2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

Planning Periodic Persistent Monitoring Trajectories for Sensing Robots in Gaussian Random Fields

Xiaodong Lan and Mac Schwager

Abstract— **This paper considers the problem of planning a trajectory for a sensing robot to best estimate a time-changing Gaussian Random Field in its environment. The robot uses a Kalman filter to maintain an estimate of the field value, and to compute the error covariance matrix of the estimate. A new randomized path planning algorithm is proposed to find a periodic trajectory for the sensing robot that tries to minimize the largest eigenvalue of the error covariance matrix over an infinite horizon. The algorithm is proven to find the minimum infinite horizon cost cycle in a graph, which grows by successively adding random points. The algorithm leverages recently developed methods for periodic Riccati recursions to efficiently compute the infinite horizon cost of the cycles, and it uses the monotonicity property of the Riccati recursion to efficiently compare the cost of different cycles without explicitly computing their costs. The performance of the algorithm is demonstrated in numerical simulations.**

I. Introduction

In this paper we investigate the problem of controlling a sensing robot to persistently monitor a continually changing field in its environment [1], [2], [3]. Consider, for example, an Unpiloted Aerial Vehicle (UAV) with a radiation sensor that must continually fly over the site of a nuclear accident to maintain an estimate of the time changing levels of radiation in the region. The objective of the vehicle is to execute a trajectory that optimizes, over an infinite horizon, the estimation accuracy. For this purpose, we propose a new randomized path planning algorithm, which we call Rapidly-expanding Random Cycles (RRC), that returns periodic trajectories with guaranteed infinite horizon sensing performance.

We model the environmental field as a Gaussian Random Field (GRF) with linear dynamics driven by Gaussian white noise, and our robot's sensor takes measurements that are linear combinations of the field value with additive Gaussian white noise. This is a suitable model for a broad range of environmental scalar fields [4], [5], [6]. The sensing robot estimates the field using a Kalman filter, which is well known to minimize the mean squared estimation error for a GRF with linear dynamics. However, our problem is distinguished from a typical estimation setting by the fact that our robot's sensing performance is a function of its position. Intuitively, the robot only obtains information about the part of the environment that is closest to it. Since the environment is dynamic, the robot must perpetually move to maintain an estimate of the field value at different places. Our algorithm finds a trajectory for the agent to execute in order to minimize the asymptotic supremum (the $\limsup$) of a measure of the quality of its field estimate. We specifically focus on finding a periodic trajectory because recent results in optimal sensor scheduling [7], [8] have shown that an optimal infinite horizon sensor schedule can be arbitrarily closely approximated by a periodic schedule.

This work was supported by ONR grant N00014-12-1-1000.

X. Lan and M. Schwager are with the Department of Mechanical Engineering and the Division of Systems Engineering, Boston University, Boston, MA 02215, USA, xlan@bu.edu schwager@bu.edu.

Finding the optimal infinite horizon trajectory is intractable in general, however our algorithm produces a series of periodic trajectories with monotonically decreasing cost, that are the minimum cost cycles in a randomly expanding graph. More specifically, our algorithm expands a tree of randomly generated nodes over the environment, a strategy inspired by recent randomized planning algorithms [9]. The algorithm maintains a record of the minimal cost cycle that can be created in this tree by adding a single edge. When a new point is added to the tree, the cost of several new cycles must be evaluated and compared with the current optimal cycle. This is accomplished by leveraging a technique, based on structure-preserving algorithms [10], for finding the asymptotic solution to periodic Riccati recursions efficiently. We also propose a new efficient test to compare the cost of one cycle with another without explicitly calculating its asymptotic cost. Specifically, the main contributions of this paper are as follows:

- We propose the Rapidly-exploring Random Cycles (RRC) algorithm to find periodic trajectories that are suitable for persistent monitoring of a Gaussian Random Field (Algorithm 1).
- We prove that the RRC algorithm gives the minimal cost simple cycle in an expanding graph of random points in the environment (Theorem 1).
- We adapt a structure-preserving algorithm to evaluate the cost of cycles when needed, and we give a procedure for comparing the cost of two cycles that does not require their costs to be computed explicitly.
- We demonstrate the algorithm in numerical simulations.

The rest of this paper is organized as follows. The problem is formulated in Section II. Our trajectory planning algorithm is described in Section III. Efficient computation of the asymptotic cost of periodic trajectories is described in Section IV. Section V presents the results of numerical simulations, and we discuss conclusions in Section VI.

II. Problem Formulation

A. Persistent Environmental Monitoring

Consider an environmental field that is modeled as a dynamic Gaussian Random Field (GRF), $\{\phi_t(q_i)\}_{q_i \in D}, t =$

$0, 1, \ldots$, where D is a discrete set of points of interest. Hence, for a fixed t, ϕ_t is a $n \times 1$ Gaussian column vector, where n is the number of points of interest. Let the dynamics of the environment be described by a linear time invariant system

$$\phi_{t+1} = A\phi_t + w_t$$

where A is a known $n \times n$ matrix and w_t is the process noise which is assumed to be white and with known Gaussian distribution $w_t \sim N(0, Q)$. This is a well-established model for spatial environmental processes [6], [11]. There are other models however, for example [4], [5] use a linear combination of basis functions for estimation rather than using a GRF.

