Cooperative Patrolling via Weighted Tours: Performance Analysis and Distributed Algorithms

Fabio Pasqualetti, Joseph W. Durham, and Francesco Bullo

***Abstract*—This work focuses on the problem of patrolling an environment with a team of autonomous agents. Given a set of strategically important locations (*viewpoints*) with different *priorities*, our patrolling strategy consists of (i) constructing a tour through the viewpoints, and (ii) driving the robots along the tour in a coordinated way. As performance criteria, we consider the *weighted refresh time*, i.e., the longest time interval between any two visits of a viewpoint, weighted by the viewpoint's priority. We consider the design of both optimal trajectories and distributed control laws for the robots to converge to optimal trajectories. First, we propose a patrolling strategy and we characterize its performance as a function of the environment and the viewpoints priorities. Second, we restrict our attention to the problem of patrolling a non-intersecting tour, and we describe a team trajectory with minimum weighted refresh time. Third, for the tour patrolling problem and for two distinct communication scenarios, namely the *Passing* and the *Neighbor-Broadcast* communication models, we develop distributed algorithms to steer the robots towards a minimum weighted refresh time team trajectory. Finally, we show the effectiveness and robustness of our control algorithms via simulations and experiments.**

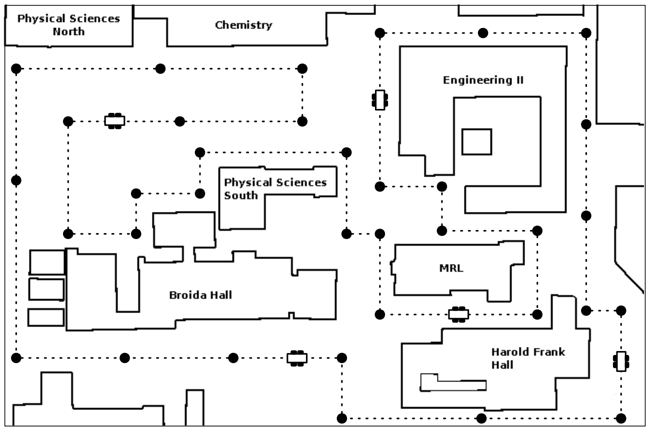


Fig. 1. This figure represents a part of the UCSB campus. For the surveillance of the buildings in the map by a team of autonomous robots, a set of 35 important locations (viewpoints) has been identified, and a tour through the viewpoints has been computed. The robots repeatedly patrol the tour to guarantee complete and persistent surveillance of the buildings. We propose the Equal-Time-Spacing trajectory, which minimizes the longest priority-weighted time gap between any two visits of the same viewpoint.

I. Introduction

Coordinated teams of autonomous agents can effectively complete tasks requiring repetitive execution, such as monitoring oil spills [1], detecting forest fires [2], tracking border changes [3], and surveilling an environment [4]. Surveillance of an environment requires that the robots persistently travel around the area, and the challenge consists in scheduling the robots trajectories to optimize a certain performance criteria.

Related work. The problem of patrolling an environment with a team of autonomous robots has recently received attention from scientists interested in mobile robotics. In patrolling problems, it is a common approach to associate a non-negative uncertainty value representing some measure of interest with each point in the environment [5], [6]. This uncertainty value grows with time, unless the physical location is covered by the sensor footprint of a robot. Additionally, a discrete representation of the environment is typically obtained by selecting a set of important *viewpoints*, and by creating a robotic roadmap with the viewpoints as vertices. The robots are constrained to move along this roadmap, and the performance of the team is measured according to a frequency of visit criteria which reflects the amount of uncertainty in the environment. In [7], two classes of patrolling strategies, based respectively on space decomposition and traveling salesperson tour computation, are presented and qualitatively compared. In [4] and [8], an efficient and distributed solution to the perimeter patrolling problem is proposed. In [9], the computational complexity of the patrolling problem is studied in relation to the roadmap representing the environment. Additionally, exact patrolling algorithms, as well as constant factor approximations, are proposed and analyzed. The possibility of storing information at important locations in the environment is exploited in [10] to design minimalist patrolling algorithms. Notice that in these last works the focus is mainly on finding optimal trajectories and in scheduling the motion of the robots. A different setup is considered in [11], [12], where the robots are constrained to move along pre-specified tours, and the goal is to control the robots velocities so as to optimize the patrolling performance.

Our setup and approach We assume that the uncertainty growth of the viewpoints is linear, with a possibly different rate (*priority*) for each viewpoint, and that the uncertainty value becomes zero as soon as the viewpoint is covered by the sensor footprint of a robot. For a given set of connected viewpoints with priorities, we first construct a minimum spanning tree through the viewpoints. Then, by means of standard graph theory techniques, we compute a tour through the viewpoints. Finally, we constrain the robots on the tour and compute an optimal team trajectory. For optimality criteria, we consider the *weighted refresh time*, which, loosely speaking, is the longest time interval between any two visits of a viewpoint, weighted by the corresponding viewpoint's priority. The idea of moving the robots along a tour of the viewpoints has been previously considered in [7]. Here we extend the results in [7], e.g., by considering viewpoints with priorities, and by providing distributed control algorithms. This works differs also from [11], [12] in the following ways. First, we focus on the design of optimal team trajectories, as opposed to optimal single-vehicle trajectories or suboptimal team trajectories. Second, we consider a discrete set of viewpoints. Third, we consider real valued priorities, instead of discretized ones. Finally, with respect to [6], we consider multiple robots on a closed path, as opposed to a single robot on an open path, and we consider different cost functions and uncertainty models.

Contributions The main contributions of this work are as follows. First, we introduce and formalize the concept of weighted refresh time for a team trajectory (Section II). We remark that designing team trajectories with minimum weighted refresh time is an *NP-hard* optimization problem. We propose a computationally efficient patrolling strategy, and we characterize its weighted refresh time performance. Second, we state the patrolling optimization problem of a weighted tour (Section III). For this problem, we show that a minimum weighted refresh time team trajectory can be computed by solving a convex,[1] in fact linear, optimization problem. We characterize a solution to this optimization problem, and we propose a minimum refresh time team trajectory. Third, we develop and characterize two distributed control algorithms to steer the robots

This material is based upon work supported in part by NSF grant IIS-0904501, NSF grant CPS-1035917 and ARO grant W911NF-11-1-0092.

Fabio Pasqualetti, Joseph W. Durham, and Francesco Bullo are with the Center for Control, Dynamical Systems and Computation, University of California at Santa Barbara, {fabiopas,joey,bullo}@engineering.ucsb.edu

[1]Several solutions to this optimization problem may exist.

towards a minimum weighted refresh time team trajectory (Section IV). For the first algorithm, we assume that two robots communicate only when they occupy the same location (*Passing* communication model), and we allow for the presence of a leader or base station to control the robots. For the second control algorithm, instead, we assume that, at some times, each robot communicates with its clockwise and counterclockwise neighbors (*Neighbor-Broadcast* communication model). We characterize the convergence properties of both algorithms. Finally, we validate our findings through a simulation study and an experiment (Section V): we use the Player/Stage simulation software to show the effectiveness and the robustness of our second patrolling procedure in a campus environment, and we conduct an experiment with real hardware in an indoor environment with obstacles. The experiments confirm the robustness of the proposed strategies against noise and unmodeled dynamics.

II. Problem setup and preliminary results

A. Robotic model and preliminary concepts

We are given a team of $m > 0$ identical robots, capable of sensing, communicating, and moving in a path-connected environment $\mathcal{E} \subseteq \mathbb{R}^2$. Our communication model for distributed control will be discussed in Section IV. Instead, we now discuss our combined sensing and motion model.

