A Risk-Aware Motion Planning Framework with Nonlinear Risk-Constrained Optimization

Elie Randriamiarintsoa, Johann Laconte, Charifou Orou Mousse, Benoit Thuilot, Romuald Aufrère

A Risk-Aware Motion Planning Framework with Nonlinear Risk-Constrained Optimization

Elie Randriamiarintsoa[1], Johann Laconte[2], Charifou Orou Mousse[1], Benoit Thuilot[1] and Romuald Aufrère[1]

***Abstract*— In this paper, we present a risk-aware motion planning framework for resilient navigation in occupancy grid maps. Navigating in unknown environments involves complex interactions with the surroundings that must be effectively managed by the robot. Recently, these interactions are frequently expressed as risk constraints, where risk is defined as potential threats that could hinder the robot from accomplishing its objectives. However, when the risk constraint involves nonlinearities, common numerical solvers are severely hampered in finding a feasible solution and are prone to failure. Therefore, we present a novel risk-aware navigation strategy based on motion primitives and the Nonlinear Model Predictive Control (NMPC) method to address nonlinear risk constraints within discrete maps. We demonstrate the effectiveness of our approach through a practical application of a robust risk assessment method that takes into account both the state of the environment and the state of the robot. In addition to enhancing the decision-making capabilities of the robot, our framework offers a more resilient motion planning process that enables the robot to navigate risky scenarios where standard optimizers are likely to fail and lead to dangerous trajectories.**

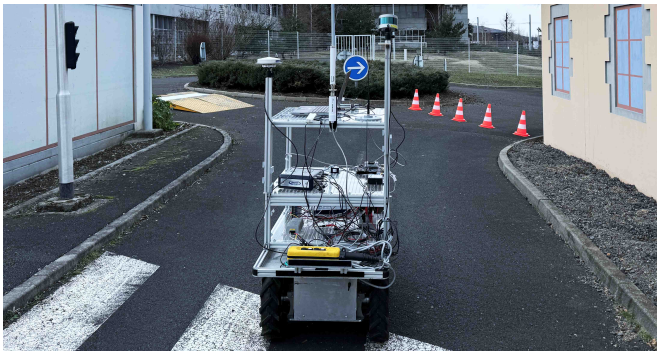

Fig. 1. An illustration of a scenario where a robot must successfully navigate through a speed bump to reach a position behind the roundabout. Using our framework, the vehicle is able to cross this environment while considering the risks involved during navigation.

I. INTRODUCTION

As autonomous robots start to roam the world, it becomes paramount to consider potential risks during navigation, especially in unknown environments. These risks encompass various scenarios, demanding continuous monitoring and adaptive responses. In Figure 1, the robot must consider not only avoid collisions with obstacles but also, in a more general view, it has to assess the severity of damage that the interactions with the environment can cause when it moves towards its goal. With a goal located behind the roundabout, the robot must cross a speed bump to avoid colliding with the traffic cones, but it needs to adjust its speed to ensure safe traversal of the speed bump. Therefore, developing a resilient motion planning framework necessitates the incorporation of a sophisticated risk function that not only assesses the likelihood of collision but also evaluates their potential severity.

To address this challenge, traversability analysis is crucial, and occupancy grid maps are the most popular choice for mapping the environment. Standard techniques using occupancy grid maps divide the environment into traversable and non-traversable regions, but are prone to errors, and also limit the nuance of behaviors in complex environments, as the one shown in Figure 1. As such, novel techniques have been developed to assess more complex risk on grid map, taking into account both the state of the environment and the state of the robot. This leads to a more nuanced analysis, though it comes with increased complexity.

In recent years, Model Predictive Control (MPC) has become more and more popular thanks to the ever-growing computational capabilities of embedded computers. Unlike classical approaches, MPC methods are optimizing the control of the robot within a specified time-window, enabling anticipation of changes in the trajectory or the environment. While the MPC is very efficient for linear problems, challenges arise when dealing with nonlinear problems. Systems with small nonlinearities can often be linearized, but highly nonlinear models require more advanced techniques to solve. In the context of risk-aware navigation, the Sequential Quadratic Programming (SQP) [1] method is commonly used to address nonlinear optimization problems. Furthermore, SQP-type approaches have been widely adopted in robotics due to their ability to operate in real time. However, a common criticism of SQP-based MPC (and nonlinear MPC methods in general) is that they can be prone to local minima or fail to find a feasible solution. In this work, we introduce a new framework based on a motion library and a NMPC approach to handle a highly nonlinear risk constraint on continuous trajectories traversing the environment. Our contributions are i) a resilient NMPC framework for risk-aware navigation; ii) an analysis of this architecture using the Lambda-Field framework [2]; and iii) a demonstration of the effectiveness of our framework for risk management on a robot in a real-world application.

