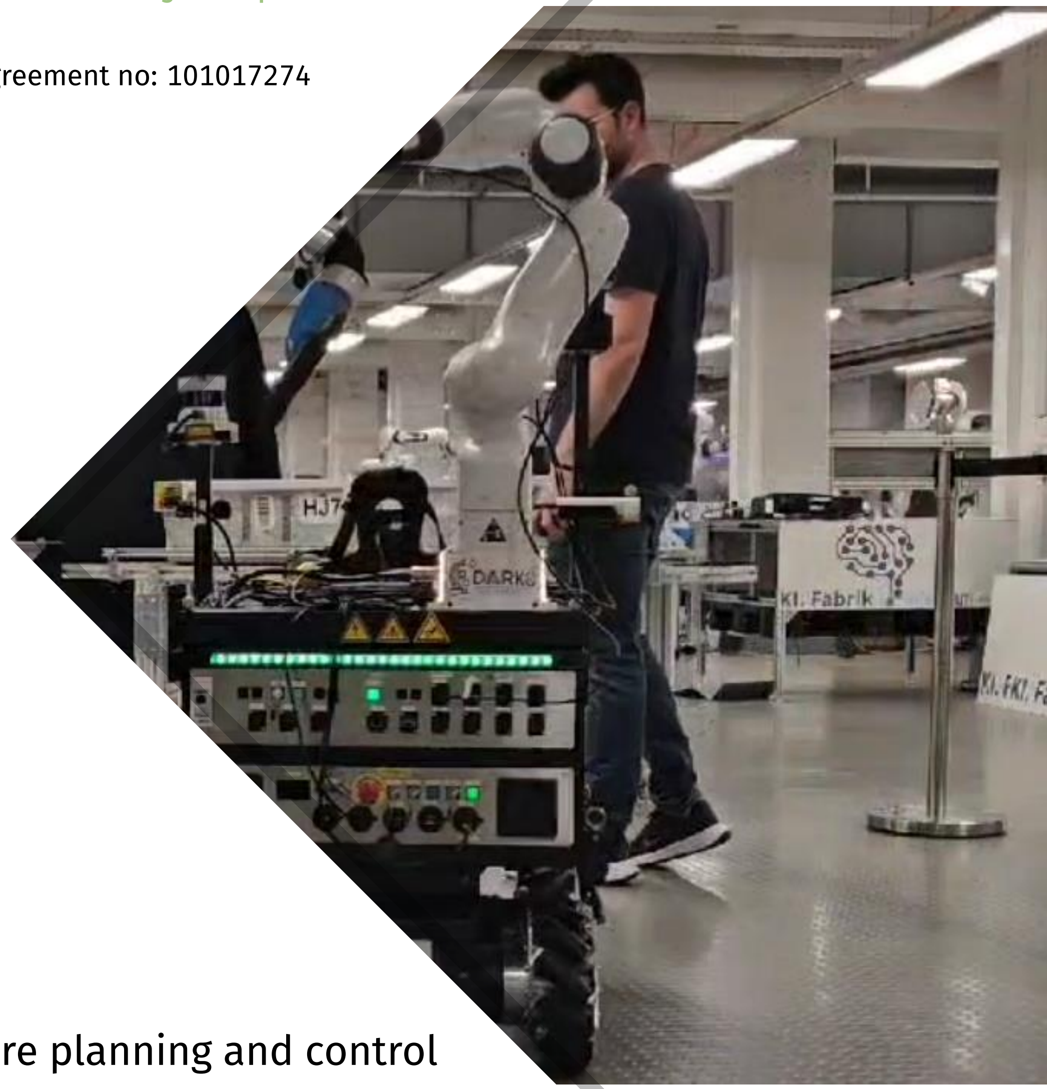
Dynamic Agile Production Robots That Learn and Optimise Knowledge and Operations

H2020-ICT-2020-2 Grant agreement no: 101017274



DELIVERABLE 6.2
Report on risk-aware planning and control

Dissemination Level: PUBLIC
Due date: month 42 (December 2024)
Deliverable type: Report
Lead beneficiary: UNIPI

1 Introduction

The results presented in this report directly contribute to the objectives of the DARKO project, specifically: **(O2)** Efficiency in human–robot coproduction, and **(O4)** Risk-aware operation for safety and efficiency. This deliverable describes the work carried out within task T6.2, specifically on the design of risk-aware strategies to cope with uncertainty and risk factors during all stages of robot navigation. By incorporating heightened awareness of uncertainty, robotic systems can effectively prevent undesirable situations, thus enhancing both safety and performance.

This task is closely related to other components of the navigation stack: task T6.1, concerning context and predictive local planning, task T6.3 addressing efficient and context-aware global planning, and finally the safety layer proposed in T6.4. We will explain how the work carried out in T6.2 is integrated in the navigation motion planning pipeline in section 2. Furthermore, this work builds on the risk factors identified in task T7.1.

In the risk management literature, risk is defined as an uncertain event that, if it occurs, has a negative impact on the success of a task. It is typically expressed as a combination of probability and severity. In the state-of-the-art in motion planning, risk is often considered solely as the probability of a collision. In contrast, our approach extends the risk concept in motion planning, including together with the collision risk other factors that could compromise task success. These additional risks include potential delays caused by human presence along the selected route, loss of localization, or navigation through an area with objects difficult to detect. To address these challenges, we explicitly incorporate uncertainty and implement strategies at both the global route selection and local planning levels.

At the global planning level, a key source of uncertainty for a robot operating in a human-robot shared environment is the unpredictable presence of humans within the workspace. For example, if a robot encounters a person while traversing a constrained area, it may need to stop or slow down to ensure safe interaction, thereby increasing the risk of delays. Consequently, encountering a person during navigation is treated as a risk factor. This risk factor will have as probability the probability of encounters and as severity a measure of the impact of having an encounter at that location. We describe the risk-aware route decision maker developed in Section 3.

At the local planner level, we introduce multiple risk maps, each representing a different space-dependent (static or dynamic) risk factor probability. These maps allow the planner to assess and respond to localized risks more effectively. In addition, we design an active sensing layer to minimize uncertainties related to localization and object detection, thus reducing the risk of task failure. The methodology for addressing risk factors within the local planner is discussed in Section 4. Our research also focused on formally evaluating the probability of collision along continuous paths (Section 5).

Finally, Section 6 summarizes the key findings and insights from this work and discusses ongoing and future works.

2 Overview

In this section, we describe how this work integrates with the other components of the navigation motion planning pipeline.

The DARKO navigation architecture was designed according to the following requirements:

- Make use of the many *contextual cues* that the robot can perceive. These cues

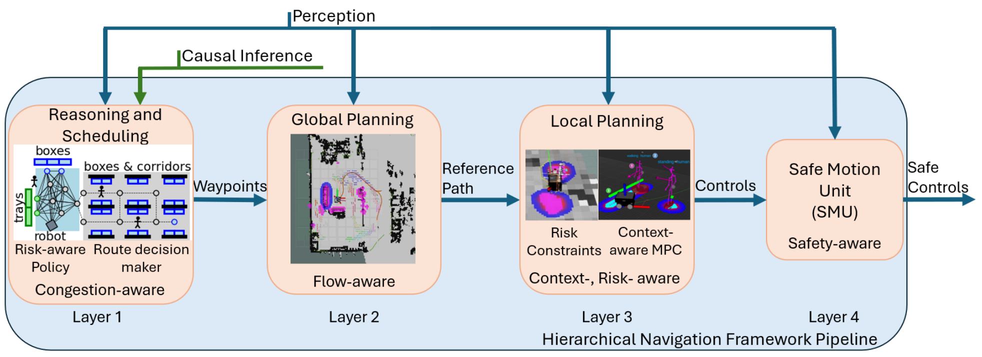


Figure 1: Architecture overview of our navigation system. The components described in this deliverable are the route decision maker and the risk constraints at the local planner level.

include detected positions, poses and activities of the moving people, static and dynamic scene maps, congestion risks estimations and human-robot spatial interaction components.

- Decompose the complex planning problem into a *hierarchical system* of layers, each effectively handling a different planning horizon from risk-aware task scheduling to safe velocity control in close proximity to people.

- Make the navigation safe and efficient by considering *predictions and risk assessments* on every layer. The generated plans should reduce the risk of collision or in general robot's unsafe operations to a minimum.

According to these requirements, we have developed a multi-layered navigation stack, see Figure 1. The architecture follows a *predictive planning* setup: differently from a traditional *sense-plan-act* one [25], our architecture plugs on different types of predictions of surrounding humans and dynamic objects inside the different layers, considering different time scales. Similarly, we allow the usage of several contextual cues across the several layers, e.g., risks, human activities. Our architecture is composed of four main layers: (1) Reasoning and Scheduling (WP7, T6.2), (2) Global planning (T6.3), (3) Local planning (T6.1, T6.2), (4) Safety layer for safe vehicle motion (T6.4).

The work in this task covers 1) the risk-aware route decision maker in the Reasoning and Scheduling block and 2) the risk constraints integrated in the context-aware MPC (Model Predictive Control) developed in T6.1.

The reasoning and scheduling layer are responsible for scheduling the next task and selecting a rough path through the large environment with many ways to reach the desired location. The risk-aware policy (WP7) aims to define the next operational actions, such as selecting which object to retrieve or engage with and determining strategic waypoints for efficient task execution. Instead, the risk-aware route decision-maker or route planner, see Section 3, is used to perform long navigation tasks. The goal of this module is to provide a policy that selects routes minimizing the risk of performance degradation, such as delays caused by necessary slowdowns during human-robot interactions, to ensure safety. The problem is formulated as a Markov Decision Process (MDP), where states include potential observations the robot can make at intersection points. This formulation allows for the modeling of dynamic re-planning, obstacle visibility, alternative route costs, and the severity of potential encounters, offering a more adaptive and robust navigation strategy.

We use heterogeneous maps to incorporate the concept of risk in the local planner. The risk maps include a dynamic collision risk map that accounts for dynamic uncertainties in obstacle detection and localization, and static risk maps, e.g., alignability maps [5], which encode the risk of loss of localization accuracy for all traversable regions. The MPC integrates these risk maps as weighted soft constraints, adjusted based on the severity of the corresponding risk factor.

In addition to the risk maps, we plan to incorporate an active sensing layer into the MPC framework, as described in [15, Ch. 6]. This layer will enhance safety by reducing collision and localization uncertainties by optimizing sensory information flow metrics [16]. This enhancement will equip the robot with dynamic re-planning capabilities to mitigate task failures and facilitate recovery from localization loss. This framework will be presented at the 2025 European Robotics Forum [26].

3 Risk-aware route planning

Indoor environments typically have well-defined layouts and maps that allow efficient path planning, routing, and navigation. However, a route or path that traverses dynamic, shared, and constrained spaces (e.g., corridors in a warehouse) can quickly degrade the robot's performance (e.g., delaying the reaching of the final destination) when these spaces are occupied by people or temporary obstructions. To move safely along humans, autonomous robots need to adjust their trajectory near people locally, to reduce their velocity, or to change route, resulting in some performance degradation. Furthermore, collaborative robots should also ensure that the interaction is perceived safe by the human during the encounter [22]. Consequently, when an autonomous robot enters a setting where humans may be present, it is prudent to recognize the potential risks associated with potential encounters throughout the various phases of motion planning. The number and location of humans in the environment are typically uncertain, and the location of the encounter is associated with different levels of severity. Autonomous robots are typically equipped with onboard sensors and detection modules that can gather real-time data on congestion levels as the robot approaches critical decision points where multiple paths or routes to the final destination are available. This allows for online adjusting the path/route if less risky alternatives are available. We investigated how to exploit, in a routing problem, with some prior information, the robot's ability to perform local observations to anticipate situations potentially at high risk of degrading the robot's overall performance (apart from safety).

Consider the problem of planning the best strategy for a robot to move from a starting position S to a goal G. Let us assume, for simplicity, that the probability distributions of the costs of each edge are independent. Notably, the solution can be found with one of many algorithms from the literature [9] and is given by a static path *path* $\mathscr{P}^*$ from S to G, characterized by the minimum possible expected cost

$$\mathscr{P}^* = \arg\min_{\mathscr{P}} C(\mathscr{P}). \tag{1}$$

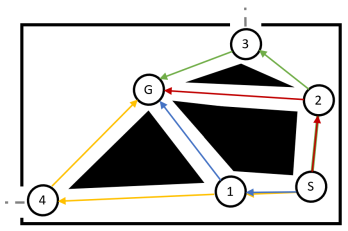
Consider, for example, the simple case in Figure 2, where the graph represents the homotopy classes of a room with three obstacles, a starting point, a goal, and two doors, which translate into five *decision points* and eight *edges*.