Regarding sensing, we assume that the environment can be completely covered by simultaneously placing a robot at each of $n > m$ *viewpoints* in the configuration space. In other words, if $m = n$ robots were available and placed at the n viewpoints, then the union of the sensor footprints of the robots would provide complete sensor coverage of the environment. We assume that each viewpoint is required for complete sensor coverage. Since we assume that $n > m$, at least one robot must visit multiple viewpoints for the entire environment to be monitored over time. Additionally, we associate the positive priority $\phi_\alpha \in \mathbb{R}_{>0}$ with the α-th viewpoint, and we let $\Phi = \{\phi_1, \dots, \phi_n\}$ denote the priority set. Let $\phi_{\min} = \min\{\phi : \phi \in \Phi\}$ and $\phi_{\max} = \max\{\phi : \phi \in \Phi\}$.

Regarding motion, we assume that the robots are holonomic, i.e., modeled as first order integrators, and move at most at unit speed. We constrain the motion of the robots to the robotic roadmap $G = (V, E)$, where $V = \{v_1, \dots, v_n\}$ denotes the set of viewpoints, and where the undirected edge $(v_\alpha, v_\beta) \in E$ denotes the possibility for a robot to travel between v_α and v_β. We associate a unique path connecting any two neighbors in G, and we adopt the path length as edge weight.[2]

A *team trajectory* X is an array of m continuous and piecewise-differentiable trajectories $x_1(t), \dots, x_m(t)$ defined by the motion of the robots on the roadmap G, i.e., $x_i : [0, \infty) \mapsto G$, for $i \in \{1, \dots, m\}$. We say that the viewpoint v_α is visited at time t by robot i if $x_i(t) = v_\alpha$. Let $\mathcal{A}(\alpha, i)$ and $\mathcal{D}(\alpha, i)$ be, respectively, the set of times at which robot r_i arrives at and departs from the viewpoint v_α. Specifically, for a sufficiently small $\varepsilon \in \mathbb{R}_{>0}$, we define

$$\mathcal{A}(\alpha, i) = \{t \in [0, \infty) \ : \ x_i(t) = v_\alpha \text{ and } x_i(t - \varepsilon) \neq v_\alpha\},$$
$$\mathcal{D}(\alpha, i) = \{t \in [0, \infty) \ : \ x_i(t) = v_\alpha \text{ and } x_i(t + \varepsilon) \neq v_\alpha\}.$$

The *(weighted) refresh time* of a team trajectory X, which we denote by $\mathrm{RT}(X)$, is the longest weighted time interval between any two consecutive visits of any viewpoint, i.e.,

$$\mathrm{RT}(X) = \max\big\{\phi_\alpha\big(t_a(\alpha, t_d) - t_d\big) \ : \ \alpha \in \{1, \dots, n\}, \ t_d \in \mathcal{D}(\alpha, i) \text{ for any } r_i\big\}, \quad (1)$$

where $t_a(\alpha, t_d)$ is the earliest arrival time by any robot at node α after departure at time t_d:

$$t_a(\alpha, t_d) = \min\{t \in \cup_{i=1}^m \mathcal{A}(\alpha, i) \ : \ t \geq t_d\}.$$

It should be observed that the expression in (1) is undefined if no viewpoint is visited by the robots. This situation is not of interest, since we aim at designing trajectories that *persistently* visit the viewpoints, that is team trajectories in which every viewpoint is persistently visited in the interval $[0, \infty)$.

Problem 1: **(Cooperative Patrolling)** Given a set of viewpoints with priorities, and a team of $m \geq 2$ robots, design a persistent team trajectory with minimum refresh time.

To conclude this section we remark that the cooperative patrolling problem is, in general, *computationally hard*, even if all the priorities have the same value [9]. Hence, we will not propose an optimal solution to the Cooperative Patrolling problem (Problem 1), and we will describe instead an efficient and distributed patrolling strategy with performance guarantees.

B. Cyclic patrolling strategy and optimality bound

Our patrolling strategy consists of three steps. First, we construct a *minimum spanning tree* of the roadmap G [13]. The sensor-based construction of a minimum spanning tree can be achieved via distributed computation, e.g., see [13], [14]. Second, we construct a non-intersecting tour visiting the viewpoints by doubling the edges of the computed minimum spanning tree. Third, we let the robots continuously travel the tour in a coordinated way. In particular, for a tour of length L, a team of m robots, and a set of initial positions, define the *Equal-Spacing* trajectory to be such that (i) the robots continuously travel the tour at maximum speed in the same direction, and (ii) the distance between any two consecutive robots is L/m. Notice that the refresh time of the Equal-Spacing trajectory equals $\phi_{\max} L/m$. Following [7], we refer to the above three steps patrolling strategy as *weighted-cyclic strategy*. Observe that, for a given weighted tour, the Equal-Spacing trajectory may not be a minimum refresh time team trajectory. We will discuss this problem in Section III. The next theorem generalizes a result of [7] from unweighted cyclic strategies to weighted cyclic strategies.

Theorem 2.1: **(Optimality of weighted-cyclic strategy)** Let G be a robotic roadmap on the viewpoints V with priorities Φ. Let $X(t)$ be the team trajectory generated by the weighted-cyclic strategy on G. Then,

$$\frac{\mathrm{RT}(X)}{\mathrm{RT}^*} \leq 2(1 + \gamma)\frac{\phi_{\max}}{\phi_{\min}},$$

where RT^* denotes the minimum refresh time on G, and γ denotes the ratio of the longest to the shortest edge of G.

Proof: We use a similar line of reasoning as in [7, Thoerem 2]. Let $\mathrm{RT}^*_{\phi_{\min}}$ be the minimum refresh time when all the priorities equal $\phi_{\min}$. Notice that $\mathrm{RT}^*_{\phi_{\min}} = \phi_{\min}\mathrm{RT}^*_1$, where RT^*_1 denotes the minimum refresh time when all the priorities are unitary. Observe that $\mathrm{RT}^* \geq \phi_{\min}\mathrm{RT}^*_1$. On the other hand, it can be shown that there exists a path cover of cardinality m for G of length at most RT^*_1 [9]. Hence, the length of a tour obtained by doubling the edges of a minimum spanning tree is $2m\mathrm{RT}^*_1 + 2(m-1)\gamma \underline{w}$, where $\underline{w}$ denotes the length

[2]We select these paths so that the set of path lengths, adopted as edge weights, verify the triangle inequality. For example, the shortest paths between viewpoints constitute a suitable choice.

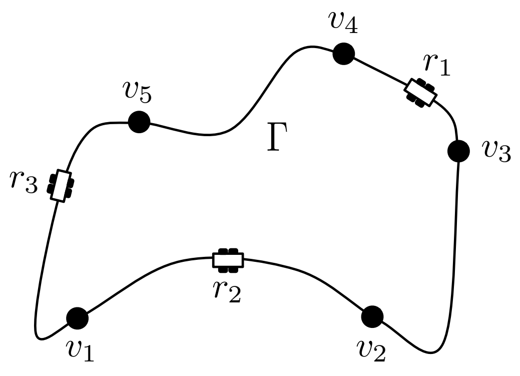


Fig. 2. The robots $\{r_1, r_2, r_3\}$ patrol the non-intersecting tour Γ with viewpoints $\{v_1, \dots, v_5\}$. Robots move in counter-clockwise direction.

of the shortest edge. Consequently, $\mathrm{RT}(X) \leq 2\phi_{\max}(\mathrm{RT}_1^* + \gamma \underline{w})$. By taking the ratio of the two bounds, we obtain

$$\frac{\mathrm{RT}(X)}{\mathrm{RT}^*} \leq \frac{2\phi_{\max}(\mathrm{RT}_1^* + \gamma \underline{w})}{\phi_{\min}\mathrm{RT}_1^*} \leq 2(1+\gamma)\frac{\phi_{\max}}{\phi_{\min}},$$

where the last inequality holds because $\mathrm{RT}_1^* \geq \underline{w}$. Indeed, since we assume $m < n$, at least one robot needs to travel one edge for all the viewpoints to be visited. ■

Observe that, if γ and $\phi_{\max}/\phi_{\min}$ are bounded by constants, then the weighted-cyclic strategy is a constant factor approximation to the Cooperative Patrolling problem (Problem 1). Moreover, the above patrolling strategy improves upon the chain heuristic presented in [9, Theorem VII.5]. In what follows we will not discuss the computation of a minimum spanning tree, since solutions to this problem already exist [13], [14]. Instead, we will focus on the problem of patrolling a tour with minimum weighted refresh time.