[1]Université Clermont Auvergne, Clermont Auvergne INP, CNRS, Institut Pascal, F-63000 Clermont-Ferrand, France. elie.randriamiarintsoa@uca.fr

[2] Université Clermont Auvergne, INRAE, UR TSCF, 63000, Clermont-Ferrand, France johann.laconte@inrae.fr

II. RELATED WORK

A key issue in navigation tasks is finding a feasible motion plan that takes into account the robot capabilities to traverse its environment while ensuring safety. To address this challenge, it is important to accurately capture the local environment around the robot. A widely used representation is the Bayesian Occupancy Grid (BOG) [3], which consists in tessellating the environment into a grid and storing occupancy information for each cell. BOG is commonly used in motion planning framework to identify collision-free paths by analyzing the probability of encountering obstacles along potential path. However, evaluating the probability of collision alone is not sufficient to exploit the full potential of the robot to traverse its environment. For instance, a mobile robot might navigate over a rough terrain or uneven surface at a reduced speed, even if it results in minor collisions. To make more informed trajectory decisions, several approaches propose incorporating risk into motion planning through the use of risk maps. In contrast to the conventional BOG, a risk map stores the cost of traversing a given position. For instance, Schroder *et al.* [4] proposed a risk map tailored for cognitive vehicles, which prevents robots from getting to close to vehicles or pedestrians. Conversely, Pereira *et al.* [5], employed bathymetry and historical shipping traffic data from coastal areas to develop a risk map specifically designed for underwater vehicles. On this map, each position likely to be occupied is assigned a probabilistic risk value, reflecting the likelihood of hazards in those areas. Primatesta *et al.* [6] also employed a risk map to assess the likelihood of an unmanned aerial vehicle causing fatal incidents in populated areas. Although these methods produce useful results, they are based on the assumption that risk is solely a function of the position of the robot in the environment. However, it is intuitive that the risk also depends on the sate of the robot and its capability to traverse the environment.

As such, recent works propose new methods to exploit both the state of the environment and the state of the robot. For instance, Hakobyan *et al.* [7] assessed the risk of collisions with randomly moving obstacles by formulating a risk-constrained trajectory optimization problem, where the distance to obstacles must be equal to or greater than a fixed threshold. This approach enables efficient trajectory planning in dynamic 3D environments, allowing for varying levels of cautious movement based on the specified threshold. Fan *et al.* [8] enhanced the framework by incorporating different risk factors, such as rough terrain, slopes, overhangs, and narrow passages. They define a tail risk map by condensing risks from these different sources, employing the Conditional Value-at-Risk (CVaR) function. The resulting risk map is then used to assess the risks at each position. Cai *et al.* [9] proposed another approach for evaluating risk by transforming a learned speed distribution map into a risk cost metric, where the traversability speed is captured via experienced trajectories. They used the Model Predictive Path Integral (MPPI) approach, introduced in [10], to solve their optimization problem. Koval *et al.* [11] developed another risk-aware path planning approach, which also uses a priori maps.

Although previous approaches have shown effectiveness, they evaluate risk by calculating the collision probability at individual positions. However, Laconte *et al.* [12] demonstrated that this approach falls short when assessing risk along continuous paths. They showed that the probability of collision, calculated as the joint probability that each cell of a path is free of obstacles, is significantly affected by the grid resolution. To address this limitation, Laconte *et al.* [12] introduced the Lambda-Field framework, a generic approach that allows for natural physical risk assessment along continuous paths. In contrast to the BOG framework, the Lambda-Field framework employs a novel mapping approach where each cell quantifies the density of potential collisions that may pose a risk to the robot. The Lambda-Field framework facilitates the integration of risk along a given path, enabling the evaluation of risks with physical significance. The Lambda-Field framework has been extended by Randriamiarintsoa *et al.* [2] to address 3D environments. This extension introduces a new measure of physical risks that allows the robot to traverse 3D obstacles such as speed bumps or small road curbs at an appropriate speed.

In this work, we address the challenge of incorporating the physics-based risk constraint introduced by Randriamiarintsoa *et al.* [2] into a nonlinear constrained trajectory optimization problem.

III. A RISK-AWARE MOTION PLANNING WITHIN GRID MAPS

A. The Navigation Problem

Suppose a robot must reach a user-defined global position $y_F \in \mathbb{R}^2$ in an environment about which it has no prior knowledge. Assume next that a global planner provides a dynamic local position goal $y_G \in \mathbb{R}^2$, located in the furthest navigable region perceived by the robot. This local position is intended to guide the robot from its current position $y_k \in \mathbb{R}^2$ to the global position y_F. Let $x_k \in \mathbb{R}^n$ and $u_k \in \mathbb{R}^{n_u}$ denote the state and the control input, respectively, of a wheeled robot at time k. The discrete kinematics model of the wheeled robot can be written as:

$$\begin{aligned} x_{k+1} &= f(x_k, u_k) \\ \text{with} \quad u_{\min} &\leq u_k \leq u_{\max} \end{aligned} \tag{1}$$

where $f : \mathbb{R}^n \times \mathbb{R}^{n_u} \to \mathbb{R}^n$ represents the kinematic function that describes the motion of the robot, and $(u_{\min}, u_{\max})$ encompasses the control limits of the robot.

Define $x_{k:N} = \{x_k, x_{k+1}, \cdots, x_{k+N}\}$ as a sequence of $N+1$ discrete states, $u_{k:N-1} = \{u_k, u_{k+1}, \cdots, u_{k+N-1}\}$ as a sequence of N control inputs, and $m = (m^{(1)}, m^{(2)}, \cdots, m^{(M)})$ denotes a grid map with M cells. The risk value r, which measures the interaction between the robot and the environment, is given by:

$$r = \mathcal{R}(x_{k:N}, u_{k:N-1}, m) \tag{2}$$

where $x_{k:N}$ represents the predicted trajectory obtained by applying the control input $u_{k:N-1}$, and $\mathcal{R}(\cdot)$ is a nonlinear

risk function that captures the state of the grid map as well as the motion of the robot within this map. The details of this function are discussed in Section III-B. To coherently combine navigation risk with the local objective of reaching the goal y_G and the local perception, we define the following optimization problem:

$$\begin{aligned} &\text{minimize} && z(u_{k:N-1}) \\ &\text{subject to} && g(u_{k:N-1}) \leq 0 \\ & && \mathcal{R}(x_{k:N}, u_{k:N-1}, m) \leq r_{\text{th}}, \end{aligned} \tag{3}$$

where $z(u_{k:N-1})$ represents the cost associated with the mission of the robot, $g(u_{k:N-1})$ encodes the controller limits and r_{th} denotes a user-defined risk threshold aligned with both the user preferences and the capabilities of the robot. The main objective is then to find the optimal control $u_{k:N-1}$ that minimizes the cost while satisfying the risk and kinematic constraints.

