To appear in the journal of IEEE Robotics and Automation Letters (RA-L 2018)

Resource-Performance Trade-off Analysis for Mobile Robot Design

M. Lahijanian, M. Svorenova, A. A. Morye, B. Yeomans, D. Rao, I. Posner,
P. Newman, H. Kress-Gazit, M. Kwiatkowska

***Abstract*— The design of mobile autonomous robots is challenging due to the limited on-board resources such as processing power and energy. A promising approach is to generate intelligent schedules that reduce the resource consumption while maintaining best performance, or more interestingly, to trade off reduced resource consumption for a slightly lower but still acceptable level of performance. In this paper, we provide a framework to aid designers in exploring such resource-performance trade-offs and finding schedules for mobile robots, guided by questions such as "what is the minimum resource budget required to achieve a given level of performance?" The framework is based on a quantitative multi-objective verification technique which, for a collection of possibly conflicting objectives, produces the Pareto front that contains all the optimal trade-offs that are achievable. The designer then selects a specific Pareto point based on the resource constraints and desired performance level, and a correct-by-construction schedule that meets those constraints is automatically generated. We demonstrate the efficacy of this framework on several robotic scenarios in both simulations and experiments with encouraging results.**

I. Introduction

Mobile robotics is a fast growing field with a broad range of applications such as home appliance, aerial vehicles, and space exploration. The main feature that makes these robots very attractive from the application perspective is their ability to operate remotely with some level of autonomy. The very same factors, however, introduce a challenge from the design angle due to the limited on-board resources such as processing power and energy source. For example, the Curiosity Mars rover operates on a CPU with less computational power than a today's typical smartphone CPU [1], resulting in slow movements and limited capabilities of the rover. In drones, the weight and the capacity of the on-board battery directly influences the robot's ability to stay airborne.

Mobile autonomy is enabled through *localization*, *perception* and *planning* modules, where localization and perception provide information about the robot's location and surroundings, respectively, and the planning module generates a trajectory. Most mobile robots treat these modules as separate processes, which are run simultaneously, and often continuously, for best performance. These algorithms are complex and consume computational resources in addition to the energy cost of the robot's motion (motors). An example of CPU usage by the modules for a mobile ground robot is shown in Fig. 1. By intelligently scheduling the modules [2], it may be possible to reduce the resource consumption while maintaining best performance. More interestingly, it may be possible to *trade off* reduced resource consumption for a slightly lower but still acceptable level of performance.

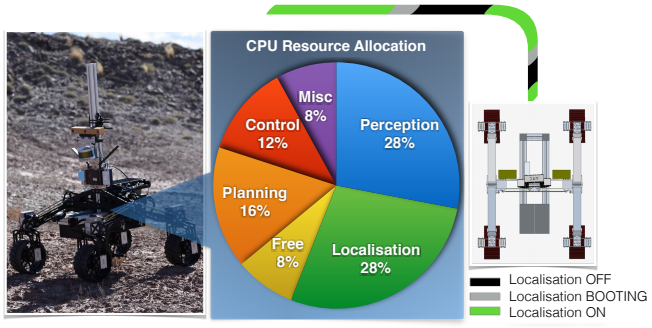


Fig. 1: A robot with a stereo camera with an on-board processor running various modules required for mobile autonomy. The color-coded pie chart is an example representation of per-module CPU usage. By intelligently scheduling each module, we are likely to free resources, not only to reduce the resource budget, but also to free resources for other modules.

Examples include switching localization on and off to save energy or restricting the continuous calls of the planner to free resources for other modules. One issue with this approach is that the objectives, *i.e.,* to reduce resource usage and to improve performance, are naturally competing, and by optimizing for one objective, the values for the other may become suboptimal. For instance, by turning localization off throughout the trajectory, the robot may minimize its energy consumption, while at the same time increase the probability of collision. On the other hand, keeping localization on for the duration can lead to excessive energy usage.

The aim of this paper is to provide a framework to aid the designers in exploring such *resource-performance trade-offs* and finding schedules for mobile robots, guided by questions such as "given a resource budget, what guarantees can be provided on achievable performance?" and, more interestingly, "what is the minimum resource budget required to achieve a given level of performance?". To this end, we exploit a technique from formal methods known as quantitative *multi-objective* verification and controller synthesis [3], [4], which, for a given scenario and a set of quantitative objectives, *e.g.,* constraints on computation power, time, energy, or probability of collisions, produces the so-called *Pareto front*, a set of Pareto optimal points that represents all the optimal trade-offs that are achievable. The designer then selects a specific Pareto point based on the resource constraints and desired performance level, and a correct-by-construction schedule that meets those constraints is *automatically* generated. We illustrate the potential of the framework on a generic resource cost function in conjunction with two performance criteria of safety and target-reachability in the context of a localization schedule but emphasize that the technique is generally applicable.

Multi-objective techniques have been studied for probabilistic models endowed with costs such as energy or time. Off-the-shelf tools exist to support Pareto front computation [5], [6], but are limited to discrete models such as Markov decision processes (MDP). The challenge here, therefore, is to adapt the techniques to the *continuous-domain*, both time and space, scenario typical for mobile robots, while also capturing the uncertainty that the autonomy modules must deal with. We thus propose a novel *abstraction* method, which considers noise in both the robot dynamics and its sensors. Given a set of control laws, the method obtains a *discretization* of the continuous robot motion as an MDP. We lift the robot's resource consumption to a cost on this MDP, and through the multi-objective techniques, we compute the Pareto front that encodes all optimal trade-offs between the resource requirements and the task-related performance guarantees. Having this set available to the designer, they have the freedom to choose a Pareto point based on their design criteria. For the selected point, we generate a schedule. We demonstrate the efficacy of this framework on several robotic scenarios with various dynamics and resources. Summarizing, the main contributions of this work are:

- a general framework for the exploration of resource-performance trade-offs in mobile robotics based on multi-objective optimization with various (possibly conflicting) objectives,
- a discretization of the robot dynamics that enables reduction to the multi-objective problem, and
- validation of the techniques by both simulation and experiments of complex robotic scenarios with encouraging results.

II. Literature Review

Most existing resource allocation works in robotics focus on single objective problems, namely optimization for QoS or energy. Examples include [7] in wireless systems, [8] in multi-robot localization, and [2] in perception. Such problems usually involve scheduling sensor usage, and the typical performance criterion is the "goodness" of the state estimation, *e.g.,* [9], [10]. Apart from [11], where computation is offloaded to the cloud to improve performance, trade-off analysis between resource and performance objectives has been little studied. Our framework not only performs such analysis, but it also allows multiple objectives for various resources, *e.g.,* energy, time, and computation power, and performance criteria, *e.g.,* safety and target reachability.

Since mobile robots are increasingly often employed in safety- and performance-critical situations, techniques from formal methods that offer high-level temporal logic planning [12], correct-by-construction controller synthesis [13] and performance guarantees [14], [15] have been gaining attention. However, these are task specific and lack the generality and flexibility of the proposed framework, which enables systematic exploration of trade-offs.

Trade-off analysis techniques have been studied in verification, mostly from the theoretical perspective, and include quantiles and multi-objective methods. Quantiles can express the cost-utility ratio but not the Pareto front, and have been applied for energy analysis of low-level protocols [16]. Multi-objective (or multi-criteria) optimization has been extensively studied in operations research and stochastic control [17]. More recently, techniques that combine high-level temporal logic specifications with multi-objective optimization have been formulated for discrete probabilistic models, including probabilistic [3] and expected total reward [4], [18] properties used in this paper. They have been employed, e.g., to analyze human-in-the-loop problems [19]. The works [20], [21] consider problems with budget constraints for discrete models from the theoretical point of view.

III. Problem formulation

The focus of this work is an optimal trade-off analysis between a robot's resource usage and its task guarantees through the use of a localization module. We consider robot dynamics (plant model) given by:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \mathbf{w}), \tag{1}$$

where $\mathbf{x} \in X \subseteq \mathbb{R}^{n_x}$ is the state of the robotic system, $\mathbf{u} \in U \subseteq \mathbb{R}^{n_u}$ is the control input, $\mathbf{w} \in \mathbb{R}^{n_w}$ is the process (motion or input) noise given by a normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{Q}_w)$ with zero mean and covariance $\mathbf{Q}_w \in \mathbb{R}^{n_w \times n_w}$, and $f : X \times U \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x}$ is a continuous integrable function that describes the evolution of the robot in the space. The robot is equipped with two sets of sensors that enable the measurement of its state, *e.g.,* odometry and high-accuracy localization sensors. The first set (odometry) uses a negligible amount of resources but provides inaccurate noisy measurements. By deploying the second set, which we refer to as the localization module or simply *localization*, additional information is obtained, and the measurements become more accurate at a higher cost of resources.