The sensing robot moves along a path in the environment and takes measurements at fixed time intervals. Let $X \subset \mathbb{R}^d$ and $U \subset \mathbb{R}^m$ be the state space and space of control inputs, respectively, of the agent. The dynamics of the agent is given by

$$x_{t+1} = x_t + u_t, \quad \|u_t\| \leq \eta, \quad x(0) = x_0$$

where $x_t \in X$, $u_t \in U$, and $\eta > 0$ is the largest distance the agent can travel in one time step. We consider these simple dynamics so as not to complicate the scenario, although more complex dynamics $x_{t+1} = f_t(x_t, u_t)$ can be accommodated with some modifications. The agent has a sensor which gives measurements at each time step according to the sensing model

$$y_t = C(x_t)\phi_t + v_t$$

where v_t is the measurement noise which is white, independent from w_t, and with known Gaussian distribution $v_t \sim N(0, R)$. Here $C(x_t)$ depends on the agent state x_t, because the agent's sensor has a measurement quality which diminishes with the distance to a point of interest in the GRF. For example, we may let the ith entry of C be given by a Gaussian decay, $c_i(x_t) = e^{-\|x_t - q_i\|^2/2\sigma_c^2}$, or we could let the ith entry be 1 inside some radius from the agent and zero outside the radius. Our results do not depend on the specific form of the dependence on x_t. Our goal is to drive the sensing agent to estimate ϕ_t while minimizing some measure of the estimation uncertainty.

We use the Kalman filter for estimation which is known to be the optimal estimator for this model in the sense that it minimizes the mean squared estimation error [12]. Let us denote the estimate of ϕ_t by a random vector $\bar{\phi}_t$, with an expected value $\hat{\phi}_t$. The *a priori* and *a posteriori* covariance matrices associated with this estimate are denoted by Σ_t and Σ_t^+, respectively, and the filter maintains a Gaussian estimate $\bar{\phi}_t \sim N(\hat{\phi}_t, \Sigma_t^+)$. From the Kalman filter equations, we can get a well-known Riccati equation between two consecutive steps which updates the *a priori* error covariance matrix

$$\Sigma_{t+1} = A\Sigma_t A^T - A\Sigma_t C(x_t)^T \times \left(C(x_t)\Sigma_t C(x_t)^T + R\right)^{-1} C(x_t)\Sigma_t A^T + Q$$

We will be interested in asymptotic solutions to this recursion, which are known not to depend on the initial condition Σ_0.

B. Performance Metric and Optimality

We choose the spectral radius (i.e. the largest eigenvalue) of the *a priori* error covariance matrix, $\rho(\Sigma_t)$, as the objective function. The spectral radius is an indication of the worst error covariance over all linear combinations of the points of interest in the environment. Our goal is to design a trajectory for the agent such that when it moves along this trajectory the maximum over time of the spectral radius of the error covariance matrix will be minimized. However, in order not to be influenced by initial transient effects we consider the asymptotic limit of this maximum

$$J(\sigma, \Sigma_0) = \limsup_{t \to \infty} \rho(\Sigma_t^\sigma), \tag{1}$$

where σ denotes a trajectory for the agent, and Σ_t^σ is the error covariance attained at time t along the trajectory σ from initial condition Σ_0. Note that the limit of the supremum always exists even though Σ_t^σ does not necessarily converge as t goes to infinity.

We define the Riccati map $g_{x_t} : \mathcal{A} \mapsto \mathcal{A}, t \in \{1, 2, \ldots\}$ as

$$g_{x_t} := A\Sigma_t A^T - A\Sigma_t C(x_t)^T \times \left(C(x_t)\Sigma_t C(x_t)^T + R\right)^{-1} C(x_t)\Sigma_t A^T + Q \tag{2}$$

where $\mathcal{A}$ is the cone of semi-definite matrices. Initially, when $t = 0$, then $g_{x_0} = A\Sigma_0 A^T + Q$. Consider the error covariance over an interval of time steps $\tau_1, \tau_1 + 1, \ldots, \tau_2$, then we define the repeated composition of g_{x_t} over the interval as

$$G_x^{\tau_1:\tau_2}(\Sigma_{\tau_1}) := g_{x_{\tau_2}}(g_{x_{(\tau_2-1)}}(\cdots g_{x_{\tau_1}}(\Sigma_{\tau_1}))), \tag{3}$$

so that $\Sigma_{(\tau_2+1)} = G_x^{\tau_1:\tau_2}(\Sigma_{\tau_1})$.

One important property of the Riccati recursion (3) (which was proven in [8], Corollary 1) is restated here in the following Lemma.

Lemma 1 (Monotonicity of Composite Riccati Maps): For any $\Sigma_1, \Sigma_2 \in \mathcal{A}$, $\alpha \in [0, 1]$, and $\tau_1, \tau_2 \in \{0, 1, 2, \ldots\}$, where $\tau_1 < \tau_2$, we have: if $\Sigma_1 \preceq \Sigma_2$, then $G_x^{\tau_1:\tau_2}(\Sigma_1) \preceq G_x^{\tau_1:\tau_2}(\Sigma_2)$.