B. The Risk Function

To better understand the risk function used in this framework, we present in this section a brief summary of the Lambda-Field framework we use to assess a risk on a continuous trajectory. In contrast to the BOG framework, which stores collision probabilities for each position in the environment, the Lambda-Field framework stores the intensity of a potentially dangerous event for a robot for each region of the environment, enabling the computation of the risk associated with these regions. For a more detailed overview of the framework, refer to the works of Laconte *et al.* [13] and Randriamiarintsoa *et al.* [2].

Based on previous work, 3D obstacles, such as those depicted in Figure 1, can be handled by computing a grid map which is called Lambda-Field. For this, a Difference Elevation Map (DEM) of the environment is first computed by using an accumulation of 3D lidar sensor data. Then, the Lambda-Field is assessed by evaluating the cells as follows: a cell $m^{(i)}$ is measured as hazardous if its DEM value exceeds a threshold value $H_{\text{safe}} \in \mathbb{R}_{\geq 0}$; otherwise, it is measured as safe. The threshold H_{safe} reflects the capability of the robot to navigate over slightly elevated obstacles, therefore its value highly depends on the size of the wheels of the robot. When evaluating the Lambda-Field, an error region of area $e \in \mathbb{R}_{>0}$ is assigned to each lidar beam which propagates the measurement to neighboring cells. The intensity λ_i of each cell is then computed by using an expectation-maximization approach:

$$\lambda_i = \frac{1}{e} \ln \left(1 + \frac{h_i}{s_i}\right) p_i \tag{4}$$

where s_i is the number of times that a cell is measured as safe and h_i is the number of times that a cell is measured as hazardous. The harmful probability, denoted as p_i in Equation 4 is given by:

$$p_i = \min\left(\frac{|H_i|}{R}, 1\right), \tag{5}$$

where H_i is the DEM value of the cell $m^{(i)}$, and R is the radius of the robot wheels. The weighting p_i ensures that only obstacles higher than the wheel radius are taken into consideration.

The risk function $\mathcal{R}(\cdot)$ shown in Equation 2, which aligns with the capability of the robot to traverse elevated obstacles, is defined as the maximum potential energy that the robot wheels will absorb in the event of a collision with the environment. Specifically, the risk value r at the position of a collision is expressed in Joule and defined as:

$$r(x_i, u_i, m) = \frac{1}{2} k_r \left(\frac{v \cos(\Psi)}{\sqrt{k_r / m_R}}\right)^2 \tag{6}$$

where k_r is the stiffness of the robot wheels, v is the linear speed of the robot, Ψ is the angle of attack of the collision and m_R is the mass of the robot.

From this risk definition and the Lambda-Field map, the expected value of the risk over a trajectory crossing the cells $\{m^{(i)}\}_{0:L-1}$ is computed by:

$$r = \mathcal{R}(x_{k:N}, u_{k:N-1}, m) = \sum_{i=0}^{L-1} K_i r(x_i, u_i, m), \tag{7}$$

where

$$K_i = \exp\left(-\Delta a \sum_{j=0}^{i-1} \lambda_j\right) (1 - \exp(-\Delta a \lambda_i)) \tag{8}$$

is the probability of encountering a harmful event at the position i of the trajectory $x_{k:N}$, L is the number of crossed cells and $\Delta a \in \mathbb{R}_{>0}$ is the area of the cells.

One can note that the Lambda-Field framework is particularly well-suited for risk assessment on continuous trajectories, as the cell area Δa appears in Equation 8, ensuring independence from grid resolution.

C. The Motion Primitives

The optimization problem defined in Equation 3 can be addressed by using MPC methods. MPC is an iterative optimization-based control strategy that systematically solves an optimization problem over a specified time horizon of size N. At each iteration, the first optimal control input from the resulting control sequence is applied to the robot. This process is repeated at each sampling time interval Δt, allowing the controller to continuously adapt to the evolving robot dynamics and environmental changes.

Usually, the control functions u used in a classical MPC approach are piecewise constant functions. Let N represent the number of control intervals over the control horizon, and n_u denote the dimension of the input vector of the robot. While a large number N is preferred in the context of autonomous navigation (i.e., a larger time window), the computational cost of solving the associated optimization problem quickly becomes intractable. To overcome this issue and to exploit all the robot capabilities, we reduce the dimensionality of the problem by using parametrized trajectories [14]. A control family, denoted as $\xi(p)$, maps the parameter $p \in \mathbb{R}^{n_p}$ to the control inputs $u_{k:N-1}$ over a time horizon of size N. The complexity of the trajectory, derived from

applying the control inputs $u_{k:N-1}$ of the control family to the direct kinematic model of the robot, is significantly influenced by the number of parameters and the choice of functions defining the control family.