Let $A = \{a_{\mathsf{start}}, a_{\mathsf{boot}}, a_{\mathsf{on}}, a_{\mathsf{off}}\}$ denote the set of all possible localization module actions (status). Action a_{start} sends a signal to start the localization, and it takes time $T_{\mathsf{boot}} \in \mathbb{R}_{\geq 0}$ for it to turn on. The action a_{boot} refers to the booting status during T_{boot}. Action a_{on} indicates that localization is on, and a_{off} turns it off. The resulting measurement model is:

$$\mathbf{z} = \begin{cases} h^{\mathsf{od}}(\mathbf{x}, \mathbf{v}^{\mathsf{od}}) & \text{if } a \in \{a_{\mathsf{start}}, a_{\mathsf{boot}}, a_{\mathsf{off}}\}, \\ h^{\mathsf{lo}}(\mathbf{x}, \mathbf{v}^{\mathsf{lo}}) & \text{if } a = a_{\mathsf{on}}, \end{cases} \tag{2}$$

where $\mathbf{z} \in Z \subseteq \mathbb{R}^{n_z}$ is the state measurement, and $\mathbf{v}^{\mathsf{od}}$, $\mathbf{v}^{\mathsf{lo}} \in \mathbb{R}^{n_v}$ are the measurement noise terms under the first and second set of sensors (odometry and localization), respectively. In high accuracy mode, the noise is given by $\mathbf{v}^{\mathsf{lo}} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_v)$, where covariance $\mathbf{Q}_v \in \mathbb{R}^{n_v \times n_v}$, whereas in low accuracy mode, no restriction is imposed on the distribution of $\mathbf{v}^{\mathsf{od}}$.

The robot moves in an environment (workspace) $W \subset \mathbb{R}^{n_W}$, where $n_W \in \{2, 3\}$, with a set of obstacles W_O and a target region W_G. Colliding with an obstacle constitutes failure; hence, the robot's task is to avoid obstacles and reach the target by following a precomputed reference trajectory φ. We assume that φ is given as a sequence of waypoints

$\varphi = (\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_{|\varphi|})$, where $\tilde{\mathbf{x}}_i \in X$. We denote the initial state of the robot by $\tilde{\mathbf{x}}_0$. The robot is equipped with an on-board controller that generates a sequence of control laws in order to follow φ (see Sec. IV-A).

We assume that, when the localization module is used, *i.e.,* $a = a_{\mathsf{on}}$, the i-th control law drives the robot to a proximity of waypoint $\tilde{\mathbf{x}}_i$ (see Sec. IV-A). When the robot turns its localization off, it may deviate from φ due to imprecise measurements. Therefore, even if φ is an obstacle-free trajectory, by turning off the localization module, there is a probability that the robot may collide with an obstacle or may not reach the target. We specify these probabilities formally in Sec. IV-C.

The aim is, given φ, to schedule the use of localization over time in a way that optimizes the trade-off between the robot's resource usage and its task guarantees. Let ς denote a localization schedule, which simply speaking, indicates which localization action the robot needs to apply at any given time (see Sec. IV-B). The resource consumption of the robot under schedule ς is the sum of the resources required for it to run the localization module and the resources consumed by the rest of the system. To allow the inclusion of different types of resources, *e.g.,* computational power, time, or energy, we define a general cost function $c : X \times U \times A \to \mathbb{R}_{\geq 0}$ (see Sec. IV-C.1 for details and Sec. VI for examples) to represent total resource usage. The formal problem definition is then as follows.

Problem 1: Given a mobile robot model as in (1) and (2) in an environment W with a set of obstacles W_O and a target W_G, a reference trajectory φ with its corresponding control laws, and resource cost function c, compute a localization schedule ς such that

- the expected total resource cost is minimized,
- the probability of collision is minimized, and
- the probability of reaching the target is maximized.

These objectives may be competing, and there may not exist a localization schedule that globally optimizes all of them. In this work, we are interested in the set of all optimal trade-offs between the objectives, which introduces an additional level of complexity to the problem. Another major challenge is dealing with a continuous robotic system with noisy measurements, *i.e.,* partial observability. This leads to reasoning in belief space, which is generally a computationally infeasible domain. We propose a framework to address these challenges in two steps. First, we overcome the complexity of belief space through a suitable finite abstraction and then use formal techniques to generate the set of all optimal trade-offs between the objectives.

This framework is general in that its structure is independent of the choices of objectives. It can also handle multiple resource cost functions. Here, we present the concrete design of the two steps of the framework for the particular choices in Problem 1 but note that, in addition to more objectives, it can also be adapted to schedule other modules such as perception or different motors.

IV. System Description

Due to both process and measurement noise, robot's motion is stochastic, and its exact state cannot be known. They, however, can be described as probability distributions. The probability distribution of $\mathbf{x}_t$ at time $t \in \mathbb{R}_{\geq 0}$ is referred to as the *belief* of $\mathbf{x}_t$, denoted by b_t, and given by

$$\mathbf{x}_t \sim b_t = \mathcal{P}_{\mathbf{x}}(\mathbf{x}_t \mid \mathbf{x}_{t_0}, \mathbf{u}_{t_0:t}, \mathbf{z}_{t_0:t}, a_{t_0:t}), \tag{3}$$

where $\mathcal{P}_{\mathbf{x}}$ denotes the (conditional) probability density function of $\mathbf{x}$, $\mathbf{x}_{t_0}$ is the distribution at the initial time t_0, and $\mathbf{u}_{t_0:t}$, $\mathbf{z}_{t_0:t}$, and $a_{t_0:t}$ are the sequences of control inputs, measurements, and statuses of the localization used from t_0 to t. We denote the belief space containing all possible beliefs by $\mathbb{B}$, *i.e.,* $b_t \in \mathbb{B}\ \forall t$.

A. Control Laws

Recall that, starting from the initial position $\tilde{\mathbf{x}}_0$, the robot follows reference trajectory $\varphi = (\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_{|\varphi|})$ using a series of *control laws*. For $1 \leq i \leq |\varphi|$, control law $\tilde{\mathbf{u}}_i = (g_i, \xi_i)$ consists of a feedback controller $g_i : \mathbb{B} \to U$ designed to drive the robot towards $\tilde{\mathbf{x}}_i$ and a termination rule (trigger) ξ_i that indicates when to terminate the execution of g_i. We assume that, when localization is on, g_i is able to stabilize the state belief b around waypoint $\tilde{\mathbf{x}}_i$. The construction of such a controller for robotic systems is detailed in [22]. In short, g_i is generally a concatenation of two controllers: reachability and stabilizer. The reachability controller drives the system to a neighborhood of $\tilde{\mathbf{x}}_i$, and then the stabilizer controller stabilizes b to a predefined belief $\tilde{\mathbf{b}}_i$ that corresponds to $\tilde{\mathbf{x}}_i$. This stabilization is typically achieved by an LQG controller on the linearized dynamics around $\tilde{\mathbf{x}}_i$ and defining $\tilde{\mathbf{b}}_i = \mathcal{N}(\tilde{\mathbf{x}}_i, \mathbf{Q}_{\tilde{\mathbf{x}}_i})$, where $\mathbf{Q}_{\tilde{\mathbf{x}}_i}$ is the steady-state covariance given by steady-state Kalman filter (solution to an algebraic Riccati equation [23]). Note that the convergence to $\tilde{\mathbf{b}}_i$ is guaranteed if the linearized dynamics are controllable and observable [23]. For a full discussion on the construction of such controllers for various systems, including nonholonomic systems, see [22]. When localization is off, g_i uses only the reachability part of the controller.

Let Δt_i be the duration that it takes the reachability controller to move the robot from $\tilde{\mathbf{x}}_{i-1}$ to (a neighborhood of) $\tilde{\mathbf{x}}_i$ under localization on. We design the trigger ξ_i to fire, *i.e.,* $\xi_i = 1$, when the belief of the robot state converges to $\tilde{\mathbf{b}}_i$ if the localization is on (in practice, an ϵ-convergence is sufficient, *i.e.,* $|b_t - \tilde{\mathbf{b}}_i| < \boldsymbol{\epsilon}$). In turn, if the localization is not on, the robot applies g_i for the duration of Δt_i. Formally,

$$\xi_i = \begin{cases} \mathbb{1}_{<\boldsymbol{\epsilon}}(|b_t - \tilde{\mathbf{b}}_i|) & \text{if } a = a_{\mathsf{on}}, \\ \delta(t - (t_{i-1} + \Delta t_i)) & \text{if } a \in \{a_{\mathsf{start}}, a_{\mathsf{boot}}, a_{\mathsf{off}}\}, \end{cases} \tag{4}$$

where $\mathbb{1}$ and δ are the Indicator and Dirac delta functions, respectively, and t_{i-1} is the time mark of the initialization of $\tilde{\mathbf{u}}_i$. Furthermore, $\boldsymbol{\epsilon} = (\epsilon_{\mathsf{mean}}, \epsilon_{\mathsf{var}})^T \in \mathbb{R}^2_{>0}$, and for $b_t = \mathcal{N}(\hat{\mathbf{x}}_t, \mathbf{Q}_{\tilde{\mathbf{x}}_i})$, where $\hat{\mathbf{x}}_t$ and $\mathbf{Q}_{\tilde{\mathbf{x}}_i}$ are the state estimate and covariance at time t, the indicator function $\mathbb{1}_{<\boldsymbol{\epsilon}}(|b_t - \tilde{\mathbf{b}}_i|) = 1$ if $|\hat{\mathbf{x}}_t - \tilde{\mathbf{x}}_i| < \epsilon_{\mathsf{mean}}$ and $|\mathbf{Q}_{x_t} - \mathbf{Q}_{\tilde{\mathbf{x}}_i}| < \epsilon_{\mathsf{var}}$; otherwise 0.

Therefore, from trajectory φ, the following series of control laws, which enable the robot to follow φ, is obtained:

$$\varphi = (\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_{|\varphi|}) \quad \Rightarrow \quad \mathbf{u} = (\tilde{\mathbf{u}}_1, \ldots, \tilde{\mathbf{u}}_{|\varphi|}). \tag{5}$$

B. Localization Schedule

The robot makes a decision on the use of its localization based on its belief b_t. This decision is referred to as the *localization schedule*. Let $\mathcal{D}(\cdot)$ denote the set of all probability distributions over a given set. Localization schedule is a function $\varsigma : \mathbb{B} \to \mathcal{D}(A)$ that assigns to a belief b_t a probability distribution over the localization actions from the set A.

