Resource-Performance Trade-off Analysis for Mobile Robots

Morteza Lahijanian[1], Maria Svorenova[1], Akshay A. Morye[2], Brian Yeomans[2], Dushyant Rao[2], Ingmar Posner[2], Paul Newman[2], Hadas Kress-Gazit[3], and Marta Kwiatkowska[1]

***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 that is automatic and quantitative 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 achievable optimal trade-offs. 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.**

***Index Terms*—Optimization and Optimal Control, Planning, Scheduling and Coordination, Formal Methods in Robotics and Automation, Autonomous Vehicle Navigation.**

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, 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.

Manuscript received: Sep., 10, 2017; Revised Dec., 22, 2017; Accepted Jan., 19, 2018.

This paper was recommended for publication by Editor Dezhen Song upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by by EPSRC Mobile Autonomy Program Grant EP/M019918/1.

[1]Authors are with Dept. of Computer Science, University of Oxford, Oxford, UK {*firstname.lastname*}@cs.ox.ac.uk

[2]Authors are with Oxford Robotics Institute, University of Oxford, Oxford, UK {akshay, brian, dushyantr, ingmar, pnewman}@robots.ox.ac.uk

[3]Author is with Dept. of Mechanical Engineering, Cornell University, Ithaca, NY hadaskg@cornell.edu

Digital Object Identifier (DOI): see top of this page.

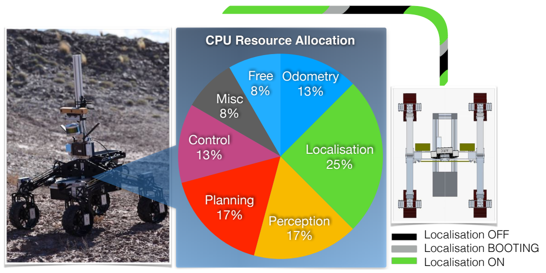


Fig. 1: ARC Q14 robot pictured performing autonomy in Utah, USA, as part of the EPSRC Program Grant EP/M019918/1. The pie chart is an example representation of per-module CPU usage on a Compulab uSVR C3555 CPU.

Mobile autonomy is enabled through several modules such as *localization*, *perception* and *planning*. Localization and perception modules 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 [1], 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. 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, energy consumption may be minimized, but at the same time, probability of collision may be increased. On the other hand, keeping localization on for the duration can lead to excessive energy usage.

The aim of this work 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 [2], [3], 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. In this paper, we particularly focus on scheduling the localization module and illustrate the potential of the framework on a generic resource cost function in conjunction with two performance criteria of safety and target-reachability but emphasize that the technique is not limited to this module and is general.

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 [4], [5], but are limited to discrete models such as Markov decision processes (MDP). The challenge here, therefore, is to adapt the techniques to the space of robotic systems, where both time and space are *continuous domains*, 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: (1) a general framework for the exploration of resource-performance trade-offs in mobile robotics based on multi-objective optimization with various (possibly conflicting) objectives, (2) a discretization of the robot dynamics that enables reduction to the multi-objective problem, and (3) illustration of the benefits and efficacy of the framework in both simulation and experiments of complex robotic scenarios.

II. Literature Review

Most existing resource allocation works in robotics focus on single objective problems, namely optimization for quality of service or energy. Examples include [6] in wireless systems, [7] in multi-robot localization, and [1] in perception. Such problems usually involve scheduling sensor usage, and the typical performance criterion is the "goodness" of the state estimation, *e.g.,* [8], [9]. Apart from [10], where computation is offloaded to the cloud to improve performance, trade-off analysis between resource and performance objectives through scheduling autonomy modules 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 [11], correct-by-construction controller synthesis [12] and performance guarantees [13], [14] 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 [15]. Multi-objective (or multi-criteria) optimization has been extensively studied in operations research and stochastic control [16]. More recently, techniques that combine high-level temporal logic specifications with multi-objective optimization have been formulated for discrete probabilistic models, including probabilistic [2], [17] and expected total reward [3], [18] properties. They have been employed, e.g., to analyze human-in-the-loop [19] and specification revision [20] problems. The works [21], [22] consider budget constraint problems.

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, referred to as the localization module or simply *localization*, additional information is obtained, and the measurements are more accurate at a higher resource cost.

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 $\mathbf{v}^{\mathsf{od}}$.