In the context of a car-like mobile robot, we define the parameter p as $p = (p^1, p^2, p^3, p^4)^\intercal \in \mathbb{R}^4$ and the control input as $u_k = (v_k, \delta_k)^\intercal \in \mathbb{R}^2$, where v_k and δ_k are respectively the speed and steering angle of the robot at time k. We model our control family using linear functions. In this representation, the parameter p^1 represents the target speed to be reached at time t_r^v, while p^3 represents the target steering angle to be reached at time t_r^δ. Analogously, p^2 and p^4 respectively represent the target speed and target steering angle to be reached at time t_N. In Figure 2, we depict two trajectories generated by distinct control family parameters, $p^{\mathrm{B}} = (2.3, 0.7, 5.0, -11.0)^\intercal$ and $p^{\mathrm{O}} = (1.6, 1.6, -5.0, 11.0)^\intercal$, shown in blue and orange, respectively. The trajectories are computed by applying the control family to the discrete Ackermann vehicle model:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + \Delta t \begin{bmatrix} v_k \cos\theta_k \\ v_k \sin\theta_k \\ v_k \tan\delta_k / l \end{bmatrix} \tag{9}$$

where l is the length of the wheelbase of the robot, (x_k, y_k) is the position of the rear axle center, and θ_k is the yaw angle of the robot in an absolute frame. As shown in Figure 2, the selection of linear functions parametrized by p^1, p^2, p^3, and p^4 is sufficient to effectively capture the movement capabilities of a car-like mobile robot. We define the parameters t_r^v and t_r^δ to reflect the acceleration and angular speed capabilities of the robot, broadening the spectrum of car-like mobile robots we can control. Note that adjusting the values of parameters p, t_r^v, t_r^δ, and t_N for both control inputs distinctly influences both the shape and length of the trajectories.

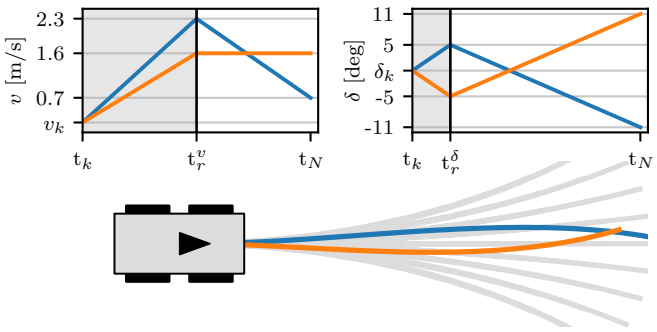

Fig. 2. Example of trajectories generated with our control family. All trajectories are considering the kinematic constraint, the current speed v_k, and the current steering angle δ_k of the robot. Two trajectories generated from the control family are depicted in blue and orange. The trajectories depicted in light gray are generated by others parameters.

D. Optimization Architecture

Using the parametrized control family, the optimization problem defined in Equation 3 over a prediction time of size N is detailed as:

$$\begin{aligned} \text{minimize} \quad & Z = Z_1 + Z_2 + Z_3 \\ \text{subject to} \quad & p^1 \in (0, v_{\max}) \\ & p^2 \in (0, v_{\max}) \\ & \delta_{\min} \leq p^3 \leq \delta_{\max} \\ & \delta_{\min} \leq p^4 \leq \delta_{\max} \\ & d_{\max} \leq \Delta v \leq a_{\max} \\ & -\omega_{\max} \leq \Delta\delta \leq \omega_{\max} \\ & \mathcal{R}(x_{k:N}, u_{k:N-1}, m) \leq r_{\text{th}} \end{aligned} \tag{10}$$

where $(\delta_{\min}, \delta_{\max})$ are respectively the minimum and maximum steering angle the robot can achieve, and $v_{\max}$ is the maximum speed it can reach. Between two sampling times, we constrain the speed variation Δv and the steering angle variation $\Delta\delta$ to not exceed the maximum deceleration $d_{\max}$, maximum acceleration $a_{\max}$, and maximum angular speed $\omega_{\max}$ of the robot.

The penalty term Z_1 is used to drive the robot to the local reference position goal y_G:

$$Z_1 = (y_G - y_N)^\intercal Q (y_G - y_N), \tag{11}$$

where $Q \in \mathbb{R}^{2\times 2}$ is the symmetric positive definite weight matrix used to penalize the error between the local reference position goal y_G and the last predicted position y_N of the robot. The penalty term Z_2 is used to encourage the robot to drive fast when it is feasible:

$$Z_2 = \sum_{i=0}^{N-1} (v_{\max} - v_i) w_v (v_{\max} - v_i), \tag{12}$$

where $w_v \in \mathbb{R}_{>0}$ is the weighting coefficient that penalizes low speed and v_i is the speed of the robot at time i. The penalty term Z_3 is used to prevent frequent changes in direction.

$$Z_3 = \sum_{i=0}^{N-1} (\delta_k - \delta_i) w_\delta (\delta_k - \delta_i) \tag{13}$$

where $w_\delta \in \mathbb{R}_{>0}$ is the weighting coefficient that penalizes the direction changes and δ_i is the steering angle of the robot at time i.

Following the standard MPC approach, the optimization problem defined in Equation 10 can be solved by iteratively optimizing an initial parameter guess p^0 to the optimal parameter p^*. As depicted in Figure 3, at step k the obtained optimal parameter p_{k+1} is used to compute the optimal control $u_{k:N-1}$ over the horizon of prediction N using a block function named `Control Family` that maps the control parameters to the control inputs. The next optimal control u_{k+1} that will be applied to the robot is simply the first term from the resulted optimal control inputs.