Remark 1: **(Tour with repeated viewpoints)** Given a spanning tree, a non-intersecting tour through the viewpoints is created as follows: (i) compute a tour by doubling the edges, and (ii) add virtual viewpoints with zero priority for each viewpoint appearing multiple times along the tour. Notice that some viewpoints may be visited several times along the tour. In this case neither the Equal-Spacing trajectory nor the Equal-Time-Spacing trajectory in Section III may achieve minimum weighted refresh time. However, the refresh time we characterize for tours without repeated viewpoints is an upper bound for the actual refresh time with repeated viewpoints, and the bound in Theorem 2.1 is valid for both cases. In what follows we consider only tours without repeated viewpoints.

C. Patrolling of a weighted tour

Motivated by the discussion in Section II-B, in the remainder of the paper we focus on the following patrolling problem.

Problem 2: **(Cooperative Tour Patrolling)** Given a set of viewpoints with priorities on a non-intersecting tour Γ, and a team of $m \geq 2$ robots moving on Γ in uniform direction, design a persistent team trajectory with minimum refresh time.

Remark 2: **(Single robot tour patrolling)** If $m = 1$, then a solution to the Cooperative Tour Patrolling problem (Problem 2) consists of letting the robot move at maximum speed along the tour. The refresh time of such a trajectory is $\phi_{\max}L$.

Without affecting generality, we label the robots *clockwise* according to their initial position, and we let the robots move *counterclockwise* on the tour Γ. A team trajectory is said to be *order-invariant* if the ordering of the robots is constant over time. A *Stop-Go* team trajectory is an order-invariant team trajectory in which each robot moves at maximum speed, except, possibly, when its position coincides with a viewpoint.

Definition 1: **(Stop-Go team trajectory)** A team trajectory is said to be *Stop-Go* if it is order-invariant and, for each robot r_i, it holds $\dot{x}_i(t) \in \{0, 1\}$ with $x_i(t) \in V$ whenever $\dot{x}_i(t) = 0$.

Lemma 2.2: **(Optimality of Stop-Go team trajectories)** For any team trajectory X on a non-intersecting tour, there exists a Stop-Go team trajectory $\tilde{X}$ satisfying $\mathrm{RT}(\tilde{X}) \leq \mathrm{RT}(X)$.

Proof: Define the permutation matrix $P(t)$ so that its (i,j)-th entry is 1 if, at time t, the i-th robot occupies the j-th position in the team of robots, and it is 0 otherwise. The discontinuities of $P(t)$ coincide with the instants at which the ordering of the robots changes. Let $\bar{X}(t) = P^{-1}(t)X(t)$, and observe that $\mathrm{RT}(\bar{X}(t)) = \mathrm{RT}(X(t))$. Suppose that the speed of robot r_i is not unitary along the edge e of length ℓ, and let $\bar{\ell}$ be the travel time of r_i along e. Notice that (1) does not increase if the robot moves at unitary speed along e and increases the waiting time at the endpoint of e by $\bar{\ell} - \ell$. ■

In the next section we design Stop-Go team trajectories for the Cooperative Tour Patrolling problem (Problem 2).

III. Team trajectories with minimum refresh time

A Stop-Go team trajectory with minimum weighted refresh time is presented in this section. Notice that a Stop-Go team trajectory can be entirely described by specifying the initial positions of the robots and a sequence of *waiting intervals* for each robot at each viewpoint, i.e., the time intervals during which a robot stops at a viewpoint. Let $x_i(0)$ denote the initial position on Γ of the i-th robot, and let $\delta_\alpha^i(k)$ be the duration of the k-th waiting interval, with $k \in \mathbb{N}$, of robot r_i at the viewpoint v_α. For notational convenience, we partition the viewpoints according to their priorities. In particular, let $V = V_{\max} \cup V_{\min}$, where $V_{\max}$ contain the m viewpoints with highest priority, $V_{\max} \cap V_{\min} = \emptyset$, and, accordingly, $\Phi = \Phi_{\max} \cup \Phi_{\min}$. Assume that the viewpoints are labeled such that $\phi_1 = \max\{\phi : \phi \in \Phi_{\min}\}$. Notice that, since we focus on Stop-Go team trajectories, each viewpoint is visited by robot r_{i+1} after being visited by robot r_i (where r_{m+1} is r_1). Also, after visiting a viewpoint, each robot moves to the closest viewpoint in counterclockwise direction at unit speed.

As a solution to the Cooperative Tour Patrolling (Problem 2), we propose the *Equal-Time-Spacing* trajectory formally described in Trajectory 1. We now give an informal description.

(Informal description) The Equal-Time-Spacing trajectory is a Stop-Go team trajectory, where the waiting intervals at each viewpoint are equal among the robots and constant in time. In other words, we have $\delta_\alpha^i(k) = \delta_\alpha^j(k+1)$ for every iteration $k \in \mathbb{N}$, for every pair of robots r_i and r_j, and for every viewpoint v_α. Moreover, the viewpoints with nonzero waiting intervals are contained in $V_{\max}$, and, being RT_T^* the minimum refresh time on the tour for the given configuration, the initial position of robot r_i is chosen such that $x_i(t) = x_{i+1}(t + \frac{\mathrm{RT}_\mathrm{T}^*}{\phi_1})$. Notice that in the Equal-Time-Spacing trajectory the robots are equally spaced in time along a common trajectory.

Theorem 3.1: **(Optimality of Equal-Time-Spacing trajectories)** Given a set of viewpoints with priorities on a non-intersecting tour of length L, and a team of $m \geq 2$ robots, let $\Phi_{\max}$ denote the set of m largest priorities. Then,

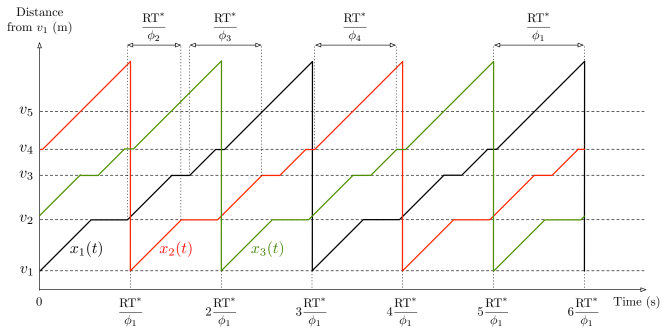
(i) the Equal-Time-Spacing trajectory $X(t)$ in Trajectory 1 has minimum refresh time, and

(ii) $\mathrm{RT}(X) = \mathrm{RT}_\mathrm{T}^* = \dfrac{L}{\sum_{\phi \in \Phi_{\max}} \phi^{-1}}$.