In this work, we assume that the localization decisions are made right before applying each control law $\tilde{\mathbf{u}}_i$. In other words, the granularity of the localization decisions corresponds to the waypoints in φ, which is user-defined. Therefore, given a reference trajectory φ and a localization schedule ς, the induced robot trajectory can be described by a sequence of state beliefs:

$$b_{t_0} \xrightarrow{(a_{t_0:t_1}, \tilde{\mathbf{u}}_1)} b_{t_1} \xrightarrow{(a_{t_1:t_2}, \tilde{\mathbf{u}}_2)} \ldots \xrightarrow{(a_{t_{|\varphi|-1}:t_{|\varphi|}}, \tilde{\mathbf{u}}_{|\varphi|})} b_{t_{|\varphi|}}, \tag{6}$$

where the localization action of a_{t_i} is assigned according to the probability distribution $\varsigma(b_{t_i})$ over A. Note that it is possible for the localization to become active during the execution of $\tilde{\mathbf{u}}_i$ if $a_{t_{i-1}} = a_{\mathsf{boot}}$. That means that, at some point between t_{i-1} and t_i, booting is complete. At this point, the measurements become more accurate, and $\tilde{\mathbf{u}}_i$ drives the robot's state belief to $\tilde{\mathbf{b}}_i$.

We assume that, without loss of generality, the localization is initially on. Therefore, in Problem 1, we are interested in computing a *feasible* schedule, in which it holds that: the first action applied by the schedule is a_{on} or a_{off}; every action a_{start} is immediately preceded by a_{off}; every action a_{boot} is immediately preceded by a_{start} or a_{boot}; if a_{start} is used at time point $t_i, i \geq 1$, then action a_{boot} is used at all time points $t_{i+1}, \ldots, t_j$ such that $t_j - t_i < T_{\mathsf{boot}} \leq t_{j+1} - t_i$; and every action a_{on} is immediately preceded by a_{boot} or a_{on}.

Since we are interested in efficient schedules, we assume that all schedules avoid turning off the localization module while booting is in progress and starting the booting process if it cannot be completed by the time $t_{|\varphi|}$.

C. Objectives

1) Resource Consumption: One of the objectives that influences the choice of ς is the resource consumption. Let $c : X \times U \times A \to \mathbb{R}_{\geq 0}$ denote a resource consumption function, which represents the amount of resources used by the robotic system given its state, control input, and localization action. An example of such resource cost is the amount of computations required by different system modules (including localization). We are interested in the total expected resource cost under ς, which we denote by $E_{\mathsf{cost}}(\varsigma)$.

Let $\mathbf{b}_t$ denote the *expected belief*, where expectation is taken over observations, *i.e.,*

$$\mathbf{b}_t = \mathbb{E}_Z(b_t \mid \mathbf{x}_{t_0}, a_{t_0:t}) \tag{7}$$

$$= \int_{Z_{t_0:t}} \mathcal{P}_{\mathbf{x}}(\mathbf{x}_t \mid \mathbf{x}_{t_0}, a_{t_0:t}, \mathbf{z}_{t_0:t}) pr(\mathbf{z}_{t_0:t}) d\mathbf{z}_{t_0:t} \tag{8}$$

$$= \mathcal{P}_{\mathbf{x}}(\mathbf{x}_t \mid \mathbf{x}_{t_0}, a_{t_0:t}), \tag{9}$$

where $\mathbb{E}_Z$ is the expectation over domain Z, and pr is the probability measure. Then, given the localization action a_t and control $\mathbf{u}_t$, the expected cost at time t is given by

$$\mathbb{E}_X(c(\mathbf{x}_t, \mathbf{u}_t, a_t)) = \int_X c(\mathbf{x}_t, \mathbf{u}_t, a_t)\, \mathbf{b}_t\, d\mathbf{x}_t, \tag{10}$$

where $\mathbb{E}_X$ is the expectation over domain X. The total expected cost for the whole trajectory under ς is

$$E_{\mathsf{cost}}(\varsigma) = \int_{t_0}^{t_{|\varphi|}} \sum_{a_t \in A} \varsigma(\mathbf{b}_t)(a_t)\, \mathbb{E}_X(c(\mathbf{x}_t, \mathbf{u}_t, a_t))\, dt. \tag{11}$$

2) Task Performance: The task objectives of obstacle avoidance and target reachability need to be reasoned about probabilistically by using beliefs. Given that all collisions with obstacles are terminal, the fragments of the beliefs that are in collision remain in obstacles as $\mathbf{b}_t$ evolves. Hence, the probability of collision along φ under schedule ς is

$$P_{\mathsf{coll}}(\varsigma) = pr\big(\mathbf{x}_{t_0:t_{|\varphi|}} \in X_O\big) = \int_{X_O} \mathbf{b}_{t_{|\varphi|}}\, d\mathbf{x}_{t_{|\varphi|}}, \tag{12}$$

where $X_O \subset X$ is the set of states that correspond to the obstacles in W_O. Similarly, the probability that the robot is in the target region at the end of the trajectory execution is

$$P_{\mathsf{targ}}(\varsigma) = pr(\mathbf{x}_{t_{|\varphi|}} \in X_G) = \int_{X_G} \mathbf{b}_{t_{|\varphi|}}\, d\mathbf{x}_{t_{|\varphi|}}, \tag{13}$$

where $X_G \subset X$ is the set of states that correspond to the target region W_G.

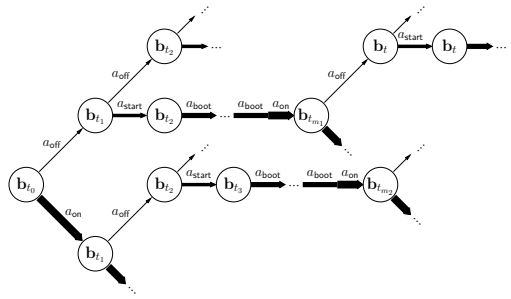
V. Solution Method

Here, we detail our abstraction method for the robotic system in Problem 1 and the reduction of the problem to a multi-objective optimization over an MDP.

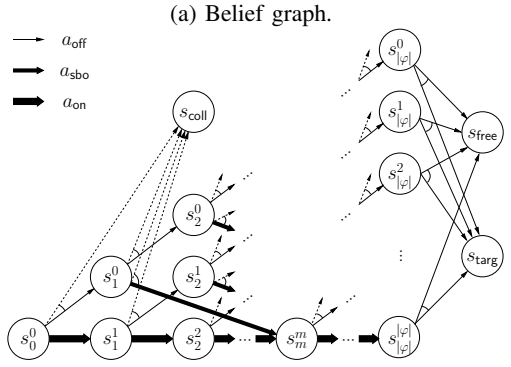
A. Abstraction

1) Belief Space Discretization: Recall that, due to motion and observation noise, the robot has to base its localization schedule on state beliefs. These beliefs are generally hard to reason about because they are continuous in both time and space. One of the novelties of this work is a discretization method that reduces the complexity of this reasoning from continuous to a discrete, finite space. The purpose is to capture all possible behaviors of the system in expectation; thus, we focus on the expected beliefs. The key observation is that localization decisions are only made right before applying each control law, and there is only a finite number of control laws (waypoints) and localization actions. Hence, the number of belief states that the robot has to make a decision on is, in turn, finite.

Technically, the robot's belief evolves sequentially based on the applied control law and localization action in each step



(a) Belief graph.



(b) Markov decision process.

Fig. 2: Structure of the belief graph and the MDP. The thickness of edges indicates their action label. In (b), action a_{sbo} represents the sequence of actions needed to fully activate localization, which begins with a_{start}, continues with a_{boot}, and ends with a_{on}.

as shown in (6). Initially, the belief is $\mathbf{b}_{t_0} = \delta(\mathbf{x}_{t_0} - \tilde{\mathbf{x}}_0)$. For $1 \le i \le |\varphi|$ and action $a_{t_{i-1}}$ given according to $\varsigma(\mathbf{b}_{t_{i-1}})$ for the time window $[t_{i-1}, t_i)$, the (expected) belief at time t_i is:

$$\mathbf{b}_{t_i} = \begin{cases} \mathcal{P}_{\mathbf{x}}(\mathbf{x}_{t_i} \mid \mathbf{b}_{t_{i-1}}, \tilde{\mathbf{u}}_i, a_{\text{on}}) & \text{if } a_{t_i^-} = a_{\text{on}}, \\ \mathcal{P}_{\mathbf{x}}(\mathbf{x}_{t_i} \mid \mathbf{b}_{t_{i-1}}, \tilde{\mathbf{u}}_i, a_{\text{off}}) & \text{if } a_{t_i^-} \neq a_{\text{on}}, \end{cases} \tag{14}$$

where $a_{t_i^-}$ is the final status of the localization in $[t_{i-1}, t_i)$. Therefore, a discrete *belief graph*, which is a directed graph that captures all possible beliefs of the robot at the localization decision points, can be constructed as in Fig. 2a. The nodes of the graph are the beliefs $\mathbf{b}_{t_i}$, at which the localization schedule is called, and each edge is labeled with a localization action and directs to the next belief.

