Collision Avoidance for Persistent Monitoring in Multi-Robot Systems with Intersecting Trajectories

Daniel E. Soltero Stephen L. Smith Daniela Rus

***Abstract*— Persistent robot tasks such as monitoring and cleaning are concerned with controlling mobile robots to act in a changing environment in a way that guarantees that the uncertainty in the system (due to change and to the actions of the robot) remains bounded for all time. Prior work in persistent robot tasks considered only robot systems with collision-free paths that move following speed controllers. In this paper we describe a solution to multi-robot persistent monitoring, where robots have intersecting trajectories. We develop collision and deadlock avoidance algorithms that are based on stopping policies, and quantify the impact of the stopping times on the overall stability of the speed controllers.**

I. Introduction

In this paper we consider the problem of persistently monitoring or sweeping a changing environment using a group of robots equipped with sensors with finite footprints. In our prior work, we developed methods for computing closed monitoring paths for robots [1], and for computing the speed with which each robot should follow its path [2]. This prior work assumes the paths are non-intersecting. However, most efficient monitoring paths may intersect. Thus, a collision avoidance procedure is needed. In this paper we develop a collision avoidance procedure for persistent monitoring, and analyze the effect of this procedure on the stability of the persistent controller.

Building on our prior work [2], we assume a changing environment modeled as a scalar valued function, called the *accumulation function*. The function captures the uncertainty at each point in the environment for a sensing task, or the quantity of material at each point for a cleaning task. The accumulation function grows at a constant rate at points outside the range of a robot, and decreases at a constant rate at points within its range. The rate of growth and decrease can be different at different points in the environment. The model captures the accumulation of material in a sweeping or cleaning task, and provides an approximation for the uncertainty in a monitoring task.

In [2] we assume that paths are given for each robot, and describe how to control the speed along the path to keep the accumulation function bounded. We developed a method for computing a speed controller for each robot that maximizes the *stability margin* of the system: The larger the stability margin, the more tolerance the robots will have to unmodeled changes in the accumulation function.

We assume we are given a group of robots, a closed path for each robot, and a speed controller for each robot along its path. We would like for each robot to continually follow its path using its prescribed speed profile, while avoiding collisions with the other robots. The collisions should be avoided in such a way that we can still guarantee boundedness of the accumulation function.

Persistent monitoring is related to sweep coverage [3], and patrolling problems [4], [5], [6] where robots with finite sensor footprints must sweep their sensor over every point in the environment. The problem is also related to environmental monitoring research such as [7], [8], [9], [10], [11]. In this prior work, the authors often use a probabilistic model of the environment, and estimate the state of that model using a Kalman-like filter. Then robots are controlled so as to maximize a metric on the quality of the state estimate. The collision avoidance problem is not addressed in these works. In addition, due to the complexity of the models used, performance guarantees are difficult to obtain. This makes it difficult to characterize the effect of collision avoidance on the system performance. In this paper, we use a more tractable model, and in doing so we can provide guarantees on the boundedness of the accumulation function.

There is a wealth of literature on collision avoidance. A common method is to use artificial potential fields [12], which repel robots from each other and from obstacles. A recently proposed method relies on velocity obstacles [13]. Such methods result in the robots deviating from their prescribed paths and from their desired speed profiles. For the persistent application, the effect of such deviations on the accumulation function is difficult to characterize. A more tractable approach is to constrain each robot to remain on its path. In this context the most closely related work is on path traversal problems where each robot is given a path. The goal is to minimize the time until the robots reach their destination points on their paths, while avoiding collisions [14]. In [15], the authors consider a variation in which the full trajectory (path and speed along the path) is specified for each robot. Collisions can be avoided only by changing the start times of each robot along its path. Our approach to collision avoidance is closely related [15]. However, since our paths are closed, and are repeatedly traversed by the robots, we avoid collisions by repeatedly stopping robots.

The main contribution of this paper is to enable the persistent speed controllers developed in [2] to operate

This material is based upon work supported in part by ONR-MURI Award N00014-09-1-1051, the NSF Graduate Research Fellowship Award 0645960 & The Boeing Company.

