Experimental Robotics*; Springer International Publishing: Cham, Switzerland, 2014; pp. 821–834.
33. Dhawale, A.; Yang, X.; Michael, N. Reactive Collision Avoidance Using Real-Time Local Gaussian Mixture Model Maps. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Madrid, Spain, 1–5 October 2018; pp. 3545–3550.
34. Gerkey, B.; Konolige, K. Planning and control in unstructured terrain. In Proceedings of the ICRA Workshop on Path Planning on Costmaps, Pasadena, CA, USA, 19–23 May 2018
35. Francis, G.; Ott, L.; Ramos, F. Functional Path Optimisation for Exploration in Continuous Occupancy Maps. In Proceedings of the ICRA Workshop on Informative Path Planning and Adaptive Sampling, Brisbane, Australia, 21–16 May 2018
36. Slavík, A. *Product Integration, Its History and Applications*; Matfyzpress Prague: Matfyzpress, Prague, 2007; Volume 1.
37. Rohou, S.; Jaulin, L.; Mihaylova, L.; Bars, F.L.; Veres, S.; Rohou, S.; Jaulin, L.; Mihaylova, L.; Bars, F.L.; Reliable, S.V.; et al. Reliable non-linear state estimation involving time uncertainties. *Automatica* **2018**, *93*, 379–388. [CrossRef]
38. Badrinarayanan, V.; Kendall, A.; Cipolla, R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2481–2495. [CrossRef] [PubMed]
39. Lalonde, J.F.; Vandapel, N.; Huber, D.F.; Hebert, M. Natural terrain classification using three-dimensional ladar data for ground robot mobility. *J. Field Robot.* **2006**, *23*, 839–861. [CrossRef]
40. Senanayake, R.; Ramos, F. Directional grid maps: Modeling multimodal angular uncertainty in dynamic environments. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Madrid, Spain, 1–5 October 2018; pp. 3241–3248.
41. Thrun, S.; Burgard, W.; Fox, D. *Probabilistic Robotics*; MIT Press: Cambridge, MA, USA, 2005; p. 647.