Information Sciences 471 (2019) 164–184

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Multi-goal motion planning using traveling salesman problem in belief space

Ali Noormohammadi-Asl, Hamid D. Taghirad[1,*]

Advanced Robotics and Automated Systems (ARAS) Faculty of Electrical Engineering, K. N. Toosi University of Technology, Iran

ARTICLE INFO

Article history:
Received 23 January 2018
Revised 25 July 2018
Accepted 25 August 2018
Available online 30 August 2018

Keywords:
Motion planning
Uncertainty
Belief space
Traveling salesman problem
Search and exploration

ABSTRACT

In this paper, the multi-goal motion planning problem of an environment with some background information about its map is addressed in detail. The motion planning goal is to find a policy in belief space for the robot to traverse through a number of goal points. This problem is modeled as an asymmetric traveling salesman problem (TSP) in the belief space using Partially Observable Markov Decision Process (POMDP) framework. Then, feedback-based information roadmap (FIRM) algorithm is utilized to reduce the computational burden and complexity. By generating a TSP-FIRM graph, the search policy is obtained and an algorithm is proposed for online execution of the policy. Moreover, approaches to cope with challenges such as map updating, large deviations and high uncertainty in localization, which are more troublesome in a real implementation, are carefully addressed. Finally, in order to evaluate applicability and performance of the proposed algorithms, it is implemented in a simulation environment as well as on a physical robot in which some challenges such as kidnapping and discrepancies between real and computational models and map are examined.

1. Introduction

Simultaneous localization and planning is the basis of many robotics applications such as exploration, search and coverage. A proper planning helps a robot to make use of the available information and generate or select right motion commands and actions. Action selection criteria vary with the goals of the application [44]. For example, in an earthquake affected building, motion commands should guide the robot to search and explore the environment in a rapid and successful way using the information such as rough layout of the building [46]. As another example, consider a service mobile robot which operates in an office-like environment and should do a sequence of tasks in different locations, e.g., different rooms. Therefore, a proper action selection and planning, based on the overview-map, is necessary to overcome its tasks successfully and efficiently [28,40]. In this paper, we assume some prior information is available about the environment, e.g., the layout of the environment or an initial map.

In the absence of uncertainty, the planning for search, exploration, and coverage is mainly concerned with investigating the methods to speed up and improve the robot's performance [24,36,47,49]. Xu and Stentz [47] propose a graph based method to find a short path for the coverage of an environment and the problem is modeled as a rural postman problem.

* Corresponding author.
E-mail addresses: a.noormohammadi@ieee.org (A. Noormohammadi-Asl), Taghirad@kntu.ac.ir, hamid@cim.mcgill.ca (H.D. Taghirad).
[1] website: aras.kntu.ac.ir

https://doi.org/10.1016/j.ins.2018.08.062
0020-0255/

Additionally, it suggests an algorithm for updating the graph in the case of discrepancy between the actual and the initial map. Pasqualetti et al. [36] use a graph-based solution for patrolling in an environment. Yehoshua et al. [49] study the coverage problem, and the cell decomposition method with high-risk cells is used to find a low-risk path for the robot. A prevalent approach to cope with these kinds of problems is formulating them as a traveling salesman problem (TSP) and exploiting the background information to make a proper plan in order to guide the robot to search the environment faster [7,15,26,35]. Kulich et al. [26] and Faigl et al. [15] model the exploration problem by a single and multi-robot as a TSP which its edges' costs are proportional to the distance between goal points. In the method proposed by Oßwald et al. [35], the robot explores the environment based on a policy obtained by solving a TSP to speed-up the exploration, but there is not a predefined and certain path, and the robot moves between manually selected goal points using a local exploration algorithm. It also suggests a solution for replanning and updating TSP solution in the case of finding a difference between the initial and the actual map.

More importantly, what makes planning more difficult and sophisticated is the presence of motion and observation uncertainties which are usually ignored in most studies. Uncertainty is closely tight to robotics problems and ignoring it may lead to choosing a short path with the low probability of success. Motion and observation uncertainties cause the lack of full state information for decision making and planning. However, a filter (e.g., Kalman and particle filter) can estimate a probability distribution function (PDF) over all possible states, called belief or information state, using the dynamical model of states and measured values by sensors [1,3]. This means the planning and decision making in the aforementioned examples should be done in the belief space. Freundlich et al. [16] study the path planning and the resource allocation for the multi-goal problem, under motion and sensor uncertainties, but it is restricted to the discrete space and is not applicable in large and real environments as well as long-term operations. Faigl et al. [14] study the autonomous multi-goal inspection by adding auxiliary navigation way points around each goal to reduce the uncertainty and applying self-organizing map algorithm to the TSP [13]. In this method, the uncertainty and the probability of colliding obstacles on the path between every two goals are not considered.

Another major challenge in multi-goal motion planning is the necessity of online replanning ability. In situations such as a change in the map or an update in the robot's belief (localization), it is required that the robot make a replanning online which are not considered in [14,16]. This challenge is more evident in a real application where the computational models of the system such as motion, sensor and map differ from real models. Replanning in belief space is more challenging and requires an approach which is suitable for a real application in the sense of time and complexity.

Motion Planning Under Uncertainty (MPUU) is the basis of the multi-goal motion planning, and many studies recently have been done in this area [21,22,37,39]. Pilania and Gupta [37] propose a sampling-based algorithm for motion planning of a mobile manipulator under uncertainty and consider effects of uncertainty on the manipulator motion. Janson et al. [22] present the MCMP (Monte Carlo Motion Planning) approach that considers probabilistic collision avoidance constraints, and is suitable for the real-time implementation. The MPUU problem is extended to the multi-robot belief space planning in unknown environments by Regev and Indelman [39], and a decentralized sampling-based planning is proposed to address this problem. Owing to successful performance of sampling based methods in motion planning problems, they are expanded to motion planning under uncertainty problems [5,10,37]. In LQG-MP algorithm presented by Berg et al. [8], the best path is chosen from a group of path obtained by RRT, based on their performance in the presence of an LQG controller. Bry and Roy [9] also use RRT* to find the nominal optimized path. Prentice and Roy [38] and Huynh and Roy [20] use breadth-first search method on the roadmap constructed by PRM, to find the best path. The aforementioned methods suffer from the lack of optimal substructure property meaning the cost of each edge affects the travel cost of other edges. Consequently, the constructed roadmap depends on the start point and in every query should be reconstructed. In other words, they are single query algorithms. Moreover, in the case of deviation from the nominal path, constructing a new roadmap is necessary which makes mentioned algorithms unsuitable for real applications.

MPUU, i.e., motion and sensor uncertainty, is an instance of sequential decision making under uncertainty. This problem can be formulated as a Partially Observable Markov Decision Process (POMDP) [23]. POMDP is a prevalent framework for modeling the sequential decision process. Many real-world problems in fields such as industry, ubiquitous computing, business, ecology, control and robotics [11,12,25,27,30] can be modeled as a POMDP problem. The curse of dimensionality and history make the POMDP problem arduous and computationally intractable [23]. They are even more troublesome in the continuous state space. Many methods have been proposed for solving POMDP problem [42,45,48]. However, they are mostly restricted to the problems with a small set of states, they are not applicable in the continuous space or they are computationally expensive. In order to mitigate the abovementioned problems and extending the POMDP applications to real-world and long-term operations, methods such as [3,18,41] have been proposed. Agha-mohammadi [2] propose the Feedback-based Information RoadMap (FIRM) algorithm where the optimal substructure property is held. FIRM turns the intractable POMDP problem into a tractable DP on the FIRM graph. The optimal substructure property of FIRM and the ability to solve the POMDP online enable online replanning in FIRM. Furthermore, the FIRM feedback-based structure makes it a robust algorithm. FIRM also considers all possible future observations in decision making, which provides a reliable plan. FIRM algorithm has been used successfully in motion planning problems and has been implemented on a physical robot and results show its efficiency [4,31].

Although much theoretical work exists on the MPUU, the multi-goal MPUU is not well studied, and the uncertainty is omitted in the most of them which may well cause an improper planning and increase the probability of failure. In addition, they have not consider any approach for online replanning to cope with unpredicted events such as finding new

obstacles which are prevalent in a real environment. In this paper, therefore, we involve the uncertainty in planning, and we model the search and exploration problem as an asymmetric TSP in the belief space. In addition, environment uncertainty is considered as high-risk areas. There are, however, some limitations which make the problem challenging and intractable to solve: (i) the path between two goals point is not the direct line connecting them, because the length of the path is not the only factor in the planning, (ii) the edges' costs is not deterministic, (iii) the cost of moving between nodes is not symmetric meaning the cost of moving from node A to B is not equal to the cost of moving from node B to A and (iv) the planning to move from a goal node to another goal node should be done in the belief space.

Briefly, the problem is a combination of a decision making and multi-goal motion planning in the belief space. The first challenge is to simplify the problem in order to obtain an optimal policy. Then, it is necessary to propose an approach for implementing the policy and provide tools for the robot to cope with the challenges of a real environment. Consequently, toward reducing the difficulty of the problem and achieving a proper policy in both offline and online phase:

- the problem is formulated as an asymmetric TSP in the belief space.
- a graph including a set of sampled nodes and goal nodes is constructed.
- an offline algorithm based on the FIRM is proposed to find an optimal search policy. In this paper we also consider the environment uncertainty in planning.
- an online algorithm is presented to enable the robot to execute the generated policy. In addition, we propose an algorithm for online replanning to overcome issues such as map changes.
- Finally, the proposed algorithms are modified for nonholonomic unicycle robots in the real and simulation experiments and a switching based controller is designed for the posture stabilization.

The outline of this study is as follows. In Section 2, we explain a brief introduction of POMDP problem, FIRM algorithm, and TSP. In Section 3, the multi-goal motion planning problem is presented as an aTSP in the belief space and the TSP-FIRM graph is presented to simplify it. In Section 4, we extend the TSP-FIRM to the unicycle nonholonomic robots. Section 5 provides the implementation details and results in the simulation and real environment. Finally, the paper is concluded in Section 6.

2. Preliminaries

The presentation and the notation of POMDP problem and FIRM algorithm in this section follow that of Aghamohammadi et al. [3].

2.1. POMDP problem

Partially observable Markov decision processes is a general framework for sequential decision making under uncertainty. POMDP is the generalization of the Markov Decision Process (MDP) in which only imperfect state information is available. A POMDP problem can be formulated as a belief MDP problem where the state is a belief state. As mentioned earlier, the motion planning under uncertainty can be formulated as a POMDP problem. To clarify the POMDP problem, we illustrate it with a simple example. Consider a temperature control system in which the temperature value is measured with an imperfect sensor. The heater accuracy is inversely proportional to the heater temperature, and the probability of its damage increases as works with high power for a long time. The goal is that the system temperature reaches and maintains in a desired range with a high probability of success and in a short time (the minimization of fuel consumption can also be added). In the following, some required notations for expressing the POMDP problem are presented.

State, control, and process model: By discretizing the time into equal sections, $x_k \in \mathbb{X}$ and $u_k \in \mathbb{U}$ show the system state and control at time k, respectively. $\mathbb{X}$ and $\mathbb{U}$ are state space and control space which can be continuous. The control history, $u_{0:k} := \{u_0, u_1, \ldots, u_k\}$, includes all of the control actions up to step k. The state evolution of the system, called process model (motion model), $x_{k+1} = f(x_k, u_k, w_k)$, shows the evolution of the states after applying a control action in the presence of process noise with probability density function $w_k \sim p(w_k|x_k, u_k)$. This evolution also can be shown by the transition pdf $p(x' \mid x, u) : \mathbb{X} \times \mathbb{U} \times \mathbb{X} \to \mathbb{R}_{\geq 0}$ where $\mathbb{R}$ denotes the Euclidean space. In the mentioned example, the temperature (state) of the system is x, and the input, u, is the heat generated by the heater. The process noise depends on the generated heat and may well increase if generated heat increases. This means $p(x'|x, u)$ will probably have a lower value if the heater works with high power.

Observation and observation model: Unlike the MDP problem, in POMDP, the states of the system are unknown, and decision-making should be done based on the imperfect measurement of the states. $z_k \in \mathbb{Z}$ denotes observation of the system, measured by the sensors, at time step k. $\mathbb{Z}$ is observation space of the system and can be continuous. The observation model, $z_k = h(x_k, v_k)$, describes how the system state x_k is related to the measured value by sensors. v_k is the observation noise distributed according to the pdf $v_k \sim p(v_k|x_k)$. $p(z \mid x) : \mathbb{X} \times \mathbb{Z} \to \mathbb{R}_{\geq 0}$ is another representation of the observation model. In the temperature control example, z is the measured temperature value by the sensor. The observation model describes how the system temperature value can be related to the measured value by the noisy sensor.