D. E. Soltero and D. Rus are with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge MA 02139 (soltero@mit.edu, rus@csail.mit.edu). S. L. Smith is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo ON, N2L 3G1, Canada (stephen.smith@uwaterloo.edu)

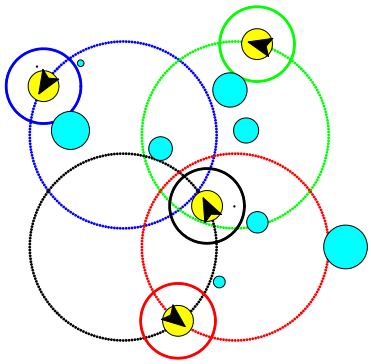


Fig. 1: Four robots (yellow-filled circles) performing a persistent task where their paths produce many collision zones. Each robot r has a unique color to represent its path γ_r and its footprint. The paths are the four large circles, and the footprints are the small circles which are centered around each respective robot. In the persistent task, the robots follow a speed profile which seeks to keep the accumulation function of each point of interest bounded. In this figure, the points of interest are colored in light blue and the value of the accumulation function at each point is proportional to the size of that point.

when multiple robots have intersecting trajectories. Here, we mean intersection in the sense that the robot bodies could collide. We develop a collision avoidance procedure based on stopping, and quantify its effect on the stability of these controllers. The collision avoidance operates by identifying *collision zones* in which collisions could occur. We then avoid collisions by stopping and restarting robots so that at most one robot occupies a given collision zone at any moment in time. We also design a procedure to avoid *deadlocks*; a situation in which a group of robots are all stopped, and are waiting for each other to move before resuming motion. We identify several different *stopping policies* and perform extensive simulations to determine the most effective policy. We also present results from a distributed implementation for two ground robots performing a persistent monitoring task.

The organization of this paper is as follows. In Section II we formulate the problem, and in Sections III and IV we introduce collision zones and how they avoid both collisions and deadlock. In Section V we introduce three different stopping policies and in Section VI we derive a condition under which the accumulation function will remain bounded when using our collision avoidance procedure. Finally, in Section VII we present stopping policy simulation results and our distributed implementation.

II. Problem Formulation

We are given n robots, indexed by $r \in \{1, 2, \ldots, n\}$. Each robot is constrained to move along a pre-determined path $\gamma_r : [0, 2\pi] \to \mathbb{R}^2$, where $\gamma_r(0) = \gamma_r(2\pi)$. Path γ_r is parametrized by $0 \leq \theta_r \leq 2\pi$, which is assumed to be the arc-length parametrization. The robot's position at time t can be described by $\theta_r(t)$, its position along the curve γ_r. Each robot has a sensor/sweeping footprint $\mathcal{B}_r(\theta_r(t))$. This footprint could be thought of, for example, as the cleaning surface of a sweeping robot. The environment contains a finite number of points of interests $\mathbf{q} \in Q$. These finite points could be the discretization of a continuous environment. A scalar field $Z(\mathbf{q}, t) \geq 0$ is defined over the points of interest $\mathbf{q} \in Q$. The field (called the accumulation function) behaves analogously to dust accumulating over a floor. At a point of interest $\mathbf{q} \in Q$, the field increases at a constant production rate of $p(\mathbf{q})$ when not covered by any robot footprints, and it is consumed at a constant rate of $c_r(\mathbf{q})$, by each robot r whose footprint is covering $\mathbf{q}$. More specifically,

$$\dot{Z}(\mathbf{q}, t) = \begin{cases} p(\mathbf{q}) - \sum\limits_{r \in \mathcal{N}_{\mathbf{q}}(t)} c_r(\mathbf{q}), & \text{if } Z(\mathbf{q}, t) > 0, \\ \Big(p(\mathbf{q}) - \sum\limits_{r \in \mathcal{N}_{\mathbf{q}}(t)} c_r(\mathbf{q})\Big)^+, & \text{if } Z(\mathbf{q}, t) = 0, \end{cases} \tag{1}$$

where $\mathcal{N}_{\mathbf{q}}(t)$ is the set of robots whose footprints are over the point $\mathbf{q}$ at time t, $\mathcal{N}_{\mathbf{q}}(t) := \{r \mid \mathbf{q} \in \mathcal{B}_r\big(\theta_r(t)\big)\}$.