In other words, Theorem 3.1 states that the Equal-Time-Spacing trajectory is an optimal solution to the Cooperative Tour Patrolling problem (Problem 2). Consider the weighted-cyclic patrolling strategy

Trajectory 1: *Equal-Time-Spacing trajectory*

```
Input    : Viewpoints $V$, Priorities $\Phi_{\text{max}}$ and $\phi_1$, Tour length $L$;
Require  : $\phi_1 = \max\{\phi : \phi \in \Phi_{\text{min}}\}$;
Output   : A Stop-Go team trajectory as specified by $\delta_\alpha^i(k)$, $x_i(0)$;
1 $\text{RT}_\text{T}^* := \frac{L}{\sum_{\phi\in\Phi_{\text{max}}} \phi^{-1}}$;
  for each robot $r_i$ and iteration $k$ do
      for $v_\alpha \in V_{\text{min}}$ do
2         $\delta_\alpha^i(k) := 0$;
      for $v_\alpha \in V_{\text{max}}$ do
3         $\delta_\alpha^i(k) := \frac{(\phi_\alpha - \phi_1)}{\phi_\alpha \phi_1} \text{RT}_\text{T}^*$;
4 $x_1(0) := v_1$;
5 for each robot $r_i$ do
6     $x_i(0) := x_1((m+1-i)\,\text{RT}_\text{T}^*/\phi_1)$;
```



Distance from v_1 (m)
$\frac{\text{RT}^*}{\phi_2}$
$\frac{\text{RT}^*}{\phi_3}$
$\frac{\text{RT}^*}{\phi_4}$
$\frac{\text{RT}^*}{\phi_1}$
v_5
v_4
v_3
v_2
v_1
$x_1(t)$
$x_2(t)$
$x_3(t)$
0
$\frac{\text{RT}^*}{\phi_1}$
$2\frac{\text{RT}^*}{\phi_1}$
$3\frac{\text{RT}^*}{\phi_1}$
$4\frac{\text{RT}^*}{\phi_1}$
$5\frac{\text{RT}^*}{\phi_1}$
$6\frac{\text{RT}^*}{\phi_1}$
Time (s)

Fig. 3. This figure shows the Equal-Time-Spacing trajectory for a set of 5 viewpoints, with $\phi_5 \le \phi_1 < \phi_4 < \phi_3 < \phi_2$, and a team of 3 robots. Notice that (i) the robots trajectories have the same shape, (ii) the team trajectory is a periodic Stop-Go trajectory, and (iii) the robots trajectories are equally spaced within the period of the team trajectory.

defined in Section II-B, and assume that the robots travel along the Equal-Time-Spacing trajectory instead of along the Equal-Spacing trajectory. Denote this new strategy as *optimal weighted-cyclic* strategy. Then, the performance bound in Theorem 2.1 is also valid for the *optimal weighted-cyclic* strategy, which, in fact, generates an approximate solution to the Cooperative Patrolling problem (Problem 1). A proof of Theorem 3.1 is the Appendix.

Example 1: **(Equal-Time-Spacing trajectory)** Consider a tour of 5 viewpoints with priorities $\{\phi_1, \dots, \phi_5\}$, and a team of 3 robots. Assume that $\phi_5 \le \phi_1 < \phi_4 < \phi_3 < \phi_2$. The Equal-Time-Spacing trajectory for this configuration is in Fig. 3. Notice that the robots do not stop at v_1 and v_5, while they stop for a certain time interval at v_2, v_3, and v_4.

Remark 3: **(Equal-Time-Spacing and Equal-Spacing trajectories)** From the discussion leading to Theorem 2.1, recall that the refresh time of the Equal-Spacing trajectory equals $\phi_{\text{max}} L/m$, and notice that it grows linearly with ϕ_{max}. On the other hand, recall from Theorem 3.1 that the refresh time of the Equal-Time-Spacing trajectory is $L/\sum_{\phi\in\Phi_{\text{max}}} \phi^{-1}$, and notice that it remains bounded as long as there exists a finite priority in the set Φ_{max}. We conclude that Equal-Time-Spacing trajectories performs, in general, much better than Equal-Spacing trajectories. Consequently, the bound in Theorem 2.1 is conservative for the optimal weighted-cyclic strategy (cf. Fig. 4). Finally, Equal-Spacing and Equal-Time-Spacing trajectories coincide when $\phi_{\text{max}} = \min\{\phi : \phi \in \Phi_{\text{max}}\}$.

IV. Distributed Control Algorithms

In this section we describe algorithms to equally space robots in time along a shared trajectory using minimal communication.

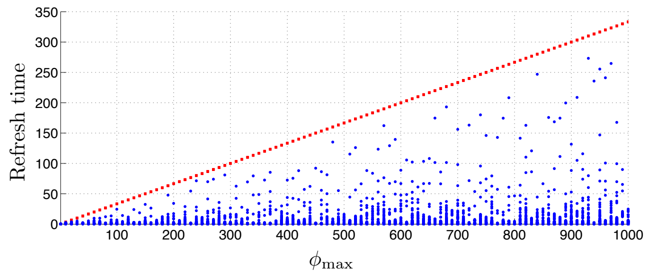


Fig. 4. Consider the configuration in Fig. 2 with $L = 1$, $\phi_1 = 1$, $\phi_5 < 1$, and $\phi_2 = \phi_{\text{max}}$. For each value of $\phi_{\text{max}} \in [1, 10, 20, \dots, 1000]$, we generate 100 priority sets by letting ϕ_3 and ϕ_4 be uniformly distributed within the interval $[1, \phi_{\text{max}}]$. We plot the refresh time of the Equal-Time-Spacing trajectory and the Equal-Spacing trajectory ($\phi_{\text{max}}/3$). The Equal-Time-Spacing trajectory (blue dots) performs generally better than the Equal-Spacing trajectory (red squares). Consequently, the bound in Theorem 2.1 is also valid for the optimal weighted-cyclic strategy, and it is conservative for most priority sets.

Algorithm 2: *Leader-Based control law (leader robot)*

```
Input    : Viewpoint $v_{\text{max}}$, Priorities $\Phi_{\text{max}}$ and $\phi_1$, Tour length $L$;
Require  : $\phi_1 = \max\{\phi : \phi \in \Phi_{\text{min}}\}$, $t :=$ current time;
1 $\text{T}^* := \frac{L}{\phi_1 \sum_{\phi\in\Phi_{\text{max}}\setminus\{\phi_{\text{max}}\}} \phi^{-1}}$;
  while true do
2     $T_{\text{last}} :=$ time of last departure of a robot from viewpoint $v_{\text{max}}$;
      if robot arrives at viewpoint $v_{\text{max}}$ then
3         stop robot at viewpoint $v_{\text{max}}$ for $\max\{0, T^* - (t - T_{\text{last}})\}$;
```

We consider two different communication models: (i) the *Passing* communication model, which assumes that two robots communicate only when they occupy the same position,[3] and (ii) the *Neighbor-Broadcast* communication model, where sporadically robots $i+1$ and $i-1$ synchronously send their position on the shared trajectory to robot i. Notice that, due to these communication constraints, a control algorithm is needed to synchronize the robots along a desired trajectory.

A. Passing communication model

For the Passing communication model, our approach to generating the communication necessary for coordination is to use one robot as a stationary leader at the vertex v_{max} with highest priority ϕ_{max}. The first robot to reach v_{max} will become the leader. When the other robots pass v_{max}, the leader will communicate with them and take control actions to generate the desired spacing. By keeping track of the arrival times at v_{max}, the leader adjusts the waiting intervals of other robots to enforce the desired spacing. We say that a robot follows the Equal-Time-Spacing trajectory if it moves at maximum speed and stops at the viewpoints as specified in Trajectory 1. Our *Leader-Based* control law is in Algorithm 2 and Algorithm 3.

Lemma 4.1: **(Leader-Based control performance)** Given a set of viewpoints with priorities on a non-intersecting tour of length L, and a team of $m \ge 2$ robots with Passing communication model, let Φ_{max} denote the set of m largest priorities. Let $X(t)$ be the team trajectory generated by the Leader-Based control law. Then,

$$\text{RT}(X(t \ge t_0)) = \text{RT}_\ell^* = \frac{L}{\sum_{\phi\in\Phi_{\text{max}}\setminus\{\phi_{\text{max}}\}} \phi^{-1}},$$

[3]Although conservative, this assumption allows us to derive control and communication laws implementable on robots with minimal capabilities. Any additional communication could only improve the refresh time performance.