We will use this monotonicity property of the composite Riccati map to compare the asymptotic cost of different periodic trajectories without explicitly computing their infinite horizon cost.

Denote the free state space and the obstacle region by X_{free} and X_{obs}, respectively. A feasible trajectory is $\sigma : [0, \infty) \mapsto X_{free}$, that is, $\sigma(t) \in X_{free}$ for $t \in [0, \infty)$. In our case, trajectories are characterized by a sequence of discrete waypoints, $(x_0, x_1, x_2, \ldots, x_T)$, connected by straight lines which satisfy $\|x_{i+1} - x_i\| \leq \eta$. Denote by $M^\infty(x_0)$ all the feasible trajectories with initial condition $x_0 \in X_{free}$. Denote the covariance matrix along a trajectory by Σ_t^σ. We now give a formal statement of our persistent environmental monitoring problem.

Problem 1 (Persistent Monitoring of a GRF): Given a bounded and connected environment X with a dynamic Gaussian Random Field $\phi_t(q_i)$, over a set of points of interest $q_i \in D$, a mobile sensing agent $x_{t+1} = x_t + u_t$, with initial state x_0, and with a measurement system

$y_t = C(x_t)\phi_t + v_t$, and with cost $J(\sigma, \Sigma_0)$ given by (1), find a feasible trajectory σ^* such that

$$\sigma * \in \underset{\sigma \in M^\infty(x_0)}{\operatorname{argmin}} \; J(\sigma, \Sigma_0) \tag{4}$$

subject to

$$\begin{aligned}\Sigma_{t+1}^\sigma &= A\Sigma_t^\sigma A^T - A\Sigma_t^\sigma C(x_t)^T \\ &\times \left(C(x_t)\Sigma_t^\sigma C(x_t)^T + R\right)^{-1} C(x_t)\Sigma_t^\sigma A^T + Q \\ \eta &\geq \|x_{t+1} - x_t\|.\end{aligned}$$

III. Trajectory Planning

In this section we propose an incremental sampling-based planning algorithm to give approximate solutions to Problem 1 by finding periodic trajectories with finite periods. Our algorithm is similar to the RRT* algorithm [9]. However, unlike RRT*, we maintain a record of the minimum cost cycle that can be obtained from the random tree by adding a single edge.

Our search for only periodic trajectories is motivated by several powerful results proven in [7] concerning optimal infinite horizon sensor schedules. Let $\tilde{\sigma}$ be a periodic trajectory with period T, so that $\tilde{\sigma}(t) = \tilde{\sigma}(t+T)$ for all $t = 1, 2, \ldots$. Let the T waypoints in $\tilde{\sigma}$ be given by $(x_1, x_2 \ldots, x_T)$, and denote the composite Riccati map that starts at x_i and ends at x_i after moving around the cycle once by $G_{x_i}^{\tilde{\sigma}}$. Using this notation, we summarize the extension of the results from [7] to our Problem 1 in the following corollary.

Corollary 1: For any $\delta > 0$, Σ_0 and x_0, if there exists an optimal trajectory for Problem 1 with optimal cost J^*, then there exists a periodic trajectory $\tilde{\sigma}$ consisting of waypoints $(x_1, x_2, \ldots, x_T)$ with a finite period $T(\delta) \in \mathbb{Z}_{>0}$ and an infinite horizon cost $J(\tilde{\sigma})$ such that

1) $0 \leq J(\tilde{\sigma}) - J^* \leq \delta$,
2) The trajectory of the covariance matrix $\Sigma_t^{\tilde{\sigma}}$ converges exponentially to a unique limit cycle, $\Sigma_\infty^{x_1}, \Sigma_\infty^{x_2}, \ldots, \Sigma_\infty^{x_T}$, where $\Sigma_\infty^{x_i}$ is the fixed point of the composite Riccati map $\Sigma_\infty^{x_i} = G_{x_i}^{\tilde{\sigma}}(\Sigma_\infty^{x_i})$, for all $i = 1, 2, \ldots T$. So $J(\tilde{\sigma}) = \rho(\Sigma_\infty^{\tilde{\sigma}}) = \max_{i \in \{1,2,\ldots,T\}} \rho(\Sigma_\infty^{x_i})$,
3) J^* is independent of Σ_0.

Proof: The proof of the first two parts follows directly from theorem 4 in [7], and the proof of the third part follows from theorem 2 in [7]. ■

Since the cost of a periodic trajectory can be arbitrarily close to the optimal cost of Problem 1, we propose an algorithm to search an expanding tree for the best feasible periodic path.

A. Primitive Procedures

Before describing the operation of the algorithm in detail, we provide several primitive procedures that will be used in the algorithm. All of these except for FindCycle and PersistentCost are identical to the primitive procedures from [9]. Hence we very briefly describe the primitive procedures that are the same as in [9] below (please see [9] for more details on these). We then give more detail on the primitive procedures that are unique to our algorithm.