Our earlier work [2] studied the problem of controlling the speed of each robot along its path in order to keep the field (accumulation function) $Z(\mathbf{q}, t)$ bounded for all time t, and for all points $\mathbf{q}$. We showed that a necessary and sufficient condition for stability of the field is that

$$\sum_{r=1}^{n} \frac{\tau_r(\mathbf{q})}{T_r} c_r(\mathbf{q}) - p(\mathbf{q}) > 0,$$

for all points of interest $\mathbf{q} \in Q$, where T_r is the period (or cycle time) of robot r along its path γ_r, $\tau_r(\mathbf{q})$ is the amount of time per period that robot r's footprint is covering the point of interest $\mathbf{q}$. We then developed a method for producing a speed controller $v_r(\theta_r)$ for each robot r, which maximizes the *stability margin*

$$\min_{\mathbf{q} \in Q} \left(\sum_{r=1}^{n} \frac{\tau_r(\mathbf{q})}{T_r} c_r(\mathbf{q}) - p(\mathbf{q}) \right). \tag{2}$$

Figure 1 shows four robots performing a persistent monitoring task. This set of paths generates multiple potential collision locations. Therefore, when implementing the system, if no collision avoidance procedure is used, the robots would collide frequently. The objective of this paper is to present and prove a practical collision avoidance procedure for a multi-robot persistent monitoring task that seeks to minimize its effects on the stabilizing persistent speed controller.

III. Collision Avoidance

For each robot r we define a safety radius $\rho_r > 0$, and the corresponding safety disk $B(\gamma_r(\theta_r), \rho_r) = \{x \in \mathbb{R}^2 : \|x - \gamma_r(\theta_r)\| \leq \rho_r\}$.[1] We say that a collision occurs between robots i and j at locations (θ_i, θ_j) if $B(\gamma_i(\theta_i), \rho_i) \cap B(\gamma_j(\theta_j), \rho_j) \neq \emptyset$. To avoid robot collisions, we must first know where collisions can occur. There are a number of ways to do this. For example, we could search all (θ_i, θ_j) pairs between any two robots i and j and obtain the set of collision configurations between any two robots. The set of collision configurations between robot i and j is defined as

[1]For simplicity of presentation we use a safety disk, but our collision procedure works for any safety set containing the robot's current position.

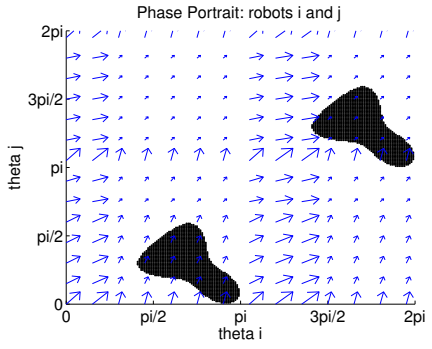


Fig. 2: Phase portrait for robots i and j. The axis are θ_i and θ_j, which are the parametrized positions of robots i and j. In this figure, the black-colored sets are the set of collision configurations (θ_i, θ_j) for robots i and j. The arrows in this plot correspond to the phase portrait, i.e. flow lines of the state (θ_i, θ_j) through time. If no collision avoidance is used, the robots will eventually collide due to the flow lines leading (θ_i, θ_j) into the black sets.

$P_{i,j} = \{(\theta_i, \theta_j) \in [0, 2\pi]^2 : B(\gamma_i(\theta_i), \rho_i) \cap B(\gamma_j(\theta_j), \rho_j) \neq \emptyset\}$.

An example of $P_{i,j}$ can be seen in Figure 2. This figure shows the phase portrait of θ_i vs. θ_j for a given γ_i and γ_j. The flow lines depict speed profiles that stabilize the system for a given set of points of interest. The set $P_{i,j}$ is given by the black-colored regions. There are various flow lines that lead (θ_i, θ_j) to enter $P_{i,j}$, resulting in collisions. One way to avoid collision is to not allow (θ_i, θ_j) to enter $P_{i,j}$ by forcing (θ_i, θ_j) to move along the edge of $P_{i,j}$. However, there are a few problems with this approach. First, since $P_{i,j}$ can have an arbitrary geometric shape, a collision avoidance of this type will in general require robots to move backwards. Moving backwards affects the shape of the phase portrait by including flow lines at angles less than zero or greater than $\pi/2$. When considering multiple robots, such backward motion will require additional collision avoidance procedures. Second, we are interested in solutions that can be implemented in a distributed manner, and thus we would like the robots to utilize pair-wise decisions in order to avoid collisions. However, it is possible for such pair-wise decisions to contradict each other. This can result in a *deadlock* situation in which a group of robots are all blocking each other.