Starting from node S, four alternative paths exist to reach node G: the blue, the red, the green, and the yellow one. By computing the expected cost ($\mathbb{E}[C]$) of the four paths we get

$$\mathbb{E}[C_{blue}] = c(S,1) + \mathbb{E}[C_{(1,G)}] = 12.7,$$

$$\mathbb{E}[C_{red}] = c(S,2) + \mathbb{E}[C_{(2,G)}] = 15.75,$$

$$\mathbb{E}[C_{green}] = c(S,2) + \mathbb{E}[C_{(2,3)}] + \mathbb{E}[C_{(3,G)}] = 19.1\,,$$



Edge	Cost
$(S,1)$	1.5
$(1,G)$	$\begin{cases} c_1 = 4,\ p_1 = 0.8 \\ c_2 = 40,\ p_2 = 0.2 \end{cases}$
$(1,4)$	$\begin{cases} c_1 = 6,\ p_1 = 0.7 \\ c_2 = 40,\ p_2 = 0.3 \end{cases}$
$(4,G)$	$\begin{cases} c_1 = 6,\ p_1 = 0.7 \\ c_2 = 40,\ p_2 = 0.3 \end{cases}$

Edge	Cost
$(S,2)$	2
$(2,G)$	$\begin{cases} c_1 = 5,\ p_1 = 0.75 \\ c_2 = 40,\ p_2 = 0.25 \end{cases}$
$(2,3)$	$\begin{cases} c_1 = 3,\ p_1 = 0.85 \\ c_2 = 40,\ p_2 = 0.15 \end{cases}$
$(3,G)$	$\begin{cases} c_1 = 3,\ p_1 = 0.85 \\ c_2 = 40,\ p_2 = 0.15 \end{cases}$

Figure 2: Stochastic graph where the edge cost becomes known in the node at the tail of the arc. From node S to node G there are four different paths: the blue, the red, the green, and the yellow.

$$\mathbb{E}[C_{yellow}] = c(S,1) + \mathbb{E}[C_{(1,4)}] + \mathbb{E}[C_{(4,G)}] = 33.9\,.$$

Therefore, the minimum cost is 12.7, which is obtained, on average, by the blue path.

Let us assume now that, when the robot approaches a *decision point*, its sensors provide some information y about the congestion of adjacent *routes*. Consequently, it is reasonable that the probabilities p_j associated with the cost C_i assuming values c_j may change to some new values

$$\mathbb{P}(C_i = c_j | y) = \hat{p}_j\,. \tag{2}$$

As new information is acquired while moving, the optimal *path* $\mathscr{P}^*$ will not be static, but will adapt online according to the performed observation. In this scenario, it is more appropriate to talk about policies. In general, a policy Π is a function

$$\Pi = \Pi(v) : V \to E\,, \tag{3}$$

that given the current configuration of the robot, represented by a node v, yields the next action, here represented by the edge e.

The solution of (1) can be extended to a static policy Π_s that assigns the action corresponding to the minimum prior expected cost at each node. However, this policy is, of course, not optimal as it does not take into account the new information acquired online.

In this case, a common approach [6] is to start to follow the optimal path, and then re-calculate a new static path $\mathscr{P}_y^*$, which is optimal based on the new information y. This generates a naive dynamic policy Π_d. However, such a policy is not always optimal. Consider again our toy example in Figure 2, where each edge cost becomes known when the robot reaches its outgoing node. The optimal path is the blue one, so the robot would take edge $(S,1)$. It could happen that after the robot has executed the first step of the originally optimal path $\mathscr{P}^*$, it observes that an obstacle is present on the edge $(1,G)$, thus updating the graph costs, and, if edge $(1,4)$ is free (so its value is 6), it produces a novel policy Π_y^* for which the next optimal step is $(1,4)$. Such a strategy would lead to an expected cost

$$\begin{aligned}\mathbb{E}[C]_{\Pi_d} = c(S,1) + p_1(1,G)c_1(1,G) + p_2(1,G)[p_2(1,4)\cdot \\ \cdot c_2(1,G) + p_1(1,4)(c_1(1,4) + \mathbb{E}[C_{(4,G)}])] = 10.21.\end{aligned}$$

However, the overall optimal dynamic policy (Π^*) for the toy example is as follows: the robot takes edge $(S,2)$. Here, if the edge $(2,G)$ has cost 5 or if both edges $(2,G)$ and $(2,3)$ have cost 40, the robot takes edge $(2,G)$. In the other case, it takes edge $(2,3)$. Indeed, computing the expected cost of this policy results in

$$\begin{aligned}\mathbb{E}[C]_{\Pi^*} = c(S,2) + p_1(2,G)c_1(2,G) + p_2(2,G)[p_2(2,3)\cdot \\ \cdot c_2(2,G) + p_1(2,3)(c_1(2,3) + \mathbb{E}[C_{(3,G)}])] = 9.70\,.\end{aligned}$$

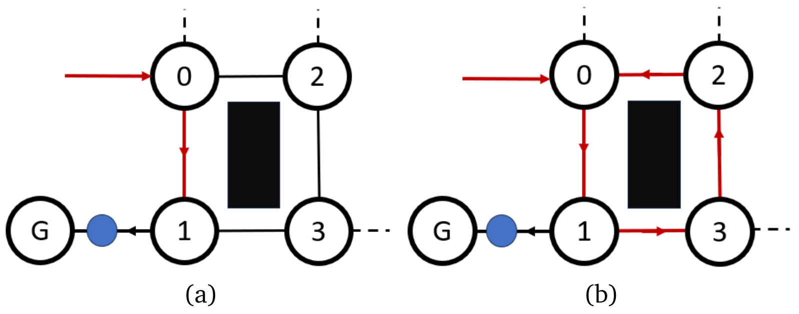


Figure 3: Suppose a scenario where the only way to reach the goal from node 1 is to take the edge on its left. The robot movements are represented by red arrows, the blue dot is a static person performing a task on the edge: (**a**) If the robot arrives in 1 and observes an obstacle on the left edge, and none on the right one, if the severity of passing in $(1, G)$ is higher than the expected cost of going to node 3 and return to node 1, the robot would follow such strategy. (**b**) If the obstacle is static, however, this is not true, and, returned to node 1, the robot will make the same observation entering a loop.

This problem belongs to a class known as "stochastic shortest path with recourse" (SSPR) [20]. In general, finding a policy that is optimal under the above-mentioned assumptions is not a trivial task. Indeed, whenever the graph $\mathcal{G}$ contains cycles, which happens very often in practical cases (see e.g., Figure 4), solving SSPR can be NP-Hard [20].

In [21], authors demonstrate that solving SSPR for networks with cycles can be done with polynomial complexity only if the stochastic graph has the property that each time the agent returns to the same node, the cost of its outgoing star arcs is independent of previous realizations (otherwise the problem is, at best, NP-complete). This is known as the *reset* property. However, assuming *reset* for networks with cycles can lead to policies that get stuck in loops when the real use-case problem does not have this property.

An example of this problem is shown in Figure 3. There, the only way to reach G from node 1 is to take the edge $(1, G)$, whereas, by assuming the reset property, it seems less costly to loop through $(1,3),(3,2),(2,0),(0,1)$ and back to node 1 “in hope" of a better observation. This clearly leads to policies that are not useful in practice, e.g., in the case of static obstacles whose position is stochastically defined.

Therefore, our goal is to make such solutions usable also in those cases, by proposing a suitable mitigating strategy. Moreover, in the classical SSPR problem formulation, as in our toy example, the edge cost would become known when the agent reaches its tail $\mathbb{P}(C_i = c_j|y) = 1$. Instead, in a shared dynamic environment, this is not true, as people may enter while the robot is already moving in the corridor, or might not be visible from the decision point. We therefore need to explicitly compute $\mathbb{P}(C_i = c_j|y) = \hat{p}_j$.

3.0.1 Cost Metric

We consider as a cost metric the sum of the path length resulting from the selected policy, l_{path}, and the sum of edge-dependant penalties $c_p[edge]$ (the severity associated with the risk in that location).

$$C(\mathscr{P}) = l_{path} + \sum_{h=0}^{k}(c_{p,h}[edge_h]) \tag{4}$$

where k is the total number of people encountered while reaching the goal.

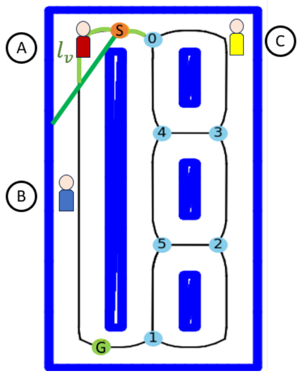


Figure 4: Consider a scenario where 3 people $\{A, B, C\}$ are present in the robot environment. Our assumptions will imply that if the robot is in node S: person A is observable, person B is not observable, and person C's presence is not considered in node S as he/she isn't in an edge of the outgoing star of node S. Person C will instead be observable from nodes $\{0, 3\}$.

3.1 Problem Formulation as an MDP

We consider an unknown number $n \in \mathbb{N} \cap [0, N_p]$ of people in the environment. A complete representation of the problem would require including the possible number of people in each corridor in the state space and solving the problem as a Partially Observable Markov Decision Process (POMDP), where the robot position is known and the human positions are partially observable. However, this method would become quickly intractable. Even without discussing the computational complexities in solving a POMDP [19], consider an environment with 50 corridors. Assuming 20 people, the number of states (just considering the possible human distributions in the corridors) would be: $n_h = \binom{69}{49}$ giving a number of states of the order of 10^{17}.

Therefore, our idea is not to consider the number of people in each corridor as part of the state, but we propose to include in it what would have been the observations of the POMDP, namely the possible observation the robot can make from each intersection point in the map. In this way, we obtain an MDP with a number of states that scales only linearly by increasing the number of intersection points and that does not depend on the number or possible distribution of people in the environment. A Markov Decision Process (MDP) is a mathematical framework used to model decision-making problems in situations where an agent interacts with an environment over a sequence of discrete time steps. The components to fully describe an MDP are:

- **States ($\mathscr{S}$):** A set of possible situations or configurations of the system.
- **Actions ($\mathscr{A}$):** A set of possible decisions that the agent can make.
- **Transition probabilities matrix (P):** It is a three-dimensional matrix that describes the probability of transition from one state to another given a particular action. The element $P[a, s, s']$ represents the probability of transitioning from state s to state s' when action a is taken. P will have dimensions $(a \text{ x } s \text{ x } s)$.
- **Rewards matrix (R):** A matrix that specifies the immediate rewards the agent receives for transitioning from one state to another by taking a specific action. The entry $R[a, s]$ represents the reward associated with the selected action a from state s.

For simplicity, we consider that when the robot is in a node $v \in V$ it makes observations only in the adjacent corridors ($e \in E \mid e = (v, u)$ for some $u \in V$) (Figure 4). Suppose a

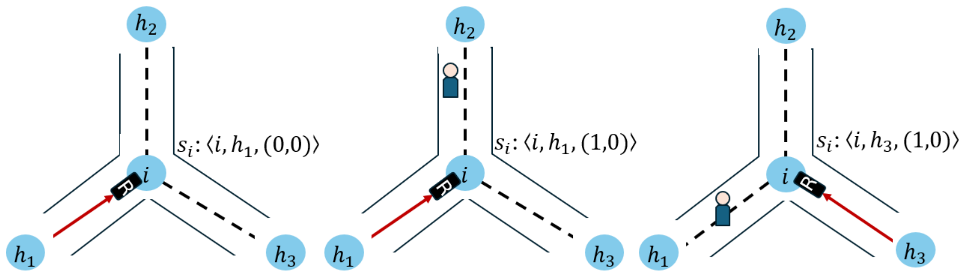


Figure 5: Examples of different states the system can be when the robot is in node i.

person is in $l_v(v,e)$, the visible part of the corridor that the robot is observing from node v (there are no obstacles between the human position and that of the robot). In that case, it can detect the person correctly (Figure 4). Moreover, to avoid a loop as in Figure 3 can be generated by simply moving back and forward on the same edge, we impose that the robot can change route at intersection points, but never turn back where it came from.

3.1.1 States

The state includes the intersection point at which the robot is making the decision and the incoming node from which it comes from. To achieve a decision planner that takes into account the possibilities and quality of replanning routes, we then propose to add as part of the state a tuple representing the possible observations the robot can make in the current node. Observation "1" means that the robot detects at least one person on the observed edge of the homotopies graph, and observation "0" means that no people are seen in the corridor.

The generic state of node i, with predecessor i_p, and observations $(i_o^1, i_o^2 ...)$ can be formally expressed as:

$$\mathscr{S}_i = \langle i, i_p, (i_o^1, i_o^2 ...)\rangle \tag{5}$$

For example, the states associated with node i having three incoming nodes $h_i, i = 1, 2, 3$ are:

$$\begin{aligned} i : \{&\langle i, h_1, (0,0)\rangle, \langle i, h_1, (0,1)\rangle, \langle i, h_1, (1,0)\rangle, \langle i, h_1, (1,1)\rangle, \\ &\langle i, h_2, (0,0)\rangle, \langle i, h_2, (0,1)\rangle, \langle i, h_2, (1,0)\rangle, \langle i, h_2, (1,1)\rangle, \\ &\langle i, h_3, (0,0)\rangle, \langle i, h_3, (0,1)\rangle, \langle i, h_3, (1,0)\rangle, \langle i, h_3, (1,1)\rangle\}. \end{aligned}$$

Where i is the current node, h_1, h_2, h_3 are the possible predecessors of node i, the combination of 0 and 1 are the possible observations the robot can make on the two remaining edges (not the predecessor). The elements of the observation vector are ordered by the indices of the neighboring nodes, starting from the smallest to the largest. Some examples are reported in Figure 5.