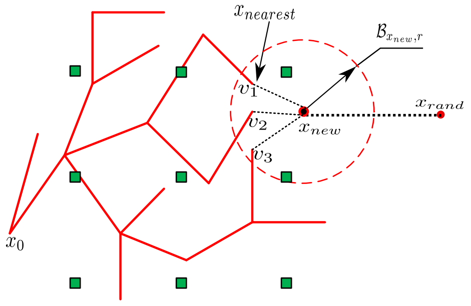
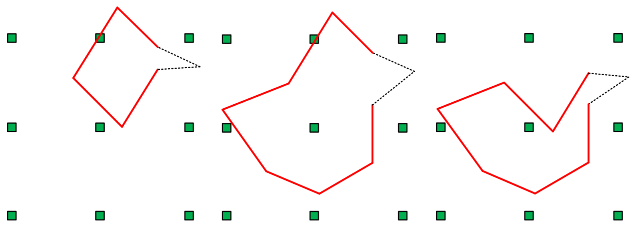


Fig. 1. This figure shows the expanding tree in the RRC algorithm. The red lines represent edges in the tree, and the green squares indicate the points of interest in the field.



(a) Potential cycle 1. (b) Potential cycle 2. (c) Potential cycle 3.

Fig. 2. This figure shows the new potential cycles found by adding the new point x_{new} to the tree in Figure 1.

SampleFree : $\mathbb{Z}_{>0} \to X_{free}$ returns uniformly distributed samples from X_{free}. In Figure 1, it returns x_{rand}. Nearest : $(G, x) \to v$ returns a vertex $v \in V$ which is closest to x in terms of the Euclidean distance. In Figure 1, it returns $x_{nearest}$. Steer : $(x, y, \eta) \to z$ returns a point $z \in X_{free}$ such that $\|z - y\|$ is minimized while $\|z - x\| \leq \eta$. In Figure 1, it returns x_{new}. Near : $(G, x, r) \to V' \subseteq V$ returns the vertices in V that are contained in a closed ball of radius r centered at x. In Figure 1, it returns v_1, v_2, v_3. CollisionFree(x, y) returns True if the straight line segment between x and y lies in X_{free} and False otherwise. Now we describe in more detail the procedures that are unique to our algorithm.

Find Cycles: Given a spanning tree $T = (V, E')$ of a graph $G = (V, E)$, and a point $x \in X_{free}$, the function FindCycle : $(v_1, v_2, x) \to C_{v_1,v_2,x}$ returns the index of the vertices in a cycle, where v_1 and v_2 are any two vertices in the spanning tree $T = (V, E')$, and $C_{v_1,v_2,x}$ is a cycle which includes vertices v_1, v_2 and x. In Figure 1, it returns 3 cycles, shown in Figure 2.

Calculating Cycle Cost: Given a cycle returned by the function FindCycle, the function PersistentCost returns the infinite horizon cost of the cycle. We will discuss how to calculate this cost in detail in Section IV. That is, PersistentCost : $C_{v_1,v_2,x} \to \mathbb{R}_{>0}$.

B. Generating a Periodic Path

We now describe the algorithm in detail, using the primitive procedures introduced above. The algorithm starts with an initial position x_0. We also initialize the algorithm by $minCycleIndex = \emptyset$ and $minCycleCost = \infty$. The variable $minCycleIndex$ is used to store the index of all the vertices in the cycle with minimum cost, and $minCycleCost$ is used to store the cost of the cycle. In this algorithm, we will keep a tree structure. At each step, we generate a new random point x_{rand} by the function SampleFree. With this new random point and the function Nearest, we find the nearest vertex $x_{nearest}$ of the graph to the random point. Then we apply the Steer function to get a potential new vertex x_{new} for the tree. These procedures are the same with RRT* from [9].

From the function Near, we get a set X_{near} which includes the near neighbors of x_{new}. If there is only one vertex in X_{near}, i.e. $x_{nearest}$, then we just connect x_{new} to $x_{nearest}$. If there are more than one vertices in X_{near}, then we do several connecting tests. That is, we do a test by connecting x_{new} to any two vertices inside X_{near}. By connecting the new point with two vertices, we form a single cycle, which is made up of the two edges connecting the vertices to the new point plus the unique path through the tree between the two vertices. If there are k vertices in X_{near}, then we will have to consider $\frac{k(k-1)}{2}$ cycles formed in this way. We compare the cost of these cycles and choose the one with the minimum cost. We can use the procedure PersistentCost to calculate these costs.