Algorithm 3: *Leader-Based control law (i-th robot)*

```
Input   : Viewpoint $v_{\max}$, Priorities $\Phi_{\max}$ and $\phi_1$, Tour length $L$;
Require : $\phi_1 = \max\{\phi : \phi \in \Phi_{\min}\}$, $t :=$ current time;

1 compute the Equal-Time-Spacing trajectory with $|\Phi_{\max}| - 1$ robots and
  priority set $\Phi_{\max} \setminus \{\phi_{\max}\}$;
2 follow the Equal-Time-Spacing trajectory unless differently instructed
  by the leader when passing viewpoint $v_{\max}$;
```

Algorithm 4: *Stop-When-Ahead control law (i-th robot)*

```
Input   : Viewpoints $V$; Priorities $\Phi_{\max}$ and $\phi_1$; Tour length $L$;
          Gain $q \in (0, 1/2)$; Period $T = L + \sum_{\alpha=1}^{n} \delta_\alpha^1(1)$;
Require : $\phi_1 = \max\{\phi : \phi \in \Phi_{\min}\}$; $t :=$ current time;

while true do
1   set $\tau_i(t) = \max\{\tau : \tau \le t, x_i(\tau) = v_1\}$;
    if communication occurs then
2       $\tau_i^c := \mathrm{mod}(\tau_{i+1}(t) - \tau_i(t), T) - \mathrm{mod}(\tau_i(t) - \tau_{i-1}(t), T)$;
3       robot $i$ stops for $\max\{0, q\tau_i^c\}$;
4       set $\tau_i(t) := \tau_i(t) + \max\{0, q\tau_i^c\}$;
```

i.e., the team trajectory $X(t)$ restricted to the interval $[t_0, \infty)$ has refresh time RT_ℓ^*. Moreover,

$$t_0 \le 2L + 2\sum_{\alpha=1}^{n} \delta_\alpha^1(1),$$

where, for $i = 1, \dots, n$, $\delta_i^1(1)$ is defined in Trajectory 1.

Proof: Notice that each robot completes one lap around Γ in time $T^* = L + \sum_{i=1}^{n} \delta_i^1(1)$. Moreover, it takes at most time T^* after the first communication for the leader to generate the correct spacing. Then, within time $2T^*$, the trajectory generated by the Leader-Based control law coincides with an Equal-Time-Spacing trajectory. Observe now that, after time $2T^*$, the refresh time of the trajectory generated by the Leader-Based control law equals the refresh time of the Equal-Time-Spacing trajectory with $m - 1$ robots and high priority set $\Phi_{\max} \setminus \{\phi_{\max}\}$. The statement follows from Theorem 3.1. ■

Notice that the refresh time of a team trajectory generated by the Leader-Based control law is equivalent to the minimum refresh time for $m - 1$ robots and high priorities $\Phi_{\max} \setminus \{\phi_{\max}\}$.

B. Neighbor-Broadcast communication model

We now consider the case of sporadic Neighbor-Broadcast communications. Notice that in an Equal-Time-Spacing trajectory every robot trajectory is T-periodic, with $T = L + \sum_{\alpha=1}^{n} \delta_\alpha^1(1)$. Let $\tau_i(t) = \max\{\tau : \tau \le t, x_i(\tau) = v_1\}$. We assume that, upon communication, robot i knows the value $\tau_{i-1}(t)$ and $\tau_{i+1}(t)$. Our *Stop-When-Ahead* control law is in Algorithm 4, where we assume that each robot moves according to the Equal-Time-Spacing trajectory, except when otherwise specified by the control law, that robot i does not initiate any additional communication until its control action is terminated (i.e., the wait time has elapsed), and that for each pair of robots r_i and r_j, it holds $\tau_i(t) \neq \tau_j(t)$ at all times.[4]

Lemma 4.2: **(Stop-When-Ahead control performance):** Given a set of viewpoints with priorities on a non-intersecting tour of length L, and a team of $m \ge 2$ robots with Neighbor-Broadcast communication model, let $\Phi_{\max}$ denote the set of m largest priorities. Let each robot implement the Stop-When-Ahead control law, and let $X(t)$ be the resulting team trajectory. Assume the existence of a duration ρ such that each robot communicates within the interval $[t, t + \rho]$, for each time t. Then,

$$\lim_{t_0 \to \infty} \mathrm{RT}(X(t \ge t_0)) = \mathrm{RT}_{\mathrm{T}}^* = \frac{L}{\sum_{\phi \in \Phi_{\max}} \phi^{-1}},$$

i.e., the team trajectory $X(t)$ asymptotically converges to a minimum refresh time team trajectory.

Proof: Let $\tau(t)$ be the vector of $\tau_i(t)$, and notice that, the evolution of $\tau(t)$ is described by a consensus system $\tau(t+1) = P(t)\tau(t)$. Notice that $P(t)$ is row-stochastic, and non-degenerate [15]. Then, because of the existence of a duration ρ such that each robot communicates within the interval $[t, t+\rho]$, the vector τ converges to the subspace spanned by the vector of all ones [15, Theorem 1.63]. Then, each control variable $\tau_i^c 0$ converges to zero, and the statement follows. ■

Observe that the Stop-When-Ahead control law generates a minimum refresh time team trajectory.

[4]If two robots occupy the same location, then they first spread and then implement the Stop-When-Ahead control law.

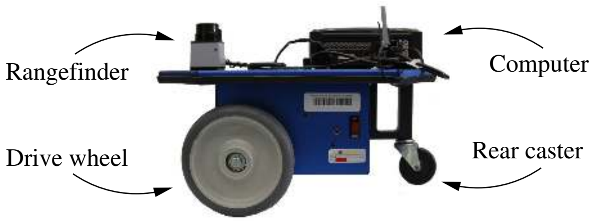


Fig. 5. Erratic mobile robot with URG-04LX laser rangefinder.

V. Simulation and experimental results

To demonstrate the effectiveness of our Stop-When-Ahead law with real robots, we implemented it in version 2.1 of the open-source Player/Stage robot software system [16]. This section describes a simulation study and a hardware experiment. Our results show that our Stop-When-Ahead law is effective with real robot motion and noisy sensor data.

A. Simulation and experiment setup

Robot hardware: We use Erratic mobile robots from Videre Design; see Fig. 5. The vehicle platform has a roughly square footprint (40cm × 37cm), with two differential drive wheels and a single rear caster. Each robot carries an on-board computer with a 1.8Ghz Core 2 Duo processor, 1 GB of memory, and 802.11g wireless communication. For navigation and localization, each robot is equipped with a Hokuyo URG- 04LX laser rangefinder. The rangefinder scans 683 points over 240° at 10Hz with a range of 5.6 meters. For simulations, the virtual robots are modeled off of our hardware.

Localization: We use the *amcl* driver in Player which implements Adaptive Monte-Carlo Localization [17]. The physical robots are provided with a map of our lab with a 15cm resolution and told their starting pose within the map. We set an initial pose standard deviation of 0.9m in position and 12° in orientation, and request updated localization based on 50 of the sensors range measurements for each change of 2cm in robot position or 2° in orientation. We use the most likely pose estimate output by amcl as the location of the robot. We let simulated robots access perfect localization information.

Navigation: Each robot uses the *snd* driver in Player for the Smooth Nearness Diagram navigation [18]. For the hardware, we set the robot radius parameter to 22cm, obstacle avoidance distance to 0.5m, and maximum speed to 0.2m/s. For our simulation, we set the maximum speed to 5m/s. We let a robot achieve its target location when it is within 10cm of the target.

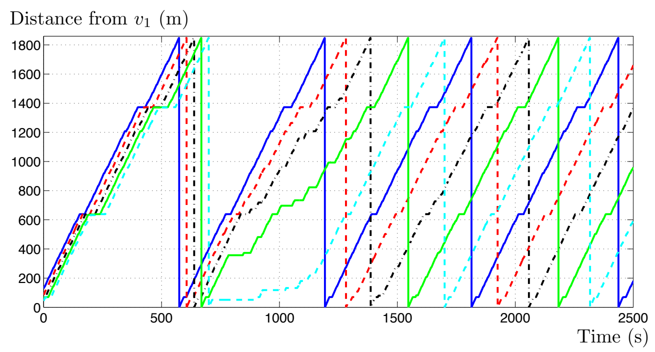


Fig. 6. For the environment in Fig. 1, this plot shows the trajectories of 5 robots as obtained from the Stop-When-Ahead control algorithm. The control algorithm becomes active after the robots have completed one lap, and it steers the robots towards an Equal-Time-Spacing trajectory. Notice that, as specified in Trajectory 1, the waiting intervals at some viewpoints are nonzero.