The number of states will be for each node $u_i 2^{(u_i-1)}$, where u_i is the number of outgoing edges that refer to the node. This applies to all nodes except for the start and goal. For the goal, we will only have one state, in which the process remains with probability 1 after the system reaches the goal. As the start and goal are added by identifying the edge on the homotopy classes graph on which they lie, they are each always connected to exactly two other nodes. For the start node, we will not have a predecessor node, so we will only have the four states $s : \{\langle s, (0,0)\rangle, \langle s, (0,1)\rangle, \langle s, (1,0)\rangle, \langle s, (1,1)\rangle\}$ associated to the two edges to which the start node is connected. The total number of states will be $n_s = \sum_{i=1}^{N}(u_i 2^{u_i-1}) + 5$, where N is the number of nodes in the homotopies graph,

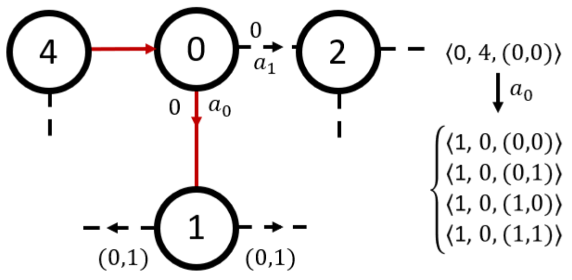


Figure 6: Possible next states by taking action a_0 from state $\langle 0, 4, (0, 0)\rangle$.

excluding the start and goal. In this way, the number of states increases only linearly when augmenting the number of nodes (when the map complexity increases). Moreover, it does not depend on the number of people in the environment.

3.1.2 Actions

We consider as actions the choice to go through the different edges belonging to the outgoing star from each node (except from the one that returns to the predecessor node). The actions will not be the same for each state, a_0 would be the action that sends the robot to the minimum index node connected to the current node, a_1 to the second minimum index adjacent node, and so on. For example, in all the states of node “0” with predecessor node “4” in Figure 6 we will have two actions:

$$0 : \{a_0 : \text{“To node 1"}, a_1 : \text{“To node 2"}\}.$$

The total number of actions will be $n_a = \max_{i \in V}\{u_i - 1\}$.

Figure 6 shows all the states the robot may reach by choosing action a_0, starting from state $\langle 0, 4, (0, 0)\rangle$. Once the action is chosen, the next node and the previous node are deterministically known, the uncertainty is in the possible observations the robot will make from the next node.

3.1.3 Transition probabilities matrix

The P matrix encodes the probability of making a tuple of observations at the next node, starting from a node with known tuples of observations. We represent the generic element of P as: $P[a_z, \langle i, i_p, (i_o^1, i_o^2 ...)\rangle, \langle j, j_p, (j_o^1, j_o^2 ...)\rangle]$, where a_z is the considered action, $\langle i, i_p, (i_o^1, i_o^2 ...)\rangle$ is the state corresponding to the current node i, $\langle j, j_p, (j_o^1, j_o^2 ...)\rangle$ is the state we may reach by taking action a_z.

The P computation algorithm is reported in algorithm 1. When we select an action, the next graph node is deterministic, so $P[a_z, \langle i, i_p, (i_o^1, i_o^2, \ldots)\rangle, \langle j, j_p, (j_o^1, j_o^2, \ldots)\rangle]$ would be 0 if $j \neq i(a_z)$ or $j_p \neq i$ (line 6).

When $j = i(a_z)$ and $j_p = i$, $P[a_z, \langle i, i_p, (i_o^1, i_o^2, \ldots)\rangle, \langle j, i, (j_o^1, j_o^2, \ldots)\rangle] = \prod_{e=0}^{u_j} p_o(obs = j_o^e)$ (line 8), where e are all the u_j edges from the node j, p_o is the probability of making a certain observation, and obs $= j_o^e$ is the observation (0 or 1) that the robot will make from node j on that edge. By doing so, we are approximating observations in different edges as independent.

For every edge starting from j, the probability of making an observation equal to 1 is the probability that at least one person is in the visible part of the corridor looking from node j. For every possible number of people in the environment n, the probability that

$obs = 0$ is the probability that all n people are outside the visible region of the selected corridor.

Once the goal is reached, the process remains in the goal state with probability 1, so the probability to reach any other node is 0. (lines 10-11).

Algorithm 1: P Matrix

```
Data: actions, states, $p_o(obs)[node][edge]$
Result: $P(a \times s \times s)$
1  for $a_z \leftarrow 0$ to $actions$ do
2      for $i$ in $graph\ nodes$ do
3          if $i \neq goal$ then
4              for $j$ in $graph\ nodes$ do
5                  if $j \neq i(a_z)$ or $j_p \neq i$ then
6                      $P[a_z, \langle i, i_p, (i_o^1, i_o^2, ..)\rangle, \langle j, j_p, (j_o^1, j_o^2, ..)\rangle] = 0$;
7                  else
8                      $P[a_z, \langle i, i_p, (i_o^1, i_o^2, ..)\rangle, \langle j, j_p, (j_o^1, j_o^2, ..)\rangle] = \prod_{e=0}^{u_j} p_o(obs = j_o^e)$ ;
9          else
10             $P[a_z, \langle i\rangle, \langle i\rangle] = 1$ ;
11             $P[a_z, \langle i\rangle, \langle j, j_p, (j_o^1, j_o^2, \ldots)\rangle] = 0$ ;
```

3.1.4 Rewards matrix

The rewards matrix (R) with dimensions ($s \times a$) will have the following form:

$$R = \begin{bmatrix} -c_{S_0,a_0} & -c_{S_0,a_1} & \cdots & -c_{S_0,a_n} \\ \vdots & \vdots & \ddots & \vdots \\ -c_{S_n,a_0} & -c_{S_n,a_1} & \cdots & -c_{S_n,a_n} \\ 0 & 0 & \cdots & 0 \end{bmatrix}, \tag{6}$$

where c_{S_i,a_j} represents the expected cost of the edge that we will choose from the node corresponding to state S_i if action a_j is selected. The last line, the one associated with the goal state, is 0 because it corresponds to the fact that once the goal is reached, the system remains in a fictitious state with cost 0.

The edge cost if k humans are on it, $c_{k,e}$, according to (4) is:

$$c_{k,e} = l_e + k\, c_p[edge]. \tag{7}$$

The expected edge costs that we need to compute as elements of the R matrix, will be:

$$c = l_e + \sum_{k=0}^{N_p} p(k|obs)\, k\, c_p[edge], \tag{8}$$

where l_e is the length of the path connecting the two nodes, $p(k|obs)$ is the probability that there are k people in the corridor given the observation obs made in the current node, $c_p[edge]$ is the severity coefficient for that edge (see Section 3.2.1).

In our scenario, given a current state and an action, the edge to be traversed is deterministically identified, so the R matrix entries will not depend on the arriving state.

We show explicitly how matrices P and R can be computed in Section 3.2.5. Once the P and R matrices are computed, the *MDP-O* optimal policy can be found by solving the

Bellman equation with classical methods, for example using the Value Iteration algorithm. The Bellman equation provides a recursive decomposition of the value function, which the Value Iteration algorithm uses to update the value of each state until convergence iteratively. We use $\gamma = 0,9999$ as the discount factor to ensure that long-term rewards are well considered.

We refer to this formulation as *MDP-O* (MDP augmented with Observations).

3.1.5 Modifying the Optimal Policy Online

This formulation has undoubted advantages in terms of the tractability of the problem; however, it has some limitations. An MDP, by its inherent property, does not take past history into account (similarly to, e.g., the *reset* property). Therefore, the probability of making certain observations will not depend on the observations already made. This implies that, if the severity of passing in an occupied edge is higher than the expected cost of returning to the same node by a different path, and here making a different observation, the *MDP-O* can underestimate the expected cost of taking that route and perform a loop during the execution of the path (Figure 3).

To account for this phenomenon, the *MDP-O* is solved the first time before starting, and an optimal policy is computed. This policy gives the optimal actions that should be taken from each possible state. Then, during the online execution of the path, the *MDP-O* is run again every time a person is seen, by adjourning the probability of observing someone for that edge as $p(obs = 1|\hat{h}) = 1$ where "$\hat{h}$" represents the event "human seen in the edge". We adjourn P by changing $p(obs)$ for both nodes on that edge, regardless of visibility. Then, we rerun the Value Iteration algorithm and compute the new optimal policy. This strategy is efficient in avoiding loops. However, in the policy computed offline, in the presence of the previously described conditions, the nodes' cost may be underestimated, as the expected cost of moving along the loop and performing a different observation is accounted for instead of passing on the occupied edge.

3.2 Risk-aware routing in a shared dynamic environment

In the following, we present how our approach can be applied to account for the risk associated with human encounters in route selection in a shared dynamic environment. We outline a method to output the best strategy to follow starting from a black-and-white picture of the environment, and some information on human distributions.

The method is divided into four phases:

I. Homotopy classes Graph Computation: in this phase, from the image of the environment, we generate our homotopy classes graph and compute static properties (Section 3.2.1).

II. Adding Start and Goal: in this step, we add the starting point, the goal and connect them as nodes to the previously computed static graph, modifying the properties of the involved edges accordingly.

III. Compute the optimal policy: determining the homotopy class that minimizes the risk of encounters in the environment corresponds to finding the "best" policy along the homotopy classes graph. In this phase, we model the problem as an *MDP-O* (Section 3.2.5).

IV. Modifying Optimal Policy Online: as discussed in Section 3.1.5 the produced offline policy may be suboptimal and generate loops in certain conditions due to the reset property. As a consequence, in this phase, the optimal policy is modified online each time a human is detected.

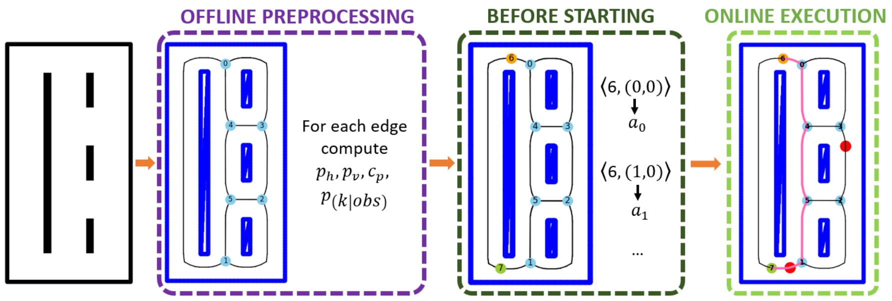


Figure 7: Methodology overview: starting from a black-and-white picture of the environment the homotopy classes graph is generated offline. In this step, we also compute graph edge properties and expected costs. When a start and goal are selected, the graph and its properties are locally modified. The *MDP-O* resolution algorithm is run to find the optimal policy. Online, the robot moves with the precomputed policy, which is modified online when needed.

The methodology overview is reported in Figure 7.

3.2.1 Homotopy classes graph

To compute and characterize the homotopy classes graph, we first compute the edges of the generalized Voronoi diagram of the static map. Given a set $O = o_1, o_2, .., o_n$ of geometrical objects such as lines and polygons (representing walls and obstacles in our scenario) a generalized Voronoi diagram is a partition of the plane such that each region contains the closest points to a particular geometrical object o_i. A Voronoi edge is the boundary between two regions. We take the Voronoi edges from the generalized Voronoi diagram and use them to create a graph $\mathcal{G} = (V, E)$. To this aim, we select the intersection points of three or more branches of the generalized Voronoi diagram. These intersection points together with the goal and the starting position of the robot will be the nodes in V of the homotopy classes graph. The edges E would result from the chains of the Voronoi edges that connect them. Using this graph, we consider the path with the lowest static collision risk (i.e., the path with the highest distance from obstacles) for each homotopy class. We will use this single candidate to compute all the static properties of the corridor (such as length and visibility) that will be needed to model the problem.

We start by computing static properties depending on the homotopy's graph geometry.

3.2.2 p_h - Human Probability

The human probability, p_h, represents the relative probability that a person is on an edge, instead of that on the others. For example, by assuming that people are uniformly distributed, we obtain:

$$p_h[e] = \frac{l_e}{l_t} \tag{9}$$

where l_e is the homotopies graph edge length, and l_t is the length sum of all edges. However, any probability distribution can be used as long $\sum_{e \in E} p_h[e] = 1$.

3.2.3 p_v - Visibility

When there are people on the edge, the probability of seeing them is equal to the probability that they are on the percentage of the edge visible from the node from which the robot makes the observation. A point from the edge is considered visible from the node when the line connecting them does not encounter any static obstacle (see Figure 4 for $l_v[S][(S,G)]$ visualization). For example, in the case of uniform human presence probability along the corridor, the probability that a single person is in the visible region of the homotopy class would be:

$$p_v[v][e] = \frac{l_v(v,e)}{l_e}. \tag{10}$$

l_v is computed by discretizing the Voronoi chains and by checking if each edge point is visible from the adjacent nodes.

3.2.4 c_p - Severity Coefficient

The severity coefficient c_p is the penalty we add in our metric each time an encounter between the robot and a human happens. To quantify this variable, we consider the consequences of a human-robot encounter. The possible consequences are:

- The robot must slow down or stop for safety reasons.
- If there is enough space the robot can locally replan a new route in the current homotopy class.
- To pass, the robot has to enter the human personal space, causing stress and mental fatigue [22].
- If the robot is transporting a huge amount of load, the interaction can be perceived as not safe.

All these undesired events depend on the human-robot distance. Therefore, the performance and the possibility of performing a safe replanning are strictly connected with the width of the corridor.

The severity of encountering a human will not be the same for all corridors on the map. It will depend on the robot's size, its load, and the geometric width of the corridor.