Belief and belief evolution: In the partially observable system, where the states of the system are unknown, the history of observations and control actions are the set of available data for decision making. The set of this available data is shown

by $\mathcal{H}_k = \{z_{0:k}, u_{0:k-1}\}$. However, the compressing of $\mathcal{H}_k$ in a conditional probability distribution will preserve required information for decision making, meaning $b_k = p(x_k \mid \mathcal{H}_k)$. b_k is the system belief or information state at time step K. In other words, the planning is done in the belief space or space of probabilistic state estimations. To obtain the belief evolution model a recursive state estimation method such as Bayes filter can be used. Thus, based on the Markov assumption and using bayesian filters, $b_k = p(x_k \mid \mathcal{H}_k)$ can be rewritten as a function of the last belief, last control action and the current observation:

$$\begin{aligned} b_{k+1} = \quad & p(z_{k+1} \mid \mathcal{H}_k, u_k)^{-1} p(z_{k+1} \mid x_{k+1}) \\ & \int_{\mathbb{X}} p(x_{k+1} \mid x_k, u_k) b_k dx_k =: \tau(b_k, u_k, z_{k+1}) \end{aligned} \tag{1}$$

In the mentioned example, based on the observation and motion model, a filter (e.g., Kalman filter) can be used to estimate the temperature value.

Policy and cost-to-go: In a partially observable system, the policy, $\pi_k \in \Pi$, is a function that uses the available history of data for generating an action, u_k, as its output. Π is the space of all possible policies. As mentioned before, compressing of $\mathcal{H}_k$ in a conditional probability distribution will preserve required information for decision making. Thus, the policy can be defined as a function returning the action u_k based on the belief b_k, $u_k = \pi_k(b_k)$. In order to obtain an optimal policy, a cost function called cost-to-go is required. To determine a cost function, first we need to define the one-step-cost, $c(b, u) = \mathbb{E}[c(x, u)|\mathcal{H}] = \int_{\mathbb{X}} c(x, u) p(x|\mathcal{H}) dx$, which is the cost of taking action u at belief b. Now, the cost-to-go can be defined as:

$$\begin{aligned} & J^{\pi}(b_0) = \sum_{k=0}^{\infty} \mathbb{E}[c(b_k, \pi(b_k))] \\ & s.t. \quad b_{k+1} = \tau(b_k, \pi(b_k), z_{k+1}), \quad z_k \sim p(z_k \mid x_k) \end{aligned} \tag{2}$$

where $\mathbb{E}$ is the expectation operator. The cost-to-go in the temperature control example can be simply defined as zero when the system is in the desired range and a positive value when the system is out of the desired range.

POMDP problem: Subsequently, the POMDP problem challenge is finding an optimal policy to minimize the cost-to-go function from every belief in the belief space. In other words, the solution of the POMDP, called optimal policy π^*, is as follows:

$$\begin{aligned} & J(\cdot) = \min_{\Pi} J^{\pi}(\cdot) \\ & \pi^* = \underset{\Pi}{argmin} J^{\pi}(\cdot), \end{aligned} \tag{3}$$

where J is the optimal cost-to-go function. Using dynamic programming, the optimal solution of this problem can be obtained. The DP equations for the MDP in the belief space are as follows:

$$\begin{aligned} & J(b) = \min_{u} \left\{ c(b, u) + \int_{\mathbb{B}} p\left(b' \mid b, u\right) J\left(b'\right) db' \right\}, \quad \forall b \in \mathbb{B} \\ & \pi^*(b) = \underset{u}{argmin} \left\{ c(b, u) + \int_{\mathbb{B}} p\left(b' \mid b, u\right) J\left(b'\right) db' \right\}, \forall b \in \mathbb{B}. \end{aligned} \tag{4}$$

Solving this DP equation is extremely difficult due to the definition over the entire belief space and the curse of history. In many problems such as motion planning, it is necessary to add some constraints such as obstacles and time limitation to the cost function, which makes the problem more difficult. We can take these constraints as a failure set, F, meaning the policy fails if the system state hits it.

2.2. Review on FIRM

FIRM is a multi–query graph based algorithm for path planning under uncertainty. FIRM helps to simplify the complex POMDP problem into a tractable MDP problem. The DP equation in (4) is defined over an infinite dimensional belief space and thus is extremely difficult to solve.

2.2.1. FIRM graph

In the first step, it is assumed that there is no constraint and obstacle, $F = \emptyset$, and the motion planning over the entire belief space is reduced into a graph constructed in the belief space. Therefore, the main MDP problem can be rewritten as a tractable MDP over a graph.

In order to construct this graph, the first step is sampling a set of stabilizers $\{\mu^j\}$. A stabilizer is a closed-loop controller that helps to stabilize in a predefined belief. A stabilizer is a combination of a filter (e.g. Kalman filter) and a separated controller, and is a mapping from belief space to control space. To construct stabilizers, an underlying PRM (Probabilistic Roadmap) is formed in the state space with a set of nodes and edges, $\{v, e\}$ and then the stabilizer corresponding to each PRM node is produced. A unique FIRM node, B, is obtained for each PRM node which is a small region $B = \left\{b : \|b - b'\| \leq \gamma\right\}$ around the sampled belief b', and γ determines the size of this region. $\mathbb{V} = \left\{B^i\right\}_{i=1}^{N}$ is the set of all FIRM nodes. The FIRM nodes should be chosen such that satisfy the reachability condition. To define the reachability in this concept, the region

B^j is called reachable under the stabilizer μ^j starting from b if $\mathbb{P}\left(B^j \mid b, \mu^j\right) = 1$. The reachability condition is discussed in detail by Agha-mohammadi [2]. Each edge of the FIRM graph is a closed loop local controller referred as a local controller, μ^{ij}. Local controller's task is to steer a belief from the FIRM node B^i to the target node of the edge, B^j. $\mathbb{M} = \left\{\mu^{ij}\right\}$ is the set of all local controllers. Local controller design is problem-dependent and will be later explained in detail.

The constructed graph is known by the set of nodes $\mathbb{V} = \left\{B^i\right\}_{i=1}^{N}$, and the set of edges or local controllers $\mathbb{M} = \left\{\mu^{ij}\right\}$. In the FIRM, the policy is formed by the concatenation of local policies performing in continuous space. The feedback feature of the local policies provides this ability to compensate the deviation of the belief from the planned path and drive it to a stopping region.

2.2.2. Belief semi-Markov decision process (SMDP)

In a Semi-Markov Decision Process (SMDP) actions take a random amount of time. In FIRM graph the transition time from a belief to a graph node is random. Thus, with the construction of the FIRM graph, the motion planning over the entire belief space is reduced to planning over a subset of the belief space, i.e., the union of graph nodes. The one-step-cost $c(b, u) : \mathbb{B} \times \mathbb{U} \to \mathbb{R}_{\geq 0}$ is changed to one-step SMDP cost $C^s(b, u) : \mathbb{B} \times \mathbb{M} \to R_{\geq 0}$ which is defined as follows:

$$C^s(b, \mu) := \sum_{t=0}^{\mathcal{T}} c(b_t, \mu(b_t) \mid b_0 = b), \tag{5}$$

where $\mathcal{T}$ represents the time when the belief hits $\Psi = \cup_j B^j$ by starting from b and invoking local controller μ. Now, the POMDP problem defined in (4) as a DP problem is reduced to the following semi-Markov decision process in belief space, called SMDP belief or Semi-POMDP:

$$J^s(b) = \min_{\mu \in \mathbb{M}(i)} C^s(b, \mu) + \int_{\Psi} p\left(b' \mid b, \mu\right) J^s\left(b'\right) db', \ \forall b \in B^i, \ \forall i \tag{6}$$

and unlike Eq. (4) where the integration is over the entire belief space, the integration is reduced to over the sampled nodes.

2.2.3. FIRM MDP

Although the DP equation in (6) is more tractable than the initial POMDP, it is difficult to solve due to the integration over the continuous neighborhood of the nodes. For sufficiently small and sufficiently smooth cost functions, it can be assumed that the cost-to-go of all beliefs in B^i are approximately equal. Therefore, the transition cost C^g can be defined on the FIRM graph which is the cost of applying local controller μ^{ij} at the FIRM node B^i. Similarly, the transition probability, $\mathbb{P}^g\left(B^j \mid B^i, \mu^{ij}\right)$, is defined on the FIRM graph which is the transition probability from B^i to B^j under the local controller μ^{ij}. Thus,

$$\forall b \in B^i, \forall i, j \begin{cases} C^g\left(B^i, \mu^{ij}\right) := C\left(b_c^i, \mu^{ij}\right) \approx C\left(b, \mu^{ij}\right) \\ \mathbb{P}^g\left(\cdot \mid B^i, \mu^{ij}\right) := \mathbb{P}\left(\cdot \mid b_c^i, \mu^{ij}\right) \approx \mathbb{P}\left(\cdot \mid b, \mu^{ij}\right), \end{cases} \tag{7}$$

where b_c^i is a point in B^i (e.g., center of B^i). Using this approximation, b_c^i will be a representative of every belief in the region B^i. Consequently, it can be shown that the DP Eq. (6) is simplified to the following DP equation in which obstacles are also considered:

$$\begin{aligned} J^g\left(B^i\right) &= \min_{\mu \in \mathbb{M}(i)} C^g\left(B^i, \mu\right) + J^g(F)\mathbb{P}^g\left(F \mid B^i, \mu\right) + \sum_j \mathbb{P}^g\left(B^j \mid B^i, \mu\right) J^g\left(B^j\right), \ \forall i \\ \pi^g\left(B^i\right) &= \underset{\mu \in \mathbb{M}(i)}{argmin}\, C^g\left(B^i, \mu\right) + J^g(F)\mathbb{P}^g\left(F \mid B^i, \mu\right) + \sum_j \mathbb{P}^g\left(B^j \mid B^i, \mu\right) J^g\left(B^j\right), \ \forall i \end{aligned} \tag{8}$$

where $J^g(F)$ is the cost-to-go of the failure region and is set to a high value. $\mathbb{P}^g\left(F \mid B^i, \mu\right)$ is the probability of hitting obstacles or generally the failure set before reaching to the stopping region. Consequently, the original POMDP equation is simplified to a finite N_v-state (over FIRM nodes $\mathbb{V} = \left\{B^i\right\}_{i=1}^{N}$) MDP in (8), referred as FIRM MDP. The following is the overall policy of FIRM which is a combination of the global and local policy

$$u_k = \pi(b_k) = \begin{cases} \mu_k^*(b_k), \ \mu_k^* = \pi^g\left(B_{k-1}^*\right) & \text{if} \quad b_k \in B_{k-1}^* \\ \mu_k^*(b_k), \ \mu_k^* = \mu_{k-1}^* & \text{if} \quad b_k \notin \Psi \end{cases} \tag{9}$$

meaning that the active local controller μ_{k+1}^* is chosen based on the global policy on the graph if the belief hits a stopping region, else it continues to be unchanged $\mu_{k+1}^* = \mu_k^*$, and generates the control input according to the belief at that time step.

The algorithm of generic construction of the FIRM graph and the generic planning (replanning) on the constructed graph are presented by Agha-mohammadi et al. [3, Algorithm 3] and Agha-mohammadi et al. [3, Algorithm 4], respectively which are omitted for the sake of brevity. Replanning is an important ability of the FIRM algorithm. There are situations such as a large deviation from the planned path, or discrepancies between the actual and computational models and map where the

robot should do replanning to overcome them. Therefore, the robot should be able to move from any new node to a node of the graph. Proposed algorithm by Agha-mohammadi et al. [4, Algorithm 1] and Agha-mohammadi et al. [4, Algorithm 2] help the robot to do replanning.

2.3. Traveling salesman problem (TSP)

Traveling salesman problem (TSP) is a well-known problem in the graph theory concerning with finding the most efficient Hamiltonian cycle that a salesman can visit every city once and returns to the starting point (Gutin and Punnen [19]). One type of TSP problem is known as asymmetric Travelling Salesman Problem (aTSP) in which the anti-parallel edges do not have same weights. An aTSP can be defined on a complete directed graph $G = (V, A)$. $V = \{1, \ldots, N\}$ is the set of vertex and $A = \{(i, j) : i, j \in V, i \neq j\}$ is its corresponding arc set. The cost matrix of the graph is defined as $C = c_{ij}$ on A and satisfies the triangularity meaning $c_{ij} \leq c_{ik} + c_{kj}, \forall i, j, k \in V$. aTSP can be formulated as:

$$\begin{aligned} \min_{X} \quad & \sum_{i \neq j} c_{ij} x_{ij} \\ s.t. \quad & \sum_{j=1}^{N} x_{ij} = 1 \quad (i \neq j, i \in V) \\ & \sum_{i=1}^{N} x_{ij} = 1 \quad (i \neq j, j \in V) \\ & \sum x_{ij} \leq |s| - 1 \qquad (s \subset V \quad , \quad 2 \leq |s| \leq N - 2) \\ & x_{ij} \in \{0, 1\} \qquad (i, j) \in A \end{aligned} \tag{10}$$

where the two first constraints mean that salesman can reach each city only from one other city and left it once. The third constraint is the subtour elimination constraint, which also can be written as MTZ formulation

$$\begin{aligned} & u_i - u_j + N x_{ij} <= N - 1; \ i, j \in \{2, \ldots, N\}, i \neq j \\ & u_i \in \mathcal{R}; i \in V, i > 1. \end{aligned} \tag{11}$$

3. TSP-FIRM for multi-goal motion planning

In the multi-goal motion planning problem discussed in this paper, we assume there exists background information about the environment helping the robot in decision making. In other words, we are seeking a method to exploit available data not only to speed up the environment search or exploration but also to increase the success probability. It is obvious that if more data of the environment is available, better decision making will be possible. In a search environment, some parts have priority over others due to the more information they provide for the robot. Therefore, goal points are selected based on the importance and information they provide. For instance, in a floor of a building, rooms can be considered as goal points. After selecting the goal points, the robot should find a policy for searching them. In this paper, choosing the goal points is done manually. However, there are some algorithms for this purpose [15,26,29]. As mentioned before, if the robot knows its location and position of the goals exactly, then it can move on its path precisely. This case can be modeled as a simple TSP, but in a real situation the motion and sensing uncertainty cannot be ignored, and the data is available in the belief space.