Based on the above discussion, we propose a collision avoidance method that relies on stopping the robots. When a robot stops in order to avoid collision, the collision avoidance procedure pauses the speed controller, and un-pauses it to resume robot motion. When the speed controller is un-paused, it is as if the system is re-started with a new set of initial conditions. This method for avoiding collision is tractable in a persistent task because the speed profile prescribed by [2] is proven to stabilize the field for any set of initial conditions. Therefore, any increase in the accumulation functions of the field while a robot is stopped will eventually be consumed, maintaining the system stable. However, if trajectories are selected in such a way that collision avoidance is needed very frequently, then the stabilizing effect of the speed controller may not be "strong" enough to overcome the frequent stops.

We describe an algorithm for stopping robots that ensures no deadlock amongst stopped robots while enabling the persistent operation of the system without collision. The intuition is to compute, for each robot, the region in space where the robot might collide with any other robot.

Since the path for each robot is known, we can search for all possible collision locations along the path. The set of all possible collision locations for robot i is defined as $P_i = \{\theta_i \in [0, 2\pi] : B(\gamma_i(\theta_i), \rho_i) \cap B(\gamma_j(\theta_j), \rho_j) \neq \emptyset$ for some $j \neq i\}$. For example, in Figure 2, P_i would be set containing the projection of $P_{i,j}$ onto the θ_i axis, and P_j would be the set containing the projection onto the θ_j axis. We can decompose P_i into a collection of w_i connected sets C_i^k, which are pairwise disjoint. Thus, we have $P_i = \cup_{k=1}^{w_i} C_i^k$, and $C_i^k \cap C_i^{k'} = \emptyset$ for any $k, k' \in \{1, \ldots, w_i\}$. As an example, in Figure 2, both P_i and P_j consist of two disjoint sets.

We would now like to determine the following: if robot i enters the set C_i^k, which sets $C_j^{k'}$ must robot j avoid? We can relate individual collision zones by constructing an undirected graph where each C_i^k is a node, $\forall i, k$. We define an edge between two nodes C_i^k and $C_j^{k'}$ if $B(\gamma_i(\theta_i), \rho_i) \cap B(\gamma_j(\theta_j), \rho_j) \neq \emptyset$, for some $\theta_i \in C_i^k$ and some $\theta_j \in C_j^{k'}$. We will refer to this graph as the *collision graph*.

Definition III.1 (Collision Zone)**.** Given the collision graph, a collision zone CZ^m is defined for each connected component in the graph, where m will range from 1 to the total number of connected components in the graph. CZ^m is defined as a tuple $\text{CZ}^m = (\text{CZ}_1^m, \text{CZ}_2^m, \ldots, \text{CZ}_n^m)$, where CZ_i^m is the union of nodes C_i^k, $\forall k$, present in the connected component corresponding to CZ^m

Note that CZ_i^m is disjoint from $\text{CZ}_i^{m'}$ for all i, where $m \neq m'$, because if they were not, then $\exists \theta_i$ for some i such that $\theta_i \in C_i^k$ and $\theta_i \in C_i^{k'}$, where $C_i^k \in \text{CZ}_i^m$ and $C_i^{k'} \in \text{CZ}_i^{m'}$. However, by definition C_i^k and $C_i^{k'}$ are disjoint. Therefore, CZ_i^m and $\text{CZ}_i^{m'}$ are also disjoint. Example III.2 and Figure 4 illustrate these mathematical constructions.