We consider a linear dependence of the severity coefficient with the corridor width and propose the following formulation:

$$c_p = \max(0, \min(-k_1(w_c - w_r) + k_2, k_3)) \tag{11}$$

where w_c is the corridor's minimum width, w_r is the robot's width, and the coefficients are chosen according to the specific application scenario (robot mass, robot target speed, human familiarity with the scenario) so that the severity coefficient for each corridor spans between 0 and k_3. w_c is computed from the generalized Voronoi diagram by checking the distance to the nearest fixed obstacle on discretized points along the Voronoi graph. This severity criterion encompasses both the time the robot will incur by decelerating or halting to ensure a smooth and risk-free local re-planning, as well as the psychological strain experienced by the human operator.

3.2.5 MDP-O Formulation

We now first compute P and R matrices by assuming people are static during the robot route execution (e.g., they are working in specific locations of the environment), and then extend to a scenario where some people may enter the corridor while the robot is already moving in it. For simplicity, we approximate static and dynamic people in the environment modeled by two known independent distributions. We consider an encounter to happen when a person is already in the chosen corridor and when a human enters the corridor in the opposite direction while the robot moves in it. In the following, we denote the probabilities relative to the static people distribution as $p_s(\cdot)$, the ones relative to the moving people distribution as $p_m(\cdot)$, and the probabilities accounting for both as $p(\cdot)$.

For example, in case the presence of static humans in all the environment follows a binomial distribution, given n_{mean} mean number, N_p maximum number, and 0 minimum number of people in the environment, the probability that there are n static people is:

$$p_n = \binom{N_p}{n} \left(\frac{n_{mean}}{N_p}\right)^n \left(1 - \frac{n_{mean}}{N_p}\right)^{N_p - n}. \tag{12}$$

3.2.6 P matrix

To explicitly compute matrix P with algorithm 1 we need to compute $p_o(obs)$. The probability that all n people are out ($obs = 0$), or at least one is in ($obs = 1$) the visible region of the corridor is:

$$p_{s,o}(obs) = \sum_{n=0}^{N_p} p_n((1 - p_v p_h)^n)^{1-obs}(1 - (1 - p_v p_h)^n)^{obs}. \tag{13}$$

3.2.7 R matrix

To compute R matrix, according to (8), we need to explicitly compute $p(k|obs)$. By applying the Bayes rule, $p(k|obs)$ becomes:

$$p(k|obs) = \frac{p(obs|k)p_k(k)}{p_o(obs)},$$

where $p_k(\cdot)$ represents the probability of encountering k people while passing through the corridor.

The probability of making a certain observation from a node on an adjacent edge is the same already computed in (13). The probability of making an observation of 0 or 1 when in the corridor there are k people, $p_s(obs|k)$, depends on the visibility probability computed in (10). If there are no people ($k = 0$) $p_s(obs = 0|k = 0) = 1$. Otherwise, ($k > 0$), $p_s(obs = 0|k)$ is the probability that all persons are in the not visible portion of the corridor. $p_s(obs = 1|k)$ is the probability that at least one person is visible. In general form, it would be:

$$p_s(obs|k) = (1 - p_v)^{(k(1-obs))}(1 - (1 - p_v)^k)^{obs}. \tag{14}$$

The probability that there are k people in the corridor would depend on human presence probability (9). For each possible number of people in the environment n, the probability that k of n humans are in the corridor is

$$p_{s,k}(k) = \sum_{n=0}^{N_p} p_n \binom{n}{k} (p_h)^k (1 - p_h)^{(n-k)}. \tag{15}$$

3.2.8 Extension in presence of Moving People

We model the presence of moving people alongside stationary ones by considering them as independent distributions. Let us denote p_e as the probability that in a time unit (e.g., $1s$), a person enters a corridor in the direction opposite to the robot's movement. Calling T the time (rounded to the nearest integer) the robot takes to traverse the corridor, the probability p_m that, while the robot is in the corridor, k people will enter the corridor itself is:

$$p_{m,k}(k) = \binom{T}{k} p_e^k (1 - p_e)^{T-k}. \tag{16}$$

The probability of making a certain observation from a node does not change if some persons enter the corridor, as the observation is instantaneous and the visible portion of the corridor is the same. The P matrix will therefore maintain the same formulation

$$p_o(obs) = p_{s,o}(obs). \tag{17}$$

To compute the total probability that the robot will encounter k people in the corridor, we notice that of these k people, a certain number, j will be static and already in the corridor when the robot chooses it (in the visible part or not), and $k - j$ would not be yet in the corridor, but will enter while the robot is traversing the corridor. This probability can be written as:

$$p_k(k) = \sum_{j=0}^{k} p_{s,k}(j) p_{m,k}(k - j). \tag{18}$$

The probability of making a certain observation assuming that we will encounter k people in the corridor $p(obs|k)$ takes now into account the fact that, as we modeled the problem so far, the $(k - j)$ people not yet in the corridor are certainly not visible. The probability that if the robot encounters k people, j are static is:

$$p(j|k) = \frac{p(k|j)p(j)}{p(k)} = \frac{p_{m,k}(k - j)p_{s,k}(j)}{p_k(k)}. \tag{19}$$

As a consequence, we have:

$$p(obs|k) = \sum_{j=0}^{k} p(j|k) p_s(obs|j). \tag{20}$$

To computed P and R matrices, we solve the *MDP-O* with the Value Iteration algorithm as described in Section 3.1.

3.3 Validations

We tested our problem modeling in three different environments with increasing map complexity. Each map highlights different aspects of our formulation. The first and the second maps are used to compare our approach with more classical algorithms, and the effects of varying the severity coefficients c_p, human density, and moving people's presence in the overall performances.

To compare our approach with reactive strategies, we define an expected cost graph $\mathcal{G}_{EC}(V, E)$, having the same nodes and edges of $\mathcal{G}$, but the weight of each edge is given by:

$$c_e = l_e + \sum_{k=0}^{N_p} p_k(k)\, k\, c_p[edge]. \tag{21}$$

We can compute the path with the minimum expected cost, approximating edge costs as independent, for example, by applying an A* algorithm to $\mathcal{G}_{EC}$.

We also use, for comparison, a modified A* algorithm that every time a $\mathcal{G}_{EC}$ node is reached (including S), the cost of its outgoing edges star are updated as:

$$c'_e = l_e + \sum_{k=0}^{N_p} p(k|obs)\, k\, c_p[edge], \tag{22}$$

Table 1: Small Map: Costs, Encounters and Lengths for the paths obtained with 10000 simulations, $p_e = 0$ and $hum_{info} = (0, 2, 10)$

	$c_p = 5$			$c_p = 20$			$c_p = 40$		
	Cost	Encounters	Length	Cost	Encounters	Length	Cost	Encounters	Length
MDP-O	**20.59 ± 0.06**	0.42 ± 0.01	18.50 ± 0.01	**25.85 ± 0.20**	0.21 ± 0.01	21.65 ± 0.06	**29.78 ± 0.37**	0.20 ± 0.01	21.64 ± 0.06
A*	21.07 ± 0.07	0.55 ± 0.01	18.32 ± 0.00	29.70 ± 0.29	0.57 ± 0.01	18.32 ± 0.00	40.86 ± 0.57	0.56 ± 0.01	18.32 ± 0.00
A* mod	**20.59 ± 0.06**	0.42 ± 0.01	18.50 ± 0.01	26.34 ± 0.24	0.38 ± 0.01	18.80 ± 0.03	33.89 ± 0.48	0.38 ± 0.01	18.81 ± 0.03

and the new minimum-expected cost path is computed. Once an edge is traversed, it is removed from the expected costs graph. To study the different effects of the c_p choice for the first two maps, we use different values of c_p, fixed for all the corridors. Note that, as we consider people as uniformly distributed in the environment, the path with the minor probability of encountering a person, and so, in case of fixed c_p, the one with the minimum expected cost, is also the minimum length path. The third map, instead, is used to study the scalability of our approach, and the effects of having different severity coefficients on portions of the environment. For the third map, the severity coefficient for each corridor is found using (11). The framework has been implemented in Python on a i7 2000 MHz 12-th Gen Intel processor with 16 GB of RAM memory.

To obtain the generalized Voronoi diagram from the environment map picture we use the code available in [12] [1]. The *MDP-O* is solved using the Value Iteration algorithm with the Markov Decision Process Toolbox for Python [2].

For each simulation, a random number n of people between 0 and N_p are considered as static in the environment (e.g., performing a task in a certain position), with n_{mean} the mean number of static humans that will be present in the environment. From now on, this information will be expressed as $hum_{info} = (0, n_{mean}, N_p)$. The n extracted people are randomly located along the homotopy classes graph with uniform probability. For simulations where we consider the presence of moving people, to compute the probability that k humans will enter the corridor while the robot is in it, we use (16), by taking p_e constant and equal for all warehouse corridors. Robot velocity is considered 1 m/s, so T is each corridor's integer rounded path length. The performances of the three algorithms are compared for each random human configuration.

For each scenario varying hum_{info}, p_e, c_p we perform 10000 simulations when only static people presence is considered, and 20000 when we account for people entering the corridor while the robot traverses it (in this case, the degree of freedom in each simulation is higher).

Unless otherwise specified the results in the tables are presented as the sample mean $\pm 1.96SE$, where SE is the standard error of the mean. This interval corresponds to a 95% confidence level, assuming the sampling distribution of the mean is approximately Gaussian, in accordance with the Central Limit Theorem.

3.3.1 Small Map - Replanning Possibilities and Visibility

In this map, we consider fixed start and goal positions. The path associated with the minimum expected cost according to the A* algorithm is the one that directly connects the start and end nodes. However, this path does not present escaping possibilities when a person is in it. Moreover, from the start node, only a portion of the corridor is visible, so even if the observation is 0, there is still a high probability of making an encounter. The map has eight nodes, six from the homotopy classes graph, plus the start and goal, for a total of 77 states and 2 actions.

[1] https://github.com/ross1573/generalized_voronoi_diagram
[2] https://pymdptoolbox.readthedocs.io/en/latest/

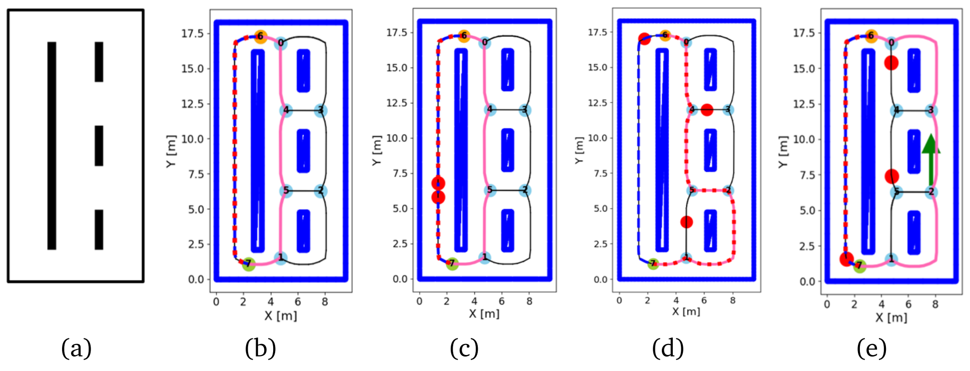


Figure 8: 8a): Map of the small warehouse. **8b-8e)**: Examples of path results in the small map from a fixed starting position (in orange) to a fixed goal (in green). Severity: $c_p = 30$, $hum_{info} = (0, 2, 5)$, $p_e = 0.005$. Path colors: MDP-O (Pink), A* (Blue), A* mod (Red). Static people are presented as red dots, moving people entering the corridor while the robot is in it are presented as a green arrow indicating the moving direction.

Table 2: Small Map: Costs, Encounters and Lengths for the paths obtained with 10000 simulations, $p_e = 0$ and $c_p = 30$.

	$hum_{info} = (0,1,3)$			$hum_{info} = (0,2,5)$			$hum_{info} = (0,4,8)$		
	Cost	Encounters	Length	Cost	Encounters	Length	Cost	Encounters	Length
MDP-O	**22.78 ± 0.16**	0.07 ± 0.01	20.83 ± 0.05	**27.52 ± 0.27**	0.19 ± 0.01	21.73 ± 0.06	**40.73 ± 0.46**	0.60 ± 0.01	22.67 ± 0.06
A*	26.50 ± 0.29	0.27 ± 0.01	18.32 ± 0.00	34.46 ± 0.40	0.54 ± 0.01	18.32 ± 0.00	52.19 ± 0.58	1.13 ± 0.02	18.32 ± 0.00
A* mod	24.10 ± 0.25	0.19 ± 0.01	18.49 ± 0.02	29.71 ± 0.34	0.37 ± 0.01	18.76 ± 0.03	42.19 ± 0.49	0.76 ± 0.02	19.48 ± 0.05

The map is presented in Figure 8a, together with its homotopy classes graph. It has dimensions 10 × 17.5 m. Some paths, resulting for the three algorithms with different conditions and human configurations, are reported in Figure 8.

In Table 1 we report the results obtained with 10000 simulations, by varying the severity coefficient c_p for $hum_{info} = (0, 2, 5)$ and $p_{mov} = 0$. For a low severity value ($c_p = 5$), the algorithm derived from the *MDP-O* formulation behaves like the *A* mod* algorithm, passing on the left side of the big wall (unless a person is present in the visible part of the corridor). In this scenario, the paths resulting from the two algorithms are the same. When severity increases, *MDP-O policy* is to pass in the central corridor, even if the length is higher. This increases the possibility of avoiding an encounter, resulting, on average, in a lower cost.