After we find out the cycle with minimum cost among the $\frac{k(k-1)}{2}$ cycles, we give the value of the minimum cost to $cycleCost$. We also look for the indices of the vertices inside this cycle using the function FindCycle, and give them to $cycleIndex$. The variable $cycleVertexIndex$ is used to store either of the two vertices which belong to X_{near} and $cycleIndex$. That is, $cycleVertexIndex \in (X_{near} \cap cycleIndex)$. Then we connect x_{new} to the vertex in $cycleVertexIndex$. So we only add one new edge to the spanning tree, and the new graph will still be a spanning tree. After this we compare the cost of this cycle with $minCycleCost$. If it is smaller, then we update the $minCycleCost$ and make it equal to the cost of the new cycle. That is, $minCycleCost = cycleCost$. We also update $minCycleIndex$ by $minCycleIndex = cycleIndex$. We repeat this process for $numSteps$ times and we get a tree with $numSteps + 1$ vertices and a fundamental cycle of this tree, i.e., $minCycleIndex$. This cycle is the periodic trajectory for Problem 1. A pseudo-code implementation of the algorithm is shown in Algorithm 1. The following theorem characterizes the performance of the algorithm.

Theorem 1 (Properties of RRC algorithm): At each iteration, the RRC algorithm gives the minimal cost feasible simple cycle that can be created from the graph G by adding a single edge. This cost is monotonically non-increasing in the number of iterations.

Proof: The proof is by induction. Let $\tilde{\sigma}_i$ be the trajectory given by the algorithm at iteration i, and denote by Ξ_i the set of all feasible simple cycles that can be created by adding one edge to G_i. Assume that $\tilde{\sigma}_i = \text{argmin}_{\tilde{\sigma}\in\Xi_i} J(\tilde{\sigma})$. In iteration $i+1$ a single vertex x_{new} is added to the graph, during which all feasible cycles that include this vertex are compared. Denote these new cycles by $\Delta\Xi_{i+1}$, and note that $\Xi_{i+1} = \Xi_i \cup \Delta\Xi_{i+1}$. The cycle with minimal cost among $\{\tilde{\sigma}_i, \Delta\Xi_{i+1}\}$ is taken to be $\tilde{\sigma}_{i+1}$, hence $\tilde{\sigma}_{i+1} = \text{argmin}_{\tilde{\sigma}\in\{\tilde{\sigma}_i,\Delta\Xi_{i+1}\}} J(\tilde{\sigma}) = \text{argmin}_{\tilde{\sigma}\in\Xi_i\cup\Delta\Xi_{i+1}} J(\tilde{\sigma}) = \text{argmin}_{\tilde{\sigma}\in\Xi_{i+1}} J(\tilde{\sigma})$. The initial case for the induction follows from the fact that x_0 is the minimum cost cycle in its own trivial graph. To prove monotonicity, notice that $\Xi_{i+1} \supset \Xi_i$, therefore $\min_{\tilde{\sigma}\in\Xi_{i+1}} J(\tilde{\sigma}) \leq \min_{\tilde{\sigma}\in\Xi_i} J(\tilde{\sigma})$. ■

Algorithm 1 Rapidly-exploring Random Cycles (RRC)

```
1: $V \leftarrow x_0$; $E \leftarrow \emptyset$; $minCycleIndex \leftarrow \emptyset$; $minCycleCost \leftarrow \infty$;
2: for $i = 1$ to $numSteps$ do
3:   $x_{rand} \leftarrow$ SampleFree;
4:   $x_{nearest} \leftarrow$ Nearest$(G = (V, E), x_{rand})$;
5:   $x_{new} \leftarrow$ Steer$(x_{nearest}, x_{rand}, \eta)$;
6:   if CollisionFree$(x_{nearest}, x_{new})$ then
7:     $X_{near} \leftarrow$ Near$(G = (V, E), x_{new}, r)$;
8:     $V \leftarrow V \cup x_{new}$;
9:     if $size(X_{near}) == 1$ then
10:      $E \leftarrow E \cup (x_{nearest}, x_{new})$;
11:    else
12:      $cycleCost \leftarrow \inf$;
13:      $cycleVertexIndex \leftarrow x_{nearest}$;
14:      $cycleIndex \leftarrow \emptyset$;
15:      for any two $(x^1_{near}, x^2_{near}) \in X_{near}$ do
16:        if CollisionFree$(x^1_{near}, x_{new})$ and CollisionFree$(x^2_{near}, x_{new})$ then
17:          $\tilde{\sigma} \leftarrow$ FindCycle$(x^1_{near}, x^2_{near}, x_{new})$;
18:          $J(\tilde{\sigma}) \leftarrow$ PersistentCost$(\tilde{\sigma})$;
19:          if $J(\tilde{\sigma}) < cycleCost$ then
20:            $cycleVertexIndex \leftarrow x^2_{near}$ (or $x^1_{near}$);
21:            $cycleCost \leftarrow J(\tilde{\sigma})$;
22:            $cycleIndex \leftarrow \tilde{\sigma}$;
23:          end if
24:        end if
25:      end for
26:      $E \leftarrow E \cup (cycleVertexIndex, x_{new})$;
27:      if $cycleCost < minCycleCost$ then
28:        $minCycleCost \leftarrow cycleCost$;
29:        $minCycleIndex \leftarrow cycleIndex$;
30:      end if
31:    end if
32:  end if
33: end for
34: return $G = (V, E)$, $minCycleIndex$, $minCycleCost$;
```