Example III.2. Figure 3 shows three trajectories that intersect each other at several places. After obtaining all the disjoint collision sets C_i^k for all robots, we construct the collision graph in Figure 4, which shows five connected components, i.e. five collision zones CZ^m, $m \in \{1, 2, 3, 4, 5\}$, where:

(i) $\text{CZ}_1^1 = C_1^1$, $\text{CZ}_2^1 = C_2^1$, $\text{CZ}_3^1 = (C_3^1 \cup C_3^2)$
(ii) $\text{CZ}_1^2 = C_1^2$, $\text{CZ}_2^2 = \emptyset$, $\text{CZ}_3^2 = C_3^3$
(iii) $\text{CZ}_1^3 = C_1^3$, $\text{CZ}_2^3 = \emptyset$, $\text{CZ}_3^3 = C_3^4$
(iv) $\text{CZ}_1^4 = \emptyset$, $\text{CZ}_2^4 = C_2^2$, $\text{CZ}_3^4 = C_3^5$
(v) $\text{CZ}_1^5 = \emptyset$, $\text{CZ}_2^5 = C_2^3$, $\text{CZ}_3^5 = C_3^6$

The following shows the importance of collision zones.

Theorem III.3. *If any two robots i and j collide, then $\theta_i \in \text{CZ}_i^m$ and $\theta_j \in \text{CZ}_j^m$, for some m.*

Proof. Suppose there is a collision between robots i and j, when robot i is at θ_i and robot j is at θ_j. This implies that $B(\gamma_i(\theta_i), \rho_i) \cap B(\gamma_j(\theta_j), \rho_j) \neq \emptyset$. By definition, $\theta_i \in C_i^k$ and $\theta_j \in C_j^{k'}$ for some k and k'. Also by definition, there

Algorithm 1: COLLISION AVOIDANCE FRAMEWORK, FOR ROBOT i

Data: θ_i is at the entering edge of CZ_i^m
Result: No collisions occur.
if flag_m *is raised* **then**
  Stop trajectory until flag_m is lowered.
else
  Robot i can enter CZ_i^m and raise flag_m.
  When robot i exits CZ_i^m, then flag_m is lowered.

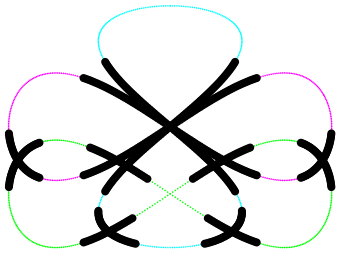


Fig. 3: Plot of paths for the three robots, whose collision zones were obtained in Figure 4. The final five collision zones CZ^m are plotted in segments of black, while the rest of the trajectories are plotted in different light colors.

exists an edge in the collision graph between nodes C_i^k and $C_j^{k'}$. This, in turn, means that $\theta_i \in \text{CZ}_i^m$ and $\theta_j \in \text{CZ}_j^m$ for some m. □

By Theorem III.3, no collisions are possible in CZ^m if there exists at most one robot i such that $\theta_i \in \text{CZ}_i^m$. Therefore, the collision avoidance framework will allow at most one robot to travel through CZ^m at any moment in time, for each m. Let flag_m be a flag which is raised if a robot i is currently inside CZ_i^m. The collision avoidance framework is given as Algorithm 1. A requirements for Algorithm 1 to work is that if flag_m is raised by some robot i, then it is lowered only by robot i once it exits CZ_i^m.

Note that the converse of Theorem III.3 is not true. That is, if $\theta_i \in \text{CZ}_i^m$ and $\theta_j \in \text{CZ}_j^m$, for some m, this does not mean that the robots have collided. This is because, in general, a connected set of (θ_i, θ_j) that results in collision is not equal to the cartesian product of its projections on the θ_i and θ_j axis, although it is a subset. This means that this collision avoidance algorithm is conservative. However, the trade-off is a solution without backward motion and

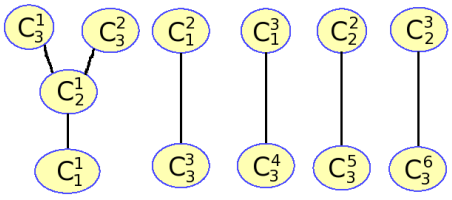


Fig. 4: Example of a collision graph used to construct the collision zones CZ^m. In this graph, there are 12 nodes corresponding to the disjoint connected sets in P_1, P_2, P_3, for robots 1, 2 and 3, respectively. The result from this graph is that there are five disjoint collision zones, i.e. CZ^m, $m \in \{1, 2, 3, 4, 5\}$. These five collision zones are mapped to their respective paths in Figure 3.

conflicting decisions where a robot stops while blocking the path of another, potentially causing a deadlock.