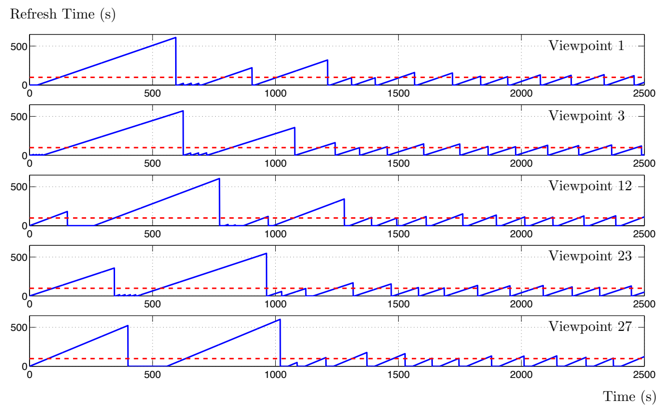


Fig. 7. For the team trajectory in Fig. 6, this figure shows the time intervals, weighted by the corresponding priority, during which the viewpoints in $\Phi_{\max}$ are not visited. The largest of these values corresponds to the refresh time of the team trajectory. The dashed line identifies the minimum refresh time predicted by Theorem 3.1. The Stop-When-Ahead control algorithm steers the robots towards an *almost-minimum* refresh time team trajectory. The difference from the predicted refresh time is due to nonholonomic robot motion.

B. A simulation of tour patrolling

In this section we simulate the Stop-When-Ahead control algorithm in a campus environment. The purpose of this simulation is twofold: (i) to illustrate the effectiveness of the control law in coordinating the robots towards the desired trajectory, and (ii) to show that the control law is robust to modeling uncertainties and the dimension of the problem.

The environment used for this simulation is a part of the UCSB campus and it is depicted in Fig. 1. For this environment, we create a tour through 35 viewpoints, and we consider a team of 5 robots. The largest 5 priorities are $\{1, 1.04, 1.09, 1.19, 1.31\}$, ϕ_1 equals 1, and the tour length is 1848 m. The robot trajectories obtained through the Stop-When-Ahead control law are reported in Fig. 6. Fig. 7 shows the refresh time of the viewpoints with highest priority. Observe that the refresh time achieved by the control algorithm is slightly larger than the value predicted in Theorem 3.1. This discrepancy is due to the fact that, unlike the ideal model in Section II, the simulated robots are non-holonomic and subject to motion constraints. The Stop-When-Ahead control law achieves satisfactory performance also in this scenario.

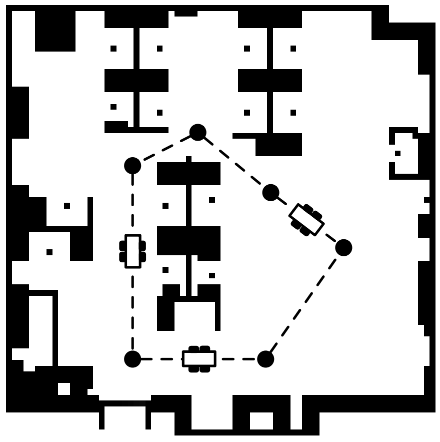


Fig. 8. An indoor environment with 3 robots and a tour with 5 viewpoints.

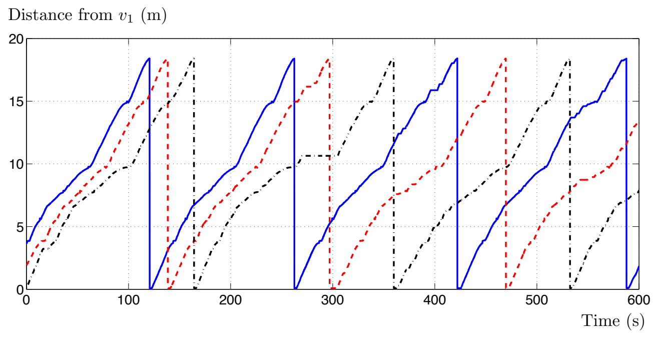


Fig. 9. For the environment in Fig. 8, this figure depicts the trajectories of 3 robots as obtained from the Stop-When-Ahead control algorithm. The control algorithm starts at time 250 and steers the robots towards an Equal-Time-Spacing trajectory. Because of motion and localization uncertainty and of the presence of obstacles, the robots do not always move at maximum speed.

C. An experiment of tour patrolling

In this section we present the results of a patrolling experiment. The goal of this experiment is to show that the Stop-When-Ahead control law is effective with real robots and sensors moving through an indoor environment. The map in Fig. 8 is of our lab and for this experiment we used our team of 3 Erratic robots. A set of 6 viewpoints is chosen as in Fig. 8, and a tour through the viewpoints is computed. The priorites set is $\{1, 1.05, 1, 1, 1, 1.08\}$, and the tour length is 18.5 m. The robots implement the Stop-When-Ahead control law while traveling the tour. The robots trajectories and the refresh time of the viewpoints are reported in Fig. 9 and Fig. 10, respectively. The control law achieves satisfactory performance even in the presence of motion and localization uncertainty.

VI. Conclusion

The problem of patrolling a set of weighted viewpoints to minimize their weighted refresh time has been considered. The proposed approach consists of (i) creating a tour through the viewpoints by means of graph-theoretic techniques, and (ii) instructing the robots to travel according to an Equal-Time-Spacing trajectory. Performance bounds for the proposed patrolling strategy have been proven. Additionally, distributed control algorithms have been designed for the robots to self-organize along our proposed trajectory. Finally, our results are validated through simulations and experiments.

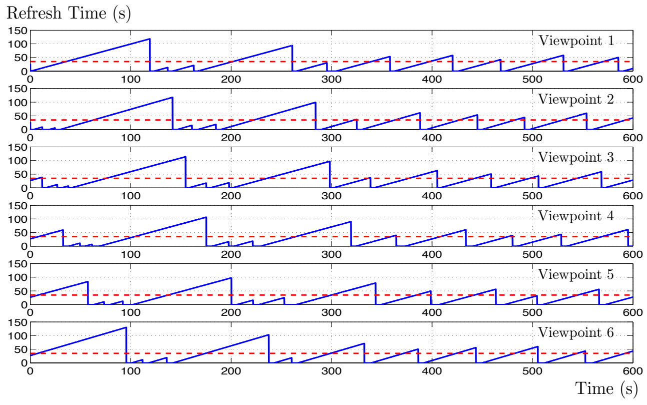


Fig. 10. For the team trajectory in Fig. 9, this figure shows the time intervals, weighted by the corresponding priority, during which the viewpoints are not visited. The dashed line identifies the theoretical bound predicted by Theorem 3.1. The difference between the actual and the predicted refresh time is due to motion uncertainty, localization uncertainty, and the presence of obstacles.

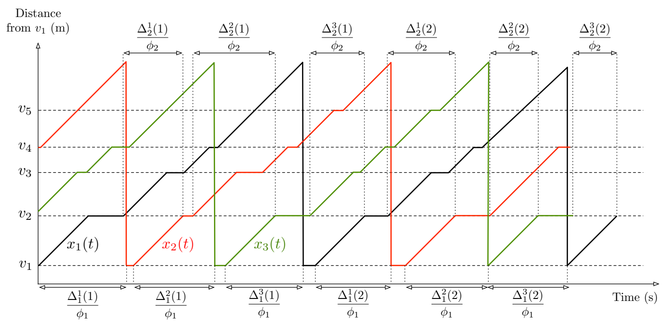


Fig. 11. For the configuration in Fig. 2, the figure shows a team trajectory and the time intervals during which the viewpoints are not visited. The refresh time of a viewpoint corresponds to the longest time interval during which it is not visited, weighted by its priority. The refresh time of the team trajectory is the largest among the refresh times of the viewpoints.