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 $\tilde{\mathbf{x}}_i \in X$, and the initial state of the robot is $\tilde{\mathbf{x}}_0$. 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 (formally defined 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-C1). 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 cumulative 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 (POMDP). This leads to reasoning in belief space, which is generally a computationally infeasible domain [23]. 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 initial state, 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.,* $\forall t,\ b_t \in \mathbb{B}$.

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 [23]. 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 [24]). Note that the convergence to $\tilde{\mathbf{b}}_i$ is guaranteed if the linearized dynamics are controllable and observable [24]. For a full discussion on the construction of such controllers for various systems, including nonholonomic systems, see [23]. 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 (ϵ-convergence is required to ensure the trigger fires in finite time, *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,

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

where $\mathbf{1}$ 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, $\boldsymbol{\epsilon} = (\epsilon_{\mathsf{mean}}, \epsilon_{\mathsf{var}}) \in \mathbb{R}^2_{>0}$, and for $b_t = \mathcal{N}(\hat{\mathbf{x}}_t, \mathbf{Q}_{x_t})$, where $\hat{\mathbf{x}}_t$ and $\mathbf{Q}_{x_t}$ are the state estimate and covariance at time

t, the indicator function $\mathbf{1}_{<\epsilon}(|b_t - \tilde{\mathbf{b}}_i|) = 1$ 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. Here, a suitable matrix norm is the max-entry norm. From trajectory φ, hence, the following series of control laws that 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 in 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 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}.

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 denoted by $E_{\mathsf{cost}}(\varsigma)$.

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

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

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

$$\mathbb{E}_X(c(\mathbf{x}_t, \mathbf{u}_t, a_t)) = \int_X c(\mathbf{x}_t, \mathbf{u}_t, a_t)\, \mathbf{b}_t \, d\mathbf{x}_t, \tag{8}$$

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

$$E_{\mathsf{cost}}(\varsigma) = \int_{t_0}^{t_{|\varphi|}} \sum_{a_t \in A} \varsigma(\mathbf{b}_t)(a_t)\, \mathbb{E}_X(c(\mathbf{x}_t, \mathbf{u}_t, a_t))\, dt, \tag{9}$$

where $\varsigma(\mathbf{b}_t)(a_t)$ is the probability of choosing action a_t.

2) Task Performance: The task objectives of obstacle avoidance and target reachability need to be reasoned about probabilistically by using expected 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 ς is

$$P_{\mathsf{coll}}(\varsigma) = Prob(\mathbf{x}_{t_0:t_{|\varphi|}} \in X_O) = \int_{X_O} \mathbf{b}_{t_{|\varphi|}}\, d\mathbf{x}_{t_{|\varphi|}}, \tag{10}$$

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

$$P_{\mathsf{targ}}(\varsigma) = Prob(\mathbf{x}_{t_{|\varphi|}} \in X_G) = \int_{X_G} \mathbf{b}_{t_{|\varphi|}}\, d\mathbf{x}_{t_{|\varphi|}}, \tag{11}$$

where $X_G \subset X$ is the set of states corresponding to target 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. Since Problem 1 requires reasoning about the behavior of the system in expectation (see (9)-(11)), 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 as shown in (6). Initially, the belief is $\mathbf{b}_{t_0} = \mathcal{N}(\tilde{\mathbf{x}}_0, \mathbf{Q}_{x_0})$. For $1 \leq i \leq |\varphi|$ 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 t_i is:

$$\mathbf{b}_{t_i} = \begin{cases} \mathcal{P}_{\mathbf{x}}(\mathbf{x}_{t_i} \mid \mathbf{b}_{t_{i-1}}, \tilde{\mathbf{u}}_i, a_{\mathsf{on}}) & \text{if } a_{t_i^-} = a_{\mathsf{on}}, \\ \mathcal{P}_{\mathbf{x}}(\mathbf{x}_{t_i} \mid \mathbf{b}_{t_{i-1}}, \tilde{\mathbf{u}}_i, a_{\mathsf{off}}) & \text{if } a_{t_i^-} \neq a_{\mathsf{on}}, \end{cases} \tag{12}$$

where $a_{t_i^-}$ is the final status of the localization in $[t_{i-1}, t_i)$. Therefore, a discrete *belief graph* in the form of a tree (a directed graph with no cycles) that captures all the expected beliefs at the localization decision points can be constructed. The nodes of the graph are the beliefs $\mathbf{b}_{t_i}$, at which the localization schedule is called, and each edge corresponds to 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 (tree) 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 (similar to [25]) as described below.