IV. DEADLOCK AVOIDANCE

Much of the previous work in deadlock avoidance is based on re-planning of the robot trajectories [16], [17], [18], [19], or schedule coordination for robots [20]. In our problem formulation, however, the robots are constrained to their prescribed paths and speed controllers, so these approaches do not apply. Instead, our approach is similar to [21], where graphs are used to detect and avoid collisions in critical sections, and to [22], where permission is given to one robot to move along a zone that can cause deadlock.

In order to avoid deadlock, we define the notion of a deadlock graph.

Definition IV.1 (Deadlock Graph)**.** A deadlock graph is a directed graph, where an edge from node i to node j encodes in robot i is stopped waiting for robot j to exit a collision zone CZ_j^m, for some m.

It is assumed that all robots have knowledge of the deadlock graph. In our application, deadlocks can be avoided by noting that they can only occur when a cycle is created on a deadlock graph. If a cycle were to exist, then deadlock is avoided by erasing one of the edges in the cycle. This corresponds to one of the robots resuming motion and breaking the deadlock.

Definition IV.2 (Stopping Policy)**.** A stopping policy, executed by robot i when about to enter a collision zone is any algorithm that returns n options ranked from best to worst. Out of n options, $n-1$ correspond to robot i stopping for each other robot j, i.e., an edge being drawn from node i to node j in the deadlock graph. The last option corresponds to robot i continuing its trajectory, i.e., no edge drawn in the deadlock graph.

Building upon Algorithm 1, if flag_m is not raised, then the robot can use a stopping policy to obtain the ranked n options. The deadlock avoidance framework consists of choosing the best-ranked option that does not cause a cycle in the deadlock graph. The collision avoidance and deadlock avoidance frameworks merged with the stopping policies, lead to the complete *collision avoidance algorithm*, seen as Algorithm 2, which ensures no collision or deadlocks will occur between any number of robots.

Algorithm 2 is executed only in two situations:

(i) when robot i is moving along its trajectory and is about to enter CZ_i^m
(ii) when robot i is stopped at the edge of CZ_i^m, and the robot j that was inside CZ_j^m just exited CZ_j^m.

The following theorem shows that Algorithm 2 avoids deadlocks.

Theorem IV.3. *Assuming all robots follow Algorithm 2, consider a robot i about to enter a collision zone CZ_i^m for any m. Then, there exists a decision by robot i which does not cause a deadlock.*

Algorithm 2: COLLISION AND DEADLOCK AVOIDANCE ALGORITHM, FOR ROBOT i

Data: θ_i is at the entering edge of CZ_i^m
Result: No collisions or deadlocks occur.
if flag_m *is raised* **then**
  Stop trajectory.
else
  Execute stopping policy. Choose best-ranked option that does not create a cycle in the deadlock graph.
  If the decision is to continue trajectory, then flag_m is raised, and any outgoing edge from robot i's node in the deadlock graph is deleted.
  Otherwise the decision edge is drawn in deadlock graph.
  When robot i exits CZ_i^m, then flag_m is lowered.

Proof. Suppose robot i is about to enter CZ_i^m. If flag_m is not raised, then it will test its n options obtained from a stopping policy, in the deadlock graph. If the best ranked option does not cause a cycle in the deadlock graph, then it is allowed and the robot chooses that option. If the option causes a cycle, then it is not allowed, and the robot tests the next best-ranked option, and so on. By default, the robot will always have the option of continuing its trajectory, which does not create an edge in the deadlock graph, which in turn does not create a cycle.

If flag_m is raised, then robot i is forced to stop because some robot j is already in CZ_j^m. Conceptually, robot i would draw an edge to robot j in the deadlock graph. However, this does not create a cycle because robot j is moving inside the collision zone, and will never stop inside it because CZ_j^m is disjoint from $\text{CZ}_j^{m'}$, $\forall m' \neq m$. □

V. STOPPING POLICIES

We are interested in stopping policies that minimize the effect on the stabilizing persistent speed controller, i.e. we would like the stopping policy to result in maximizing the stability margin of the system, which is given by Equation (2). However, Equation (2) assumes the system is periodic (a proof for this can be found in [2]). That is, it assumes that each robot takes a fixed amount of time to complete a cycle of its path. However, each time a robot stops, it breaks periodicity. In Sections V-A, V-B and V-C, we define three stopping policies, two of which use very similar versions of Equation (2) to generate their n ranked options.