However, as the risk constraint $\mathcal{R}(x_{k:N}, u_{k:N-1}, m)$ defined in Equation 10 is nonlinear, this optimization process is prone to failures. Without proper initialization, the optimizer will often fail to find a solution that satisfies the risk constraint, which can lead to hazardous behaviors. It is therefore insufficient to rely solely on the previous described

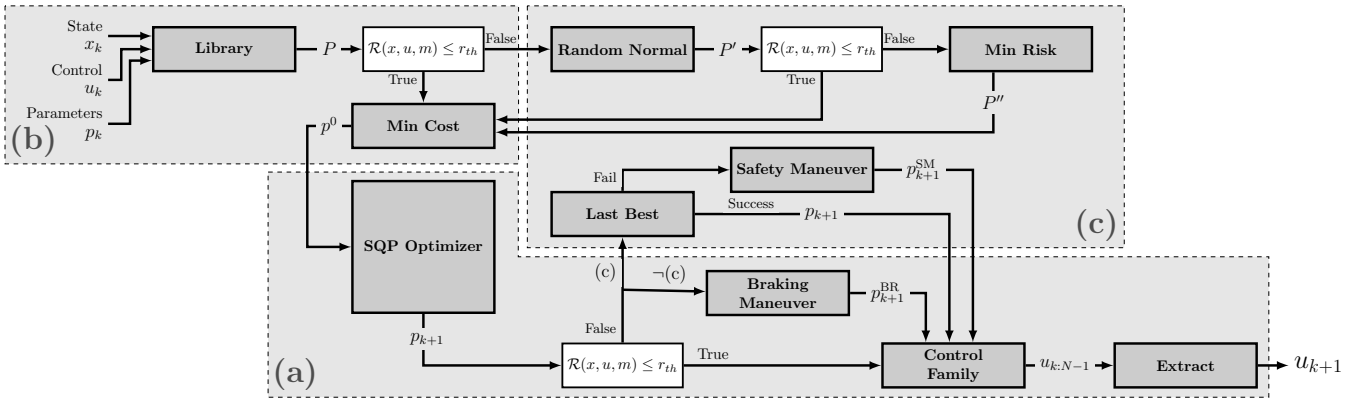

Fig. 3. Overview of the architecture. The current state x_k, control input u_k, and the last optimized control family parameter p_k are used by the system to compute the next optimal control u_{k+1} that steers the robot towards the current goal, considering the risk involved in the map m. The system integrates three key modules: a) an efficient optimization solver based on SQP; b) a trajectory selection process that improves the optimizer performance by providing a good initial guess p^0 for the solution; c) a safety maneuver trajectory module for recovery in case of failure.

as module (a) in Figure 3. To ensure that the risk constraint is not violated during the optimization process, we propose to add two modules (b) and (c).

First, a module (b) is added to the pipeline. The objective of this module is to build a set of tentacles that explores the environment in order to provide a well-informed initial guess p^0 to the optimizer. These tentacles are computed by using the control family mapping $\xi(\cdot)$ over a set of parameters p. This set of parameters named P in Figure 3 is obtained by using the `Library` block function. This function discretizes the continuous parameter space into a finite number of N_D parameters, taking into account the kinematic constraints defined in Equation 10. Note that at each iteration of the optimization process, we add the last optimized parameter p_k to the set P. Indeed, if the environment has not changed significantly, this parameter is still likely to be close to the optimal one. If multiple tentacles satisfy the risk criterion, we select the one that minimizes the cost Z and pass it to the optimizer as an intial guess p^0. The optimizer then refines this initial guess p^0 as described previously. Section IV-A demonstrates the benefits of this approach for the optimization process.

Despite these enhancements, there are still two scenarios where the approach may fail. They are addressed by module (c). The first scenario arises when none of the tentacles satisfies the risk constraint. In such cases, the parameter set P is replaced with a new set P' derived by perturbing the parameters associated with the least risky trajectory found in P using a normal distribution. This process is performed in the block function `Random Normal` in Figure 3. If the set P' results in tentacles that satisfy the risk constraint, we select the one with the highest score and pass it to the optimizer. However, if none of the tentacles meets the risk constraint, we prioritize selecting the parameter p^0 based on minimizing risk first, followed by minimizing cost.

The second scenario arises if the module (a) fails to converge to a feasible solution. In this instance, we first employ the `Last Best` block function, as illustrated in Figure 2. This function attempts to retrace the iterations of the `SQP Optimizer` until identifying a parameter p that aligns with the risk constraint. Since we retrace all the iterations of SQP, it is possible to find a solution p that both satisfies the risk constraint and minimizes the cost, taking advantage of the nature of the SQP and the proper initialization of the algorithm. However, there is a possibility that no parameter p is identified, indicating that the optimization might have started with a risky parameter as an initial guess. This situation can occur in complex scenarios where, based on its current configuration, the robot faces challenges in avoiding the hazard. In such a scenario, the `Safety Maneuver` function depicted in Figure 3 is used to generate a safety maneuver consisting in steering the robot towards the least risky region of the environment while maximizing the robot deceleration. To achieve this, we set the steering angle parameters (p^3, p^4) to values that best minimize the risk in the SQP steps, and the speed parameters (p^1, p^2) to zero values (safety maneuver parameters p^{SM}_{k+1}).

It is noteworthy that an optimal solution to Equation 10 may be found during an safety maneuver, as the robot transitions into a new configuration and may access to a fresh set of feasible inputs.

IV. EXPERIMENTS

In this section, we first conduct a quantitative evaluation of our framework through an ablation study. Then we demonstrate the effectiveness of our approach through a real-world experiment.

A. Evaluation of the optimization pipeline

In this section, we evaluate the optimization pipeline depicted in Figure 3. Our framework is designed to identify a feasible starting point to prevent optimization failures. Furthermore, in scenarios where the optimizer encounters failures, the architecture is also designed to recover and minimize potential damage.