3.1. Problem formulation

At first, we consider the problem in an obstacle free environment, $F = \emptyset$. Assuming N_g goal points, the sets V and A can be defined as $V = \{1, \ldots, N_g\}$ and $A = \{(i, j) \mid i, j \in V, i \neq j\}$, respectively. Note that, b^i, $i \in V$ is the belief of ith goal point and the set of these beliefs is $B = \{b^i \mid i = 1 : N_g\}$. B^i_{goal} is the ith goal region that the system stops when belief enters it. Each b^i belongs to a goal region, meaning $b^i \in B^i_{goal}$. The set $B_{goal} = \{B^i_{goal}\}_{i=1:N_g}$ shows all the goal regions of the goal points. The one-step-cost in the belief space is defined by

$$\begin{cases} c_i(b, u) = 0, & if \quad b \in B^i_{goal} \\ c_i(b, u) = \mathbb{E}[c(x, u)|\mathcal{H}] \\ \qquad = \int_{\mathbb{X}} c(x, u) p(x|\mathcal{H}) dx \geq \varepsilon > 0, & if \quad b \notin B^i_{goal} \end{cases} \tag{12}$$

where x is the state of the robot and ε is a positive value. The one-step-cost is set to a zero value in the goal region, and to avoid infinite cycles and stopping before reaching the goal, the cost of taking action is positive until the system satisfies

the stopping condition. Now we can formulate the problem as:

$$\begin{aligned}
\min_{\{\mathbf{p},\Pi\}} \quad & \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} p_{ij} \sum_{k=0}^{\infty} \mathbb{E}\left[c_j\left(b_k^i, \pi\left(b_k^i\right)\right)\right] \\
s.t. \quad & \sum_{j=1}^{N_g} p_{ij} = 1 \quad (i \in V) \\
& \sum_{i=1}^{N_g} p_{ij} = 1 \quad (j \in V) \\
& \sum p_{ij} \le |s| - 1 \qquad (s \subset V, \quad 2 \le |s| \le N_g - 2) \\
& p_{ij} \in \{0,1\} \qquad (i,j) \in A \\
& b_{k+1} = \tau\left(b_k, \pi\left(b_k\right), z_k\right)
\end{aligned} \tag{13}$$

where $\mathbf{p}$ determines the sequence of searching goal points and Π is the policy of searching these goal points, based on $\mathbf{p}$. $b_k^i, k \neq 0$ is the belief changes started from $b_0^i \equiv b^i$. The inner summation presents the POMDP formulation where the goal is to steer the belief started from b_0^i to the goal region B_{goal}^j. According to (12), $c_j\left(b_k^i, \pi\left(b_k^i\right)\right)$ becomes zero when the belief started from $b_0^i \in B_{goal}^i$, reaches to the jth node goal region, B_{goal}^j. Two outer summations are for TSP formulation. The goal is to find a sequence for searching these goal points and a policy to move between each of them. As stated before, solving this problem is too complex and intractable. Hence, we break it down into the two subproblems. The first and basic step is finding a path between every two goal points using FIRM algorithm. In the next step, a sequence and a policy for searching goal pints are obtained.

3.2. Belief space aTSP via FIRM

3.2.1. Multi-path TSP on the TSP-FIRM graph

The first step in the construction of TSP-FIRM graph is sampling a set of stabilizers, μ^j. Therefore, we construct the underlying PRM with the nodes $\mathcal{V} = \{\{v^j\}_{j=1}^{N_s}, \{v_{goal}^i\}_{i=1}^{N_g}\}$ where $\{v_{goal}^i\}_{i=1}^{N_g}$ is the goal points set and $\{v^j\}_{j=1}^{N_s}$ is the sampled nodes set. Then, for each PRM nodes in $\mathcal{V}$ a stabilizer is constructed. Briefly, a TSP-FIRM graph with the set of nodes $\{B^i\}_{i=1}^{N_t=N_s+N_g}$ and the edges (local controllers), $\mathbb{M} = \{\mu^{ij}\}$, is obtained in which N_t is the total number of TSP-FIRM nodes consisting goal nodes. In this graph, the assumption (7) is held.

By constructing this graph, the main aTSP in the entire of the belief space and with the nondeterministic paths, is simplified to an aTSP in the finite space with N_t states defined on the TSP-FIRM graph nodes. In this new aTSP, there is not necessarily only one way between every two TSP nodes, in other words a multi-path TSP is generated. Q_{ij} is the set of all paths between the goal points i and j, on the TSP-FIRM graph. Consequently, the aTSP defined in (13) can be written as:

$$\begin{aligned}
\min_{\{\mathbf{p},\mathbf{y},\Pi^g\}} & \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} p_{ij} \sum_{q\in Q_{ij}} \sum_{k=0}^{\infty} \mathbb{E}\left[C_j^g\left(B_k^i, \pi^g\left(B_k^i\right)\right)\right] y_{ij}^q \\
s.t. \quad & \sum_{j=1}^{N_g} p_{ij} = 1 \quad (i \in V) \\
& \sum_{i=1}^{N_g} p_{ij} = 1 \quad (j \in V) \\
& \sum p_{ij} \le |s| - 1 \qquad (s \subset V \quad , \quad 2 \le |s| \le N_g - 2) \\
& p_{ij} \in \{0,1\} \qquad (i,j) \in A \\
& y_{ij}^q \in \{0,1\} \qquad q \in Q_{ij}, \ (i,j) \in A \\
& \textstyle\sum_{q\in Q_{ij}} y_{ij}^q = 1 \qquad \text{for each } (i,j) \\
& \mathbb{P}\left(B_{k+1}^i \mid B_k^i; \pi^g\left(B_k^i\right)\right)
\end{aligned} \tag{14}$$

where $\mathbf{y}$ is a matrix with maximum of one non-zero element in each row, and it determines which path is chosen between every two goal nodes. B_0^i is the ith goal node and B_k^i, $k \neq 0$ shows the belief node change on the TSP-FIRM graph started from node B_0^i. $q \in Q_{ij}$ is a path between the goal points i and j, and y_{ij}^q shows whether this path is selected or not. $C_j^g\left(B_k^i, \pi^g\left(B_k^i\right)\right)$ is the transition cost and shows the cost of taking local controller obtained from $\pi^g\left(B_k^i\right)$, at B_k^i, and the goal is to reach jth goal node, B^j. The fifth and sixth constraint indicate that between every two goal nodes only one path should be chosen and the last constraint shows the transition probability. The goal of two inner summations is to find the best policy and path between every two goal-points.

3.2.2. Asymmetric TSP on FIRM

The obtained aTSP in the previous section is more simple and tractable than the main aTSP. However, the problem is difficult to solve still due to nonunique paths between the aTSP nodes and defining optimal policy as a decision variable. Therefore, the TSP-FIRM is used to find the optimal policy and choose the best path between the goal nodes. The DP Eq. (8) is solved between every two goal nodes and the optimal path and the policy between them is obtained. As a result, a multi-path aTSP expressed in (14) is changed to a simple aTSP problem:

$$\begin{aligned} \min_{\{\mathbf{p}\}} \quad & \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} p_{ij} J_j^g\left(b^i\right) \\ s.t. \quad & \sum_{j=1}^{N_g} p_{ij} = 1 \quad (i \neq j, i \in V) \\ & \sum_{i=1}^{N_g} p_{ij} = 1 \quad (i \neq j, j \in V) \\ & \sum p_{ij} \leq |s| - 1 \qquad (s \subset V \quad , \quad 2 \leq |s| \leq N_g - 2) \\ & p_{ij} \in \{0, 1\} \qquad (i, j) \in A \end{aligned} \tag{15}$$

where $J_i^g\left(b^i\right)$ is obtained by Eq. (8) and is the cost-to-go value of starting from node b^i when the goal is node B_{goal}^j. This aTSP can be solved using methods such as Concorde, LKH or metaheuristic optimization algorithms [6,43,50].

3.2.3. Incorporating obstacles in aTSP

In this paper, we have considered three types of obstacles. In the first type, the position of them is known based on the initial information of the environment. Second, the regions which are potentially obstacle and there is no certain information about them, F_{sus}, and the robot prefers to avoid this area. Finally, the unknown obstacles which there is no information about them and the robot will detect them during its movement (online phase). In order to consider the second type of obstacles in the robot's decision making, the one-step-cost is defined as:

$$\begin{cases} c_i(b,u) = 0, & \text{if } \left(\left(b \in B_{goal}^i\right) \text{or } (F \text{ happens})\right) \\ & \quad \text{and } (b \notin F_{sus}) \\ c_i(b,u) \geq \beta > \varepsilon > 0, & \text{if } \quad (b \in F_{sus}) \\ \beta > c_i(b,u) \geq \varepsilon > 0, & \text{if } \quad \text{otherwise} \end{cases} \tag{16}$$

where ε and β are positive values. Like Eq. (12), the one-step-cost is set to a zero value in the goal or failure region, and a positive value for other situations. However, a higher one-step-cost value is selected for potential obstacle areas to help the robot to avoid these regions as possible. Now, the cost-to-go between every two goal points is calculated using (8), and then they are used in solving the aTSP in (15).

4. TSP-FIRM For nonholonomic robots

In this section, the generic TSP-FIRM is adapted to the group of nonholonomic systems with unicycle dynamics. Here, we assume Gaussian noise for the system and design the estimator and controllers based on this assumption. The challenging part is to design controllers for the point to point stabilization and the path tracking in the belief space.

4.1. Controllability of nonholonomic systems

In the roadmap based algorithm such as PRM, it is assumed that there exists a controller to drive the robot on an edge from the starting point to the end point or its vicinity. The discrete model of a unicycle robot is

$$X_{k+1} = f(X_k, u_k, w_k) = \begin{pmatrix} x_k + (V_k + n_v)\delta t \cos\theta_k \\ y_k + (V_k + n_v)\delta t \sin\theta_k \\ \theta_k + (W_k + n_w)\delta t \end{pmatrix} \tag{17}$$

where $X_k = (x_k, y_k, \theta_k)$ is the robot state at the time step k. $u_k = (V_k, W_k)^T$ is the control vector where V_k and W_k are the linear and angular velocity of the robot, respectively. δt is the time step and the vector $w_k = (n_v, n_w)^T \sim \mathcal{N}(0, \mathbf{Q}_k)$ is the motion noise of the robot. The noise of the system is assumed to be Gaussian. It can be shown that for the posture stabilization of the robot a discontinuous or time-varying controller is required [34]. However, for trajectory tracking a linear controller can be used if $V_k \neq 0$ or $W_k \neq 0$.

In the belief space, the posture stabilization and the trajectory tracking are more challenging, because the controller should drive the robot to a neighbor of the belief node. Considering a unicycle robot with sensors to measure the range and bearing from landmarks, linearizing the motion and observation model provide an observable but uncontrollable system for

the posture stabilization. Therefore, in this paper, we use a Kalman filter and a switching controller to drive the system to a belief node.

4.2. Switching based TSP-FIRM algorithm

For implementing the TSP-FIRM algorithm for a nonholonomic robot, it is necessary to determine proper nodes and local controllers. It is also required to compute the transition probabilities and costs. In the following, we describe the construction of the TSP-FIRM graph, based on the Kalman filter and switching controller.

4.2.1. Estimator design

The extended Kalman filter is used for the state estimation or in other words, for the localization of the robot. Thus, the belief dynamics is obtained as $b_{k+1} = \tau\left(b_k, u_k, z_{k+1}\right)$.

State space model: By linearizing the system around the PRM node $\mathbf{v}$, the state space model can be represented by

$$\begin{aligned} x_{k+1} &= \mathbf{A}x_k + \mathbf{B}u_k + \mathbf{G}w_k, \quad w_k \sim \mathcal{N}(0, \mathbf{Q}) \\ z_k &= \mathbf{H}x_k + \mathbf{M}v_k, \qquad\quad v_k \sim \mathcal{N}(0, \mathbf{R}) \end{aligned} \tag{18}$$

where w_k and v_k are motion and observation noise, respectively, having a zero mean Gaussian distribution with covariances $\mathbf{Q}$ and $\mathbf{R}$.