2) MDP Construction: Recall that the robot converges to $\tilde{\mathbf{b}}_i$ when $a_{t^-_{i-1}} = a_{\mathsf{on}}$. 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 $X - X_O$:

$$\bar{\mathbf{b}}_t = \mathcal{P}_{\mathbf{x}}(\mathbf{x}_t \mid \mathbf{x}_{t_0}, \mathbf{u}_{t_0:t}, a_{t_0:t}, \mathbf{x}_t \notin X_O). \tag{13}$$

Then, $\bar{\mathbf{b}}_{t_i} = \tilde{\mathbf{b}}_i$ (up to precision ϵ) when $a_{t^-_i} = a_{\mathsf{on}}$. 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 pre-computed distribution, resulting in an independence from the history of the applied localization actions unlike $\mathbf{b}_{t_i}$. This can be viewed as a pruning technique, resulting in a lower number of unique $\bar{\mathbf{b}}_{t_i}$ beliefs, *i.e.,* lower number of unique nodes in the graph. Let $\bar{\mathbf{b}}^j_i$, $0 \le j \le i$, denote the collision-free belief at time t_i with the most recent localization action a_{on} at time t_j, *i.e.,* $a_{t^-_j} = a_{\mathsf{on}}$ and $a_t \ne a_{\mathsf{on}}$ for all $t_j < t < t_i$. The sequential evolution of the collision-free beliefs becomes:

$$\begin{cases} \bar{\mathbf{b}}^i_i = \tilde{\mathbf{b}}_i & \text{if } a_{t^-_i} = a_{\mathsf{on}}, \\ \bar{\mathbf{b}}^j_i = \mathcal{P}_{\mathbf{x}}(\mathbf{x}_{t_i} \mid \bar{\mathbf{b}}^j_{i-1}, \tilde{\mathbf{u}}_i, a_{\mathsf{off}}, \mathbf{x}_{t_i} \notin X_O) & \text{if } a_{t^-_i} \ne a_{\mathsf{on}}. \end{cases} \tag{14}$$

Unlike (12), the above evolution is not deterministic; rather, it is probabilistic. That is, under each localization action, there is a probability associated with the transition from one collision-free belief to the next, and the remaining probability mass is assigned to the collision with an obstacle. Therefore, by reasoning over $\bar{\mathbf{b}}^j_i$ 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 below) with a state structure as in Fig. 2.

An MDP is a tuple $\mathcal{M} = (S, s_{\mathsf{init}}, Act, P)$, where S is a finite set of states, $s_{\mathsf{init}} \in S$ is the initial state, Act is a 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^j_i, for $0 \le i \le |\varphi|$, $0 \le j \le i$, that correspond to beliefs $\bar{\mathbf{b}}^j_i$, where s^i_i indicates that the robot's belief is $\tilde{\mathbf{b}}_i$, and s^0_0 is the initial belief. In addition, S includes states $s_{\mathsf{coll}}, s_{\mathsf{targ}}, s_{\mathsf{free}}$, which correspond to the robot being in collision, target, and

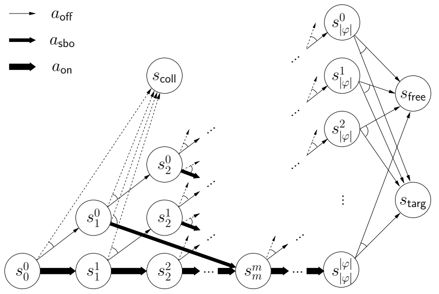


Fig. 2: Structure of the abstraction MDP.

free space, respectively. The set of MDP actions is $Act = \{a_{\mathsf{off}}, a_{\mathsf{on}}, a_{\mathsf{sbo}}\}$, 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^j_i are computed as follows. For actions a_{on} and a_{off}, the values can be computed by evolving $\bar{\mathbf{b}}^j_i$ according to (14). In practice, techniques such as Kalman Filter or Particle Filter can be employed to compute these evolutions as well as their corresponding transition probabilities. A similar computation can be performed for action a_{sbo} by considering T_{boot}. That is, these transition probabilities can be obtained by combining the previously computed probabilities of action a_{off} for the duration of T_{boot} followed by the evolution of the resulting belief with a_{on}. Once the computations for all s^j_i states are complete, the probabilities of being in the target region or in the free space for states with $i = |\varphi|$ are calculated.