Figure 4 depicts the simulated environments used for the evaluation. The initial condition is the same for each environment, with the robot positioned at the left and facing

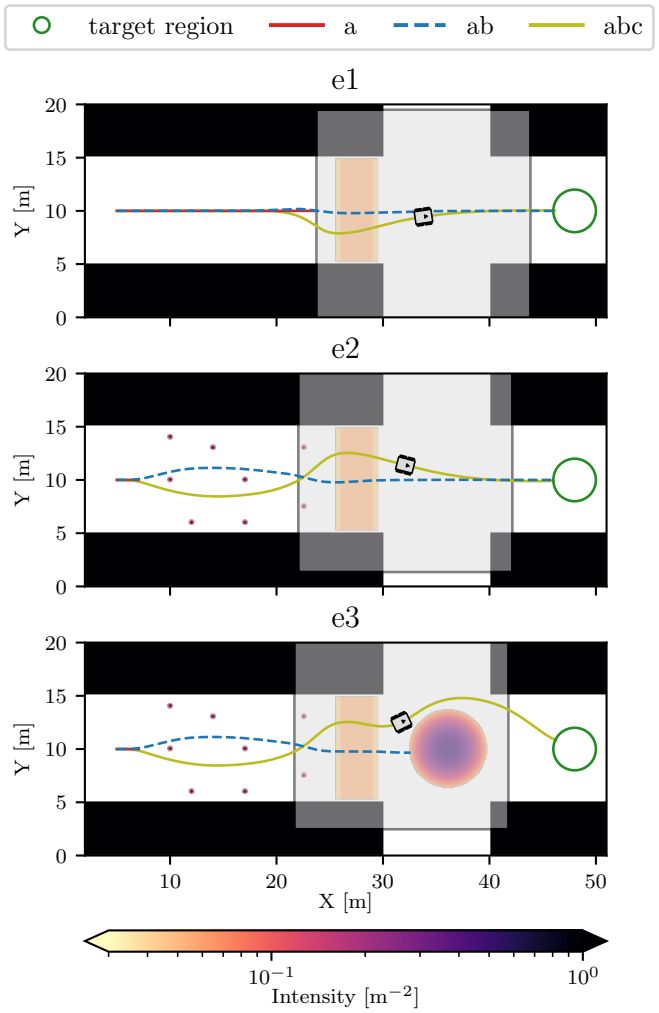

Fig. 4. Simulation environments: the robot (box) has to reach the region within the green circle while managing the risk involved in each environment. Each path computed by using the module (a), (a)-(b) and (a)-(b)-(c) are respectively depicted with red solid line, blue dashed line and light green solid line. The gray-shaded rectangle represents the area of the Lambda-Field map seen by the robot at the midpoint of the simulation.

to the right. The control input v_0 and δ_0 at initial time t_0 are set to zero. Each simulation have a duration limit of 30 s. The local map size of the robot is set to $20.1\,\mathrm{m} \times 20.1\,\mathrm{m}$ and the robot width is set to 66.5 cm. For all the environments, the risk threshold r_{th} is consistently set to 10 J and is combined with the parameters shown in Table I. The SLSQP otpimizer [15] is used as the backbone optimizer to solve Equation 10.

In Figure 4, the complexity of the test environments increases from top to bottom. The environment is becoming increasingly cluttered by different elements. The Lambda-Field map of each simulated environment is depicted in Figure 4 using a color scale. The color scale represents the intensity value of each cell of the map. The higher the intensity of a cell in the map, the more likely it is that a dangerous event will occur in that cell. On the other hand, a cell with zero intensity (in white in Figure 4) means that it will never lead to a dangerous event. In the environment `e1`, the risk only arises from the walls (in black in Figure 4)

TABLE I

PARAMETERS IN EVALUATION OF THE OPTIMIZATION PIPELINE

Parameter	Value	Parameter	Value
$v_{\max}$	$3.35\,\mathrm{m\,s^{-1}}$	N	30
$a_{\max}$	$1\,\mathrm{m\,s^{-2}}$	Δt	0.1 s
$d_{\max}$	$3\,\mathrm{m\,s^{-2}}$	Q	$\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$
$\delta_{\max}$	$23°$	w_v	2
$\delta_{\min}$	$-23°$	w_δ	4
$\omega_{\max}$	$0.2\,\mathrm{rad\,s^{-1}}$	N_D	81
R	20.5 cm	t_r^v	1.5 s
l	50 cm	t_r^δ	0.5 s

TABLE II

SUCCESS RATE IN FINDING A FEASIBLE SOLUTION AND TRAVERSAL TIME FOR EACH MODULE IN DIFFERENT SCENARIOS

	Environment		
Module	e1	e2	e3
a	26.58%-30.0 s	5.98%-30.0 s	5.98%-30.0 s
ab	84.39%-23.0 s	76.41%-27.0 s	62.13%-30.0 s
abc	**100.00%-19.7 s**	**100.00%-20.3 s**	**100.00%-21.1 s**

and the speed bump (in yellow-orange in Figure 4). In the environment `e2`, traffic cones are added to the environment to prevent the robot from starting out in a straight line. Finally, a roundabout is added in the environment `e3` to significantly increase the complexity of the environment.