By defining the matrix $\check{\mathbf{Q}}$ such that ,$\mathbf{G}\mathbf{Q}\mathbf{G}^T = \check{\mathbf{Q}}\check{\mathbf{Q}}^T$ and choosing systems with the controllable pair $\left(A, \check{Q}\right)$ and observable pair (A, H), it can be concluded that using the Kalman filter and according to its stationary behavior (stationary Kalman filter), the estimation covariance converges to the following matrix P_s which is independent of its initial covariance

$$P_s = P_s^- - P_s^-\mathbf{H}^T\left(\mathbf{H}P_s^-\mathbf{H}^T + \mathbf{M}\mathbf{R}\mathbf{M}^T\right)^{-1}\mathbf{H}P_s^- \ , \tag{19}$$

where P_s^- is the unique, symmetric and positive semidefinite solution of the Discrete Algebraic Riccati Equation (DARE)

$$P_{k+1}^- = \mathbf{A}\left(P_k^- - P_k^-\mathbf{H}^T\left(\mathbf{H}P_k^-\mathbf{H}^T + \mathbf{M}\mathbf{R}\mathbf{M}^T\right)^{-1}\mathbf{H}P_k^-\right)\mathbf{A}^T + \mathbf{G}\mathbf{Q}\mathbf{G}^T. \tag{20}$$

The estimation mean is computed as follows at each time step

$$\hat{x}_{k+1}^+ = (I - \mathbf{K}\mathbf{H})\mathbf{A}\hat{x}_k^+ + (I - \mathbf{K}\mathbf{H})\mathbf{B}u_k + \mathbf{K}z_{k+1} + (I - \mathbf{K}\mathbf{H})(I - \mathbf{A})\mathbf{v}, \tag{21}$$

where $\mathbf{K} = P_s^-\mathbf{H}^T\left(\mathbf{H}P_s^-\mathbf{H}^T + \mathbf{M}\mathbf{R}\mathbf{M}^T\right)^{-1}$. It is clear from (21) that the estimated mean is a function of observations, thus it evolves randomly.

4.2.2. Switching based controller in the belief space

As stated before, the estimation covariance of the system using an stationary Kalman filter (SKF) converges to P_s. Therefore, by designing a feedback controller to control the estimation mean and assuming that the system remains in the valid linearization region, the belief will reach to $b_c = (\mathbf{v}, P_s)$ where b_c is in the stopping region of the belief space.

In order to design a feedback controller, a switching controller is adopted as follows to steer the belief, $\hat{X}_k^+ = \left(\hat{x}_k^+,\ \hat{y}_k^+,\ \hat{\theta}_k^+\right)$, to the PRM node $\mathbf{v} = \left(v_x,\ v_y,\ v_\theta\right)$

$$\begin{cases} \text{DFL Controller}, & \text{if} \quad e_p \geq e_{th} \\ \text{Heading-Angle Controller}, & \text{if} \quad e_p < e_{th}, \end{cases} \tag{22}$$

where e_p is defined as $e_p = \sqrt{(\hat{x}_k^+ - v_x)^2 + (\hat{y}_k^+ - v_y)^2}$ and e_{th} is a positive value that depends on the robot size and the stopping region radius. Details about Dynamic Feedback Linearization (DFL) controller can be found in [34]. In the heading angle controller, the following control inputs are proposed:

$$V = 0; \quad W = K_\theta \tanh k_\theta \check{\theta}_k^+ \tag{23}$$

where K_θ and k_θ are positive values.

4.3. TSP-FIRM nodes

The first step in constructing the TSP-FIRM graph is choosing the goal points in the environment. Then, the underlying PRM including the goal points is constructed, which is known by its nodes and edges, $\{\{\mathbf{v}\}, \{\mathbf{e}^{ij}\}\}$. The next step is to linearize the system about the PRM nodes, $\mathbf{v}^j$, and then designing a stationary Kalman filter and a switching based belief controller called as jth node controller. Consequently, the belief node B^j is generated. B^j is an ϵ-ball in the belief space with the center $b_c^j \equiv (\mathbf{v}^j, P_s^j)$ where P_s^j is the covariance of the SKF obtained in (19). Mathematically, B^j can be defined as follows:

$$B^j = \left\{b \equiv (x, P) : \left\|x - \mathbf{v}^j\right\| < \delta_1,\ \left\|P - P_s^j\right\|_m < \delta_2\right\}, \tag{24}$$

where $\|\cdot\|$ and $\|\cdot\|_m$ are proper vector and matrix norms, respectively. δ_1 and δ_2 determine the TSP-FIRM nodes size and should be chosen sufficiently small that the approximation (7) stablishes.

Since the goal points are chosen manually and independent of the TSP-FIRM construction algorithm, it is possible that the system is not observable at that point. We will describe shortly how to cope with this limitation.

4.4. Local controllers

The local controller μ^{ij} is the integration of the edge controller and the node controller, which is responsible for steering the belief from the node B^i to the node B^j. The edge controller $\bar{\mu}_k^{ij}$ is responsible for driving the belief from B^i to the vicinity of the node B^j. Then, the switching based node controller described earlier, is activated. Unlike the posture stabilization, the unicycle robot is linearly controllable along the PRM edge if the linear or angular velocity is greater than zero. A linear time-varying LQG controller is adopted in order to track the PRM edge e^{ij}. For this purpose, a sequence of nominal inputs $\mathbb{U}^p = \{u_k^p\}$ are designed firstly to drive the robot from the PRM node $\mathbf{v}^i$ to the node $\mathbf{v}^j$. By applying the inputs u_k^p, a series of nominal states, $\mathbb{X}_p = X_p^k$, are produced. The cost function of the LQR controller for tracking nominal states is as:

$$\begin{aligned} J_{LQR} &= \mathbb{E}\left[\sum_{k\geq 0}\left(\hat{x}_k^+ - x_k^p\right)^T W_x\left(\hat{x}_k^+ - x_k^p\right) + \left(\hat{u}_k - u_k^p\right)^T W_u\left(\hat{u}_k - u_k^p\right)\right] \\ &= \mathbb{E}\left[\sum_{k\geq 0}\left(\hat{e}_k^+\right)^T W_x\left(\hat{e}_k^+\right) + (\delta u_k)^T W_u(\delta u_k)\right]. \end{aligned} \tag{25}$$

By defining $\check{W}_x^T\check{W}_x = W_x$, if the pair (A, B) is controllable and the $\left(A, \check{W}_x\right)$ is observable, the linear controller input minimizing the above cost function is obtained as follows:

$$\delta u_k = -L_k\hat{e}_k^+ \ , \tag{26}$$

where the feedback gain L_k is obtained using the following equations:

$$L_k = \left(B_k^T S_{k+1} B_k + W_u\right)^{-1} B_k^T S_{k+1} A_k \tag{27}$$

$$S_k = C^T W_x C + A_k^T S_{k+1} A_k - A_k^T S_{k+1} B_K L_k \quad . \tag{28}$$

Eq. (28) is the Discrete Algebraic Riccati Equation (DARE) computed recursively. The initial condition of the Eq. (28) is ,$S_N = W_x$ where N is the length of the nominal path. The final control input in the online phase can be computed as:

$$u_k^p + \delta u_k = u_k^p - L_k\hat{e}_k^+ = u_k^p - L_k\left(\hat{x}_k^+ - x_k^p\right) \quad . \tag{29}$$

4.5. Transition probabilities and costs

Although computing the transition probability $\mathbb{P}\left(\cdot \mid B^i, \mu^{ij}\right)$ and transition cost $C(B^i, \mu^{ij})$ is time-consuming, it can be tolerated due to the offline construction of TSP-FIRM graph. In order to compute the collision and absorption probabilities, the sequential Monte Carlo method is adopted. In defining the transition cost function, estimation accuracy has the main role because not only does the estimator operate better but also the controller can have better performance. To involve the estimation accuracy in the cost function, we use the weighted trace of the estimation covariances as $\Phi^{ij} = \mathbb{E}[\sum_{k=1}^{\mathcal{T}} tr(\lambda P_k^{ij})]$ where P_k^{ij} is the estimation covariance at the kth time step during running the local controller μ^{ij} and λ is defined as $\lambda = diag\left([\lambda_x, \lambda_y, \lambda_\theta]\right)$. Another factor that should be considered in the cost function is the mean stopping time of local controller, $\hat{\mathcal{T}}^{ij} = \mathbb{E}\left[\mathcal{T}^{ij}\right]$, to speed-up the search. Finally, we consider the mean time of spending in the high-risk area, i.e potential obstacle area, $\hat{\mathcal{T}}_{obs}^{ij} = \mathbb{E}\left[\mathcal{T}^{ij}\right]$, to reduce it. By a linear combination of the mentioned factors with the proper coefficients ξ_P, $\xi_\mathcal{T}$ and ξ_O the cost of invoking μ^{ij} at B^i (transition cost), is obtained as:

$$C\left(B^i, \mu^{ij}\right) = \xi_P\Phi^{ij} + \xi_\mathcal{T}\hat{\mathcal{T}}^{ij} + \xi_O\hat{\mathcal{T}}_{obs}^{ij} \quad . \tag{30}$$

4.6. Switching-based TSP-FIRM construction and planning

Algorithm 1 presents the details of the TSP-FIRM graph construction and solving the aTSP. After constructing TSP-FIRM graph, the DP Eq. (8) is solved between every two goal points, and the cost-to-go and policy is computed. The computed cost-to-go values are used to form aTSP cost matrix. By solving the aTSP, the overall search policy is obtained. Algorithm 2 presents the steps for executing the search policy obtained in the Algorithm 1. If the initial belief b_0 belongs to the TSP-FIRM graph, but not to any goal nodes, first we find the best path to one of the goal nodes. If the initial belief b_0 does not belong to any TSP-FIRM graph nodes, first we connect it to the graph. The outgoing edges from the corresponding node, B^0, are denoted by $\mathbb{M}(0)$. Then, the transition cost $C^g(B^0, \mu)$, the transition probabilities $\mathbb{P}^g\left(B^j \mid B^0, \mu\right)$ and the failure probability $\mathbb{P}\left(F \mid B^0, \mu\right)$ for invoking each local controller, $\mu \in \mathbb{M}(0)$, is computed. Then the following equation is solved for each goal node

$$\pi^g\left(B^0\right) = \underset{\mu\in\mathbb{M}(0)}{argmin}\, C_i^g\left(B^0, \mu\right) + J^g(F)\mathbb{P}^g\left(F \mid B^0, \mu\right) + \sum_j \mathbb{P}^g\left(B^j \mid B^0, \mu\right)J^g\left(B^j\right) \quad . \tag{31}$$

Algorithm 1: TSP-FIRM graph construction for a nonholonomic robot.

input : Free space map, $\mathbb{X}_{free}$
output: TSP-FIRM Graph $\mathcal{G}$
Get the set of search points, $\left\{\mathbf{v}^i_{goal}\right\}$;
Construct a PRM with nodes $\mathcal{V} = \left\{\mathbf{v}^j\right\}$ and edges $\varepsilon = \left\{e^{ij}\right\}$ where $i, j = 1, \ldots, N_s$;
Add search points to PRM nodes $\mathcal{V} \leftarrow \left\{\mathbf{v}^i_{goal}, \mathcal{V}\right\}$ and update the edges $\varepsilon = \left\{e^{ij}\right\}$ where $i, j = 1, \ldots, N_g + N_s$;
forall the *PRM nodes* $\mathbf{v}^j \in \mathcal{V}$ **do**
  Design the switching-based node-controller μ^j for stabilizing the system at $\mathbf{v}^j$;
  Generate TSP-FIRM node B^j using Eq. (24) with the center $b^j_c = \left(\mathbf{v}^j, P^j_s\right)$(Computed using Eq. (19)) ;
end
Collect all TSP-FIRM nodes $\mathbb{V} = \left\{B^i\right\}$;
forall the $\left(B^i, e^{ij}\right)$ *pairs* **do**
  Design the edge-controller $\bar{\mu}^{ij}_k$;
  Concatenate edge controller $\bar{\mu}^{ij}_k$ and node-controller μ^j_k to construct the local controller μ^{ij}_k;
  Set the initial belief b_0 equal to the center of B^i (based on the approximation in Eq. (7));
  Compute transition probabilities $\mathbb{P}^g\left(B^j \mid B^i, \mu^{ij}\right)$, transition cost $C^g\left(B^i, \mu^{ij}\right)$ and failure probabilities $\mathbb{P}^g\left(F \mid B^i, \mu^{ij}\right)$;
end
Collect all local controllers $\mathbb{M} = \left\{\mu^{ij}\right\}$;
forall the $\left\{v^i_{goal}\right\}$ **do**
  Solve the graph DP in Eq. (8) for all $j \neq i$ to compute pair$S_i = \left(\pi^{g^*}_i, J^{g^*}_i\right)$to take the robot to the B^j_{goal};
end
Collect all pairs $\mathbb{S} = \left\{S^*_i\right\}$;
Construct TSP cost matrix and solve it, $Tour^*$;
$\mathcal{G} = (\mathbb{V}, \mathbb{M}, \mathbb{S}, Tour^*)$;
return $\mathcal{G}$

Afterwards, the best initial policy is chosen to drive the robot to one of the goal points. Then, based on the policy obtained from the aTSP, the robot starts searching goal points.