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,

$$C(s^j_i, a) = \int_{t_i}^{t_{i+1}} \mathbb{E}_X(c(\mathbf{x}_t, \mathbf{u}_t, a_t))\, dt, \tag{15}$$

where $\mathbb{E}_X(c(\mathbf{x}_t, \mathbf{u}_t, a_t))$ is given by (8), $\mathbf{x}_{t_i} \sim \bar{\mathbf{b}}^j_i$, 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. 2 independently, gaining a speed-up of $\mathcal{O}(|\varphi|)$.

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 (9)-(11) for the robot become the following over $\mathcal{M}$:

$$E_{\mathsf{cost}}(\varsigma_\pi) \;=\; \mathbb{E}_\pi\Big(\sum_{k=0}^{|\varphi|-1} C(s_k, a_k)\Big)\Big), \tag{16}$$

$$P_{\text{coll}}(\varsigma_\pi) = \mathbb{P}_\pi(s_{\text{coll}}), \tag{17}$$
$$P_{\text{targ}}(\varsigma_\pi) = \mathbb{P}_\pi(s_{\text{targ}}), \tag{18}$$

where a_k is picked according to $\pi(s_k)$, and $\mathbb{E}_\pi$ 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. Problem 1, hence, reduces to a multi-objective optimization problem over $\mathcal{M}$, where the goal is to construct a policy that minimizes (16) and (17) and maximizes (18).

As mentioned in Sec. III, however, there may not exist a policy that globally optimizes all three objectives. Hence, we are interested in the set of all optimal trade-offs between the objectives, known as the Pareto front. Let $Y \subseteq \mathbb{R} \times [0,1]^2$ be the set of all achievable values for the three objectives. Point $\mathbf{y} = (y_{E_{\text{cost}}}, y_{P_{\text{coll}}}, y_{P_{\text{targ}}}) \in Y$ is *Pareto optimal* if there does not exist $\mathbf{y}' = (y'_{E_{\text{cost}}}, y'_{P_{\text{coll}}}, y'_{P_{\text{targ}}}) \in Y$ such that $y_{E_{\text{cost}}} \geq y'_{E_{\text{cost}}}$, $y_{P_{\text{coll}}} \geq y'_{P_{\text{coll}}}$, $y_{P_{\text{targ}}} \leq y'_{P_{\text{targ}}}$, and $\mathbf{y} \neq \mathbf{y}'$. *Pareto front* of Y is the set of all Pareto optimal points in Y.

There exist algorithms and off-the-shelf tools, *e.g.,* [4], that solve the above multi-objective problem for MDPs and construct the Pareto front through a value iteration procedure [3]. Given a Pareto point, the algorithms generate the corresponding policy using linear programming [2], [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 the continuous reference trajectory. As this discretization is refined, the obtained optimality approaches the true one in the continuous domain. Furthermore, there are two possible sources of approximation in the MDP abstraction. One is rooted in the ϵ-convergence of the beliefs under a_{on} and their presentation as a single MDP state. For sufficiently small ϵ, this approximation becomes negligible. The other approximation arises in the computations of the transition probabilities and costs of the MDP [23].

VI. Experimental Results

We demonstrate the efficacy of the method in two case studies. We first consider a robot with energy as a resource cost in simulations, and then we show a similar analysis on ARC Q14 planetary rover and deploy it with the obtained schedules in an experimental setup. More case studies, including an application in hardware design, can be found in [26].

A. Second-Order Unicycle

Setup: We considered a mobile robot with (second-order) unicycle dynamics given by $\dot{x}_1 = v\cos\theta$, $\dot{x}_2 = v\sin\theta$, where x_1, x_2 indicate the position, v is the speed, and θ is the heading angle of the robot. The control inputs are the acceleration $\dot{v} = u_1$, and angular velocity $\dot{\theta} = u_2$. We corrupted the dynamics by motion noise $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \sigma_w^2\mathbf{I})$, where $\sigma_w = 0.01$, 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 $\star \in \{\text{od}, \text{lo}\}$, $\sigma_{\text{od}} = 0.2$, $\sigma_{\text{lo}} = 0.03$.