For each environment, we challenge the robot to reach a target region (in green circle in Figure 4) located beyond its field of perception. The traversal time of the robot to reach the target region is shown in Table II. If the robot fails to reach the target region within the allotted time, we show in Table II the duration of the simulation instead of the traversal time. Each percentage shown in Table II represents the success rate of each module in finding a feasible solution out of the total number of optimizations performed over the 30 s of simulation. It is important to note that when the module fails, the resulting trajectory involves a risk greater than the acceptable threshold. Since the risk of crossing an element is related to the speed at which the robot crosses it, this indicates that the robot is moving too fast to cross the element safely. Therefore, the robot must adjust its speed appropriately to ensure safe traversal and avoid any potential damage. Note also in Figure 3 that when the module (c) is not in use, the `Braking Maneuver` function ensures that if no feasible solution is found, the robot safely comes to a stop.

We perform an ablation study, focusing on the added value of our modules (b) and (c). First, we use solely the module (a) which consists in the classical NMPC. We see in Figure 4 and in Table II that this module fails to drive the robot to its goal. In the environment `e1`, the optimizer fails to find a feasible solution when a speed bump obstructs the path to the goal, resulting in a success rate of only 26.58%. In the environment `e2` and the environment `e3` the success rate is worse because the optimizer is quickly confronted with a risk. As the module (a) performs poorly against risk, module

(b) is added to module (a), resulting to the couple module (a) - module (b).

The aim of the module (b) is to provide the optimizer of the module (a) with a good starting point at all times. In the environment `e1`, the couple module (a) - module (b) successfully hits the target region within the allotted time. Furthermore, as shown in Table II, it achieves this in only 23.0 s of traversal time. In constrat to the sole module (a), the couple module (a) - module (b) successfully hits the target region in the environment `e2`. However, the traversal time in this environment is greater than the traversal time in the environment `e1`. This is due to the presence of traffic cones, which cluttering up the environment. As one of the traffic cones obstructs the straight path to the target region, and these traffic cones create unacceptable risks to cross, the couple module (a) - module (b) must plan trajectories that avoid these traffic cones, forcing the couple module (a) - module (b) to take a slower but safer path. In the environment `e3`, the couple module (a) - module (b) is unable to drive the robot to the target region due to a roundabout that obstructs the path to the goal. Figure 4 shows that the optimizer fails into a local minimum. Indeed, in this case, the couple module (a) - module (b) fails to find an escape route. The only solution the couple module (a) - module (b) have found to minimize potential damage is to sufficiently slow down the robot to stop right in front of roundabout.

Finally, we add the module (c) which enable recovery from failures. As shown in Table II, the addition of the module (c) significantly reduces the traversal time of the robot from 23 s to 19.7 s in the environment `e1`, and 27 s to 20.3 s in the environment `e2`. Figure 4 and Table II show that only the full combination module enables the robot to reach the target region in the environment `e3`. We can also see in Figure 4 that, in each environment, the robot clearly optimizes its path to maximize speed while ensuring that the risks associated with the environment remain below the risk threshold r_{th} during the traversal.

Through this ablation study, we demonstrate the effectiveness of our framework in managing risk and failures. We have demonstrated the contribution of each module to managing increasingly complex environments. This study demonstrates that our framework can satisfy the risk constraint while minimizing traversal time.

B. Application to the Lambda-Field Framework

We applied our framework in the real-world scenario depicted in Figure 1, where the robot encounters different elements such as a speed-bump, traffic cones, a roundabout, sidewalks, fences, and walls. In this experiment, the global position to reach is deliberately placed beyond its field of perception. The results of this experiment are presented in Figure 5.

From the robot capabilities we set the maximum speed parameter v_{max} to $1\,\mathrm{m\,s^{-1}}$ and the steering angle range (δ_{min},δ_{max}) is set to $\pm 13°$. Considering the robot wheel radius $R = 20.5\,\mathrm{cm}$, we allow the robot to take a little amount of risk, specifically the risk threshold is set to

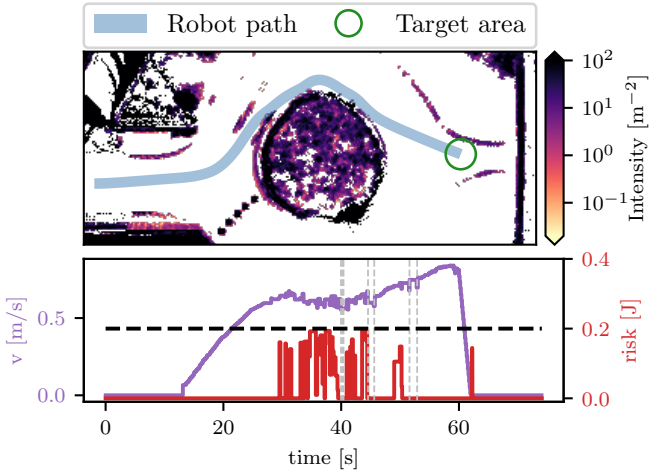

Fig. 5. The first row shows the Lambda-Field map of the environment obtained after the traversal. The path followed by the robot is shown in light blue and the target area is represented by a green circle. The last row shows the speed of the robot in purple and the predicted risk during the traversal of the environment. The risk threshold is depicted by a horizontal dashed black line. And the times when an emergency maneuver has been activated are depicted by vertical dashed gray lines.

$r_{\text{th}} = 0.2\,\mathrm{J}$. The local map size of the robot is set to $20.0\,\mathrm{m} \times 20.0\,\mathrm{m}$. The horizon time of prediction is set to 6 s and the sample time Δt for trajectory discretization is set to 0.1 s. The control loop frequency is set to 10 Hz. The coefficient w_v and w_δ are both set to 2.0, meaning that reducing the steering variation and maximizing the speed are twice as important as reaching the intermediate target position. When the final position is sufficiently close (i.e., when the robot enters the target area), the weight w_v of the cost Z_2 is set to zero, resulting into a stabilization problem. Additional settings are shown in Table III.