During the online phase execution, Algorithm 2, it is possible that the robot finds new obstacles. Algorithm 3 is presented to handle this situation which can be executed online. In this algorithm, if the transition cost, transition probability and failure probability changes are more than $\alpha_{\min}$ or an edge of the graph hits the obstacle, replanning is required. If these changes are more than $\alpha_{\max}$ or a goal node is in the obstacle region, resolving the aTSP is required too.

Remark 1. In Algorithm 3, in the case of resolving the TSP, if it is required that the robot returns to its starting position, the TSP cost matrix should be changed. We show the current position and initial position of the robot by B^r and B^s respectively. A dummy node, B^d, is added to the TSP where the cost of connecting edge of this node to B^r and the connecting edge of B^s to B^d are zero. The cost of other outgoing and incoming edges of B^d are set to a high value. The solution of this TSP will be a route including the $B^s - B^d - B^r$. Therefore, by starting from B^r, the robot reaches the starting point. In another case, if returning to the start point is not important, the cost of connecting edges of other nodes to B^r are set to zero.

Remark 2. Current belief of the robot in the replanning can be added to the graph permanently (to extend the graph) if its current belief covariance is close to the covariance of the robot system at that point.

5. Experimental results

In this section, the algorithms presented in the previous sections are implemented in a robotic simulation software and on a physical robot in order to illustrate their applicability and performance. These experiments are designed such that the ability of the algorithms in finding the search and exploration policy as well as its robustness and flexibility in situations like finding new obstacles, deviation from the planned path, observation loss, and kidnapping can be assessed. The simulation is done in Webots, a robot simulator, and on the e-Puck robot. The real experiment is performed in a floor of the Electrical Engineering department at K. N. Toosi University of Technology and on a robot named Melon.

Algorithm 2: Planning on the TSP-FIRM graph for a nonholonomic robot.

input : Initial belief b_0, TSP-FIRM Graph $\mathcal{G}$, Underlying PRM graph
if $\exists B^i \in \mathbb{V}$ *such that* $b_0 \in B^i$ **then**
  Invoke the best policy to take the robot into one of the goal nodes, $B_{start} = B^i_{goal}$;
else
  Generate a new TSP-FIRM node B_0 based on b_0;
  Connect B_0 to the graph, design its local planners and collect them in set $\mathbb{M}(0) = \{\mu^{0j}\}$;
  forall the $\mu \in \mathbb{M}(0)$ **do**
    Compute the transition cost $C^g(B^0, \mu)$, the transition probabilities $\mathbb{P}^g(B^j \mid B^0, \mu)$and the failure probability $\mathbb{P}(F \mid B^0, \mu)$;
  end
  forall the *Goal nodes* **do**
    Solve Eq (31) for the B^0 to choose the best initial local planner μ^{0j};
  end
  Invoke the best policy to take the robot into one of the goal nodes, $B_{start} = B^i_{goal}$;
end
Cur_Goal ← The neighbor of B_{start} in the *Tour** as the next goal;
forall the *Goal nodes* **do**
  while $B \neq$ *Cur_Goal* **do**
    while *(* $\nexists B^j, s.t. b_k \in B^j$ *) and "no collision"* **do**
      Apply the robot the control input $u_k = \mu^{ij}_k(b_k)$;
      Get the sensor measurement z_{k+1};
      if *Collision happens* **then return** Collision;
      Update belief: $b_{k+1} = \tau\left(b_k, \mu^{ij}_k(b_k), z_{k+1}\right)$;
    end
    Modify the current FIRM node $B_i = B_j$;
    Assign the next local controller based on the policy, $\mu^{ij} = \pi^g(B^i)$;
  end
  Cur_Goal ← The neighbor of *Cur_Goal* in the *Tour** as the next goal;
end

5.1. Implementation details

5.1.1. TSP-FIRM elements

Environment: The map of the environment is generated by MATLAB, and it generates a similar one in Webots. The markers in the experiments are black and white patterns based on a modified Hamming code and each one has a unique id. The ArUco library, provided by Garrido-Jurado et al. [17], detects each visible marker's id and provides the relative range and bearing from it.

Motion model: Both the e-Puck and the Melon robot have the unicycle dynamics as Eq. (17). The vector $w_k = (n_v, n_w)^T \sim \mathcal{N}(0, \mathbf{Q}_k)$ is the motion noise of the system and is a combination of a fixed uncertainty and a term proportional to the control input values. The motion noise covariance is

$$\mathbf{Q}_k = \mathbf{diag}\left(\left(\eta_v V_k + \sigma_b^V\right)^2, \left(\eta_w W_k + \sigma_b^w\right)^2\right) \tag{32}$$

where η_v, η_w, σ_b^V and σ_b^w are constant values.

Sensor model: As mentioned before, the robot can compute its relative range and bearing from the markers. Therefore, if we show the jth landmark in the global coordinate by ${}^jL$, the sensor model can be written as follows:

$${}^jz_k = \left[\left\|{}^j\mathbf{d}_k\right\|, atan2\left({}^jd_{2_k}, {}^jd_{1_k}\right) - \theta\right]^T + {}^j\nu, {}^j\nu \sim \mathcal{N}\left(\mathbf{0}, {}^j\mathbf{R}\right), \tag{33}$$

where ${}^j\mathbf{d}_k = \left[{}^jd_{1_k}, {}^jd_{2_k}\right]^T := [x_k, y_k]^T - L_j$. The random vector ${}^j\nu$ represents the measurement noise in the measurement of the jth landmark which is proportional to the relative distance to the landmark and the angle between the wall and the line connecting the camera to the landmark, $\phi \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$. Consequently, the sensor noise is a zero mean Gaussian noise with the following covariance:

$${}^j\mathbf{R}_k = \mathbf{diag}\left(\left(\eta_{r_d}\left\|{}^j\mathbf{d}_k\right\| + \eta_{r_\phi}|\phi_k| + \sigma_b^r\right)^2, \left(\eta_{\theta_d}\left\|{}^j\mathbf{d}_k\right\| + \eta_{\theta_\phi}|\phi_k| + \sigma_b^\theta\right)^2\right), \tag{34}$$

Algorithm 3: TSP-FIRM graph updating in finding new obstacles.

Estimate obstacle position;
Update map;
$F \leftarrow$ Retrieve surrounding edges of the obstacles;
forall the *edges,* $\mu \in F$ **do**
  $\boldsymbol{f} = \left[C^g\left(B^i, \mu\right), \mathbb{P}^g\left(B^j \mid B^i, \mu\right), \mathbb{P}^g\left(F \mid B^i, \mu\right)\right]$;
  Recompute the transition cost , transition probability and collision probability
  $\boldsymbol{f_{new}} = \left[C^g_{new}\left(B^i, \mu\right), \mathbb{P}^g_{new}\left(B^j \mid B^i, \mu\right), \mathbb{P}^g_{new}\left(F \mid B^i, \mu\right)\right]$;
end
if *any TSP-FIRM edge intersect obstacles* **then**
  if *any search node is in the obstacle area* **then**
    Delete all TSP-FIRM nodes in the obstacle area ;
    $newTSP \leftarrow true$;
  end
  Delete all edges and nodes intersect obstacles; Update F and corresponding **f**;
  $newPlanning \leftarrow true$;
end
if *exists* $\mu \in F$ *such that* $|\boldsymbol{f}_{new} - \boldsymbol{f}| \not< \boldsymbol{\alpha}_{\min}$ **then**
  if *exists* $\mu \in F$ *such that* $|\boldsymbol{f}_{new} - \boldsymbol{f}| \not< \boldsymbol{\alpha}_{\max}$ **then**
    $newTSP \leftarrow true$;
  end
  $newPlanning \leftarrow true$;
end
if *newPlanning* **then**
  Replace previous transition costs, transition probabilities and collision probabilities with new computed values;
  if *newTSP* **then**
    Recompute π^{g^*} and its corresponding J^{g^*} between each search node;
    Construct TSP cost matrix, solve it and assign new goals sequence;
  end
  Replan($b_{current}$);
else
  Graph does not change
end

where η_{r_d}, η_{θ_d}, η_{r_ϕ}, η_{θ_ϕ}, σ_b^r and σ_b^θ are constant values. Owing to the independence of the measurements, the full vector of measurements $z = \left[{}^{i_1}z^T, \ldots, {}^{i_r}z^T\right]^T$ is the concatenation of visible landmarks $\left\{L_{i_1}, \ldots, L_{i_r}\right\}$ which its model can be written as $z = h(x) + v$ where $v \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ and $\mathbf{R} = diag\left({}^{i_1}\mathbf{R}, \ldots, {}^{i_r}\mathbf{R}\right)$.

One-step-cost: The one-step-cost is defined as (30) and if the robot hits an obstacle the cost-to-go is set to a high value. *TSP-FIRM node:* As mentioned in Section 4, the TSP-FIRM node can be defined mathematically as (24). In the experiments, the TSP-FIRM node must satisfy two following conditions:

$$\begin{aligned} \left|x - v^j\right| &< \boldsymbol{\Delta_1} \\ \left|diag(P - P_s^j)\right| &< \boldsymbol{\Delta_2}, \end{aligned} \tag{35}$$

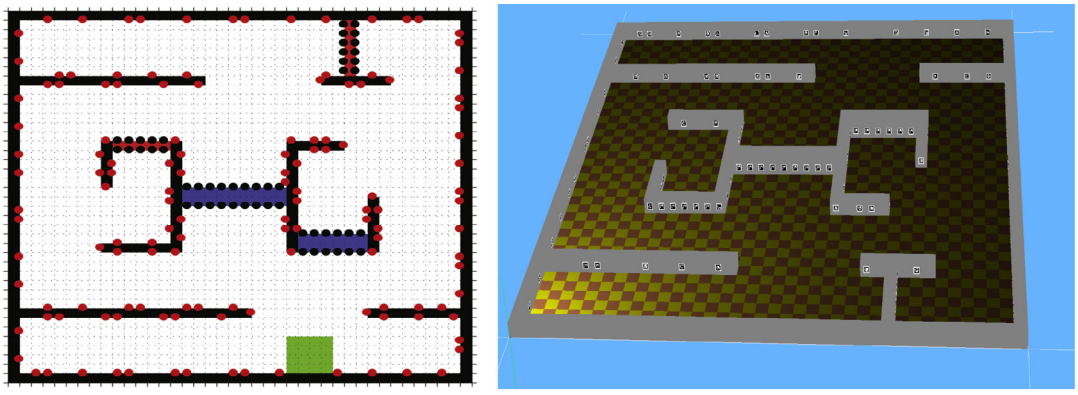
where Δ_1 and Δ_2 are proper vectors. The $|A| < |B|$ operator here means the element-wise comparison of the absolute value of A and B.

Local controller: For tracking nominal states (path) the LQG controller described in (4.4) is used and the switching based controller explained in (4.2) is adopted for designing stabilizers.

5.1.2. Algorithm robustness

There are some situations, including finding new obstacles, deviating from planned path, kidnapping and becoming highly uncertain about the position that the robot has to leave the execution of the policy obtained in the offline mode and update its map, graph, and policy.

Highly uncertain about position: As mentioned before, it is possible that the observability condition in a TSP-FIRM node (explained in (4.2)) is not met . In this situation, we take the worst possible observation and design the stabilizers based on it. Consequently, it cannot be guaranteed that the robot converges to the P_s. Moreover, it is probable that the robot cannot observe any marker in its way due to reasons such as covered markers by objects or the camera fault. Therefore,



(a) (b)

Fig. 1. Environment map in: (a) Matlab. (b) Simulation. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

the covariance matrix P_s grows up. To handle these situations, the robot stops the local controller and starts gathering information from its environment and continues it until to get a better certainty about its position and the P_s decreases.

Information gathering phase: In the information gathering phase, the robot moves slightly to observe landmarks while avoiding obstacles. The information gathering continues until the covariance matrix decreases and the mean of estimation improves. This condition can be checked by a suitable norm of matrix P_s. When the robot exits from this phase, does a replanning in its new belief (resolves aTSP if required) and continues to move based on the graph policy.

Remark 3. In the information gathering phase, the system is linearized around the estimated mean of the robot.

Deviation from the planned path: During the robot movement, it is probable that the robot deviates from its path due to the poor localization, slipping or sensor error. In order to check the occurrence of this situation, the difference between the desired state X_k^p, and the estimation mean $\hat{x}_k^+$, $|\hat{x}_k^+ - X_k^p|$ is computed in each step. If an element of this vector exceeds the threshold value for more than a specific time, $|\hat{x}_k^+ - X_k^p| \not< \delta_{\max}$, the deviation is detected. Then, a replanning is done to steer the robot toward the goal.

Finding new obstacles: In the online phase, if the robot detects a new obstacle, it should estimate the obstacle position, update its map and graph, and finally update its policy. In the experiments, we use an identical marker for unknown obstacles. After updating the map, the edges near to the obstacle are selected and according to the Algorithm 3 the graph is updated. The obstacle may affect the robot's observation too, thus the transition probability, failure probability and the transition costs are recomputed. If resolving the TSP is not required, the robot only does a replanning, else it must resolve aTSP before replanning.