It is important to note that, in addition to preserving the history of collisions, $\mathbf{b}_{t_i}$ is dependent on the history of the applied actions. In other words, $\mathbf{b}_{t_i}$ is unique to the sequence of applied localization actions up to t_i. For simplicity of presentation, we do not project this history in the notation of beliefs, but it is reflected in the graph in Fig. 2a by only one incoming edge for each belief node. This results in a state explosion in the belief graph. That is, the total number of nodes is exponential in the length of φ, *i.e.,* $\mathcal{O}(|A|^{|\varphi|})$. We drastically reduce this size (to quadratic in $|\varphi|$) by distinguishing between the collision-free and in-collision parts of the belief nodes as explained below.

2) MDP Construction: Recall that the robot converges to $\tilde{\mathbf{b}}_i$ when $a_{t_{i-1}^-} = a_{\text{on}}$. This is trivially conditioned on the fact that the robot's trajectory is collision-free. Let $\bar{\mathbf{b}}_{t_i}$ denote the collision-free part of $\mathbf{b}_{t_i}$, *i.e.,* the truncated probability distribution of $\mathbf{x}_{t_i}$ over $X - X_O$:

$$\bar{\mathbf{b}}_t = \mathcal{P}_{\mathbf{x}}(\mathbf{x}_t \mid \mathbf{x}_{t_0}, \mathbf{u}_{t_0:t}, a_{t_0:t}, \mathbf{x}_t \notin X_O). \tag{15}$$

Then, $\bar{\mathbf{b}}_{t_i} = \tilde{\mathbf{b}}_i$ (up to precision ϵ) when $a_{t_i^-} = a_{\text{on}}$. This means that, every time the localization is turned on, the truncated collision-free belief $\bar{\mathbf{b}}_{t_i}$ of the robot becomes a precomputed distribution, resulting in an independence from the history of the applied localization actions unlike $\mathbf{b}_{t_i}$. This leads to a lower number of unique $\bar{\mathbf{b}}_{t_i}$ beliefs (*i.e.,* lower number of unique nodes in the graph). Let $\bar{\mathbf{b}}_i^j$, $0 \le j \le i$, denote the collision-free belief at time t_i with the most recent localization action a_{on} at time t_j, *i.e.,* $a_{t_j^-} = a_{\text{on}}$ and $a_t \neq a_{\text{on}}$ for all $t_j < t < t_i$. The sequential evolution of the collision-free beliefs becomes:

$$\begin{cases} \bar{\mathbf{b}}_i^i = \tilde{\mathbf{b}}_i & \text{if } a_{t_i^-} = a_{\text{on}}, \\ \bar{\mathbf{b}}_i^j = \mathcal{P}_{\mathbf{x}}(\mathbf{x}_{t_i} \mid \bar{\mathbf{b}}_{i-1}^j, \tilde{\mathbf{u}}_i, a_{\text{off}}, \mathbf{x}_{t_i} \notin X_O) & \text{if } a_{t_i^-} \neq a_{\text{on}}. \end{cases} \tag{16}$$

Unlike (14), the above evolution is not deterministic but is probabilistic. That is, under each localization action, there is a probability associated with the transition from one collision-free belief to the next one, and the remaining probability mass is assigned to the collision with an obstacle. Therefore, by reasoning over $\bar{\mathbf{b}}_i^j$ instead of $\mathbf{b}_{t_i}$, the belief graph can be greatly reduced in size at the cost of introducing probabilistic transitions. This probabilistic model is, in fact, an MDP defined as follows.

An MDP is a tuple $\mathcal{M} = (S, s_{\text{init}}, Act, P)$, where S is a non-empty finite set of states, $s_{\text{init}} \in S$ is the initial state, Act is a non-empty finite set of actions, and $P : S \times Act \to \mathcal{D}(S)$ is a (partial) probabilistic transition function. A *cost function* for MDP $\mathcal{M}$ is a (partial) function $C : S \times Act \to \mathbb{R}_{\ge 0}$ such that $C(s, a)$ is defined iff $P(s, a)$ is defined.

The construction of the MDP for the evolution of the robot is as follows. The state space S consists of states of the form s_i^j, for $0 \le i \le |\varphi|$, $0 \le j \le i$, that correspond to beliefs $\bar{\mathbf{b}}_i^j$, where s_i^i indicates that the robot's belief is $\tilde{\mathbf{b}}_i$, and s_0^0 is the initial state. In addition, S includes states $s_{\text{coll}}, s_{\text{targ}}, s_{\text{free}}$, which correspond to the robot being in collision, target, and free space, respectively. The set of MDP actions is $Act = \{a_{\text{off}}, a_{\text{on}}, a_{\text{sbo}}\}$, where a_{sbo} represents the sequence of actions needed to fully activate localization, which begins with a_{start}, continues with a_{boot}, and ends with a_{on}.

The transition probabilities for states s_i^j are computed as follows. For actions a_{on} and a_{off}, the values can be computed by evolving $\bar{\mathbf{b}}_i^j$ according to (16). In practice, techniques such as Kalman Filter or Particle Filter can be employed to compute these evolutions as well as their corresponding transition probabilities. For action a_{sbo} in state s_i^j, *i.e.,* $\bar{\mathbf{b}}_i^j$, we first find the smallest index m, $i < m \le |\varphi|$, such that $\sum_{k=i+1}^{m} \Delta t_k > T_{\text{boot}}$, which indicates that booting process becomes complete at some point between t_{m-1} and t_m if initialized at t_i. Therefore, we compute the transition probabilities of a_{sbo} by combining the previously computed probabilities of action a_{off} in states $s_i^j, s_{i+1}^j, \ldots, s_{m-2}^j$ followed

by the evolution of $\bar{\mathbf{b}}^j_{m-1}$, initialized with the localization off. After the remainder of the booting time, the localization comes on and $\bar{\mathbf{b}}^m_m$ is reached. Once the computations for all s^j_i states are complete, we check the probability of being in the target region or in the free space for states with $i = |\varphi|$.

The MDP cost C at state s^j_i under $a \in Act$ is the expected resource usage by the robot starting from belief $\bar{\mathbf{b}}^j_i$ at time point t_i to the next time point t_{i+1} under the corresponding localization action. Formally,

$$C(s^j_i, a) = \int_{t_i}^{t_{i+1}} \mathbb{E}_X(c(\mathbf{x}_t, \mathbf{u}_t, a_t))\, dt, \tag{17}$$

where $\mathbb{E}_X(c(\mathbf{x}_t, \mathbf{u}_t, a_t))$ is given by (10), $\mathbf{x}_{t_i} \sim \bar{\mathbf{b}}^j_i$, and $a_t \in A$ corresponds to the MDP action $a \in Act$.

The size of the state space of this MDP is quadratic in the length of φ, *i.e.,* $|S| = (|\varphi|+2)(|\varphi|+1)/2 + 3$, and there are at most two actions per state. The implementation of the algorithm can be made parallel by constructing the diagonal levels of the triangle-shaped state space in Fig. 2b independently, further reducing the complexity from quadratic to linear in the length of φ.

B. Problem Reduction

A *policy* for MDP $\mathcal{M}$ is a function $\pi : S \to \mathcal{D}(Act)$ that associates every state with a distribution according to which the next action is chosen. From the above construction of the MDP $\mathcal{M}$ and definition of its action a_{sbo}, it follows that every feasible localization schedule for the robot corresponds to a policy for $\mathcal{M}$, and vice versa, every policy π of $\mathcal{M}$ implies a feasible localization schedule ς_π defined as $\varsigma_\pi(\mathbf{b}_{t_i}) = \pi(s^j_i)$. Thus, E_{cost}, P_{coll}, and P_{targ} in (11), (12), and (13), respectively, for the robot become the following over $\mathcal{M}$:

$$E_{\mathsf{cost}}(\varsigma_\pi) = \mathbb{E}_\pi\Big(\sum_{k=0}^{|\varphi|-1} C(s_k, \pi(s_k))\Big), \tag{18}$$

$$P_{\mathsf{coll}}(\varsigma_\pi) = \mathbb{P}_\pi(s_{\mathsf{coll}}), \tag{19}$$

$$P_{\mathsf{targ}}(\varsigma_\pi) = \mathbb{P}_\pi(s_{\mathsf{targ}}), \tag{20}$$

where $\mathbb{E}_\pi$ and $\mathbb{P}_\pi(s)$ denote the expectation over the paths of $\mathcal{M}$ and the probability of reaching state s under policy π, respectively. Therefore, Problem 1 reduces to a multi-objective optimization problem over $\mathcal{M}$, where the goal is to construct a policy that minimizes (18) and (19) and maximizes (20).

There exist algorithms and off-the-shelf tools that solve the above multi-objective problem for MDPs, *e.g.,* [5]. These algorithms construct the Pareto front, *i.e.,* set of all optimal trade-offs between the objectives through a value iteration procedure [4]. Given a Pareto point in this set, the algorithms generate the corresponding policy using linear programming [3], [18]. The complexity of this algorithm is polynomial in the size of the MDP. We first construct the Pareto front using these algorithms. Then, by the choice of the designer, we generate the desired policy, *i.e.,* localization schedule.

We note that the obtained localization schedule for the continuous system with trajectory φ is optimal with respect to the user-provided waypoints, a discretization of φ. As this discretization is refined (increasing the number of waypoints), the obtained optimality approaches the true one in the continuous domain.

VI. Experimental Results

In this section, we demonstrate the efficacy of the method in three case studies. First, we illustrate the proposed framework on a scenario involving a second-order unicycle robot with energy as a resource cost and analyze the optimal performance-energy trade-offs for three different reference trajectories in simulations. Second, we perform the same analysis on a commercially available planetary rover and deploy the rover with the obtained schedules in an experimental setup. Third, we demonstrate the potential of the framework in aiding the designer to choose the hardware components, specifically a computer, for a robotic system.