IV. Efficient Computation of Cycle Cost

In the primitive procedure PersistentCost, we need to calculate the infinite horizon cost of a cycle. Let's consider such a cycle $\tilde{\sigma}$ with waypoints $(x_1, x_2, \ldots, x_T)$ and period T, when taking measurements along this cycle, we will get a discrete-time stochastic periodic system with period T.

$$\phi_{t+1} = A\phi_t + w_t, w_t \sim N(0, Q) \tag{5a}$$

$$y_t = C(x_t)\phi_t + v_t, v_t \sim N(0, R) \tag{5b}$$

where $C(x_t) = C(x_{t+T})$. Also, by doing Kalman filtering along this cycle, we will get a discrete-time periodic Riccati equation (DPRE) with period T.

$$\Sigma_\infty^{x_{i-1}} = A\Sigma_\infty^{x_i}A^T - A\Sigma_\infty^{x_i}C(x_i)^T \times \left(C(x_i)\Sigma_\infty^{x_i}C(x_i)^T + R\right)^{-1} C(x_i)\Sigma_\infty^{x_i}A^T + Q \quad (6)$$

[13] proves that there exists a unique symmetric periodic positive semidefinite (SPPS) solution to this DPRE if and only if the periodic system (5) is stabilizable and detectable.

Next we will discuss how to solve for this DPRE numerically and efficiently. One naive way is to start with an initial covariance matrix and calculate an approximation by brute force. In this work, we will use the structure-preserving algorithm introduced in [10] to calculate the SPPS solution, because to the best of our knowledge this method is the most efficient approach.

A. Structure-preserving Algorithms for DPRE

Structure-preserving algorithms consist of structure-preserving swap and collapse algorithm (SSCA) and structure-preserving doubling algorithm (SDA). Basically, this algorithm uses the SSCA algorithm to get an equivalent Riccati equation for the periodic Riccati equation, and then it uses the SDA algorithm to solve this equivalent Riccati equation and get $\Sigma_\infty^{x_T}$, from which the other $\Sigma_\infty^{x_i}, (i = T-1, \ldots, 1)$ can be found through equation (6). More details can be found in [10].

B. Comparing the Cost of Different Cycles

In line 19 of Algorithm 1, when we compare the infinite time horizon cost of two cycles, a naive method is to use the function PersistentCost to calculate the cost of the two cycles respectively, and then compare them. However, it is always good if we can avoid calculating the infinite horizon cost of some cycles, because it increases the computation burden of our algorithm even though the structure-preserving algorithm is very efficient. As in Corollary 1, let the composite Riccati map corresponding to one complete cycle of a periodic trajectory $\tilde{\sigma}$ that starts and ends at x_i be denoted by $G_{x_i}^{\tilde{\sigma}}$, and recall that $J(\tilde{\sigma}) = \max_{i=1,2,\ldots,T} \rho(\Sigma_\infty^{x_i})$. We have the following theorem to compare the cost of two cycles.

Theorem 2 (Cycle Cost Comparison): Consider two periodic paths, $\tilde{\sigma}_x$ with period T_x and points $(x_1, x_2, \ldots, x_{T_x})$, and $\tilde{\sigma}_y$ with period T_y and points $(y_1, y_2, \ldots, y_{T_y})$, and let $k = \text{argmax}_{i=1,2,\ldots,T_x} \rho(\Sigma_\infty^{x_i})$. We have that

1) if there exists one $G_{y_i}^{\tilde{\sigma}_y}(\Sigma_\infty^{x_k}) \succeq \Sigma_\infty^{x_k}$, then $J(\tilde{\sigma}_y) \geq J(\tilde{\sigma}_x)$,
2) if $G_{y_i}^{\tilde{\sigma}_y}(\Sigma_\infty^{x_k}) \preceq \Sigma_\infty^{x_k}$ for all $i = 1, 2, \ldots, T_y$, then $J(\tilde{\sigma}_y) \leq J(\tilde{\sigma}_x)$,
3) if there exists at least one $(G_{y_i}^{\tilde{\sigma}_y}(\Sigma_\infty^{x_k}) - \Sigma_\infty^{x_k})$ remaining indefinite and there is no $G_{y_i}^{\tilde{\sigma}_y}(\Sigma_\infty^{x_k}) \succeq \Sigma_\infty^{x_k}$ for $i = 1, 2, \ldots, T_y$, then we can NOT compare $J(\tilde{\sigma}_y)$ and $J(\tilde{\sigma}_x)$.