In Table 2, we present the results of 10000 simulations, by varying hum_{info} parameter. The mean cost obtained using the *MDP-O* policy is always slightly better than the cost obtained with the reactive strategy of the *A* mod* algorithm. In Table 3 we show the results in the presence of moving people in the environment, performing 20000 simulations for each scenario. For $p_e = 0.05$, there is a high probability that people enter the corridors while the robot is traversing them. This means that the value of observations is reduced,

Table 3: Small Map: Costs, Encounters and Lengths for the paths obtained with 20000 simulations, $c_p = 30$ and $hum_{info} = (0, 2, 5)$

	$p_e = 0.005$			$p_e = 0.01$			$p_e = 0.05$		
	Cost	Encounters	Length	Cost	Encounters	Length	Cost	Encounters	Length
MDP-O	**31.09 ± 0.24**	0.31 ± 0.01	21.71 ± 0.04	**34.24 ± 0.28**	0.42 ± 0.01	21.69 ± 0.04	**58.22 ± 0.47**	1.31 ± 0.02	18.78 ± 0.02
A*	37.71 ± 0.32	0.65 ± 0.01	18.32 ± 0.00	40.57 ± 0.34	0.74 ± 0.01	18.32 ± 0.00	62.57 ± 0.49	1.48 ± 0.02	18.32 ± 0.00
A* mod	32.84 ± 0.28	0.47 ± 0.01	18.78 ± 0.02	35.75 ± 0.30	0.56 ± 0.01	18.80 ± 0.02	**58.22 ± 0.47**	1.31 ± 0.02	18.78 ± 0.02

and so, for this map, is more convenient to pass as first-choice on the left side of the wall, resulting in the same policy of the *A* mod* algorithm. In the other tested scenarios, the *MDP-O* approach performs better.

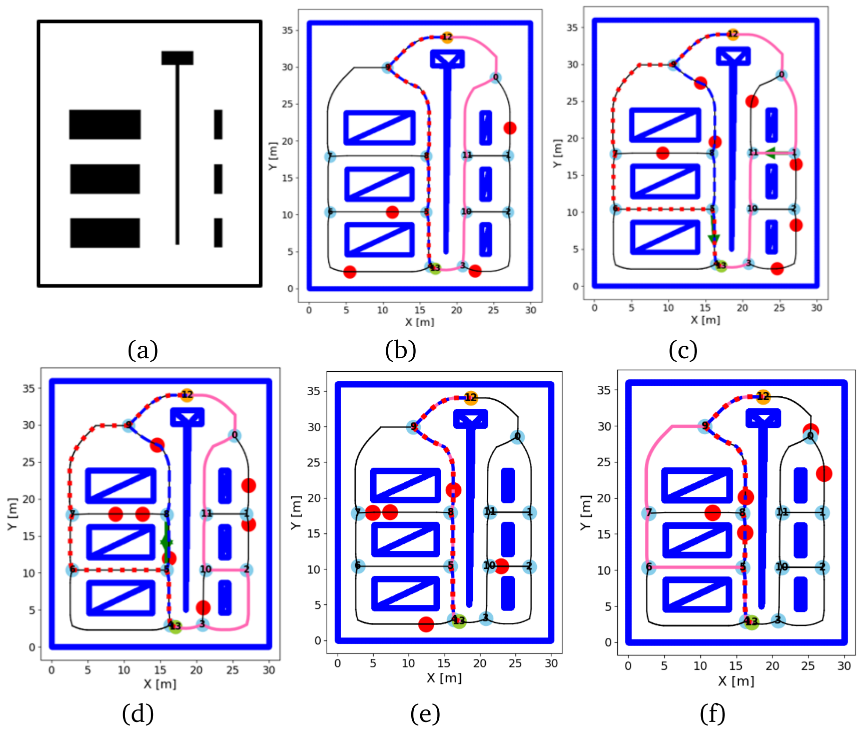


Figure 9: 9a): Map of the medium warehouse. Path colors: MDP-O (Pink), A* (Blue), A* mod (Red). Static people are presented as red dots, the entrance of moving people as green arrows. **9b-9d)**: Examples of path results in the medium map from a fixed start position (in orange) and goal (in green). Severity: $c_p = 30$, $hum_{info} = (0, 6, 12)$, $p_e = 0.005$. **9e)** For $c_p = 5$, $hum_{info} = (0, 6, 12)$, $p_e = 0$ the MDP-O executes the same policy as the A* mod algorithm. **9f)** For $c_p = 20$, $hum_{info} = (0, 6, 12)$, $p_e = 0$ the A* mod algorithm does not replan even if a human is seen in (9,8) corridor.

3.3.2 Medium Map - Cost of the Alternative Routes

Within this map, we test the ability of our approach to account for the expected cost of alternative routes. The map is shown in Figure 9, along with some path obtained by varying human configurations and parameters. The map has 12 nodes plus the start and goal, for a total of 149 states and 2 actions. The start and goal are fixed and the single path associated with the minimum expected cost is the one passing left to the long obstacle in the middle. However, by passing right, the length of the alternative routes in case the first choice corridor is occupied, is minor. The environment has dimensions 30 × 35 m, and c_p is kept constant for all the map corridors. In Tables 4, 5 are reported the results for 10000 simulations, by varying human densities and the severity coefficient. In Table table 6 the results of 20000 simulations, obtained by varying the entering probability p_e, are presented.

In this scenario, the *MDP-O* algorithm performs as the *A* mod* only for the scenario with $c_p = 5$, in the other case studies it leads to a smaller mean cost. Depending on the selected parameters, *A* mod* algorithm may choose to remain on the *A** path even if a person is seen, as the estimated expected cost of the best alternative route is higher than

Table 4: Medium Map: Costs, Encounters and Lengths for the paths obtained with 10000 simulations, $p_e = 0$ and $hum_{info} = (0, 6, 12)$

	$c_p = 5$			$c_p = 20$			$c_p = 40$		
	Cost	Encounters	Length	Cost	Encounters	Length	Cost	Encounters	Length
MDP-O	**45.37 ± 0.09**	0.94 ± 0.02	40.70 ± 0.03	**56.18 ± 0.31**	0.53 ± 0.01	45.49 ± 0.11	**67.10 ± 0.60**	0.53 ± 0.01	46.08 ± 0.12
A*	46.20 ± 0.10	1.20 ± 0.02	40.18 ± 0.00	64.76 ± 0.41	1.23 ± 0.02	40.18 ± 0.00	88.64 ± 0.82	1.21 ± 0.02	40.18 ± 0.00
A* mod	**45.37 ± 0.09**	0.94 ± 0.02	40.70 ± 0.03	59.13 ± 0.37	0.89 ± 0.02	41.25 ± 0.06	72.47 ± 0.67	0.71 ± 0.02	44.02 ± 0.12

Table 5: Medium Map: Costs, Encounters and Lengths for the paths obtained with 10000 simulations, $p_e = 0$ and $c_p = 30$

	$hum_{info} = (0, 2, 5)$			$hum_{info} = (0, 6, 15)$			$hum_{info} = (0, 10, 15)$		
	Cost	Encounters	Length	Cost	Encounters	Length	Cost	Encounters	Length
MDP-O	**46.66 ± 0.21**	0.09 ± 0.01	43.87 ± 0.09	**61.53 ± 0.46**	0.53 ± 0.01	45.57 ± 0.11	**82.66 ± 0.67**	1.21 ± 0.02	46.21 ± 0.11
A*	52.52 ± 0.36	0.41 ± 0.01	40.18 ± 0.00	76.98 ± 0.63	1.23 ± 0.02	40.18 ± 0.00	101.09 ± 0.78	2.03 ± 0.03	40.18 ± 0.00
A* mod	48.03 ± 0.27	0.20 ± 0.01	42.03 ± 0.10	65.42 ± 0.51	0.71 ± 0.02	44.22 ± 0.13	86.16 ± 0.70	1.38 ± 0.02	44.70 ± 0.13

the cost of making the encounter on the first-choice path (Figure 9f). However, this is not necessarily the best option, as more replanning possibilities are present after the first rerouting edge $(9, 7)$ is taken. In the scenario represented in the picture, *MDP-O* policy takes the path on the left from 12 as $(12, 0)$ is occupied and then decides to take $(9, 7)$ to avoid the person in $(9, 8)$.

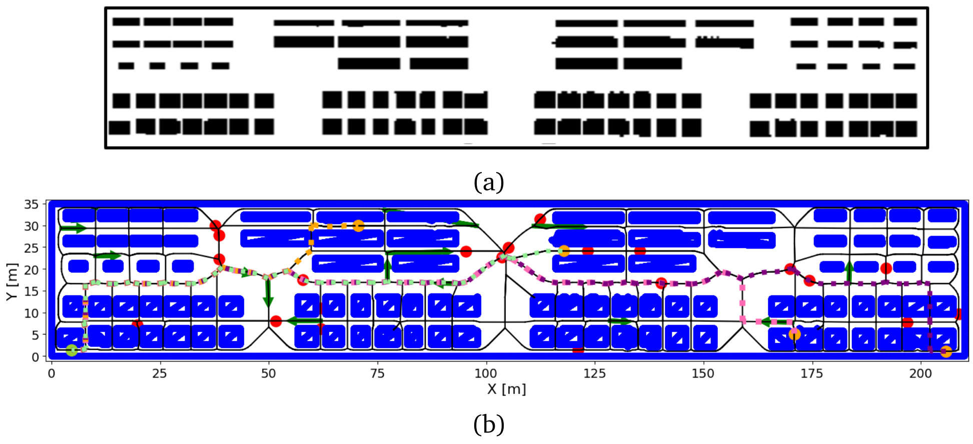


Figure 10: 10a): Map of the big warehouse. **10b)**: Path results in the big map with random starting positions (in orange) and a fixed goal point (in green). Different path colors represent different paths founded by the MDP-O policy changing the starting position in the same human configuration. Static people are presented as red dots, the entrance of moving people as green arrows. Results obtained for $hum_{info} = (0, 20, 30)$, $p_e = 0.005$.

Table 6: Medium Map: Costs, Encounters and Lengths for the paths obtained with 20000 simulations, $c_p = 30$ and $hum_{info} = (0, 6, 12)$

	$p_e = 0.005$			$p_e = 0.01$			$p_e = 0.05$		
	Cost	Encounters	Length	Cost	Encounters	Length	Cost	Encounters	Length
MDP-O	**68.29 ± 0.38**	0.76 ± 0.01	45.53 ± 0.08	**75.14 ± 0.43**	0.99 ± 0.01	45.49 ± 0.08	**128.86 ± 0.69**	2.88 ± 0.02	42.36 ± 0.03
A*	82.77 ± 0.47	1.42 ± 0.02	40.18 ± 0.00	88.73 ± 0.51	1.62 ± 0.02	40.18 ± 0.00	138.07 ± 0.72	3.26 ± 0.02	40.18 ± 0.00
A* mod	71.74 ± 0.41	0.92 ± 0.01	44.10 ± 0.09	78.75 ± 0.45	1.16 ± 0.01	44.07 ± 0.09	131.10 ± 0.70	3.01 ± 0.02	40.68 ± 0.02

Table 7: Computational times for the case study maps, 100 trials.

	$time[s]$		
	Before Starting	MDP-O Resolution	Online Modification
Small Map	$10^{-2}(2.5 \pm 0.7)$	$10^{-4}(4.0 \pm 1.2)$	$10^{-3}(1.3 \pm 0.1)$
Medium Map	$10^{-1}(1.6 \pm 0.1)$	$10^{-4}(7.8 \pm 7.6)$	$10^{-3}(7.0 \pm 4.3)$
Big Map	$10^{0}(1.5 \pm 0.3)$	$10^{-2}(3.2 \pm 1.3)$	$10^{-1}(2.7 \pm 0.8)$

3.3.3 Big Map - Scalability and Severity

In this case study, we test our approach on a complex map of big dimensions. The picture of the environment is taken from [8], and represents a real warehouse (real dimensions are not provided). We consider the map to have dimensions 210×35 m. The map has 157 nodes, for a total of 2499 states, and 3 actions. We use this map to study the scalability of the proposed approach and how severity influences the policies starting from various starting positions. Severity is static and obtained using (11). The coefficients are chosen so that if the corridor is wider than $w_r + 5\,m$ the severity is 0 if it is narrower than $w_r + 1\,m$ the severity is 50. In this way, we have: $k_1 = 12.5, k_2 = 62.5, k_3 = 50$. We take $w_r = 1\,m$.

In each simulation, a random starting position is chosen, the start and goal are added locally modifying the graph, the properties of the newly added edges are computed, and the *MDP-O* optimal policy is found. Figure 10 shows the map picture and an example of path obtained in previously described conditions, starting from different starting positions in the environment (in the same randomly extracted human presence configuration). We can see that the *MDP-O* algorithm prefers slightly longer paths that include wider corridors, where the severity, in case an encounter happens, is lower.

Table 7 reports the computational times for the *MDP-O* formulation for the Before Starting, *MDP-O* resolution, and online modification parts, for the three maps. The data are expressed as mean ± standard deviation in this case. The "Before Starting" phase includes the local modification of the static graph and its properties to include the start and goal and the first *MDP-O* solving (not the offline preprocessing phase). For the small and medium maps adding the start and goal to the map is also done by randomly selecting a different start in each simulation. Most of the time is spent computing the visibility property for the modified edges, so it is strictly dependent on the length of the edge to split. The "*MDP-O* Resolution" column highlights the computational time needed to solve the *MDP-O* using the Value Iteration algorithm. "Online Modification" indicates the time required to modify the policy online by adjourning the P matrix and rerun the *MDP-O* online when a human is seen.