A. Second-Order Unicycle

Setup: We considered a mobile robot with (second-order) unicycle dynamics given by

$$\dot{x}_1 = v\cos\theta + w_1, \qquad \dot{x}_2 = v\sin\theta + w_2, \tag{21}$$

$$\dot{v} = u_1 + w_3, \qquad \dot{\theta} = u_2 + w_4, \tag{22}$$

where $x_1, x_2 \in [0, 10]$ indicate the position, v is the speed, and θ is the heading angle of the robot. The control inputs u_1 and u_2 are the acceleration and angular velocity, respectively, and the motion noise $\mathbf{w}$ is sampled from $\mathcal{N}(\mathbf{0}, \sigma_w^2\mathbf{I})$ with $\sigma_w = 0.01$, and $\mathbf{I}$ is the identity matrix. The robot measurements were modeled as $\mathbf{z} = \mathbf{x} + \mathbf{v}^\star$, where sensor noise $\mathbf{v}^\star \sim \mathcal{N}(\mathbf{0}, \sigma_\star^2\mathbf{I})$ for $\star \in \{\mathsf{od}, \mathsf{lo}\}$. For odometry, $\sigma_{\mathsf{od}} = 0.2$, and for localization $\sigma_{\mathsf{lo}} = 0.03$.

We focused on trade-off analysis of the performance objectives, *i.e.,* collision avoidance and target reaching, and energy consumption as the resource objective. The energy consumption model was taken from [2]. Intuitively, the energy consumed by the robot is the sum of the energy consumed by the localization module and the rest of the system. Without loss of generality, we attribute the latter mainly to the motion, *i.e.,* motors, and the CPU. Formally, the energy cost function $c_{\mathsf{en}} : X \times U \times A \to \mathbb{R}_{\geq 0}$ is defined as $c_{\mathsf{en}} = c_{\mathsf{enl}} + c_{\mathsf{enm}}$. The localization energy cost (per unit time) $c_{\mathsf{enl}} : X \times U \times A \to \mathbb{R}_{\geq 0}$ is:

$$c_{\mathsf{enl}}(\mathbf{x}, \mathbf{u}, a) = \begin{cases} \frac{\mathsf{E}_{\mathsf{boot}}}{T_{\mathsf{boot}}} & \text{if } a \in \{a_{\mathsf{start}}, a_{\mathsf{boot}}\}, \\ \mathsf{P}_{\mathsf{on}} & \text{if } a = a_{\mathsf{on}}, \\ 0 & \text{if } a = a_{\mathsf{off}}, \end{cases} \tag{23}$$

for all $\mathbf{x} \in X, \mathbf{u} \in U$, where $\mathsf{E}_{\mathsf{boot}}$ and T_{boot} are the energy and time required to turn on the localization, respectively. P_{on} is its power demand when active that can be estimated as the sum of the power demand of the sensors of the localization module, taken directly from their specification, and the power consumed by the CPU for the localization algorithm that processes sensory inputs. The energy cost function for the rest of the system is defined as $c_{\mathsf{enm}}(\mathbf{x}, \mathbf{u}, a) = \mathsf{P}_{\mathsf{mot}} + \mathsf{P}_{\mathsf{CPU}}$, for all $\mathbf{x} \in X, \mathbf{u} \in U, a \in A$, where $\mathsf{P}_{\mathsf{mot}}$ and $\mathsf{P}_{\mathsf{CPU}}$ are power

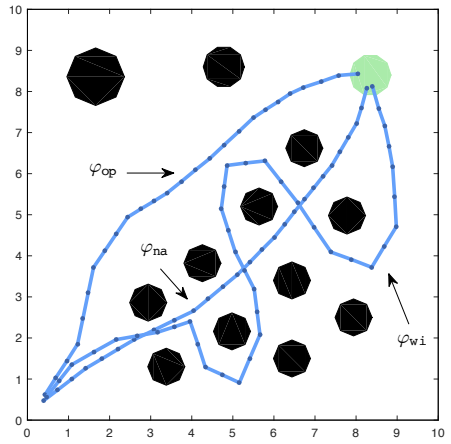


Fig. 3: Workspace of the unicycle robot. Obstacle and target regions are shown in black and green, respectively. The waypoints of reference trajectories φ_{op}, φ_{na}, and φ_{wi} are depicted as dark blue dots. In light blue, we show the corresponding robot trajectories using feedback controller in (24) and localization schedule ς_{on}.

demands of the motors and the CPU (when not executing the localization algorithm). The expected energy consumption E_{en} is then computed using equations in Sec. IV-C.1. In this example, we used the above model with the following parameters. The localization module used a camera and an algorithm to process the images at rate 16 Hz. The power demand of the module when active was $\mathsf{P}_{\mathsf{on}} = 8$ W, and the module required time $T_{\mathsf{boot}} = 5$ s and energy $\mathsf{E}_{\mathsf{boot}} = 40$ J to boot. The remaining power consumption for the robot was approximated as $\mathsf{P}_{\mathsf{mot}} + \mathsf{P}_{\mathsf{CPU}} = 42$ W.

We considered a workspace with several obstacles and a target region, and three reference trajectories as depicted in Fig. 3. The trajectories were: φ_{op} through open space with 26 waypoints; φ_{na} between narrow obstacles with 27 waypoints; φ_{wi} winding around obstacles with 39 waypoints. The waypoints are shown as dark blue dots in Fig. 3. For reachability of the waypoint $\tilde{\mathbf{x}} = (\tilde{x}_1, \tilde{x}_2)$ and stabilization of the belief for localization on, we used dynamic feedback linearization (DFL) controller to linearize the unicycle dynamics as in [24] and an LQG controller on the linearized dynamics given by:

$$\begin{aligned} u_1 &= \bar{u}_1 \cos\theta + \bar{u}_2 \sin\theta, \quad u_2 = \tfrac{\bar{u}_2}{v}\cos\theta - \tfrac{\bar{u}_1}{v}\sin\theta, \\ \bar{u}_1 &= k_1(\tilde{x}_1 - \hat{x}_1) - k_2\,\hat{x}_3 \cos\hat{x}_4, \\ \bar{u}_2 &= k_3(\tilde{x}_2 - \hat{x}_2) - k_4\,\hat{x}_3 \sin\hat{x}_4, \end{aligned} \tag{24}$$

where $\hat{x}_i$ is the ith component of the mean of the state estimate, the output of Kalman filter, and $k_1 = k_3 = 1$ and $k_2 = k_4 = 2.236$ are feedback gains. Fig. 3 depicts the robot trajectories under no noise.

Pareto Fronts: We constructed an MDP for each reference trajectory according to the abstraction algorithm in Sec. V-A by using a Particle Filter. They consisted of 656, 708, and 1 488 state-action pairs for φ_{op}, φ_{na}, and φ_{wi} respectively. We then computed the Pareto front for the three objectives of collision avoidance, target reaching and energy consumption for each reference trajectory using PRISM-games [6]. In Table I, we present the plots and list the vertices of the convex Pareto fronts and compare their values against the ones of the schedule ς_{on}, which keeps the localization module on at all times.

Every point on the surface of the Pareto front corresponds to a particular optimal trade-off between the three objectives and there exists a localization schedule that achieves it. A bound on a value of an objective, *e.g.,* the probability of collision should be at most 0.01, imposes a slice through the surface that divides the Pareto front into trade-offs which are achievable with the desired bound and those which are not. A projection of the Pareto front to a lower dimension is the Pareto front for the subset of objectives. For example, in plots in Tables Ia-c, the projection onto the bottom plane represents the optimal trade-offs between the target-reaching and collision probabilities, regardless of the expected energy consumption.

In all three cases, the localization schedule ς_{on} is not Pareto optimal. That means it is not efficient to keep localization on at all times because there exist localization schedules that have the same probabilistic guarantees as ς_{on}, *i.e.,* $P_{\mathsf{targ}} = 1$ and $P_{\mathsf{coll}} = 0$, while saving energy by turning localization off. On the other hand, the localization schedule ς_{off}, which keeps the localization off at all times, is Pareto optimal in all three cases. While ς_{off} tends to decrease the energy consumption, this may not be a desirable schedule due to low performance guarantees. Generally, Pareto-optimal schedules save 60-80% of total energy and 72-100% of localization energy compared to ς_{on}.

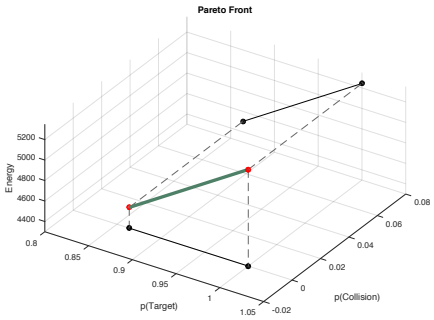
Localization Schedules: For each reference trajectory, we generated schedules for two Pareto points using PRISM [5]. We simulated 500 sample robot trajectories under each localization schedule and Fig. 4 depicts 100 of them. The first set of schedules correspond to the Pareto points with $P_{\mathsf{targ}} = 1$, $P_{\mathsf{coll}} = 0$ (top-left images in Fig. 4). As shown in these figures, to ensure that the target region is reached with probability 1, these schedule turn on localization only at the critical waypoints: near obstacles to ensure safety and right before the target to ensure ending in target. The second set of schedules correspond to ς_{off} for φ_{op} and Pareto points with $P_{\mathsf{targ}} = 0.97$ and $P_{\mathsf{coll}} = 0.03$ for φ_{na} and φ_{wi} (bottom-right images in Fig. 4). These schedules trade off the performance by a small percentage to save energy. As shown in bottom-right image in Fig. 4a, this schedule keeps localization off for the entire φ_{op}, resulting in missing the target 14% of times (loss in performance) in trade-off for 79% gain in energy. For φ_{na} and φ_{wi}, the schedules turn on the localization only at two extremely critical points; one very close to an obstacle and one before the target. These schedules trade off 3% loss in performance to gain 73% to 75% in energy. To summarize, the simulations suggest that the need for the localization to be active grows with the level of noise and proximity of obstacles.