Proof: Here we only prove part 1. The proof for part 2 is similar, and by proving part 1 and part 2, we proved part 3. Without loss of generality, let us consider point y_1 in the path $\tilde{\sigma}_y$. By part 3 of Corollary 1, $J(\tilde{\sigma}_y)$ is independent of where we start. So we choose $\Sigma_0^{y^1} = \Sigma_\infty^{x^k}$ as the initial covariance matrix for the Riccati recursion along $\tilde{\sigma}_y$. We denote the trajectory of the covariance matrix at y_1 by $\Sigma_0^{y_1}, \Sigma_{T_y}^{y_1}, \Sigma_{2T_y}^{y_1}, \Sigma_{3T_y}^{y_1}, \ldots, \Sigma_\infty^{y_1}$, so

$$\begin{aligned} \text{if } \Sigma_{T_y}^{y_1} &= G_{y_1}^{\tilde{\sigma}_y}(\Sigma_\infty^{x_k}) \preceq \Sigma_\infty^{x_k} \\ \text{then } \Sigma_{2T_y}^{y_1} &= G_{y_1}^{\tilde{\sigma}_y}(\Sigma_{T_y}^{y_1}) \preceq G_{y^1}^{\tilde{\sigma}_y}(\Sigma_\infty^{x_k}) = \Sigma_{T_y}^{y_1} \preceq \Sigma_\infty^{x_k} \\ &\vdots \\ \Sigma_\infty^{y_1} &\preceq \Sigma_\infty^{x_k} \\ &\Rightarrow \rho(\Sigma_\infty^{y_1}) \leq \rho(\Sigma_\infty^{x_k}), \end{aligned}$$

where the inequalities are an application of the monotonicity of the composite Riccati map from Lemma 1. Similarly, we can prove that $\rho(\Sigma_\infty^{y_i}) \leq \rho(\Sigma_\infty^{x_k}), i = 2, \ldots, T_y$. So we have

$$J(\tilde{\sigma}_y) = \max_{i=1,2,\ldots,T_y} \rho(\Sigma_\infty^{y_i}) \leq \rho(\Sigma_\infty^{x_k}) = J(\tilde{\sigma}_x)$$

■

C. Computational Complexity Analysis

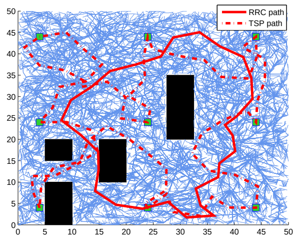
As discussed in [9], the CollisionFree procedure can be executed in $O(\log^d M)$ time, M is the number of obstacles in the environment and d is the dimension of the environment. As for the procedure Nearest, it has time complexity $O(\log|V|)$, where $|V|$ is the number of vertices; and $O(\log N)$ time is spent on the procedure Near, where N is the number of iterations. In the procedure FindCycle, we use depth-first search (DFS) to find the cycles inside the tree. For DFS in a tree, the time complexity is $O(|V|)$ [14]. Assume we have k vertices in X_{near}, then we need to find out the $\frac{k(k-1)}{2}$ cycles. Hence the time complexity of this procedure is $O(\frac{k(k-1)}{2}|V|)$. Because k is bounded, we can write the complexity of this procedure as $O(|V|)$.

Hence, in RRC at each iteration we have $O(\log^d M)$ time spent on collision checking, $O(\log|V|)$ time spent on finding nearest vertex, $O(\log N)$ time spent on finding near vertices, $O(|V|)$ time spent on finding cycles and $O(k^2)$ time spent on finding the cycle with minimum cost. So with N iterations, the time complexity of RRC is $O(N(\log^d M + \log|V| + \log N + |V| + k^2))$. Since d and M are fixed, $|V| = N + 1$ and k is bounded, the time complexity of RRC is $O(N^2)$.

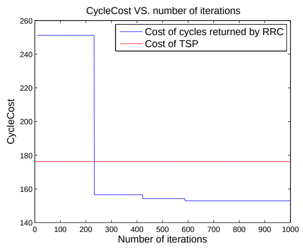
V. Numerical Simulations

This section presents numerical simulations of our algorithm. We compare the performance of our algorithm with an alternative method based on a Traveling Salesperson (TSP) tour of the points of interest.

We choose the parameters of the simulation as follows. The environment is a square region with four rectangular obstacles of varying sizes. There are 9 points of interest arranged in an even grid in the environment. We choose the dynamics of the field as $A = 0.99 I_9$, where I_9 is the 9×9 identity matrix. The reason why we choose such a dynamics for the field is that, by such an A matrix, the field is stable,



(a) Black blocks: obstacles; green squares: points of interest; blue curves: rapidly-exploring random tree; solid red closed curve: cycle generated by RRC with period equal to 29 time units; dotted red closed curve: cycle by TSP with period equal to 66 time units.



(b) This plot shows the decreasing cost of the cycle found by the RRC algorithm for the simulation in Fig. 3(a). It has smaller cost than the cycle given by TSP.

Fig. 3. Numerical results of our RRC algorithm with 10000 iterations and TSP with path given by RRT.

since all the characteristic multipliers of the system belong to the open unit disk. So the periodic systems along any of the cycles in the field is stabilizable and detectable. Hence the periodic Riccati equation has a unique symmetric periodic positive semidefinite solution. The covariance matrix of the process noise is $Q = 5I_9$. We let each entry of the measurement function be given by a Gaussian function in the distance between the agent and the point of interest, so $c_j = e^{-\|x-q_j\|^2/2\sigma_c^2}$, and $C = [c_1 \cdots c_9]$. Here the standard deviation was chosen as $\sigma_c = 6$. The covariance matrix of the measurement noise is $R = 10$.