4 Risk-aware local planning and control

At the local level, the robot must make rapid decisions to adjust its path, ensuring safe, smooth, and easily interpretable movements. To address uncertainty in coordination with the MPC planner, we propose to leverage two key components:

- Risk maps, which highlight nearby spatial risk factors based on the current level of uncertainty.
- An active sensing module, which guides the robot's movement to actively reduce uncertainty.

Note that, this level while avoiding collisions does not formally guarantee human safety.

Human safety is explicitly managed by the safety layer developed in T6.4, which ensures safe controls by reducing the velocity when necessary.

4.1 Risk maps

4.1.1 Collision Probability Map

This module aims to construct a collision probability map M using the estimated robot pose, the tracked obstacles, and the uncertainties associated with those measurements. The map assigns to each cell the probability that the robot would collide with at least one obstacle *if* the robot's center were placed in that cell with the current localization uncertainty. The map has fixed dimensions and is centered on the robot's current position, thus giving a local environment collision risk representation.

To ensure accuracy, we consider both the uncertainty of the robot's pose and the uncertainty of the obstacle's position on the map. Although these quantities originate from the same lidar measurements, they are treated as independent because of their distinct sampling processes and the assumption of no systematic errors. Moreover, to adopt a conservative approach, the robot's orientation is not taken into account. Hence, we assume the robot has a circle footprint with a radius of $r_{\text{robot}} = \frac{1}{2}\sqrt{(width\ robot)^2 + (height\ robot)^2}$. We consider the robot's real position described by a 2D normal distribution, having as variance $\sigma^2_{r,x}$ and $\sigma^2_{r,y}$, derived from the covariance matrix of the localization algorithm.

For each obstacle, instead, we consider the estimated position of the centroid and bounding box dimensions derived from a Kalman filter used for objects detection and tracking. Each obstacle position would been associated with a variance denoted as $\sigma^2_{o_i,x}$ and $\sigma^2_{o_i,y}$.

The idea behind the collision probability map is that each cell c_m of the map M is associated with the probability of collision between the robot and obstacles in the event that the center of the robot is located in c_m.

Consider the robot with center in $c_m = (X_r, Y_r)$ and an obstacle O_i with center $c_{o_i} = (X_{o_i}, Y_{o_i})$, and dimensions $(\hat{w}_i, \hat{h}_i)$. We set:

$$\Delta X = X_r - X_{o_i}, \quad \Delta Y = Y_r - Y_{o_i},$$

both treated as independent Gaussian random variables:

$$\Delta X \sim \mathcal{N}\big(\mu_{\Delta x}, \sigma^2_{\Delta x}\big), \quad \Delta Y \sim \mathcal{N}\big(\mu_{\Delta y}, \sigma^2_{\Delta y}\big),$$

where $\sigma^2_{\Delta x} = \sigma^2_{r,x} + \sigma^2_{o_i,x}$, $\sigma^2_{\Delta y} = \sigma^2_{r,y} + \sigma^2_{o_i,y}$, assuming these errors are independent. The means $\mu_{\Delta x}, \mu_{\Delta y}$ are $(X_r - X_{o_i}), (Y_r - Y_{o_i})$.

When measures are deterministic, we have a collision between the robot and the obstacle when:

$$\Big|\Delta X\Big| \le \frac{\hat{w}_i}{2} + r_{x,\text{robot}} \quad \text{and} \quad \Big|\Delta Y\Big| \le \frac{\hat{h}_i}{2} + r_{y,\text{robot}},$$

where $r_{x,\text{robot}}$ and $r_{y,\text{robot}}$ represent the projections of the robot's radius along the x and y axes, respectively, determined by the relative distance between the cell c_m and the obstacle's center c_{O_i}. So, in the presence of uncertainty, the collision probability for each obstacle is

$$p_{\text{coll},i} = p\big(|\Delta X| \le A_x\big) \times p\big(|\Delta Y| \le A_y\big),$$

where

$$A_x = \frac{\hat{w}_i}{2} + r_{x,\text{robot}}, \quad A_y = \frac{\hat{h}_i}{2} + r_{y,\text{robot}}.$$

For a 1D Gaussian variable $\Delta X \sim \mathcal{N}(\mu_{\Delta x}, \sigma^2_{\Delta x})$, the probability that $|\Delta X| \leq A_x$ is:

$$p\big((|\Delta X| \leq A_x\big) = \Phi\Big(\tfrac{A_x - \mu_{\Delta x}}{\sigma_{\Delta x}}\Big) - \Phi\Big(\tfrac{-A_x - \mu_{\Delta x}}{\sigma_{\Delta x}}\Big),$$

where $\Phi(z)$ is the CDF of the standard normal distribution, that we can numerically compute as:

$$\Phi(z) = \frac{1}{2}\Big[1 + \operatorname{erf}\Big(\frac{z}{\sqrt{2}}\Big)\Big].$$

While the collision probability for static obstacles is computed directly using their fixed position and uncertainty, dynamic obstacles require trajectory prediction over discrete time steps, with their worst-case occupancy considered in the final map M. For dynamic obstacles $O_{d,i}$, we consider the obstacle occupancy as the union (computed as the maximum for each cell) of the obstacle's estimated position after $k = 0, 1, 2$ seconds:

$$p_{\text{coll},d,i} = \max_{k\in\{0,1,2\}}\Big\{p_{\text{coll},i}\Big\}.$$

If we denote with N the number of obstacles, each with a collision probability $p_{\text{coll},i}$ (computed as above) and considering collision with different obstacles as independent events, the probability that at least one obstacle collides is the union of events:

$$p_{\text{coll}}(c_m) = 1 - \prod_{i=1}^{N}\Big(1 - p_{\text{coll},i}(c_m)\Big).$$

Figure 11 depicts an example of the computation of the probability occupancy map M, considering two static obstacles, $\hat{O}_1$ and $\hat{O}_3$, and a dynamic one, $\hat{O}_2$. Figures 11a-c illustrate M_k with $k = 0$, $k = 1$, and $k = 2$, respectively. In these Figures, the color intensity gradation represents the decreasing probability of each cell being occupied.

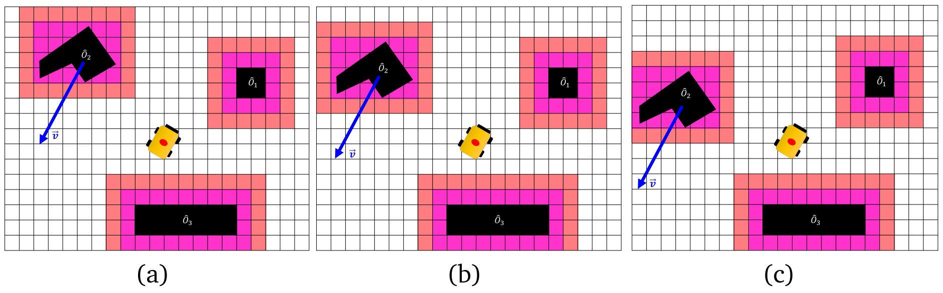


Figure 11: Obstacles Occupancy Probability Map M_k in three different times instances $k = 0, 1, 2$ in an environment with two static obstacles $\hat{O}_1$, $\hat{O}_3$, and a dynamic one $\hat{O}_2$: (**a**) M_0, (**b**) M_1,(**c**) M_2. The transition of colors from dark to light indicates a decreasing occupancy probability value.

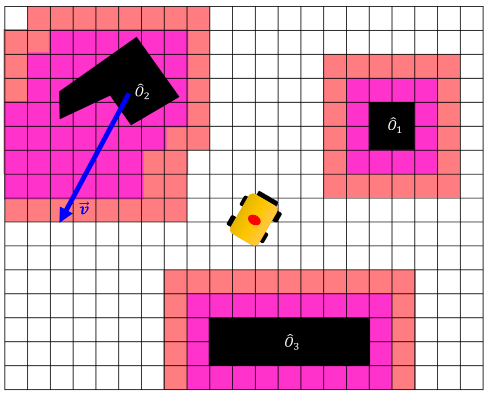


Figure 12: Obstacles Occupancy Probability Map M where each cell is associated with the maximum occupancy probability among the probabilities in M_0, M_1, and M_2.

Figure 12 presents the final map M obtained at the current time $k = 0$, taking into account the object's maximum probability across M_k with $k = 0, 1, 2$ for each cell.

4.1.2 Static risk maps

Static risk maps include a localization failure risk map and additional maps representing known risk areas. The localization risk map is a 2D discrete representation of the environment that encodes the risk of getting localization errors for all traversable regions. We presented it in [5]. This map is based on the notion of alignability [17], defined as the capacity of a given scan coming from a range-based sensor to be aligned with subsequent scans. The higher this capacity is, the lower the risk of localization failure.

Other static risk maps may include known areas where it is preferably for the robot to not enter due to the likely presence of objects difficult to detect, or high congested areas.

4.2 Active sensing layer for task failure mitigation

The successful execution of robotic tasks, particularly in uncertain environments, critically depends on the quality and quantity of the available sensory information. In such scenarios, it becomes imperative to integrate active sensing techniques to mitigate localization and environmental uncertainties, consequently reducing the risk of task failure. Traditional approaches incorporate active sensing within control frameworks by maximizing information acquisition [14, 7], often without explicitly addressing the system stability and safety. In response, we propose a novel approach centered around a new Control Barrier Function (CBF), termed *Information-aware CBF* (I-CBF). This formulation ties safety to the sensory information required for successful task execution. CBFs have found diverse applications, including maintaining Segway vehicle stability [10], enabling adaptive cruise control [2], supporting lane-keeping systems [27], advancing legged robot locomotion [11], and managing multi-robot systems [4, 13]. In our approach, however, CBFs are applied in a novel manner, where we connect the safe set to the minimum information required for task success. When integrated into a Model Predictive Control (MPC) framework, the proposed I-CBF represents an active sensing layer, enabling the robot to dynamically adjust its actions when sensory information is insufficient for accurate task execution. Moreover, by combining I-CBF with Lyapunov Control Functions (LCFs) within the MPC framework, the approach ensures both task stability and the optimization of sensory information acquisition simultaneously.

4.2.1 Optimal Active Sensing Control via I-CBF

In this section, we first introduce our I-CBF and subsequently present its integration within an MPC framework.

Let us consider a robotic system described by the following affine-in-control dynamics

$$\dot{\boldsymbol{q}}(t) = \boldsymbol{f}(\boldsymbol{q}(t)) + \boldsymbol{g}(\boldsymbol{q}(t))(\boldsymbol{u}(t) + \boldsymbol{w}(t)) \tag{23}$$

$$\boldsymbol{z}(t) = \boldsymbol{h}(\boldsymbol{q}(t)) + \boldsymbol{\nu}(t), \tag{24}$$

where $\boldsymbol{q}(t) \in \mathscr{D} \subset \mathbb{R}^n$ and $\boldsymbol{u}(t) \in \mathscr{U} \subset \mathbb{R}^m$ are the system's state and the control inputs, respectively. $\boldsymbol{z}(t) \in \mathbb{R}^p$ represents the sensor outputs (i.e., the measurements available through sensors at time t), and $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$ and $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are locally Lipshitz continuous functions. Finally, $\boldsymbol{h}(\cdot)$ represents the sensor model, $\boldsymbol{\nu}(t) \sim \mathscr{N}(\boldsymbol{0}, \boldsymbol{R}(t)) \in \mathbb{R}^p$ and $\boldsymbol{w}(t) \sim \mathscr{N}(\boldsymbol{0}, \boldsymbol{Q}(t)) \in \mathbb{R}^m$ are white, normally-distributed Gaussian noises with zero means and covariance matrices $\boldsymbol{R}(t)$ and $\boldsymbol{Q}(t)$, respectively.

During task execution, an estimator reconstructs the unknown robot state and the environmental parameters. To ensure safe task execution with an acceptable risk of failure r, the minimum amount of sensory information must meet or exceed a predefined threshold. To quantify the sensory information, we utilize the smallest eigenvalue of the Constructibility Gramian (CG), denoted as $\lambda_{\min}(\mathscr{G}_c(t))$. This metric has been demonstrated in [23] as an effective measure for evaluating information content in nonlinear dynamic systems.

Building on this foundation, we define our I-CBF by extending the framework proposed in [1].

Definition 1 (Information-aware Control Barrier Function)**.** *Let $\mathscr{C} \in \mathscr{D} \subset \mathbb{R}^n$ be the superlevel set of a continuously differentiable function $B : \mathscr{D} \to \mathbb{R}$ defined as follows*

$$B(\boldsymbol{q}) = \lambda_{\min}(\mathscr{G}_c(t)) - s\lambda_{\min}({}^d\mathscr{G}_c(t)) \tag{25}$$

with $\boldsymbol{q}$ the system state and $\lambda_{\min}({}^d\mathscr{G}_c(t))$ the desired minimum amount of information ensuring the task succession with an acceptable risk of failure r which is linked to s by Chi-Square likelihood, i.e.,

$$s = -2\ln\left(\frac{r}{100}\right).$$

Then, B is a CBF if there exists an extended class $\mathscr{K}_\infty$ function $\alpha(B(\boldsymbol{q})) = \gamma B(\boldsymbol{q})$ such that

$$\sup_{\boldsymbol{u}\in\mathscr{U}} \left[L_f B(\boldsymbol{q}) + L_g B(\boldsymbol{q})\boldsymbol{u}\right] \geq -\gamma B(\boldsymbol{q}) \tag{26}$$

with $\gamma > 0$, $L_f = \frac{\partial B(q)}{\partial q}\boldsymbol{f}(\boldsymbol{q})$, and $L_g = \frac{\partial B(q)}{\partial q}\boldsymbol{g}(\boldsymbol{q}, \boldsymbol{u})$

Therefore, any control input $\boldsymbol{u}$ that satisfies (26) ensures the minimum information to complete the task successfully within the acceptable risk level r.