Validation: In the simulations above, the average performance of the robot with respect to all three objectives was within 3% of the theoretical values (performance guarantees

TABLE I: Pareto fronts computed for the second-order unicycle. For each of the three reference trajectories, we list the corner points of the Pareto front, namely their probability of reaching the target P_{targ}, the probability of collision P_{coll} and the expected total energy E_{en} with the fraction of it consumed by the localization system E_{enl}. For comparison, we also list the performance guarantees of the schedule ς_{on} and the percentage of the total and localization energy saved by Pareto-optimal schedules compared to ς_{on}. On the right, Pareto fronts are visualized in three-dimensional space. For better readability of the images, we plot the projection of the polytopes onto two planes.

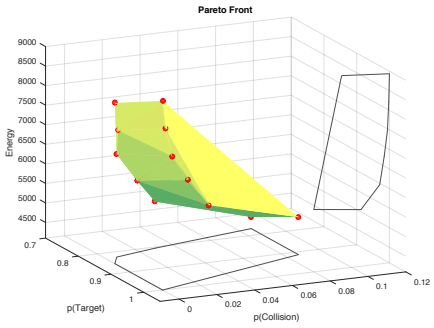
(a) Open trajectory φ_{op}.

	P_{targ}	P_{coll}	E_{en}	fraction E_{enl}		
ς_{on}	1	0	21 790.20	3486.44		
Pareto points:					E_{en} and E_{enl} saved over ς_{on}	
1 (ς_{off})	0.8640	0	4502.19	0	79.34%	100.00%
2	1	0	5243.21	176.91	75.94%	94.93%



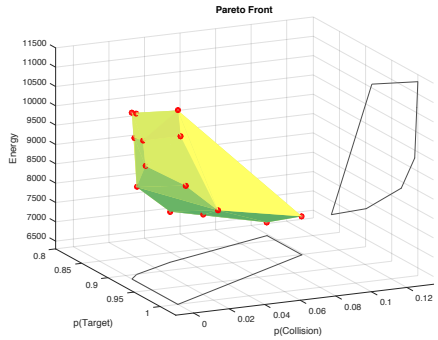
(b) Narrow trajectory φ_{na}.

	P_{targ}	P_{coll}	E_{en}	fraction E_{enl}		
ς_{on}	1	0	22 476.90	3596.30		
Pareto points:					E_{en} and E_{enl} saved over ς_{on}	
1	0.8540	0	8213.72	840.07	63.46%	76.64%
2	1	0	8935.53	1013.56	60.25%	71.82%
3	0.9980	0.0020	8211.67	869.09	63.47%	75.83%
4	0.8523	0.0020	7491.30	695.95	66.67%	80.65%
5	0.9940	0.0060	7459.44	690.57	66.81%	80.80%
6	0.8250	0.0060	6734.18	516.92	70.04%	85.63%
7	0.9841	0.0159	6748.25	519.60	69.98%	85.55%
8	0.9703	0.0297	5959.24	337.93	73.49%	90.60%
9	0.8054	0.0297	5290.72	174.74	76.46%	95.14%
10	0.8306	0.0159	6018.15	346.52	73.23%	90.36%
11	0.9131	0.0869	5060.09	159.01	77.49%	95.58%
12 (ς_{off})	0.7689	0.0869	4396.48	0.00	80.44%	100.00%

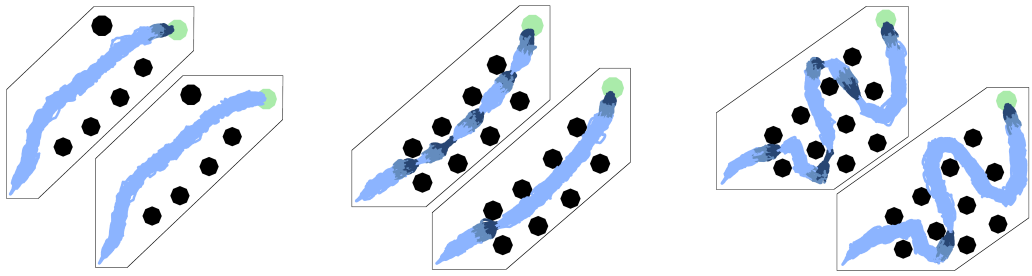


(c) Winding trajectory φ_{wi}.

	P_{targ}	P_{coll}	E_{en}	fraction E_{enl}		
ς_{on}	1	0	33 354.40	5336.70		
Pareto points:					E_{en} and E_{enl} saved over ς_{on}	
1	0.9120	0	10 596.90	837.37	68.23%	84.31%
2	0.9200	0	10 634.30	811.57	68.12%	84.79%
3	1	0	11 316.77	1011.48	66.07%	81.05%
4	0.9980	0.0020	10 633.27	876.41	68.12%	83.58%
5	0.9102	0.0020	9914.84	702.64	70.27%	86.83%
6	0.9261	0.0020	9977.65	674.78	70.09%	87.36%
7	0.9243	0.0040	9296.83	540.38	72.13%	89.87%
8	0.9940	0.0060	9297.63	539.62	72.12%	89.89%
9	0.9006	0.0060	8560.97	363.32	74.33%	93.19%
10	0.9683	0.0317	8325.83	340.42	75.04%	93.62%
11	0.8773	0.0317	7608.22	168.68	77.19%	96.84%
12	0.9012	0.0988	7294.80	157.48	78.13%	97.05%
13 (ς_{off})	0.8345	0.0988	6642.58	0	80.08%	100.00%
14	0.9012	0.0433	7640.69	179.72	77.09%	96.63%



of the Pareto points). In addition, we randomly selected 10 more Pareto points and performed similar computations. All the simulation results were within 4% of the theoretical values. We note that these error values are expected to decrease as the number of simulations and the number of particles in the particle filter increase.



(a) Open trajectory φ_{op}. (b) Narrow trajectory φ_{na}. (c) Winding trajectory φ_{wi}.

Fig. 4: Sample trajectories under two localization schedules for each trajectory φ_{op}, φ_{na}, and φ_{na} for the unicycle robot. The light, medium, and dark blue trajectory segments indicate localization status a_{off}, a_{start} and a_{boot}, and a_{on}, respectively. In the top-left images in (a)-(c), the performance guarantees are $P_{\mathsf{targ}} = 1$ and $P_{\mathsf{coll}} = 0$, the same as ς_{on}, while saving 60% to 75% energy over ς_{on}. In the bottom-right images, the performance guarantees are $P_{\mathsf{targ}} = 0.86$ and $P_{\mathsf{coll}} = 0$ for φ_{op} and $P_{\mathsf{targ}} = 0.97$ and $P_{\mathsf{coll}} = 0.03$ for φ_{na} and φ_{wi} in return for additional energy saving of 73% to 75%.

B. Rover Experiments

Setup: The robotic platform used in this experimental case study is ARC Q14 planetary rover shown in Fig. 1. It is designed to mimic the configuration and specification found on rovers deployed for planetary exploration. The rover's base is rectangular (0.8 m by 0.9 m) and has 4 wheels and 8 motors. It can operate in two kinematic modes: Ackermann steering and differential drive with maximum speed of 0.5 m/s. The robot is equipped with a Point Grey Bumblebee XB3 camera. We use Dub4 [25] as the high accuracy localisation module, while low accuracy measurements are obtained using Visual Odometry [26]. The on-board computations are carried out on MicroSVR computer. The energy consumption model of the robot is the same as the one in Sec. VI-A taken from [24] that previously studied this platform.

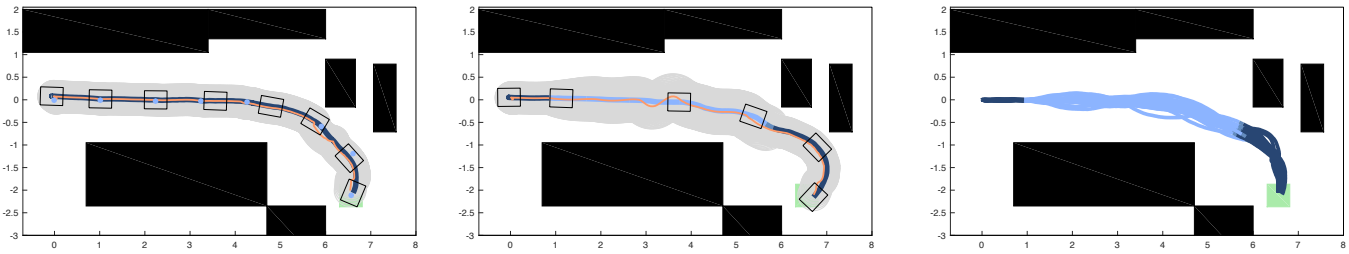
We modeled the motion of the rover as the unicycle in Sec. VI-A with constrained velocity, turn angle, and acceleration. We used the same DFL as above to linearize the dynamics and employed receding horizon controller for reachability. Kalman filter was utilized for state estimation. The online control computations were performed in MATLAB on a MacBook Pro with 2.7 GHz Intel Core i5 and 8 GB of memory, which communicated to the robot via Wi-Fi. We estimated motion and measurement noise as $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$, where $\sigma_w = 0.1$, $\sigma_{\mathsf{od}} = 0.1$, and $\sigma_{\mathsf{lo}} = 0.01$, and the frequency of sensor measurements was 4 Hz. The robot's task was to navigate from an entrance to exit door of a $10m$-by-$6m$ meeting room cluttered with various furniture pieces. The robot was first driven by a human to learn the reference trajectory φ (teach phase), during which the localization module automatically extracts waypoints of φ. The environment and these waypoints are shown in Fig. 5a.