We initialize the RRC algorithm with $x_0 = (0,0)$. For the primitive procedures, we choose $\eta = 5$ for the Steer function. The r in the function Near is chosen as $r = \min\{\gamma(\log|V|/|V|)^{1/2}, \eta\}$, where $\gamma = (6\mu(X_{free})/\zeta_2)^{1/2} + 1$, $\mu(X_{free})$ is the volume of the free space, ζ_2 is the volume of the unit ball in $\mathbb{R}^2$, and $|V|$ denotes the number of vertices in V.

As there are no current methods besides RRC to find periodic sensing trajectories for infinite horizon sensing, we use a naive TSP procedure as a point of comparison. The TSP trajectory is found by planning a TSP tour through the points of interest, and then connecting an executable path between points of interest using the standard RRT algorithm, with $\eta = 5$. An example trajectory is shown in Figure 3(a).

From Figure 3(a), we can see that the cycle returned by RRC is quite different from the cycle given by TSP. The costs of the two methods are shown as a function of the number of RRC iterations in Figure 3(b). From this figure, the cost of the cycle returned by the RRC algorithm decreases with the number of iterations, which verifies the monotonicity property of the algorithm. In the beginning, the cycle found by RRC may have larger cost than the cycle given by the TSP method, but as the tree expands RRC can find a cycle which has smaller cost than the cycle given by TSP. The lowest cost cycle in the simulation was found at about iteration 600, as can be see in Fig. 3(b). A hardware implementation with an m3pi robot executing the RRC trajectory in a motion capture environment can be found at `http://people.bu.edu/schwager/Movies/LanICRA13ExperimentMovie.mp4`.

VI. Conclusions and Future Work

In this paper we proposed an incremental sampling-based algorithm to plan a periodic trajectory for a robot that tries to minimize the largest eigenvalue of the error covariance matrix over an infinite horizon. The algorithm maintains the minimum cost simple cycle in an expanding graph. We use recent results for the periodic Riccati recursion to efficiently compute the infinite horizon cost of the cycles. We also leverage the monotonicity property of the Riccati recursion to efficiently compare the cost of cycles. Numerical simulations and hardware experiments demonstrated the effectiveness of this algorithm. In the future we plan to investigate online versions of this algorithm in which the environment dynamics are not known *a priori*. We will also extend the algorithm to multiple robots monitoring an environment simultaneously.

References

[1] S. L. Smith, M. Schwager, and D. Rus, "Persistent robotic tasks: Monitoring and sweeping in changing environments," *IEEE Transactions on Robotics*, vol. 28, pp. 410–426, April 2012.
[2] R. N. Smith, M. Schwager, S. L. Smith, B. H. Jones, D. Rus, and G. S. Sukhatme, "Persistent ocean monitoring with underwater gliders: Adapting sampling resolution," *Journal of Field Robotics*, vol. 28, pp. 714–741, September-October 2011.
[3] C. G. Cassandras, X. Ding, and X. Lin, "An optimal control approach for the persistent monitoring problem," in *Proc. of 50th IEEE Conf. Decision and Control*, (Orlando, USA), 2011.
[4] N. E. Leonard, D. A. Paley, F. Lekien, R. Sepulchre, D. M. Fratantoni, and R. E. Davis, "Collective motion, sensor networks, and ocean sampling," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 48–74, Jan. 2007.
[5] K. M. Lynch, I. B. Schwartz, P. Yang, and R. A. Freeman, "Decentralized environmental modeling by mobile sensor networks," *IEEE Transactions on Robotics*, vol. 24, no. 3, pp. 710–724, June, 2008.
[6] J. L. Ny and G. J. Pappas, "On trajectory optimization for active sensing in gaussian process models," in *Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*, (Shanghai, China), December 16-18, 2009.
[7] W. Zhang, M. P. Vitus, J. Hu, A. Abate, and C. J. Tomlin, "On the optimal solutions of the infinite-horizon linear sensor scheduling problem," in *2010 49th IEEE Conference on Decision and Control (CDC)*, (Atlanta, GA, USA), pp. 396 – 401, 15-17 Dec. 2010.
[8] M. Vitus, W. Zhang, A. Abate, J. Hu, and C. Tomlin, "On sensor scheduling of linear dynamical systems with error bounds," in *Decision and Control (CDC), 2010 49th IEEE Conference on*, pp. 1318–1323, IEEE, 2010.
[9] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, pp. 846–894, June 2011.
[10] E. K. W. Chu, H. Y. Fan, W. W. Lin, and C. S. Wangs, "Structure-preserving algorithms for periodic discrete-time algebraic riccati equations.," *International Journal of Control*, vol. 77, no. 8, pp. 767 – 788, 2004.
[11] N. Cressie, *Statistics for Spatial Data*. New York: Wiley, 1993.
[12] R. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.
[13] S. Bittanti, P. Colaneri, and G. De Nicolao, "The difference periodic ricati equation for the periodic prediction problem," *Automatic Control, IEEE Transactions on*, vol. 33, pp. 706 –712, Aug 1988.
[14] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. The MIT Press, 2001.