We now show how our I-CBF is integrated into an MPC framework for failure mitigation in task execution.

Consider the dynamics (23) with output (24), and a generic observer that recovers the state estimate $\hat{\boldsymbol{q}}(t)$ of the unknown true state $\boldsymbol{q}(t)$ during motion by exploiting the current sensor measurements $\boldsymbol{z}(t)$. The task-aware optimal active sensing control algorithm that ensures the minimum amount of information required to perform a given task with a predetermined risk is derived by solving the following optimal control problem

Problem 1 (I-CBF MPC)**.** *Given the prediction horizon L, the control input $\boldsymbol{u}(t)$, the predicted trajectory of the nominal system $\tilde{\boldsymbol{q}}$ obtained by applying $\boldsymbol{u}(t)$ starting from the state estimate*

$\hat{\boldsymbol{q}}(t_k)$ provided by the employed observer at time t_k, determine, $\forall t \in [t_k,\, t_{k+L}]$, the optimal control sequence sequence

$$\begin{aligned} \boldsymbol{u}^* = \min_{\boldsymbol{u}\in S(\Delta),\delta} \int_{t_k}^{t_{k+L}} & \|\tilde{\boldsymbol{q}}(\tau) - \boldsymbol{q}_{\mathrm{ref}}(\tau)\|_{\mathbf{Q}_\mathrm{c}} \\ & + \|\boldsymbol{u}(\tau) - \boldsymbol{u}_{\mathrm{ref}}(\tau)\|_{\mathrm{c}} \mathrm{d}\tau \\ & + \|\tilde{\boldsymbol{q}}(t_k + L) - \boldsymbol{q}_f\|_{\mathbf{Q}_\mathrm{f}} + \nu_\delta \delta^2 \end{aligned} \tag{27}$$

s.t.

$$\dot{\tilde{\boldsymbol{q}}}(t) = \boldsymbol{f}(\tilde{\boldsymbol{q}}(t)) + \boldsymbol{g}(\tilde{\boldsymbol{q}}(t))(\boldsymbol{u}(t) + \boldsymbol{w}(t)) \tag{28}$$

$$\tilde{\boldsymbol{q}}(t_k) = \hat{\boldsymbol{q}}(t_k) \tag{29}$$

$$\underline{\boldsymbol{u}} \le \boldsymbol{u}(t) \le \bar{\boldsymbol{u}} \tag{30}$$

$$L_f V(\boldsymbol{q}(t)) + L_g V(\boldsymbol{q}(t))\boldsymbol{u}(t) + \lambda V(\boldsymbol{q}(t)) - \delta \le 0 \tag{31}$$

$$L_f B(\boldsymbol{q}(t)) + L_g B(\boldsymbol{q}(t))\boldsymbol{u}(t) + \gamma B(\boldsymbol{q}(t)) \ge 0 \tag{32}$$

where $S(\Delta)$ is the family of piece-wise constant functions with sampling period Δ, and (27) *is a task-oriented objective function consisting of the state cost $|\tilde{\boldsymbol{q}}(\tau) - \boldsymbol{q}_{\mathrm{ref}}(\tau)\|_{\mathbf{Q}_\mathrm{c}} + |\boldsymbol{u}(\tau) - \boldsymbol{u}_{\mathrm{ref}}(\tau)\|_{\mathrm{c}}$ to measure the deviation from a reference trajectory and the final cost $|\tilde{\boldsymbol{q}}(t_k + L) - \boldsymbol{q}_f|_{\mathbf{Q}_\mathrm{f}}$ to steer the system to the final configuration.* (28) *is the nominal model of the system, which is used to predict the state evolution starting from the initial state* (29), (30) *are the control bounds and* (31) *is a Lyapunov constraint to ensure the stability of the task, with $\lambda > 0$ a design parameter that describes the convergence rate and δ a relaxation variable that ensures well-posedness of the optimal control problem penalized by a weight $\nu_\delta > 0$. To conclude,* (32) *is the safety constraint based on our Information-aware Control Barrier Function.*

4.2.2 Case study

To demonstrate the effectiveness of our approach, we tested it on a unicycle vehicle navigating in obstacle-rich (risky) environments. Consider a unicycle vehicle with the following kinematic equations

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \left(\begin{bmatrix} v \\ \omega \end{bmatrix} + \boldsymbol{w} \right) \tag{33}$$

where (x, y) represents the position of the vehicle in the plane, and θ denotes its heading angle. Here, v is the forward velocity, ω is the angular velocity, and $\boldsymbol{w}$ is white Gaussian white actuation noise with zero mean and covariance matrix $\mathbf{Q}$. We consider only a state estimation scenario, in which an EKF estimates the robot's state using sensor range measurements w.r.t. several landmarks. Let x_{M_i} and y_{M_i} denote the Cartesian coordinates of the i-th landmark, each sensor output is given by $h_i = \sqrt{(x - x_{\mathrm{M}_i})^2 + (y - y_{\mathrm{M}_i})^2}$

The chosen task consist in following a trajectory in an obstacle-rich environment with the possible risk of collision. During task execution, the robot exploits sensor information to avoid collisions with landmarks, safely reaching the desired goal position.

We define the reference trajectory, $\boldsymbol{p}(s)$, as a function of the parameter s with dynamics governed by $\dot{s}(t) = v_s(t)$ and $s(t) \in [0, 1]$. This parametrization yields the trajectory in the form $\boldsymbol{p}(s) = [p_x(s), p_y(s), p_\theta(s)]^\top$. Subsequently, we reformulate the task execution as a tracking problem, where a follower unicycle must track a leader unicycle. The leader unicycle position $\boldsymbol{q}_l(s)$ (which represent $\boldsymbol{q}_{\text{ref}}$ in (27)) starts from the initial configuration $\boldsymbol{q}_l(0) = [p_x(0), p_y(0), p_\theta(0)]^\top$ and moves along the trajectory $\boldsymbol{p}(s)$ with linear velocity $v_l(s)$ and angular velocity $\omega_l(s)$, derived by leveraging the flatness [3] property.

To ensure stability during task execution, we define the Lyapunov candidate as follows

$$V(e) = \frac{1}{2}(e_1^2 + e_2^2) + K(1 - \cos e_3), \qquad K > 0 \tag{34}$$

which is a function of the tracking error $\boldsymbol{e} = [x - x_l, y - y_l, \theta - \theta_l]^T$. By choosing the following control law, representing u_{ref} in (27),

$$\begin{aligned} v_{\text{ref}}(\boldsymbol{e}) &= v_l \cos e_3 - K\lambda_1 e_1 \\ \omega_{\text{ref}}(\boldsymbol{e}) &= \omega_l - \frac{e_2}{K} v_l - \lambda_2 \sin e_3, \end{aligned}$$

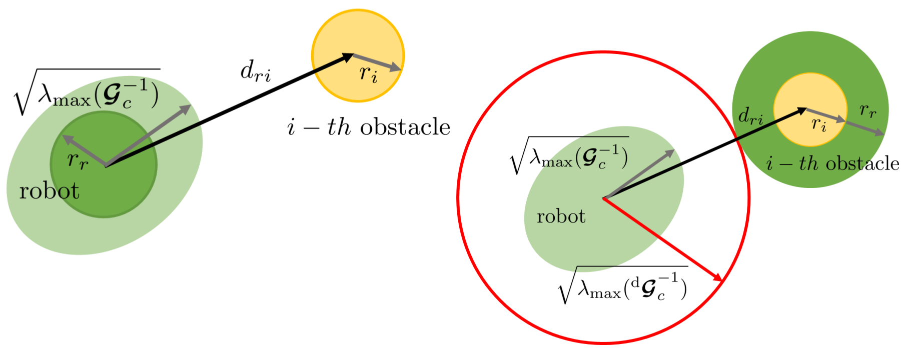
with $\lambda_1, \lambda_2 > 0$, we obtain $\dot{V}(\boldsymbol{e}) \leq 0$ and then by using the Krasowski-Lasalle principle, the G.A.S. of the equilibrium $\boldsymbol{e} \equiv 0$ can be demonstrated [24].

To safely perform task execution, we need to define our I-CBFs for obstacle avoidance. By observing Figures 13a and 13b, safety is guaranteed if all the possible robot positions remain within the desired estimation uncertainty circle (red circle in Figure 13b). As a consequence, our I-CBF is defined as

$$B_{\text{a}}^i(\boldsymbol{q}_{\text{a}}) = \lambda_{\min}(\mathcal{G}_{\boldsymbol{c}}(t)) - s\,(d_{ri}(\boldsymbol{q}) - (r_r + r_i))^{-2}. \tag{35}$$

where $d_{ri}(\boldsymbol{q}) - (r_r + r_o))$, and $d_{ri} = h_i$.

Note that, in this case study, unlike in Problem 1, v_s is introduced as an additional optimization variable. This implies that the velocities of the leader unicycle are also optimized, ensuring the most feasible and efficient task execution.


$\sqrt{\lambda_{\max}(\mathcal{G}_c^{-1})}$
d_{ri}
r_i
r_r
robot
$i - th$ obstacle
d_{ri}
r_r
r_i
$i - th$ obstacle
$\sqrt{\lambda_{\max}(\mathcal{G}_c^{-1})}$
robot
$\sqrt{\lambda_{\max}({}^{\text{d}}\mathcal{G}_c^{-1})}$

(a) Unicycle vehicle state estimation (b) Safety constraint for unicycle vehicle state estimation.

Figure 13: Obstacle avoidance scenarios for unicycle case study. The robot and the i-th obstacle sizes are represented with green and yellow circles, respectively. The estimation error ellipses are drawn in light green for the robot and light yellow for the obstacle. The red circle represents the safety boundary measured via a desired estimation uncertainty function of a desired CG (i.e., ${}^{\text{d}}\mathcal{G}_c$) which delineates the collision-free region for all the robot and obstacles possible positions.

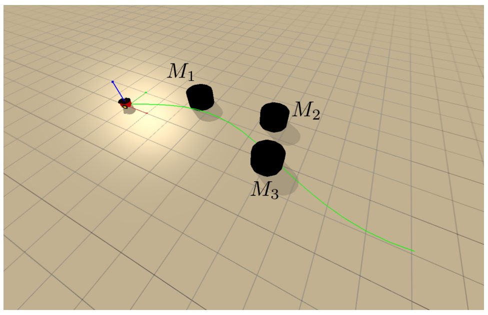


Figure 14: Task for the unicycle case study. The reference path is drawn in light green. The unicycle has to follow the path and avoid the three black cylinders located in the environment.

4.2.3 Statistical validation

In this section, we compare the results obtained by solving Problem 1 with the same problem where: 1) the active sensing component is neglected and CBF constraints (35) is replaced by the following classical inequality constraint for obstacle avoidance $d_{ri} - (r_r + r_i) \geq 0$, 2) the active sensing component is neglected and the CBF is $B^i = d_{ri} - (r_r + r_i)$, 3) the active sensing component is integrated into the controller through a classical inequality constraint, this is equivalent to set (35) equal or greater than zero.

	Mean information	Mean estimation error	Mean task execution time [s]	Mean computational time [s]	Task success rate
MPC without AS without CBF	20.064 ± 0.782	$e_x = 0.060 \pm 0.034$ m $e_y = 0.114 \pm 0.063$ m $\boldsymbol{e_\theta = 0.043 \pm 0.023}$ rad	**31.488 ± 1.053**	0.220 ± 0.004	23.5 %
MPC without AS with CBF	21.051 ± 0.782	$e_x = 0.056 \pm 0.033$ m $e_y = 0.111 \pm 0.061$ m $\boldsymbol{e_\theta = 0.043 \pm 0.023}$ rad	35.633 ± 0.900	0.230 ± 0.007	36 %
MPC with AS without CBF	82.694 ± 7.915	$e_x = 0.043 \pm 0.020$ m $e_y = 0.062 \pm 0.031$ m $e_\theta = 0.043 \pm 0.019$ rad	40 ± 3.738	0.260 ± 0.009	51 %
Information-aware CBF MPC	**389.603 ± 4.493**	$\boldsymbol{e_x = 0.045 \pm 0.021}$ m $\boldsymbol{e_y = 0.039 \pm 0.019}$ m $e_\theta = 0.057 \pm 0.028$ rad	56.996 ± 3.167	**0.157 ± 0.019**	**100 %**

Table 8: Comparison between MPC without active sensing without CBF, MPC without active sensing with CBF, MPC with active sensing without CBF, and MPC with active sensing with CBF (i.e. our methodology with the information-aware CBF). The mean values are computed on the 200 optimal solutions obtained with the different control strategies. The best results are shown in bold. Our methodology outperforms the others providing the highest task success rate, the highest mean information, the smallest mean estimation error along x and y, and the mean computational time. However, our approach also needs more time to find safe solutions, implying a high mean task execution time.

The task is illustrated in Figure 14 and involves following the reference trajectory, depicted in light green, from the starting point to the endpoint while avoiding the three black obstacle. The robot's initial configuration, $\boldsymbol{q}_0$, is aligned with the starting point of the reference trajectory. The task is performed under the assumption of a failure risk $r = 5\%$

For the evaluation, we have compared the estimation performances of the EKF (used an observer) in terms of the average minimum amount of information, i.e., $\lambda_{\min}(\boldsymbol{P})$, collected along the trajectory, and the mean estimation errors, mean task execution time, mean computational time, and task success rate (the task is considered failed if at least one of the safety constraint is violated).