APPENDIX

In this section, we derive a proof of Theorem 3.1. Notice that $\mathrm{RT}(X) = \max\{\mathrm{RT}_1, \mathrm{RT}_2\}$, where $\mathrm{RT}_1 = \mathrm{RT}(X)$ with $t_d \in \{\cup_1^m \mathcal{D}(\alpha,i) : \mathcal{D}(\alpha,i) < \mathcal{A}(1,i)\}$, and $\mathrm{RT}_2 = \mathrm{RT}(X)$ with $t_d \in T_2 = \{\cup_1^m \mathcal{D}(\alpha,i) : \mathcal{D}(\alpha,i) \geq \mathcal{A}(1,i)\}$. We first show that the Equal-Time-Spacing trajectory minimizes the cost RT_2, and then that, for the same solution, $\mathrm{RT}_1 = \mathrm{RT}_2$. Hence, the Equal-Time-Spacing trajectory is a solution to the Cooperative Tour Patrolling problem (Problem 2). We now introduce the necessary notation. For robot r_i, let $\mathcal{A}_{\min}(i) = \min\{t : t \in \mathcal{A}(1,i)\}$ and $\mathcal{A}_{\min} = [\mathcal{A}_{\min}(1) \ldots \mathcal{A}_{\min}(m)]^\mathsf{T}$. The time interval between any two consecutive visits of a viewpoint can be written as a function of the arrival times $\mathcal{A}_{\min}$ and of the waiting intervals $\delta_\alpha^i(k)$ at the viewpoints. Let $\Delta_\alpha^i(k)$ be the time gap (weighted by ϕ_α) between the k-th departure from v_α of robot r_i and the subsequent arrival at v_α of robot r_{i+1}. For instance, for $k=1$ and $\alpha=1$ (cf. Fig. 11),

$$\begin{aligned}
\Delta_1^1(1)/\phi_1 &= \mathcal{A}_{\min}(2) - \mathcal{A}_{\min}(1) - \delta_1^1(1),\\
\Delta_1^2(1)/\phi_1 &= \mathcal{A}_{\min}(3) - \mathcal{A}_{\min}(2) - \delta_1^2(1),\\
&\ \ \vdots\\
\Delta_1^m(1)/\phi_1 &= L + \mathcal{A}_{\min}(1) + \sum_{j=1}^n \delta_j^1(1) - \mathcal{A}_{\min}(m) - \delta_1^m(1).
\end{aligned}$$

Analogously, we have (cf. Fig. 11)

$$\begin{aligned}
\frac{\Delta_2^1(1)}{\phi_2} &= \mathcal{A}_{\min}(2) + \delta_1^2(1) - \mathcal{A}_{\min}(1) - \delta_1^1(1) - \delta_2^1(1),\\
\frac{\Delta_2^2(1)}{\phi_2} &= \mathcal{A}_{\min}(3) + \delta_1^3(1) - \mathcal{A}_{\min}(2) - \delta_1^2(1) - \delta_2^2(1),\\
&\ \ \vdots\\
\frac{\Delta_2^m(1)}{\phi_2} &= L + \mathcal{A}_{\min}(1) + \sum_{j=1}^n \delta_j^1(1) + \sum_{j=1}^n \delta_j^1(2) - \mathcal{A}_{\min}(m)\\
&\quad - \sum_{j=1}^n \delta_j^m(1) - \delta_1^m(2).
\end{aligned}$$

We now derive a more compact expression for these time gaps. Let I_m be the $m \times m$ identity matrix, 0_m the $m \times m$ zero matrix, and B_m the $m \times m$ matrix of zeros with a single 1 in the bottom left corner. Let $D_m = B_m - I_m$, and let Z_m be the $m \times m$ circulant matrix described by the vector $[-1\ 1\ 0\ \ldots\ 0]$.

Let e_i be the i-th column of I_m, and let $\Delta_\alpha(k) = [\Delta_\alpha^1(k)\ \ldots\ \Delta_\alpha^m(k)]^\mathsf{T}$. It can be verified that

$$\underbrace{\begin{bmatrix}\Delta_1(1)\\ \vdots\\ \Delta_n(1)\\ \Delta_1(2)\\ \vdots\\ \Delta_n(2)\\ \vdots\end{bmatrix}}_{\Delta} = \underbrace{\begin{bmatrix}\phi_1 L e_m\\ \vdots\\ \phi_n L e_m\\ \phi_1 L e_m\\ \vdots\\ \phi_n L e_m\\ \vdots\end{bmatrix}}_{b} + DM \underbrace{\begin{bmatrix}\mathcal{A}_{\min}\\ \delta_1^1(1)\\ \vdots\\ \delta_1^m(1)\\ \delta_2^1(1)\\ \vdots\\ \delta_2^m(1)\\ \vdots\\ \delta_1^1(2)\\ \vdots\end{bmatrix}}_{\delta}, \tag{A-1}$$

where

$$M = \begin{bmatrix}
Z_m & D_m & B_m & \cdots & B_m & 0_m & \cdots & 0_m & \cdots\\
Z_m & Z_m & D_m & B_m & \ddots & \ddots & \ddots & \ddots & \ddots\\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0_m & \ddots\\
Z_m & Z_m & Z_m & Z_m & D_m & B_m & \ddots & B_m & \ddots\\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots
\end{bmatrix}, \tag{A-2}$$

$$D = \text{blk-diag}(\phi_1 I_m, \ldots, \phi_n I_m, \phi_1 I_m, \ldots, \phi_n I_m, \ldots). \tag{A-3}$$

Lemma 6.1: **(Equivalent optimization problems)** Given a set of viewpoints with priorities on a non-intersecting tour of length L and a team of $m \geq 2$ robots, a team trajectory with minimum cost RT_2 is computed by solving the minimization

$$\begin{aligned}
&\min_{\delta} \quad \|b + DM\delta\|_\infty,\\
&\text{subject to:} \quad \delta \succeq 0,\ b + DM\delta \succeq 0\\
&\qquad\qquad\quad 0 = \mathcal{A}_{\min}(1) \leq \cdots \leq \mathcal{A}_{\min}(m),
\end{aligned} \tag{A-4}$$

where b, D, and M are as in (A-1), (A-2), and (A-3), and $\succeq$ denotes componentwise inequality.

Proof: The vector δ, with $\delta \succeq 0$, $b + DM\delta \succeq 0$, describes a Stop-Go trajectory ($t \in T_2$), and $\|b+M\delta\|_\infty$ denotes its refresh time RT_2. The statement follows from Lemma 2.2. ■

From Lemma 6.1, a minimum refresh time (RT_2) team trajectory can be computed by solving a linear optimization problem. The vector δ contains the robots arrivals at v_1 and the waiting intervals at each viewpoint, and it describes a Stop-Go team trajectory ($t \in T_2$). Consider the following result.

Lemma 6.2: **(Empty waiting set)** Let $m = n-1$, and let D and M be defined as in (A-1). Then, for all $i \in \{1, \dots, m\}$,

$$\mathcal{C} = \{\delta \; : \; DM\delta \prec 0, \; \delta_1^i(k) \geq 0, \; k \in \mathbb{N}\} = \emptyset.$$

Proof: Let the matrix N be such that $N\delta \geq 0$ is equivalent to $\delta_1^i(k) \geq 0$, for all $i \in \{1, \dots, m\}$ and for all iterations k. In particular, the column corresponding to $\delta_1^i(k)$ in the matrix N, for each robot r_i and iteration k, has only one nonzero entry equal to 1. Recall from [19, Motzkin's Theorem] that the set $\mathcal{C}$ is empty if and only if there exists $y_1 \succeq 0$, with $y_1 \neq 0$, and $y_2 \succeq 0$ such that $y_1 DM - y_2 N = 0$. Let $y_1 = [0 \; e_m^{\mathsf{T}} \; \dots \; e_1^{\mathsf{T}} \; 0 \dots] D^{-1}$, where 0 denotes here an m-dimensional zero vector. It can be verified that $(y_1 DM)_j = 1$ if $j = (m+1)^2$, and $(y_1 DM)_j = 0$ otherwise. Let $(y_2)_j = 1$ if $j = m+1$, and $(y_2)_j = 0$ otherwise. The statement follows from Motzkin's Theorem since $y_1 DM - y_2 N = 0$. ■