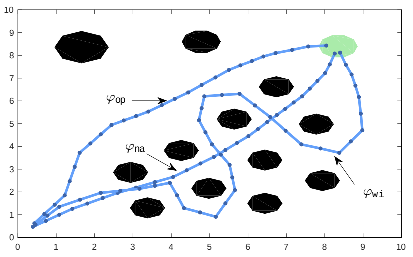


Fig. 3: Workspace of the unicycle robot. Obstacle and target regions are shown in black and green, respectively. The waypoints of φ_{op}, φ_{na}, and φ_{wi} are depicted as dark blue dots.

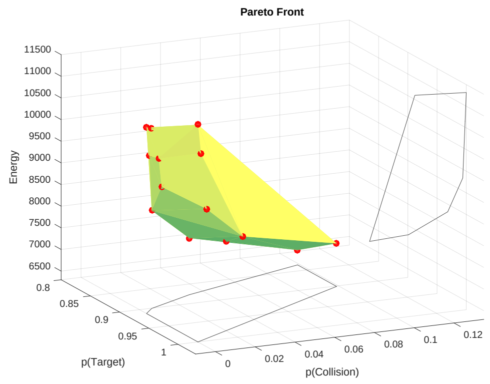


Fig. 4: Visualization of the Pareto front with its projections onto two planes for the unicycle robot on φ_{wi}. Vertices are shown in red.

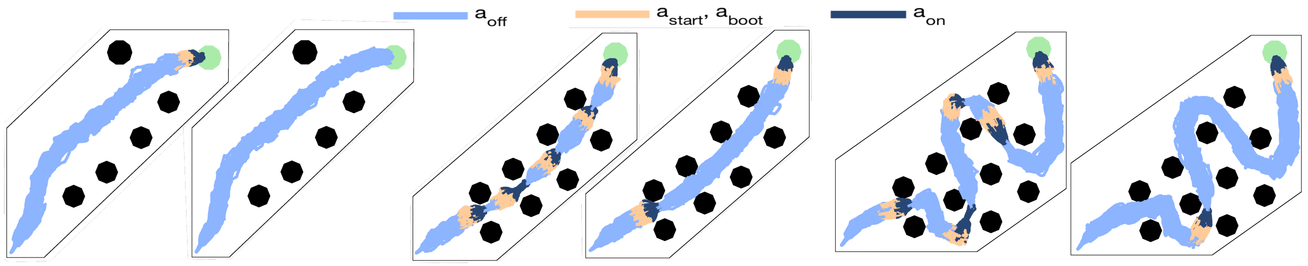
We focused on trade-off analysis of the performance objectives, *i.e.,* collision avoidance and target reaching, and energy consumption as the resource objective. The energy consumption model was taken from [1], where the localization module uses a camera and an algorithm to process the images. The power demand of the module was 8 W when active, and it required time 5 s and energy 40 J to boot. The remaining power consumption for the robot was approximated as 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. To enable reachability and belief stabilization, we designed a feedback controller using dynamic feedback linearization (DFL) and LQG. 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

TABLE I: Pareto front for the unicycle on φ_{wi}. Only the vertices with $P_{\text{targ}} \geq 0.97$ are listed. For comparison, values for ς_{off}, ς_{on}, and localization energy E_{enL} are also shown. E_{enT} is total energy.

P. Pt	P_{targ}	P_{coll}	E_{enT}	E_{enL}	E_{enT}, E_{enL} saved over ς_{on}	
1	1.000	0.000	11 316	1011	66.1%	81.1%
2	0.998	0.002	10 633	876	68.1%	83.6%
3	0.994	0.006	9297	540	72.1%	89.9%
4	0.970	0.032	8325	340	75.0%	93.6%
5 (ς_{off})	0.835	0.099	6642	0	80.1%	100.0%
ς_{on}	1.000	0.000	33 354	5337		



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

Fig. 5: Sample trajectories under two localization schedules. In left images of (a)-(c), the performance guarantees are $P_{\text{targ}} = 1$ and $P_{\text{coll}} = 0$. In right images of (a)-(c), the guarantees are $P_{\text{targ}} = 0.86$ and $P_{\text{coll}} = 0$ for φ_{op} and $P_{\text{targ}} = 0.97$ and $P_{\text{coll}} = 0.03$ for φ_{na} and φ_{wi}.

then computed the Pareto front for each reference trajectory using PRISM-games [5]. In Fig. 4, we show the convex Pareto front (surface) for φ_{wi}. In Table I, we present the vertices of this surface with $P_{\text{targ}} \geq 0.97$ and compare their values against those of the schedule ς_{on}, which keeps the localization on at all times. Every point on the surface of Pareto front corresponds to a particular optimal trade-off between the three objectives, and there exists a localization schedule that achieves it.