Kidnapping: In order to assess the robot ability in recovering from a catastrophic localization failure when the robot is suddenly moved to an arbitrary location, the kidnapping is considered in our implementations. To handle this situation, the robot should detect the kidnapping, localize itself and do the replanning. In the kidnapping, we do not consider resolving TSP; however, it is easy to check its condition and resolve it to get a new policy for searching the goal points.

5.2. Simulation in webots

In the following, we present the simulation results which is done in Webots software and on the e-Puck robot. In this simulation, the robot finds a policy to search all the goal points and starts from the initial node and returns to it at the end of the search.

Environment map: The Fig. 1a shows the constructed environment which its size is $2\text{m} \times 2\text{m}$. The blocks in black, blue, green and red show the known obstacles, potentially true obstacles, potentially false obstacles and completely unknown obstacles, respectively. The red points are the markers mounted on the wall, and the black points are markers with the same id to detect the unknown obstacles. Fig. 1b shows the corresponding environment in the Webots.

Motion and sensor model: The motion model of the robot is as (17) and the motion noise is as (32) with the parameters $\eta_v = 0.1$, $\eta_w = 0.01$. $\sigma_b^V = 1$ cm/s and $\sigma_b^w = 0.02\ rad/s$. The sensor model and the measurement noise are as (33) and (34) with the parameters $\eta_{r_d} = 0.1$, $\eta_{\theta_d} = 0.001$, $\eta_{r_\phi} = 0.1$, $\eta_{\theta_\phi} = 0.01$, $\sigma_b^r = 6$ and $\sigma_b^\theta = 0.06$.

TSP-FIRM elements: The transition cost is according to (30) and the weighting factor λ is $diag([0.5, 0.5, 2000])$. The computations in this simulation are based on the *cm* and *rad*. TSP-FIRM nodes are obtained by Eq. (35) in which $\Delta_1 = [5, 5, 0.2]$ and $\Delta_2 = [5^2, 5^2, 0.2^2]$.

Local controller: The LQG controller weights are set as $S_N = diag([0.003, 0.003, 0.03])$, $W_x = diag([0.003, 0.003, 0.03])$ and $W_u = diag([0.1, 0.1])$. In the switiching controller (stabilizer), the orientation controller parameters are $K_\theta = 2$, $k_\theta = -1$. The

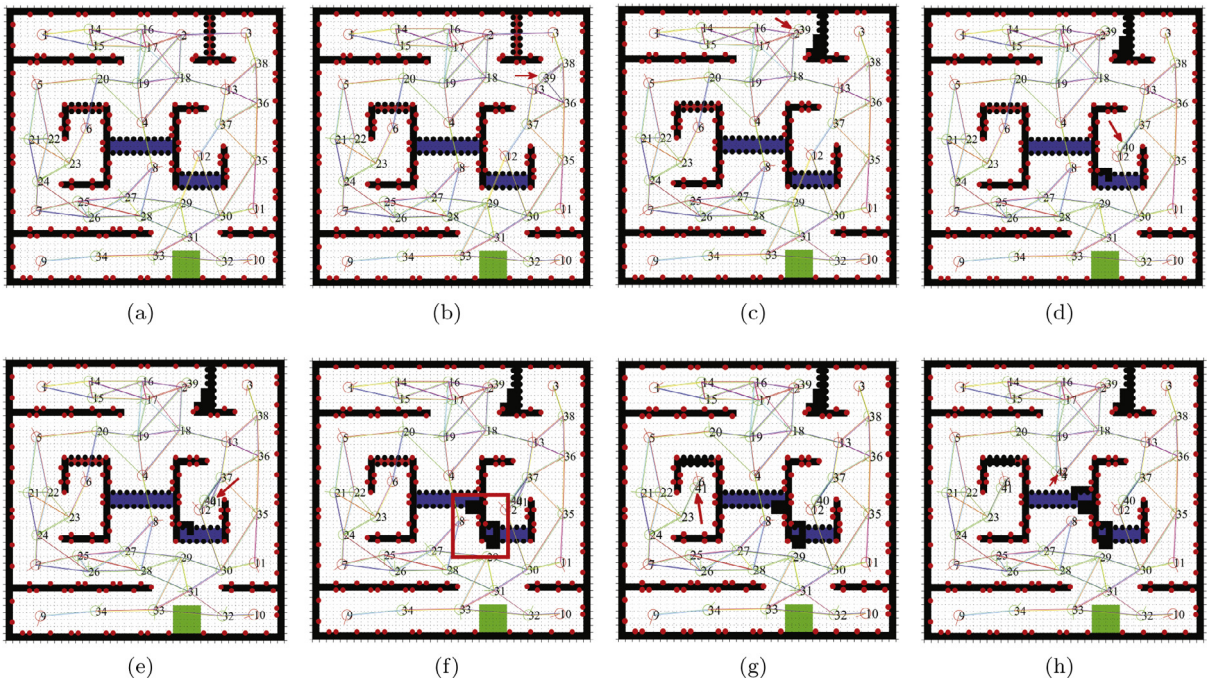


Fig. 2. Map and graph updates in simulation. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

DFL controller is proposed in [34] and its parameters are set as $k_{p1} = 2$, $k_{p2} = 12$, $k_{d1} = 3$, $k_{d2} = 7$. $e_{th} = 4\,\text{cm}$ also determines when the switching should happen.

Later, in the implementation on the physical robot, we will discuss more about setting parameters.

5.2.1. Simulation results

Offline phase: The Fig. 2a shows the constructed graph including the sampled nodes, the goal nodes and the edges. The goal nodes are [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13] shown by the red points in the graph. The transition cost, transition probability and the failure probability are computed using Monte Carlo method. Then, based on the Algorithm 1, the TSP matrix is formed and solved. By solving the aTSP, the route [1, 2, 3, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 1] is obtained. Consequently, the robot motion sequence is obtained as following in the offline mode.

[**1**, 14, 16, **2**, **3**, 38, **13**, 37, **12**, 29, 30, **11**, 30, 31, 32, **10**, 32, 33, 34,
9, 34, 33, 29, 27, **8**, 27, 25, **7**, 24, 23, **6**, 20, **5**, 20, **4**, 18, 17, 14, **1**]

Online phase: The robot is placed approximately near node 1. It starts moving toward node 2 and continues up to node 16, according to the offline policy. During moving on the edge connecting node 16 to 2, the robot is kidnapped. The robot detects the kidnapping situation and starts gathering information, thus localize itself. Node 39 is added to the graph temporarily (Fig. 2b) and the replanning is done. Then, the robot moves toward node 2 on the route [39,13,18,2]. After reaching node 2, the robot moves toward node 3, but it detects some unknown obstacles. Therefore, the edges on the obstacle are removed, and the current belief is added as permanent node 39 to the graph (according to the Remark 2). The TSP-FIRM graph is updated (Fig. 2c) and the transition cost, the transition probability and the failure probability of the new edges and the edges surrounding the obstacle is computed. By computing the cost-to-go value from node 39 to node 3, it is concluded that resolving aTSP is required to find a new policy for searching the goal nodes. aTSP matrix is formed by considering returning to the start point condition (explained in Remark 1) and solved. The new search policy is obtained as [39, 13, 3, 12, 11, 10, 9, 8, 7, 6, 5, 4, 1]. According to the new policy, the robot visits node 13 and 3 respectively. Then, it moves toward node 12, but near this point, detects a new obstacle. Node 40 is added to the graph permanently and the graph is updated (Fig. 2d). The replanning is done and the robot continues its way (resolving aTSP is not required). After reaching node 2, the next goal is node 11. However, on the connecting edge between node 12 and 37, the robot deviates from the planned path. The current belief is added to the graph as a temporary node 41 (Fig. 2e). After replanning, the robot moves toward node 11 and returns to its path and visits nodes 11, 10, 9, 8 and 7, respectively. It also detects some new obstacles and updates its map (Fig. 2f). Since these new obstacles do not change the transition probability and cost as well as the failure probability significantly, replanning is not required. Afterward, the robot moves from node 7 to 6. Near node 6, new obstacles are detected and after updating the map, permanent node 41 is added to the graph (Fig. 2g). The robot does replanning without resolving aTSP, then moves toward nodes 6, 5 and 4, respectively. Near node 4, the robot detects new

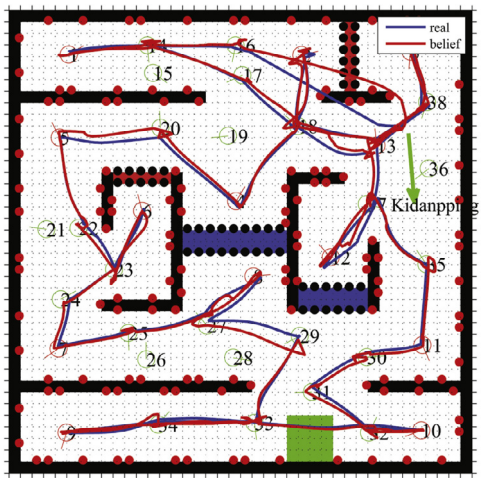


Fig. 3. Robot's beliefs and real positions during online phase. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

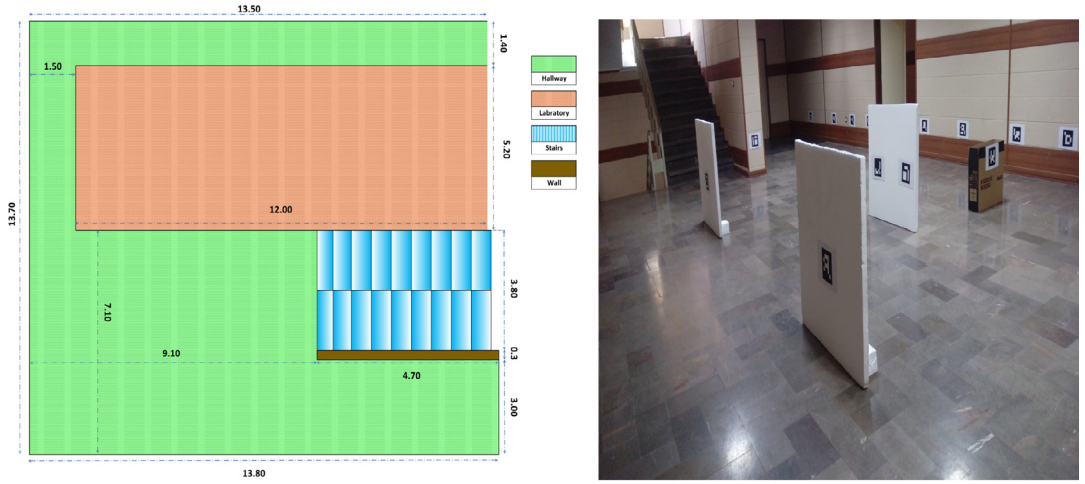


Fig. 4. Real experiment environment: (a) layout (b) overview. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

obstacles and updates its map. Permanent node 42 is added to the graph (Fig. 2h). After replanning (resolving aTSP is not required), the robot visits node 4 and returns to the start point, node 1. Fig. 3shows the real path traversed by the robot (blue) and its belief (red). In kidnapping situation, their difference is more clear. The video of this simulation is available in [32].

5.3. Experiment on a physical robot

In the real experiments, the robot starts from in front of the laboratory and searches the goal points in the hallway and returns to the laboratory. We consider the kidnapping and unknown obstacles in this experiment.

Environment map: The map of the floor of Electrical Engineering department where the experiment is done, is shown in Fig. 4a. Fig. 4b depicts an overview of the real environment. The environment size is approximately 14m $\times$ 14m, and the green area shown in Fig. 4a is used for the experiment. The sizes that are given in Fig. 4a are not accurate that causes more uncertainty in the map.

Motion and sensor model: The motion and sensor model of the robot should be close to the real models. The motion model and its noise model are as Eqs. (17) and (32), respectively. The large values of η_V and η_w mean that noise of linear/angular velocity significantly increases as the linear/angular velocity increases. σ_b^V and σ_b^w also show the fixed uncertainty in the model. In the sensor model, the measurement noise increases as the robot observes the markers from long distance and tight angle of view. Higher η_{r_d}, η_{θ_d}, η_{r_ϕ} and η_{θ_ϕ} values show that uncertainty significantly increases with the distance

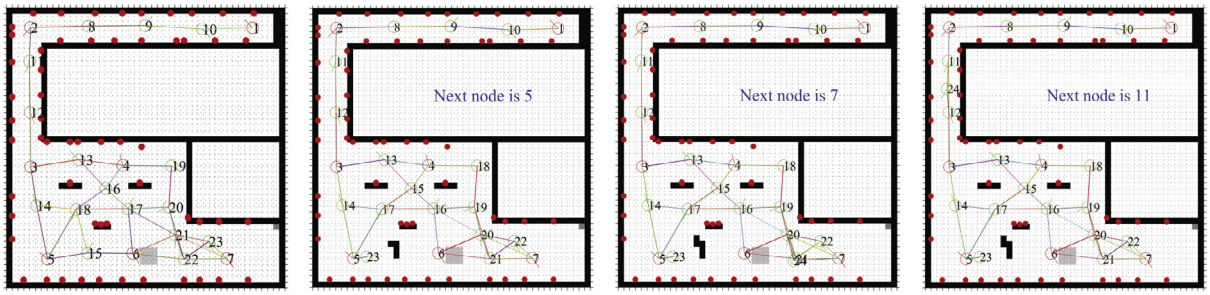


Fig. 5. Map and graph updates in real experiment. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

and angle of view value. By setting higher values to the mentioned parameters and considering higher uncertainty values, results into more conservative planning. In contrast, parameters that produce less uncertainty makes planning unreliable. In this experiment, the motion model parameters are $\eta_v = 0.1,\ \eta_w = 0.01,\ \sigma_b^V = 6\ cm/s$ and $\sigma_b^w = 0.08\ rad/s$. The sensor model parameters are $\eta_{r_d} = 0.1,\ \eta_{\theta_d} = 0.001,\ \eta_{r_\phi} = 0.1,\ \eta_{\theta_\phi} = 0.01,\ \sigma_b^r = 6$ and $\sigma_b^\theta = 0.06$, which represent a moderate values for uncertainty.