Proof of Theorem 3.1: Let $V_{\text{min}} = \emptyset$, $V = V_{\text{max}}$, and $|V| = n = m+1$. Let δ^* be the vector of waiting intervals and arrival times defined in Trajectory 1. Note that the specification of the robots initial positions implies the specification of the arrival times $\mathcal{A}_{\text{min}}$. Suppose that δ^* is not an optimal solution to the optimization problem (A-4), and let δ be a minimizer of (A-4). Then, $\|b + DM\delta\|_\infty < \|b + DM\delta^*\|_\infty$. It can be verified that the entries of $b + DM\delta^*$ indexed by V_{max} are all equal to each other. Also, $b + DM\delta \succeq 0$, $b + DM\delta^* \succeq 0$. Hence,

$$DM(\delta - \delta^*) \prec 0. \tag{A-5}$$

Notice that, since $\delta_1^i(k)^* = 0$ and $\delta_1^i(k) \geq 0$ due to the constraint in (A-4), it follows that $\delta_1^i(k) - \delta_1^i(k)^* \geq 0$. Because of Lemma 6.2, the inequalities (A-5) with the constraint $\delta_1^i(k) - \delta_1^i(k)^* \geq 0$ are infeasible. Due to convexity, we conclude that δ^* is a global minimizer of (A-4) with $V_{\text{min}} = \emptyset$.

Let $V_{\text{min}} \neq \emptyset$, and observe that, because of Lemma 2.2, the weighted refresh time for the set of viewpoints $V_{\text{max}} \cup V_{\text{min}}$ cannot be smaller than the weighted refresh time for the set of viewpoints V_{max}. Then, the vector δ^* defined in Trajectory 1 is a global minimizer of the optimization problem (A-4).

We now characterize the performance of Trajectory 1. Notice that $x_i(t) = x_{i+1}(t + \frac{\mathrm{RT}_{\mathrm{T}}^*}{\phi_1})$. Hence, the viewpoint v_α is not visited for an interval of length $\frac{\mathrm{RT}_{\mathrm{T}}^*}{\phi_1} - \delta_\alpha$, where δ_α is the waiting interval at the viewpoint v_α. We have

$$\frac{\mathrm{RT}_{\mathrm{T}}^*}{\phi_1} - \delta_\alpha = \frac{\mathrm{RT}_{\mathrm{T}}^*}{\phi_1} - \frac{\mathrm{RT}_{\mathrm{T}}^*(\phi_\alpha - \phi_1)}{\phi_\alpha \phi_1} = \frac{\mathrm{RT}_{\mathrm{T}}^*}{\phi_\alpha}.$$

From (1), we have $\mathrm{RT}(X) = \max_\alpha \phi_\alpha \frac{\mathrm{RT}_{\mathrm{T}}^*}{\phi_\alpha} = \mathrm{RT}_{\mathrm{T}}^*$. Note that

$$\frac{\Delta_1^m(1)}{\phi_1} = L + \mathcal{A}_{\text{min}}(1) + \sum\nolimits_{\alpha=1}^{n} \delta_\alpha^1(1) - \mathcal{A}_{\text{min}}(m) - \delta_1^m(1),$$

where $\Delta_1^m(1) = \mathrm{RT}_{\mathrm{T}}^*$, $\delta_1^m(1) = 0$, and

$$\mathcal{A}_{\text{min}}(1) - \mathcal{A}_{\text{min}}(m) = -(m-1)\,\mathrm{RT}_{\mathrm{T}}^*/\phi_1,$$
$$\sum\nolimits_{\alpha=1}^{n} \delta_\alpha^1(1) = m\,\mathrm{RT}_{\mathrm{T}}^*/\phi_1 - \sum\nolimits_{\phi \in \Phi_{\text{max}}} \mathrm{RT}_{\mathrm{T}}^*\,\phi^{-1}.$$

Hence, $\mathrm{RT}_{\mathrm{T}}^* = \frac{L}{\sum_{\phi \in \Phi_{\text{max}}} \phi^{-1}}$. Finally, for the Equal-Time-Spacing trajectory, it holds $\mathrm{RT}_1 = \mathrm{RT}_2 = \mathrm{RT}_{\mathrm{T}}^* = \mathrm{RT}(X)$. ■

References

[1] J. Clark and R. Fierro, "Mobile robotic sensors for perimeter detection and tracking," *ISA Transactions*, vol. 46, no. 1, pp. 3–13, 2007.

[2] D. B. Kingston, R. W. Beard, and R. S. Holt, "Decentralized perimeter surveillance using a team of UAVs," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1394–1404, 2008.

[3] S. Susca, S. Martínez, and F. Bullo, "Monitoring environmental boundaries with a robotic sensor network," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 2, pp. 288–296, 2008.

[4] Y. Elmaliach, A. Shiloni, and G. A. Kaminka, "A realistic model of frequency-based multi-robot polyline patrolling," in *International Conference on Autonomous Agents*, Estoril, Portugal, May 2008, pp. 63–70.

[5] I. I. Hussein and D. M. Stipanovic̀, "Effective coverage control for mobile sensor networks with guaranteed collision avoidance," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 4, pp. 642–657, 2007.

[6] C. G. Cassandras, X. C. Ding, and X. Lin, "An optimal control approach for the persistent monitoring problem," Aug. 2011, available at `http://arxiv.org/pdf/1108.3221`.

[7] Y. Chevaleyre, "Theoretical analysis of the multi-agent patrolling problem," in *IEEE/WIC/ACM Int. Conf. on Intelligent Agent Technology*, Beijing, China, Sep. 2004, pp. 302–308.

[8] D. B. Kingston, R. S. Holt, R. W. Beard, T. W. McLain, and D. W. Casbeer, "Decentralized perimeter surveillance using a team of UAVs," in *AIAA Conf. on Guidance, Navigation and Control*, San Francisco, CA, Aug. 2005.

[9] F. Pasqualetti, A. Franchi, and F. Bullo, "On cooperative patrolling: Optimal trajectories, complexity analysis and approximation algorithms," *IEEE Transactions on Robotics*, Jan. 2011, to appear.

[10] G. Cannata and A. Sgorbissa, "A minimalist algorithm for multirobot continuous coverage," *IEEE Transactions on Robotics*, vol. 27, no. 2, pp. 297–312, 2011.

[11] S. L. Smith and D. Rus, "Multi-robot monitoring in dynamic environments with guaranteed currency of observations," in *IEEE Conf. on Decision and Control*, Atlanta, GA, USA, Dec. 2010, pp. 514–521.

[12] S. L. Smith, M. Schwager, and D. Rus, "Persistent robotic tasks: Monitoring and sweeping in changing environments," *IEEE Transactions on Robotics*, Feb. 2011, to appear.

[13] D. Peleg, *Distributed Computing. A Locality-Sensitive Approach*, ser. Monographs on Discrete Mathematics and Applications. SIAM, 2000.

[14] A. Davoodi, P. Fazli, P. Pasquier, and A. K. Mackworth, "On multi-robot area coverage," in *Japan Conference on Computational Geometry and Graphs*, Kanazawa, Japan, Nov. 2009, pp. 75–76.

[15] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks*, ser. Applied Mathematics Series. Princeton University Press, 2009, available at http://www.coordinationbook.info.

[16] B. Gerkey, R. T. Vaughan, and A. Howard, "The Player/Stage Project: Tools for multi-robot and distributed sensor systems," in *Int. Conference on Advanced Robotics*, Coimbra, Portugal, Jun. 2003, pp. 317–323.

[17] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1-2, pp. 99–141, 2001.

[18] J. W. Durham and F. Bullo, "Smooth nearness-diagram navigation," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, Nice, France, Sep. 2008, pp. 690–695.

[19] O. L. Mangasarian, *Nonlinear Programming*. SIAM, 1994.