Pareto Front: We computed the Pareto front for this scenario by first generating the abstraction MDP and then our multi-objective algorithm. We considered the same objectives as in Sec.VI-A; the vertices of the Pareto front are shown in Table II. In this case study, both ς_{on} and ς_{off} are Pareto optimal; one gives rise to the highest P_{targ} and the other results in the smallest E_{en}. Note that it is possible to save 18%, 24%, and 32% in E_{en} by sacrificing small percentage (0.5%, 1%, and 5%, respectively) in P_{targ}.

Robot Deployment: We deployed the robot under ς_{on} and ς_3. Fig. 5b-c show the robot's trajectories, localization status in different shade of blue, state estimate in orange, and belief's variance's projection onto 2-D in gray. The robot itself is shown as black-edged rectangles along the trajectory. As evident in these figures, under ς_{on}, the robot is always safe because it is able to stay within a very close proximity of φ at all times. Under ς_3, the robot uses its localization only at the very beginning and for the last two waypoints. The use of localization at the beginning sets the robot's trajectory and belief on the right path. Once localization is turned off, the uncertainty in the robot's belief grows, but the robot is still able to continue with the path without deviating too far from the φ thanks to its initial localization. Once the robot is near a point that is dangerously close to an obstacle, and φ requires sharp maneuvers, the robot turns on its localization to reduce its uncertainty and enable itself to perform the maneuvers. Note that, under ς_3, once the localization is turned back on, on account of the increased uncertainty, the robot is required to make a sharper turn than under ς_{on} to be able to reach the target. The framework is aware of such uncertainties; therefore, under ς_3, the performance guarantee is reduced by 1% to save 24% in energy in comparison to ς_{on}, resulting in an elongation of the battery life. Fig. 5c illustrates 50 trajectories that was obtained in simulation prior to deployment of the robot. Note that this figure shows only the trajectory of the center of the robot; the robot's volume needs to be added to every point along the trajectory.

C. Robot with choices of PCs

Hardware choices in robot design affect the capabilities of the robot and can result in different achievable resource-performance trade-offs. In this example, we analyzed resource-performance trade-offs for a mobile robot with two different mini PCs. This type of analysis can aid the designer in choosing the best suitable hardware to achieve a desired level of performance.



(a) Robot trajectory under ς_{on}. (b) Robot trajectory under ς_3. (c) Simulation trajectories under ς_3.

Fig. 5: Robot trajectories in experiments and simulations. The waypoints are shown as light blue dots in (a). The light, medium, and dark blue trajectory segments indicate localization status a_{off}, a_{start} and a_{boot}, and a_{on}, respectively. In (a) and (b), robot's belief is shown in orange (state estimate) and gray (projection of variance), and robot's orientation by black-edged boxes. Under ς_{on}, the performance guarantees are $P_{\mathsf{targ}} = 1$ and $P_{\mathsf{coll}} = 0$ while they are $P_{\mathsf{targ}} = 0.99$ and $P_{\mathsf{coll}} = 0.01$ for ς_3 in trade-off for 24% reduction in E_{en}.

TABLE II: Pareto fronts for the planetary rover. E_{en} and E_{enl} denote and the total and localization energy, respectively. For comparison, we list the energy savings of Pareto-optimal schedules compared to ς_{on}.

P. Pt	P_{targ}	P_{coll}	E_{en}	E_{enl}	E_{en}, E_{enl} saved	
1	0.5000	0.0000	6155.52	507.90	29.99%	65.34%
2 (ς_{on})	1.0000	0.0000	8791.87	1465.31	0.00%	0.00%
3	0.9900	0.0100	6713.23	650.88	23.64%	55.58%
4	0.9950	0.0050	7215.16	814.58	17.93%	44.41%
5 (ς_{off})	0.4428	0.0050	4805.86	0.00	45.34%	100.00%
6	0.9552	0.0448	5969.96	325.25	32.10%	77.80%

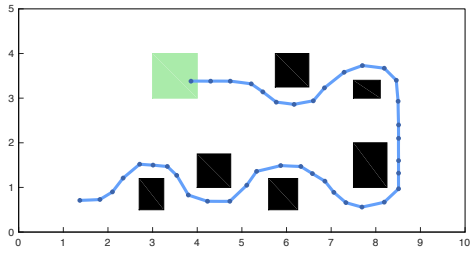


Fig. 6: Workspace and reference trajectory of the mobile robot. The waypoints of the reference trajectory are depicted as dark blue dots. In light blue, we show the corresponding robot trajectory using nominal system.

Setup: We modeled the robot dynamics as

$$\dot{x}_1 = 0.1x_2 - 0.3x_1 + u_1 + w_1,$$
$$\dot{x}_2 = 0.1x_1 - 0.3x_2 + u_2 + w_2$$

where $x_1 \in [0, 10]$ and $x_2 \in [0, 5]$ indicate the 2-D position of the robot and the process noise distribution is $\mathcal{N}(\mathbf{0}, 0.07^2\mathbf{I})$. The measurement models for odometry and localization module were $\mathbf{z} = \mathbf{x} + \mathbf{v}$ with noise distribution $\mathcal{N}(\mathbf{0}, 0.2^2\mathbf{I})$ and $\mathcal{N}(\mathbf{0}, 0.03^2\mathbf{I})$, respectively. We considered the workspace shown in Fig. 6 with obstacles and a target region and a trajectory with 40 waypoints. The controllers for the waypoints were designed by LQG method. When localization system was on, a controller was terminated when the state estimate reaches (a proximity of) the steady state distribution around the associated waypoint. The time triggers for localization off were time durations computed based on the nominal system, *i.e.,* without process and measurement noise.

We analyzed the robot running with IPC i3 Barebone (~$600) and IPC2 i5 Barebone (~$700). The two computers processed localization measurements at rates proportional to their computational power, namely 10 Hz and 16 Hz, respectively. With IPC2, the robot received feedback about its position at higher rate which allowed for faster convergence to waypoints. Thus using the localization module, the robot traversed the trajectory in less time with IPC2 than with IPC. Odometry measurements were processed at 20 Hz when the localization module was not active with both computers. We then focused on trade-off analysis for the performance objectives P_{targ} and P_{coll}, and two resource objectives of the expected energy consumption E_{en} and expected trajectory duration E_{dur}.

The energy consumption was given by the following parameters. The power consumption of localization module and motors were estimated as 5 W and 30 W, respectively. The IPC and IPC2 require power 17 W and 15 W, respectively. This amounts to an overall power consumption of 44 W and 42 W when localization module was active, respectively. When localization was not active, the power savings were estimated as 5 W for the sensors and one fifth of the computer power consumption, *i.e.,* 8 W for both computers. The module required time $T_{\mathsf{boot}} = 5$ s and energy 40 J to boot.

Pareto Fronts: The MDP constructed for the reference trajectory consisted of 1 529 state-action pairs for both computers. The convex Pareto fronts for the four objectives of target reaching, collision avoidance, energy consumption and trajectory duration had 11 vertices for IPC and 22 vertices for IPC2. In Table III, we list only the vertices for which the probability of reaching the target P_{targ} is at least 0.95.

In neither cases, localization schedule ς_{on} that keeps the localization module on at all times is Pareto optimal because there exist localization schedules that have the same probabilistic guarantees as ς_{on}, *i.e.,* $P_{\mathsf{targ}} = 1$ and $P_{\mathsf{coll}} = 0$, while saving energy and time by turning localization

off. On the other hand, localization schedule ς_{off}, which keeps the localization off at all times, is Pareto optimal for both IPC and IPC2 as it minimizes the expected energy consumption and duration. This however comes at a cost of poor performance guarantees. As indicated in Table III, generally, Pareto-optimal schedules save 34-60% of energy and 31-52% of time compared to ς_{on} for IPC, and 20-50% of energy and 17-40% of time for IPC2.

Hardware Comparison: For given performance guarantees, the robot can complete the reference trajectory faster with IPC2 than with IPC, *e.g.,* in $118.87\,\mathrm{s}$ compared to $126.42\,\mathrm{s}$ for $P_{\mathsf{targ}} = 1$ and $P_{\mathsf{coll}} = 0$. More interestingly, while the IPC2 has higher power requirements with $17\,\mathrm{W}$ relative to $15\,\mathrm{W}$ for IPC, the analysis shows that the robotic system consumes less energy with IPC2 than with IPC in completing the trajectory. That means that IPC2 offers overall better resource-performance trade-offs than IPC for the given trajectory. This, however, comes at the cost of higher dollar value for IPC2.