TABLE III
PARAMETERS IN THE REAL-WORLD SCENARIO

Parameter	Value	Parameter	Value
$a_{\max}$	$1\,\mathrm{m\,s^{-2}}$	$d_{\max}$	$3\,\mathrm{m\,s^{-2}}$
$\omega_{\max}$	$0.225\,\mathrm{rad\,s^{-1}}$	l	50 cm
Q	$\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$	N_D	82
t_r^v	1.5 s	t_r^δ	0.3 s

It can be noted in Figure 5 that the robot initially follows closely the shortest path to the global target area, since it encounters no risk along its path. As the robot approaches the roundabout, some traffic cones are preventing the robot to turn right. The controller adjusts the robot course to navigate around the roundabout in the left lane. However, a speed bump lies on this path, increasing the risk involved. As the risk threshold has been set to 0.2 J, the framework controls the risk throughout the entire traversal through the management of the robot speed. The last row in Figure 5 shows that the risk taken by the robot is systematically below the threshold, and the few emergency maneuvers never lead the robot to take hazardous paths. This experiment

demonstrates that our framework is applicable in real-world scenarios.

V. CONCLUSION

In this paper, we presented a new risk-aware motion planning framework that is designed for safe and resilient navigation in Lambda-Field grids. We have introduced a novel formulation for a NMPC controller that takes into account a nonlinear risk function as a hard constraint and supplemented it with modules to address failures of the optimization process. We have demonstrated through an ablation study that even in the event of occasional failures, our architecture enables the robot to fulfill its mission while satisfying the risk constraint in both simulation and real-world scenarios. The real-world scenario highlights the practical relevance and applicability of our approach to unknown environments. Indeed, we have succeeded in planning safe trajectories that not only guarantee safety but also actively address the risks arising from interactions with an unknown environment.

Future works will focus on incorporating a more advanced global planner to demonstrate the pertinence of the framework in larger-scale experiments. We will explore alternative risk metrics, such as CVaR, to account for rare but high-impact risk events. Sequential Convex Programming will also be explored to improve satisfaction with the risk constraint in the optimizer. Furthermore, to enhance decision-making in intricate scenarios, such as handling visibility problems due to fog, rain, or darkness and, in general, lack of knowledge, we intend to use additional risk functions derived from these sources. Indeed, enabling mobile robots with such capabilities will considerably improve their autonomy in complex and unknown environments.

ACKNOWLEDGMENT

This work has been supported by the AURA Region and the European Union (FEDER) through the LOG SSMI project of CPER 2020 MMaSyF challenge.

REFERENCES

[1] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta Numerica*, vol. 4, pp. 1–51, 1995.

[2] E. Randriamiarintsoa, J. Laconte, B. Thuilot, and R. Aufrère, "Risk-aware navigation for mobile robots in unknown 3d environments," in *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, 2023, pp. 1949–1954.

[3] C. Coué, C. Pradalier, C. Laugier, T. Fraichard, and P. Bessière, "Bayesian occupancy filtering for multitarget tracking: An automotive application," *The International Journal of Robotics Research*, vol. 25, no. 1, pp. 19–30, 2006.

[4] J. Schroder, T. Gindele, D. Jagszent, and R. Dillmann, "Path planning for cognitive vehicles using risk maps," in *2008 IEEE Intelligent Vehicles Symposium*, 2008, pp. 1119–1124.

[5] A. A. Pereira, J. Binney, B. H. Jones, M. Ragan, and G. S. Sukhatme, "Toward risk aware mission planning for autonomous underwater vehicles," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 3147–3153.

[6] S. Primatesta, G. Guglieri, and A. Rizzo, "A risk-aware path planning strategy for UAVs in urban environments," *Journal of Intelligent & Robotic Systems*, vol. 95, pp. 629–643, 2019.

[7] A. Hakobyan, G. C. Kim, and I. Yang, "Risk-aware motion planning and control using CVaR-constrained optimization," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3924–3931, 2019.

[8] D. D. Fan, K. Otsu, Y. Kubo, A. Dixit, J. Burdick, and A. Agha-Mohammadi, "STEP: stochastic traversability evaluation and planning for safe off-road navigation," *CoRR*, vol. abs/2103.02828, 2021.

[9] X. Cai, M. Everett, J. Fink, and J. P. How, "Risk-aware off-road navigation via a learned speed distribution map," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 2931–2937.

[10] G. Williams, N. Wagener, B. Goldfain, *et al.*, "Information theoretic mpc for model-based reinforcement learning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1714–1721.

[11] A. Koval, S. Karlsson, and G. Nikolakopoulos, "Experimental evaluation of autonomous map-based spot navigation in confined environments," *Biomimetic Intelligence and Robotics*, vol. 2, no. 1, p. 100 035, 2022.

[12] J. Laconte, C. Debain, R. Chapuis, F. Pomerleau, and R. Aufrère, "Lambda-field: A continuous counterpart of the bayesian occupancy grid for risk assessment," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 167–172.

[13] J. Laconte, A. Kasmi, F. Pomerleau, *et al.*, "A novel occupancy mapping framework for risk-aware path planning in unstructured environments," *Sensors*, vol. 21, no. 22, 2021.

[14] N. Morette, C. Novales, L. Josserand, and P. Vieyres, "Direct model navigation issue shifted in the continuous domain by a predictive control approach for mobile robots," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2566–2573.

[15] D. Kraft, "A software package for sequential quadratic programming," *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt*, 1988.