In all cases, the localization schedule ς_{on} is not Pareto optimal because there exist localization schedules that have the same probabilistic guarantees as ς_{on}, *i.e.,* $P_{\text{targ}} = 1$ and $P_{\text{coll}} = 0$, 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 because it minimizes the energy consumption. The performance guarantees under ς_{off}, however, are not desirable, *i.e.,* P_{targ} and P_{coll} are 0.86 and 0 for φ_{op}, 0.77 and 0.09 for φ_{na}, and 0.83 and 0.10 for φ_{wi}, respectively. 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 [4]. We simulated 500 sample robot trajectories under each localization schedule and Fig. 5 depicts 100 of them. The first set of schedules correspond to the Pareto points with $P_{\text{targ}} = 1$ and $P_{\text{coll}} = 0$ (left images). To ensure that the target is reached with probability 1 and minimum energy, these schedules turn on localization only at the critical waypoints: near obstacles and right before the target. The second set of schedules correspond to ς_{off} for φ_{op} and Pareto points with $P_{\text{targ}} = 0.97$ and $P_{\text{coll}} = 0.03$ for φ_{na} and φ_{wi}. These schedules trade off the performance by a small percentage to save energy. As shown in the right image of Fig. 5a, ς_{off} keeps localization off for the entire trajectory φ_{op}, resulting in missing the target 14% of times 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 73-75% in energy. In all simulations, the obtained values for the three objectives were within 3% of the corresponding Pareto point values, validating the framework.

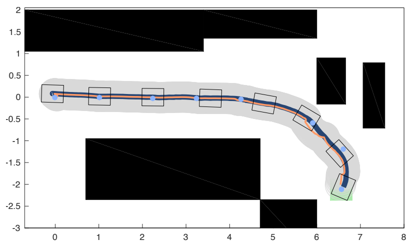
B. Rover Experiments

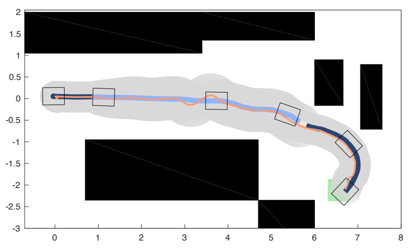
Setup: The robotic platform used in this experimental case study is ARC Q14 planetary rover shown in Fig. 1. It is designed to mimic the configuration and specification found on rovers deployed for planetary exploration. The rover has 4 wheels and 8 motors with max speed of 0.5 m/s (see [1] for full spec). We used Dub4 [27] as the high accuracy localization module, while low accuracy measurements were obtained using Visual Odometry. The energy consumption model was taken from [1] that studied this rover platform.

We modeled the motion of the rover as the unicycle in Sec. VI-A with constrained speed, angular velocity, and acceleration. We used the same DFL as above to linearize the dynamics and employed receding horizon controller for reachability and Kalman Filter for state estimation. We estimated motion and measurement noise as $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ with $\sigma_w = \sigma_{\text{od}} = 0.1$ and $\sigma_{\text{lo}} = 0.01$. The robot's task was to navigate from an entrance to exit door of a meeting room. It was first driven by a human, during which the localization module automatically extracted waypoints (see Fig. 6a).

Pareto Front: We computed the Pareto front for this scenario by first generating the abstraction MDP and then applying 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 smallest total energy E_{enT}. Note that it is possible to save 18-32% in E_{enT} by sacrificing only 0.5-5% in P_{targ}.

Robot Deployment: We deployed the robot under ς_{on} and ς_3. Fig. 6a-b show the robot's trajectories, localization status in blue, yellow, and purple, state estimate in orange, and the projection of the belief's variance 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 it thanks to its initial localization. Once the robot is near a point that is dangerously close to an obstacle, and φ requires sharp maneuvers, ς_3 turns on localization to reduce robot's uncertainty and enable it to perform the maneuvers. Note that, 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


(a) Robot trajectory under ς_{on}.


(b) Robot trajectory under ς_3.