Transition cost: In the transition cost (30), parameters ξ_P, $\xi_\mathcal{T}$ and ξ_O represent the importance of the estimation accuracy, the algorithm runtime and the time the robot moves on high-risk areas, respectively. The main goal of TSP-FIRM is to obtain a policy that has a high probability of success. Hence, Φ^{ij} has a more important role in the transition cost in comparison with the runtime. In addition, the runtime, $\hat{\mathcal{T}}^{ij}$, is more important than $\hat{\mathcal{T}}^{ij}_{obs}$, in the planning. Although it is preferred that the robot avoids high-risk areas, the replanning ability of TSP-FIRM algorithm enables to cope with the map uncertainty such as new obstacles. Selecting ξ_P, $\xi_\mathcal{T}$ and ξ_O depends on the importance of the aforementioned factors in the cost function.

TSP-FIRM nodes: In order to define the TSP-FIRM nodes, based on (35), Δ_1 and Δ_2 should be selected small enough that condition (7) is satisfied. These parameters' values depend on the size and the noise of the robot and the environment. For example, in the real implementation for the robot with diameter 55cm, $\Delta_1 = [25, 25, 0.35]$ and $\Delta_2 = \left[25^2, 25^2, 0.35^2\right]$ are considered, but in the simulation for the robot with diameter 7.4cm Δ_1 is set to $\Delta_1 = [5, 5, 0.2]$.

Replanning: In Algorithm 3, $\boldsymbol{\alpha}_{\min}$ and $\boldsymbol{\alpha}_{\max}$ determine when it is necessary to do replanning and resolve the TSP. Small $\boldsymbol{\alpha}_{\min}$ and $\boldsymbol{\alpha}_{\max}$ increase the probability of replanning and resolving TSP. Although replanning and resolving TSP cause more reliable plan, they are time-consuming and in most cases are not necessary. In contrast, the large values of $\boldsymbol{\alpha}_{\min}$ and $\boldsymbol{\alpha}_{\max}$ cause that the robot does not consider new obstacles in its planning that makes it unreliable.

Local controller: The LQG controller (edge controller) weights are $S_N = diag\,([0.003, 0.003, 0.03])$, $W_x = diag\,([0.003, 0.003, 0.03])$ and $W_u = diag\,([0.1, 0.1])$. In the switching controller (stabilizer) the parameters related to the DFL controller are $k_{p1} = 2,\ k_{p2} = 12,\ k_{d1} = 3,\ k_{d2} = 7,\ K_\theta = 2,\ k_\theta = -1$, and for the orientation controller the parameters are set as $K_\theta = 2$ and $k_\theta = -1$. In the switching controller the parameter $e_{th} = 14\,\text{cm}$ determines when to switch. In the absence of uncertainty, this parameter may be set to a very small value. However, in a real application this parameter depends on the size of the robot and environment and most importantly to the size of the stopping region. e_{th} shall be small enough to stabilize the robot in the stopping region.

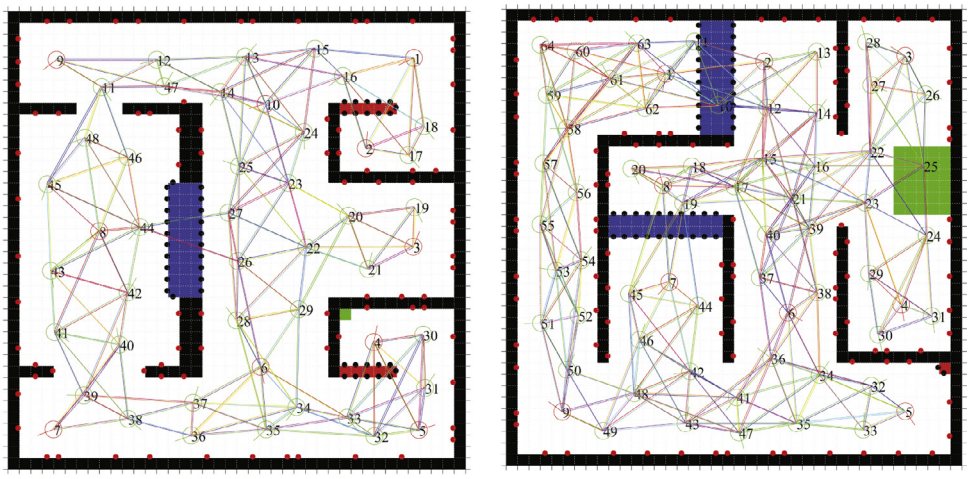
5.3.1. Results

Offline: The first step is selecting goal points and sampling PRM nodes. Then, the transition cost, the transition probability and the failure probability are computed using Monte Carlo method, and subsequently the TSP-FIRM graph is generated. Fig. 5a illustrates the constructed graph with the goal nodes [1, 2, 3, 4, 5, 6, 7, 1] (red nodes). The black and gray blocks are known and potential obstacles (false), respectively. The red points are markers position. The markers' positions are not accurate which increase the localization uncertainty. In the next step, the path between every two goal nodes and its cost-to-go is computed and used in forming the aTSP matrix. Solving aTSP provides the route between the goal nodes as [1, 2, 3, 5, 6, 7, 4, 1]. Consequently, the nominal path, i.e. nodes sequence, is obtained as follows which is possible to change in the online mode.

$$[\mathbf{1}, 10, 9, 8, \mathbf{2}, 11, 12, \mathbf{3}, 14, \mathbf{5}, 15, \mathbf{6}, 22, \mathbf{7}, 21, 17, 16, \mathbf{4}, 13, 3, 12, 11, 2, 8, 9, 10, \mathbf{1}]$$

In this sequence, the robot moves from node 6 to node 22, meaning it prefers to move from high-risk regions due to the uncertain and time-consuming alternative paths.

Online: The robot is placed near goal node 1, close to the laboratory entrance. The robot starts to move and passes through the goal points 1, 2, 3, and 5 respectively according to the route obtained in the offline phase. Then, the robot moves toward goal node 6. However, near node 15, there is an unknown obstacle. When the robot gets close to this obstacle, detects it and then stops and estimates its position. Node 15 is deleted because it is in the new founded obstacle area. Then, the current belief of the robot is added to the graph, node 23, permanently (according to remark 2). The graph and map are



(a) Simulation experiment 2 (b) Simulation experiments 3 and 4

Fig. 6. Map and graph in simulation experiments. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

Table 1
Comparison of shortest path based policy with TSP-FIRM based policy (simulation in swebots).

	Algorithm	Φ	$\hat{T}$	$\hat{T}_{obs}$	Succes %	TSP solution
Exp 1	Shortest path	71.6651	3.0435 min	3.594 sec	70	[1, 2, 3, 13, 12, 11, 9, 10, 8, 4, 5, 7, 6, 1]
	TSP-FIRM	51.4981	3.1667 min	2.736 sec	97	[1, 2, 3, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 1]
Exp	Shortest path	247.97	2.27 min	2.3 sec	75	[1, 2, 10, 3, 6, 5, 4, 7, 8, 9, 1]]
	TSP-FIRM	63	2.4 min	0 sec	97	[1, 2, 10, 3, 6, 5, 4, 7, 8, 9, 1]
Exp 3	Shortest path	445.38	1.98 min	7 sec	53	[1, 2, 3, 4, 6, 5, 9, 7, 8, 1]
	TSP-FIRM	36.4216	2.27 min	1 sec	97	[1, 2, 8, 4, 3, 6, 5, 7, 9, 1]
Exp 4	Shortest path	445.38	1.98 min	7 sec	53	[1, 2, 3, 4, 6, 5, 9, 7, 8, 1]
	TSP-FIRM	45.65	3.12 min	0 sec	97	[1, 9, 7, 5, 8, 2, 4, 3, 6, 1]

updated (Fig. 5b) and the transition cost, the transition probability as well as the failure probability of the new edges and the edges near to the obstacle is computed. According to their large changes, aTSP is resolved (based on remark 1). Therefore, the new route is obtained as [6, 7, 4, 1]. Afterward, the robot moves toward goal node 6 through the path [23, 5, 17, 16, 6]. After reaching goal node 6, the robot moves toward goal node 7 on the path [6, 22, 7], but it observes only some few markers and gets highly uncertain about its position, thus covariance matrix, P_s, grows. The robot starts information gathering phase and continues it until P_s decreases. Then, the new belief is added to the graph temporarily, node 24, (Fig. 5c), and replanning is done to steer the robot to goal node 7. After reaching goal node 7, the robot moves toward goal node 4 on the [7, 21, 17, 16, 4] path. Then it starts to return to the start point, goal node 1, but on the edge connecting node 8 to node 9, the robot is kidnapped and placed in a point between nodes 11 and 12. The robot detects the kidnapping and starts gathering information phase and continues it until the conditions to exit this phase is met. Then, temporary node 24 is added to the graph and replanning is done (Fig. 5d). Thereafter, the robot moves toward node 11 and then moves toward node 1 based on its policy. These experiments were repeated many times successfully and the robot is able to detect kidnapping in different positions, localize itself and complete its task successfully. The video of this experiment is available in [33].

5.4. TSP-FIRM highlights

5.4.1. Uncertainty impact on planning

In order to have a better understanding of the necessity of considering uncertainty in the planning, we compare the TSP-FIRM search policy with a policy obtained based on merely shortening the path. To obtain a short path we use obtained TSP-FIRM graphs, but the distance between nodes is the only factor considered in planning. To have a fair comparison, the same control method is employed in both methods to drive the robot on the planned path. In addition, in order to make the comparison more conclusive two more simulation experiments are performed with different scenarios. Fig. 6 shows the maps of these experiments. The TSP solutions are provided in Tables 1 and 2. Experiments 3 and 4 have the same map, but in experiment 4, the ξ_O in Eq. (30), is set to a high value to increase the effect of the time the robot moves on the potential obstacle area. According to the TSP solution, in experiment 4 the robot prefers to travel a longer way to avoid potentially obstacle area.

Table 2
Comparison of shortest path based policy with TSP-FIRM based policy (real experiments).

Algorithm	Φ	$\hat{\mathcal{T}}$	$\hat{\mathcal{T}}_{obs}$	Success %	TSP solution
Shortest path	1.3515×10^3	17.2703 min	18.2 sec	67	[1, 2, 3, 4, 7, 6, 5, 1]
TSP-FIRM	821.6526	18.4887 min	18.2 sec	97	[1, 2, 3, 5, 6, 7, 4, 1]

There are three main factors, including Φ, $\hat{\mathcal{T}}$ and $\hat{\mathcal{T}}_{obs}$, which are considered in this analysis and are reported in Tables 1 and 2. Φ, $\hat{\mathcal{T}}$ and $\hat{\mathcal{T}}_{obs}$ are the summation of Φ^{ij}, $\hat{\mathcal{T}}^{ij}$ and $\hat{\mathcal{T}}^{ij}_{obs}$ of all edges forming the path obtained from TSP solution. In all of the experiments, the Φ value in the shortest path method is more than the path obtained by TSP-FIRM which means TSP-FIRM provides more reliable and informative path with regard to the localization uncertainty. The Φ value in the real experiment is more than the simulation experiment owing to the bigger environment size and the greater motion noise of the real experiment. In all experiments, the shortest path runtime is less than TSP-FIRM path runtime as expected. However, there is not a significant difference because the shortest path policy spends more time for stabilization (node controller) due to the weak localization. Furthermore, in TSP-FIRM algorithm, the robot spends less time in the potential obstacle area, which is more evident in experiment 4, where the robot travels a longer path to avoid the potential obstacle area.

The obtained policies for real and simulation experiments are executed in the MATLAB thirty times and their success percentages are reported in the last column of tables. Although the runtime of TSP-FIRM is slightly more than the policy obtained based on shortest distance, TSP-FIRM provides a reliable policy with a high probability of success which is the main goal of the TSP-FIRM.