The designer may also wish to choose the computer based on the duration or energy rather than performance guarantees as the primary preference. For example, if it is sufficient to complete the trajectory in expected time $E_{\mathsf{dur}} \leq 130\,\mathrm{s}$, IPC is a better choice because it can achieve the best performance at lower dollar value. On the other hand, if the constraint is $E_{\mathsf{dur}} \leq 120\,\mathrm{s}$, only IPC2 can guarantee best performance.

We can also analyze the Pareto fronts in more detail. For example, consider time constraint $E_{\mathsf{dur}} \leq 90\,\mathrm{s}$. By projecting the Pareto fronts onto E_{en} and fixing $E_{\mathsf{dur}} = 90\,\mathrm{s}$, we obtain the convex sets of trade-offs between performance objectives P_{targ} and P_{coll} achievable by the two computers within the given time bound. For IPC, the set has vertices $P_{\mathsf{targ}} = 0.938, P_{\mathsf{coll}} = 0.062$ (for maximizing P_{targ}), and $P_{\mathsf{targ}} = 0.887, P_{\mathsf{coll}} = 0$ (for minimizing P_{coll}). For IPC2, the set has vertices $P_{\mathsf{targ}} = 0.943, P_{\mathsf{coll}} = 0.057$ (for maximizing P_{targ}), and $P_{\mathsf{targ}} = 0.897, P_{\mathsf{coll}} = 0.047$ (for minimizing P_{coll}). Thus, while finishing the trajectory within $90\,\mathrm{s}$, there exists a localization schedule for IPC that guarantees no collision while no such schedule exists for IPC2. On the other hand, with IPC2 the achievable probability of reaching the target location is higher than for IPC. Using this analysis, the designer can make a well-informed choice according to given preferences.

VII. Final Remarks and Future Work

We have introduced a general framework for the exploration of performance-resource trade-offs and demonstrated its efficacy in module scheduling and robot design on case studies. The framework can be adapted to schedule other modules such as perception or different motors. The framework can also be extended to scenarios, in which the aim is to schedule the use of more than one localization module, providing state estimates at different levels of certainty. Intuitively, the only change to the framework is in the abstraction step, *i.e.,* the MDP construction. The additional modules cause an increase in both action set and state space of the MDP.

TABLE III: Pareto fronts for the planetary rover with two different mini PCs. In the tables, E_{en} and E_{dur} denote and the energy consumption and trajectory duration, respectively. We only list points with $P_{\mathsf{targ}} \geq 0.95$. For comparison, we list the performance guarantees of the schedule ς_{on} and the energy and time savings of Pareto-optimal schedules compared to ς_{on}.

(a) IPC i3 Barebone.

	P_{targ}	P_{coll}	E_{en}	E_{dur}		
ς_{on}	1	0	9468.67	182.09		
P. Pt					E_{en}, E_{dur} saved	
1	1.0000	0.0000	6282.54	126.42	33.65%	30.57%
2	0.9980	0.0020	5759.97	117.06	39.17%	35.71%
3	0.9920	0.0080	5026.35	104.72	46.92%	42.49%
4	0.9821	0.0179	4689.18	99.67	50.48%	45.26%
5 (ς_{off})	0.8798	0.0660	3827.27	86.98	59.58%	52.23%

(b) IPC2 i5 Barebone.

	P_{targ}	P_{coll}	E_{en}	E_{dur}		
ς_{on}	1	0	7185.22	143.71		
P. Pt					E_{en}, E_{dur} saved	
1	1.0000	0.0000	5762.75	118.87	19.80%	17.28%
2	0.9980	0.0020	5214.40	110.93	27.43%	22.81%
3	0.9960	0.0040	4985.78	106.74	30.61%	25.72%
4	0.9900	0.0100	4691.53	101.92	34.71%	29.08%
5	0.9880	0.0120	4604.97	100.55	35.91%	30.03%
6	0.9841	0.0159	4470.06	97.95	37.79%	31.84%
7	0.9802	0.0198	4371.77	96.33	39.16%	32.97%
8	0.9645	0.0355	4098.98	92.87	42.95%	35.37%
9 (ς_{off})	0.8570	0.0883	3601.42	85.75	49.88%	40.33%

There are multiple directions for future work. First, in this work we focused on total expected resource costs, but more complex cost measures such as long-run average or ratio costs might be of interest. Second, the delay between making an observation and computing the corresponding state estimate introduces additional uncertainty. An attractive future direction is to explore the Pareto front by considering these delays. Finally, solving the problem for a combination of modules, *e.g.,* localization and perception, and, ultimately, localization, perception and planning, poses a great challenge, which should be explored for full trade-off analysis of the autonomy modules.

References

[1] (2008, July) RAD750 component specification. [Online]. Available: http://www.baesystems.com/en/download-en/20151124120355/1434555668211.pdf

[2] P. Ondruska, C. Gurau, L. Marchegiani, C. H. Tong, and I. Posner, "Scheduled perception for energy-efficient path following," in *ICRA*, 2015, pp. 4799–4806.

[3] K. Etessami, M. Kwiatkowska, M. Vardi, and M. Yannakakis, "Multi-objective model checking of Markov decision processes," in *TACAS*, 2007, pp. 50–65.

[4] V. Forejt, M. Kwiatkowska, and D. Parker, "Pareto curves for probabilistic model checking," in *ATVA*, 2012, pp. 317–332.

[5] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *CAV*, ser. LNCS, vol. 6806, 2011, pp. 585–591.

[6] T. Chen, V. Forejt, M. Kwiatkowska, D. Parker, and A. Simaitis, "PRISM-games: A model checker for stochastic multi-player games," in *TACAS*, 2013, pp. 185–191.
[7] R. Madan, S. P. Boyd, and S. Lall, "Fast algorithms for resource allocation in wireless cellular networks," in *IEEE/ACM Trans. Netw.*, vol. 18, Jun. 2010, pp. 973–984.
[8] E. D. Nerurkar, S. I. Roumeliotis, and A. Martinelli, "Distributed maximum a posteriori estimation for multi-robot cooperative localization," in *ICRA*, 2009, pp. 1402–1409.
[9] A. I. Mourikis and S. I. Roumeliotis, "Optimal sensor scheduling for resource-constrained localization of mobile robot formations," *IEEE Trans. on Rob.*, vol. 22, no. 5, pp. 917–931, 2006.
[10] V. Gupta, T. H. Chung, B. Hassibi, and R. M. Murray, "On a stochastic sensor selection algorithm with applications in sensor scheduling and sensor coverage," *Automatica*, vol. 42, no. 2, pp. 251 – 260, 2006.
[11] J. Salmern-Garca, P. igo Blasco, F. D. del Ro, and D. Cagigas-Muiz, "A tradeoff analysis of a cloud-based robot navigation assistant using stereo image processing," *IEEE Trans. on Aut. Sci. and Eng.*, vol. 12, no. 2, pp. 444–454, April 2015.
[12] H. Kress-Gazit, G. Fainekos, and G. J. Pappas, "Where's waldo? sensor-based temporal logic motion planning," in *ICRA*, 2007, pp. 3116–3121.
[13] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon control for temporal logic specifications," in *HSCC*, 2010, pp. 101–110.
[14] M. Lahijanian, M. Kloetzer, S. Itani, C. Belta, and S. Andersson, "Automatic deployment of autonomous cars in a robotic urban-like environment (RULE)," in *ICRA*, 2009, pp. 2055–2060.
[15] R. Luna, M. Lahijanian, M. Moll, and L. E. Kavraki, "Asymptotically optimal stochastic motion planning with temporal goals," in *WAFR*, 2014, pp. 335–352.
[16] C. Baier, C. Dubslaff, S. Klüppelholz, and L. Leuschner, "Energy-utility analysis for resilient systems using probabilistic model checking," in *PETRI NETS*, 2014, pp. 20–39.
[17] J. Climaco, *Multicriteria Analysis*. Springer, 1997.
[18] V. Forejt, M. Kwiatkowska, G. Norman, D. Parker, and H. Qu, "Quantitative multi-objective verification for probabilistic systems," in *TACAS*, 2011, pp. 112–127.
[19] L. Feng, C. Wiltsche, L. Humphrey, and U. Topcu, "Synthesis of human-in-the-loop control protocols for autonomous systems," *IEEE Trans. on Aut. Sci. and Eng.*, vol. 13, no. 2, pp. 450–462, 2016.
[20] K. Chatterjee, R. Majumdar, and T. A. Henzinger, "Controller synthesis with budget constraints," in *HSCC*, 2008, pp. 72–86.
[21] E. Tesarova, M. Svorenova, J. Barnat, and I. Cerna, "Optimal observation mode scheduling for systems under temporal constraints," in *ACC*, 2016, pp. 1099–1104.
[22] A.-A. Agha-Mohammadi, S. Chakravorty, and N. M. Amato, "Firm: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements," *The International Journal of Robotics Research*, vol. 33, no. 2, pp. 268–304, 2014.
[23] D. Bertsekas, *Dynamic programming and optimal control*. Cambridge, MA: Athena Scientific, 2007, vol. 3rd edn.
[24] G. Oriolo, A. De Luca, and M. Vendittelli, "Wmr control via dynamic feedback linearization: design, implementation, and experimental validation," *IEEE Transactions on control systems technology*, vol. 10, no. 6, pp. 835–852, 2002.
[25] C. Linegar, W. Churchill, and P. Newman, "Made to measure: Bespoke landmarks for 24-hour, all-weather localisation with a camera," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 787–794.
[26] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 1, June 2004, pp. I–652–I–659 Vol.1.