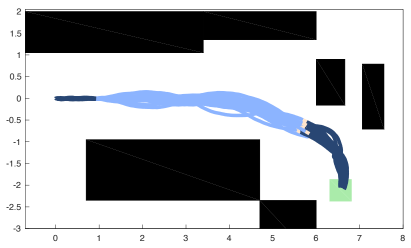

(c) Simulation trajectories under ς_3.

Fig. 6: Robot trajectories in experiments and simulations. The waypoints are shown as light blue dots in (a). The blue, yellow, and purple 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 $P_{\text{targ}} = 1$ and $P_{\text{coll}} = 0$ while they are $P_{\text{targ}} = 0.99$ and $P_{\text{coll}} = 0.01$ for ς_3 in trade-off for 24% reduction in E_{enT}.

TABLE II: Pareto front for the planetary rover. For comparison, we list the energy savings of Pareto-optimal schedules against ς_{on}.

P. Pt	P_{targ}	P_{coll}	E_{enT}	E_{enL}	E_{enT}, E_{enL} saved	
1	0.500	0.000	6156	508	30.0%	65.3%
2 (ς_{on})	1.000	0.000	8792	1465	0.0%	0.0%
3 (ς_3)	0.990	0.010	6713	651	23.6%	55.6%
4	0.995	0.005	7215	814	17.9%	44.4%
5 (ς_{off})	0.443	0.005	4806	0	45.3%	100.0%
6	0.955	0.045	5970	325	32.1%	77.8%

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. 6c shows 50 trajectories of the center of the robot obtained in simulations prior to the rover deployment.

VII. Final Remarks and Future Work

In this work, we proposed a general framework for exploration of performance-resource trade-offs and demonstrated its efficacy in robot design. The framework can be adapted to schedule other modules such as perception, different motors, or various localization modules. The future work directions include performance-resource trade-off analysis for: more complex cost measures such as long-run average cost; systems with delayed measurements; a combination of modules, *e.g.,* localization and perception, and, ultimately, localization, perception and planning.

References

[1] 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.

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

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

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] H. Kress-Gazit, G. Fainekos, and G. J. Pappas, "Where's waldo? sensor-based temporal logic motion planning," in *ICRA*, 2007, pp. 3116–3121.

[12] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon control for temporal logic specifications," in *HSCC*, 2010.

[13] 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.

[14] R. Luna, M. Lahijanian, M. Moll, and L. E. Kavraki, "Asymptotically optimal stochastic motion planning with temporal goals," in *WAFR*, 2014, pp. 335–352.

[15] 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.

[16] J. Climaco, *Multicriteria Analysis*. Springer, 1997.

[17] E. M. Hahn, V. Hashemi, H. Hermanns, M. Lahijanian, and A. Turrini, "Multi-objective robust strategy synthesis for interval Markov decision processes," in *QEST*. Springer, 2017, pp. 207–223.

[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] M. Lahijanian and M. Kwiatkowska, "Specification revision for Markov decision processes with optimal trade-off," in *CDC*. IEEE, 2016, pp. 7411–7418.

[21] K. Chatterjee, R. Majumdar, and T. A. Henzinger, "Controller synthesis with budget constraints," in *HSCC*, 2008, pp. 72–86.

[22] E. Tesarova, M. Svorenova, J. Barnat, and I. Cerna, "Optimal observation mode scheduling for systems under temporal constraints," in *ACC*, 2016, pp. 1099–1104.

[23] 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.

[24] D. Bertsekas, *Dynamic programming and optimal control*. Cambridge, MA: Athena Scientific, 2007, vol. 3rd edn.

[25] S. Patil, J. Van Den Berg, and R. Alterovitz, "Estimating probability of collision for safe motion planning under gaussian motion and sensing uncertainty," in *Int. Conf. on Rob. and Aut.* IEEE, 2012, pp. 3238–3244.

[26] M. Lahijanian, M. Svorenova, , A. A. Morye, B. Yeomans, D. Rao, I. Posner, P. Newman, H. Kress-Gazit, and M. Kwiatkowska, "Resource-performance trade-off analysis for mobile robot design," *arXiv:1609.04888 [cs.RO]*, 2017, https://arxiv.org/abs/1609.04888.

[27] C. Linegar, W. Churchill, and P. Newman, "Made to measure: Bespoke landmarks for 24-hour, all-weather localisation with a camera," in *Int. Conf. on Rob. and Aut.* IEEE, May 2016, pp. 787–794.