5.4.2. Key features

The TSP-FIRM graph has exploited the practical and theoretical merits of the FIRM algorithm such as breaking the curse of history, probabilistic completeness, efficient planning, robustness, reliability, scalability, which are discussed in [3] thoroughly. In Section 5.1.2, the robustness of the algorithm is discussed completely. In the following, some of the abovementioned features are discussed for TSP-FIRM algorithm.

Efficient planning: The TSP-FIRM graph is constructed offline, and subsequently, the solution for finding a sequence (modelled as aTSP) and a policy for moving between goal points can be obtained offline. This feature enables online planning viable and workable, especially in a real implementation.

Reliability: In the construction of TSP-FIRM graph the success probability of the robot is considered. The probability of colliding obstacles is also computed offline helping to find a more reliable plan for traversing goal points.

Scalability: TSP-FIRM graph is a multi-query roadmap in the belief space that enables it to operate for a long period of time in the same environment. In addition, in the proposed algorithm, the constructed graph can be extended, meaning that new nodes and edges can be added to it during online execution. The complexity of the TSP-FIRM graph construction is a constant multiplier of the complexity of the PRM construction like the FIRM algorithm, and in contrast to the most of belief-space planner that have exponential planning complexity. The TSP is an NP-hard problem, but solving it is executed offline. There are also algorithms, such as LKH and Concorde, that can provide an acceptable solution in a reasonable time.

These features enable TSP-FIRM algorithm to be used in a real application.

6. Conclusions

The underlying motivation of this paper is planning for a service or a rescue robot with goals such as search and exploration in an environment. Multi-goal motion planning is an approach to cope with this problem which is becoming increasingly attractive. Much effort, however, is put into planning without considering the motion and sensor uncertainties. This is while, many real robotic applications are involved with uncertainty and ignoring it has a significant impact on planning. This paper introduces TSP-FIRM, a novel framework for multi-goal motion planning under motion and sensor uncertainty. In the first step, we integrate the POMDP and TSP to formulate the problem as an asymmetric TSP in the belief space. However, the problem is notoriously difficult to solve in this form. Therefore, we break it down to a single-goal motion planning among goal nodes and planning for searching them. The FIRM algorithm is utilized to overcome the motion planning problem. Then an algorithm is proposed for the TSP-FIRM graph offline generation which helps to reduce the problem to planning over the nodes of the graph and thus obtaining an offline search policy. Moreover, an algorithm is proposed for the online execution of the obtained policy. The general TSP-FIRM is modified for nonholonomic mobile robots using a switching-based controller and the EKF estimator. To assess the capability of the proposed algorithms, they are implemented on a nonholonomic mobile robot in both simulation and real environment. Furthermore, we provide solutions to enable the robot to handle the challenging situations such as changes in the map, a large deviation from the planned path and getting highly uncertain. The experiment results and the comparison with a short path policy verify the TSP-FIRM efficiency. In addition the efficiency, reliability, robustness and scalability of TSP-FIRM are discussed. In future research, we address solving TSP online, handling dynamic obstacles and multi-hypothesis belief distributions in decision making.

References

[1] Agha-mohammadi, S. Chakravorty, N. M. Amato, Firm : feedback controller-based information-state roadmap - a framework for motion planning under uncertainty, in: Proceedings of the IEEE/RSJ International Conference IROS, 2011, pp. 4284–4291.
[2] A. Agha-mohammadi, Feedback-based information roadmap (FIRM) : graph-based estimation and control of robotic systems under uncertainty, Texas A&M Univ., 2014 Ph.D. thesis.
[3] A. Agha-mohammadi, S. Chakravorty, N.M. Amato, Firm : sampling-based feedback motion planning under motion uncertainty and imperfect measurements, Int. J. Robot. Res. (IJRR) 33 (2) (2014) 268–304.
[4] A. Agha-mohammadi, A.M. S. Agarwal, S. Chakravorty, J.D. D. Tomkins, N.M. Amato, Robust online belief space planning in changing environments: application to physical mobile robots, in: Proceedings of the IEEE International Conference on ICRA, 2014.
[5] R. Alterovitz, T. Siméon, K.Y. Goldberg, The stochastic motion roadmap: a sampling framework for planning with Markov motion uncertainty, in: Proceedings of the International Robotics Science and Systems (RSS), vol. 3, 2007, pp. 233–241.
[6] D. Applegate, R. Bixby, V. Chvátal, W. Cook, TSP Cuts Which Do Not Conform to the Template Paradigm, Comput. Comb. Optim. Lecture Notes in Computer Science, vol. 2241, Springer, Berlin, Heidelberg, 2001.
[7] U. Aybars, Path planning on a cuboid using genetic algorithms, Inf. Sci. 178 (16) (2008) 3275–3287.
[8] J.V.D. Berg, P. Abbeel, K. Goldberg, LQG-MP: optimized path planning for robots with motion uncertainty and imperfect state information, Int. J. Robot. Res. (IJRR) 30 (7) (2011) 895–913.
[9] A. Bry, N. Roy, Rapidly-exploring random belief trees for motion planning under uncertainty, in: Proceedings of the IEEE International Conference (ICRA), 2011, pp. 723–730.
[10] S. Chakravorty, S. Kumar, Generalized sampling-based motion planners, IEEE Trans. Syst. Man Cybern. B 41 (3) (2011) 855–866.
[11] X. Chen, R.D. Shachter, A.W. Kurian, D.L. Rubin, Dynamic strategy for personalized medicine: an application to metastatic breast cancer, J. Biomed. Inform. 68 (2017) 50–57.
[12] J. Cui, Y. Liu, Y. Xu, H. Zhao, H. Zha, Tracking generic human motion via fusion of low-and high-dimensional approaches, IEEE Trans. Syst. Man Cybern. Syst. 43 (4) (2013) 996–1002.
[13] J. Faigl, On the performance of self-organizing maps for the non-euclidean traveling salesman problem in the polygonal domain, Inf. Sci. 181 (19) (2011) 4214–4229.
[14] J. Faigl, T. Krajník, V. Vonásek, L. Přeučil, On localization uncertainty in an autonomous inspection, in: Proceedings of the IEEE International Conference on ICRA, 2012, pp. 1119–1124.
[15] J. Faigl, M. Kulich, L. Přeučil, Goal assignment using distance cost in multi-robot exploration, in: Proceedings of the IEEE/RSJ International Conference on IROS, 2012, pp. 3741–3746.
[16] C. Freundlich, P. Mordohai, M.M. Zavlanos, Optimal path planning and resource allocation for active target localization, in: Proceedings of the IEEE American Control Conference (ACC), 2015, pp. 3088–3093.
[17] S. Garrido-Jurado, R. Muñoz-Salinas, F.J. Madrid-Cuevas, M.J. Marín-Jiménez, Automatic generation and detection of highly reliable fiducial markers under occlusion, Pattern Recogn. 47 (6) (2014) 2280–2292.
[18] D.K. Grady, M. Moll, L.E. Kavraki, Extending the applicability of POMDP solutions to robotic tasks, IEEE Trans. Robot. 31 (4) (2015) 948–961.
[19] G. Gutin, A.P. Punnen, The Traveling Salesman Problem and its Variations, 12, Springer Science & Business Media, 2006.
[20] V.A. Huynh, N. Roy, icLQG: combining local and global optimization for control in information space, in: Proceedings of the IEEE International Conference on ICRA, 2009, pp. 2851–2858.
[21] A. Jain, S. Niekum, Leveraging task knowledge for robot motion planning under uncertainty, arXiv:1802.04205 (2018).
[22] L. Janson, E. Schmerling, M. Pavone, Monte Carlo motion planning for robot trajectory optimization under uncertainty, in: Robotics Research, Springer, 2018, pp. 343–361.
[23] L.P. Kaelbling, M.L. Littman, A.R. Cassandra, Planning and acting in partially observable stochastic domains, Artif. Int. 101 (1) (1998) 99–134.
[24] A. Kaplan, N. Kingry, P. Uhing, R. Dai, Time-optimal path planning with power schedules for a solar-powered ground robot, IEEE Trans. Autom. Sci. Eng. 14 (2) (2017) 1235–1244.
[25] V. Krishnamurthy, A. Aprem, S. Bhatt, Multiple stopping time POMDPs: structural results & application in interactive advertising on social media, Automatica 95 (2018) 385–398.
[26] M. Kulich, J. Faigl, L. Přeučil, On distance utility in the exploration task, in: Proceedings of the IEEE International Conference on ICRA, 2011, pp. 4455–4460.
[27] Y. Liu, L. Nie, L. Han, L. Zhang, D.S. Rosenblum, Action2activity: recognizing complex activities from sensor data., in: Proceedings of the IJCAI, vol. 2015, 2015, pp. 1617–1623.
[28] Y. Ma, H. Wang, Y. Xie, M. Guo, Path planning for multiple mobile robots under double-warehouse, Inf. Sci. 278 (2014) 357–379.
[29] A.A. Makarenko, S.B. Williams, F. Bourgault, H.F. Durrant-Whyte, An experiment in integrated exploration, in: Proceedings of the IEEE International Conference on ICRA, vol. 1, 2002, pp. 534–539.
[30] M. Memarzadeh, C. Boettiger, Adaptive management of ecological systems under partial observability, Biol. Conserv. 224 (2018) 9–15.
[31] F.J. Montana, J. Liu, T.J. Dodd, Sampling-based reactive motion planning with temporal logic constraints and imperfect state information, in: Critical Systems: Formal Methods and Automated Verification, Springer, 2017, pp. 134–149.
[32] A. Noormohammadi-Asl. Multi-goal belief space planning using tsp- rm with unknown obstacles (webots), 2017. URL https://www.youtube.com/watch?v=P1aoHDiZvtQ.
[33] A. Noormohammadi-Asl. Multi-goal motion planning under uncertainty using tsp- rm with unknown obstacles, 2017. URL https://www.youtube.com/watch?v=Uuo9Exle9Po.
[34] G. Oriolo, A.D. Luca, M. Vendittelli, Wmr control via dynamic feedback linearization: design, implementation, and experimental validation, IEEE Trans. Control Syst. Technol. 10 (6) (2002) 835–852.
[35] S. Oßwald, M. Bennewitz, W. Burgard, C. Stachniss, Speeding-up robot exploration by exploiting background information, IEEE Robot. Autom. Lett. 1 (2) (2016) 716–723.
[36] F. Pasqualetti, A. Franchi, F. Bullo, On cooperative patrolling : optimal trajectories , complexity analysis , and approximation algorithms, IEEE Trans. Robot. 28 (3) (2012) 592–606.
[37] V. Pilania, K. Gupta, Mobile manipulator planning under uncertainty in unknown environments, Int. J. Robot. Res. 37 (2–3) (2018) 316–339.
[38] S. Prentice, N. Roy, The belief roadmap: efficient planning in belief space by factoring the covariance, Int. J. Robot. Res. (IJRR) 28 (11–12) (2009) 1448–1465.
[39] T. Regev, V. Indelman, Decentralized multi-robot belief space planning in unknown environments via identification and efficient re-evaluation of impacted paths, Auton Robots 42 (4) (2018) 691–713.
[40] L. Santos, C. Christophorou, E. Christodoulou, G. Samaras, J. Dias, Development strategy of an architecture for e-health personalised service robots, Int. J. Comput. Sci. Inf. Syst. 9 (2014) 1–18.
[41] K.M. Seiler, H. Kurniawati, S.P. Singh, An online and approximate solver for POMDPs with continuous action space, in: Proceedings of the IEEE International Conference on ICRA, 2015, pp. 2290–2297.
[42] M.T. Spaan, N. Vlassis, Perseus: randomized point-based value iteration for POMDPs, J. Artif. Intell. Res. 24 (2005) 195–220.
[43] É.D. Taillard, K. Helsgaun, Popmusic for the travelling salesman problem, Eur. J. Oper. Res. (2018) Available online 23 June 2018, doi:10.1016/j.ejor.2018.06.039.
[44] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics, MIT press, 2005.

[45] T. Veiga, M.T. Spaan, P.U. Lima, C. Brodley, P. Stone, Point-based POMDP solving with factored value function approximation, in: Proceedings of the Twenty-Eighth Conference on Artificial Intelligence (AAAI), 2014, pp. 2513–2519.
[46] A. Wallar, E. Plaku, D.A. Sofge, Reactive motion planning for unmanned aerial surveillance of risk-sensitive areas, IEEE Trans. Autom. Sci. Eng. 12 (3) (2015) 969–980.
[47] L. Xu, A.T. Stentz, A fast traversal heuristic and optimal algorithm for effective environmental coverage, in: Proceedings of the International Conference on Robotics Science and Systems (RSS), Pittsburgh, PA, 2010.
[48] N. Ye, A. Somani, D. Hsu, W.S. Lee, Despot: online POMDP planning with regularization, J. Artif. Intell. Res. 58 (2017) 231–266.
[49] R. Yehoshua, N. Agmon, G.A. Kaminka, Safest path adversarial coverage, in: Proceedings of the IEEE/RSJ International Conference on IROS, 2014, pp. 3027–3032.
[50] Y. Zhong, J. Lin, L. Wang, H. Zhang, Hybrid discrete artificial bee colony algorithm with threshold acceptance criterion for traveling salesman problem, Inf. Sci. 421 (2017) 70–84.