Table 8 shows that our methodology outperforms the other approaches, providing the highest task success rate, the highest mean information, the smallest mean estimation errors along x and y, and mean computational time. The MPC without active sensing without/with CBF presents the smallest task success rate. This approach only considers the robot's and landmarks' sizes but not the robot state estimation uncertainty. In other words during the planning, the estimation uncertainty about the robot state is not considered leading to trajectories that could be safe for the estimated state but not for the real one. A slight improvement is obtained through the MPC with active sensing without CBF. The inequality constraints do not always find safe trajectories, while CBFs improve the feasibility of the problem. This is proved by the task success rate, greater for the approaches with CBFs. Finally, with our Information-aware CBF, the highest task success rate is obtained which is equal to 100%. However, finding safe solutions requires time, and hence with our approach, the task is accomplished with the highest mean task execution time. This aspect can be mitigated, for example, by increasing the risk r (which is 5% here) or by introducing a slack variable in the CBF constraint.

4.3 Integration with the navigation stack

To encompass the concept of risk at the motion planning level, we integrate the described risk maps with the local planner (the MPC (model predictive control) developed in T6.1). The risk maps are embedded as soft constraints within the MPC formulation, with their weights adjusted according to the potential severity of the associated risk factors. Figure 15 and Figure 16 show the robot moving during integration tests at KI.Fabrik in Munich. The robot moves taking into account human 3D poses, activities, trajectory prediction and risk maps.

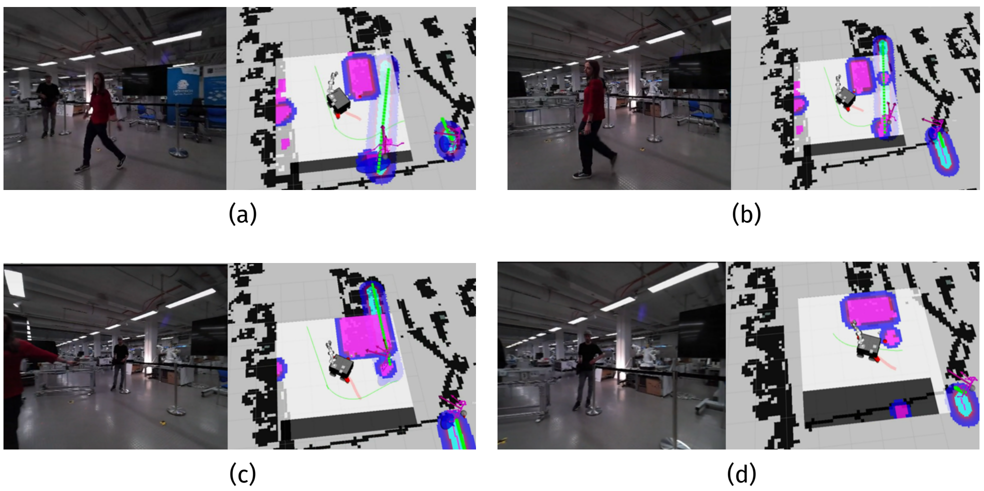


Figure 15: Rviz visualization and onboard camera image, while the robot moves in the KI.Fabrik in Munich. The risk map is centered on the robot and includes detected objects, and static known risky areas (near the retractable belt stanchions).

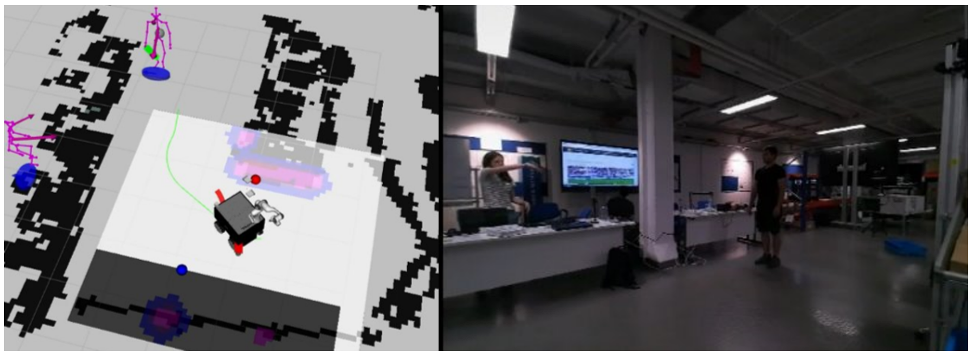


Figure 16: Focus on human skeletons detection and the constraints (blue and red markers) derived from the risk maps.

Integration of the other components described in this deliverable is still ongoing. When the robot has to perform a long navigation task across multiple corridors the route planner will give as output the next routes intersection point to be reached by the global planner. We are integrating the active sensing module in the navigation framework as a constraint for the MPC similarly to as done for the risk maps.

5 On the evaluation of collision probability along a path

The evaluation of collision probability (CP) is a critical component of risk assessment in robotics, particularly for applications involving autonomous systems operating in dynamic and uncertain environments. Grid-based methods, which discretize the naturally continuous environment, have previously been proposed for CP estimation during motion. However, these methods face a fundamental limitation - as the grid cells approach infinitesimal size to better approximate the continuous space, the CP estimation invariably approaches 1, rendering the approach ineffective. While Monte Carlo (MC) simulations offer an alternative with high accuracy, their computational demands make them unsuitable for real-time applications, and their inherent discontinuities present challenges for smooth optimization.

In this section, we explain the research work that we carried out to propose a method that combines computational efficiency with accuracy, addressing the continuous nature of robot trajectories and environmental uncertainties. For a detailed derivation of the mathematical formulation, proofs, and extensive experimental validation, the reader is referred to the complete work in [18].

We propose a novel metric, termed Risk Density, to address the limitations of existing methods for evaluating CP along a continuous path. The method assumes a robot that navigates along a continuous trajectory modeled as a parametric path. The position of the robot and obstacles are both modeled as Gaussian distributions, accounting for uncertainties in localization and obstacle positioning. Given these assumptions, the goal is to estimate the probability of collision along the entire trajectory.

The work provides a framework to examine how different assumptions about the interdependence of collision events lead to valid CP approximations. Risk Density serves as a theoretical bridge between two of these assumptions: by computing the sensitivity of CP to the size of the combined robot-obstacles object, the same quantity is obtained starting from either assumptions. This quantity in turn is used to effectively approximate CP and risk variations without the need for excessive sampling or discretization.

Mathematically, the risk density is a functional, expressed as a unidimensional line integral of the probability density function of the combined robot-obstacle object over the trajectory. Given a trajectory $\gamma_d(s)$, a Gaussian pdf $\mathcal{N}$ representing uncertainty, and its associated covariance Σ_d, the Risk Density is:

$$r_d(\gamma_d) = 2\int_0^1 \mathcal{N}(\gamma_d(s)|0, \Sigma_d)\left|\frac{d\gamma_d}{ds}\right| ds. \tag{36}$$

The associated CP approximation then is simply

$$P_a(C, r) \simeq r_d(\gamma_d)r, \ r \text{ radius of combined object.} \tag{37}$$

This formulation avoids the need for parameter tuning and offers a computationally efficient approximation of the CP, making it suitable for real-time applications.

The proposed method has been validated through extensive numerical simulations and benchmark comparisons against traditional approaches:

- Monte Carlo Simulations: The Risk Density approximation closely matches MC results with significantly reduced computation time while also maintaining differentiability.
- Grid-Based Approaches: Unlike grid-based methods, the proposed approach avoids issues arising from discretization, providing consistent results without requiring tuning of grid resolution.
- Stagewise Approximations: The method outperforms estimations based on Boole's Lemma, often used in chance-constrained optimization, which tends to overestimate collision probabilities due to double counting of events.

The experiments demonstrate that the proposed method effectively balances accuracy and computational efficiency across different trajectories and levels of uncertainty. Moreover, its formulation makes it particularly suitable for trajectory optimization tasks where probabilistic constraints must be enforced and multiple obstacles are present.

6 Conclusions and ongoing work

In this task, we have explored several strategies to incorporate risk reasoning and uncertainty awareness throughout all levels of the motion planning pipeline. In particular, we developed:

- A risk-aware route decision maker that effectively includes possible future observations in the model to minimize a risk factor concerning human encounters.
- Heterogenous risk maps to evaluate risk factors locally.
- An active sensing layer to reduce uncertainty when needed.
- A novel method to evaluate the probability of collision along a continuous path.

While some components have already been integrated into the DARKO navigation framework, the integration of the remaining strategies is an ongoing effort and part of our future work.

References

[1] Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European Control Conference (ECC)*, pages 3420–3431, 2019.

[2] Aaron D. Ames, Jessy W. Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs with application to adaptive cruise control. In *53rd IEEE Conference on Decision and Control*, pages 6271–6278, 2014.

[3] Logan E Beaver and Andreas A Malikopoulos. Optimal control of differentially flat systems is surprisingly easy. *Automatica*, 159:111404, 2024.

[4] Urs Borrmann, Li Wang, Aaron D. Ames, and Magnus Egerstedt. Control barrier certificates for safe swarm behavior. *IFAC-PapersOnLine*, 48(27):68–73, 2015. Analysis and Design of Hybrid Systems ADHS.

[5] Manuel Castellano Quero, Tomasz P. Kucner, and Martin Magnusson. Alignability maps for the prediction and mitigation of localization error. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS) Workshop on Closing the Loop on Localization*, 2023.

[6] Devin Connell and Hung Manh La. Dynamic path planning and replanning for mobile robots using rrt. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1429–1434. IEEE, 2017.

[7] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza. Pampc: Perception-aware model predictive control for quadrotors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8, 2018.

[8] Seyedshams Feyzabadi and Stefano Carpin. Risk-aware path planning using hirerachical constrained markov decision processes. In *2014 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 297–303. IEEE, 2014.

[9] Harary Frank. Shortest paths in probabilistic graphs. *operations research*, 17(4):583–599, 1969.

[10] Thomas Gurriet, Andrew Singletary, Jacob Reher, Laurent Ciarletta, Eric Feron, and Aaron Ames. Towards a framework for realizable safety critical control through active set invariance. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pages 98–106, 2018.

[11] Shao-Chen Hsu, Xiangru Xu, and Aaron D. Ames. Control barrier function based quadratic programs with application to bipedal robotic walking. In *2015 American Control Conference (ACC)*, pages 4542–4548, 2015.

[12] JunTak Lee, Tae-Won Kang, Yong-Sik Choi, and Jin-Woo Jung. Clearance-based performance-efficient path planning using generalized voronoi diagram. *International Journal of Fuzzy Logic and Intelligent Systems*, 23(3):259–269, 2023.

[13] Anqi Li, Li Wang, Pietro Pierpaoli, and Magnus Egerstedt. Formally correct composition of coordinated behaviors using control barrier certificates. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3723–3729, 2018.

[14] V. Murali, I. Spasojevic, W. Guerra, and S. Karaman. Perception-aware trajectory generation for aggressive quadrotor flight using differential flatness. In *2019 American Control Conference (ACC)*, pages 3936–3943, 2019.

[15] Olga Napolitano. *Shaping the Information Flow in Robotic Systems: from Measures of Information to Active Sensing Control Strategies*. Phd thesis, Uni. Pisa, 2024.

[16] Olga Napolitano, Daniele Fontanelli, Lucia Pallottino, and Paolo Salaris. Gramian-based optimal active sensing control under intermittent measurements. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[17] Simona Nobili, Georgi Tinchev, and Maurice Fallon. Predicting alignment risk to prevent localization failure. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1003–1010, 2018.

[18] Lorenzo Paiola, Giorgio Grioli, and Antonio Bicchi. On the evaluation of collision probability along a path. *IEEE Transactions on Robotics*, 2024.

[19] Christos H Papadimitriou and John N Tsitsiklis. The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

[20] George H Polychronopoulos and John N Tsitsiklis. Stochastic shortest path problems with recourse. *Networks: An International Journal*, 27(2):133–143, 1996.

[21] J Scott Provan. A polynomial-time algorithm to find shortest paths with recourse. *Networks: An International Journal*, 41(2):115–125, 2003.

[22] Matteo Rubagotti, Inara Tusseyeva, Sara Baltabayeva, Danna Summers, and Anara Sandygulova. Perceived safety in physical human–robot interaction—a survey. *Robotics and Autonomous Systems*, 151:104047, 2022.

[23] P. Salaris, M. Cognetti, R. Spica, and P. Robuffo Giordano. Online optimal perception-aware trajectory generation. *IEEE Transactions on Robotics*, 35(6):1307–1322, 2019.

[24] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control*. Advanced Textbooks in Control and Signal Processing. Springer London, 2010.

[25] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer Science & Business Media, 2008.

[26] Elena Stracca, Andrey Rudenko, Luigi Palmieri, and Paolo Salaris et al. Darko-nav: Hierarchical risk- and context-aware robot navigation in complex intralogistic environments (accepted). In *European Robotics Forum 2025*. Springer Nature Switzerland.

[27] Xiangru Xu, Jessy W. Grizzle, Paulo Tabuada, and Aaron D. Ames. Correctness guarantees for the composition of lane keeping and adaptive cruise control. *IEEE Transactions on Automation Science and Engineering*, 15(3):1216–1229, 2018.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 101017274