For the following policies, executed by robot i, option_i corresponds to robot i continuing its trajectory, and option_j corresponds to robot i stopping for robot j. These policies assume no other collision procedures are taking place while the current collision is being resolved. Although this is not always true, it provides a quick and inexpensive approximation, compared to the price of obtaining the exact information.

Remark V.1 (Limitation of locally optimizing). All of the following stopping policies look ahead in time, up to the point where the collision is avoided, and they optimize the system, according to their metric for that look-ahead time. Therefore, these policies are limited to optimizing locally in time, and there could be the case where a decision is optimal for the policy, but suboptimal over a longer time horizon.

Algorithm 3: MIN-TIME POLICY, FOR ROBOT i

Initialize: $\text{option}_i = \infty$
foreach *robot* $j \neq$ *robot* i **do**
  /* If stopping is necessary */
  if $T^i_{exit} > T^j_{enter}$ **then**
    $\text{option}_i = \min(T^i_{exit} - T^j_{enter}, \text{option}_i)$
    $\text{option}_j = T^j_{exit}$
  else
    $\text{option}_j = \infty$
Rank the options in ascending order of their values.

A. Minimum Time Policy

This policy ranks the options based on how much time a robot spends stopped to avoid the collision. For robot i, the policy takes into account all of the robots' current positions and trajectories, and calculates the amount of time robot i would have to stop while the other robots enter and exit the collision zone, and the amount of time the other robots would have to stop while robot i exits the the collision zone. The options are ranked in ascending order of stopping time. Algorithm 3 contains the pseudocode for this policy, which is called the *Min-Time* policy, where:

- T^i_{enter} is the time robot i would take to get from where it is right now until it enters the collision zone.
- T^i_{exit} is the time robot i would take to get from where it is right now until it exits the collision zone.

B. All Time Policy

This stopping policy approximates Equation (2) with the *empirical stability margin* of the system up until the current time t. This is done by constructing the same expression in Equation (2), where T_r becomes the current time t, and $\tau_r(\mathbf{q})$ becomes the total coverage time on point $\mathbf{q}$. The policy estimates the empirical stability margin at the time when the collision is avoided, and ranks the options in descending order of estimated empirical stability margin. Algorithm 4 contains the pseudocode for this policy, which is called the *All-Time* policy, where:

- $TC^i_{enter}(\mathbf{q})$ is the time robot i covers point $\mathbf{q}$ while it moves from its current position until it enters CZ_i^m
- $TC^i_{exit}(\mathbf{q})$ is the time robot i covers point $\mathbf{q}$ while it moves from its current position until it exits CZ_i^m
- The average consumption on point $\mathbf{q}$ from all robots, except robots i and j, up to time t is obtained by

$$\sum_{r \neq i,j} \frac{\tau_r(\mathbf{q})}{t} c_r(\mathbf{q}) \tag{3}$$

- The average consumption on point $\mathbf{q}$ from robots i and j at the time the collision is avoided, assuming robot i

Algorithm 4: ALL-TIME POLICY, FOR ROBOT i

```
Initialize: option_i = −∞
Initialize: V_r(q) = −p, ∀r
foreach robot j ≠ robot i do
    V_i(q) = −p
    /* If stopping is necessary */
    if T^i_exit > T^j_enter then
        /* case: robot i continues moving */
        foreach q do
            Add to V_i(q) the average consumption on point q from all robots, except robots i and j, given by equation (3).
            Add to V_i(q) the average consumption on point q, by robots i and j, given by equation (4).
        /* case: robot j continues moving */
        foreach q do
            Add to V_j(q) the average consumption on point q from all robots, except robots i and j, given by equation (3).
            Add to V_j(q)the average consumption on point q, by robots i and j, given by equation (5).
        option_i = max(min_q V_i(q), option_i).
        option_j = min_q V_j(q).
    else
        option_j = −∞.
Rank the options in descending order of their values.
```

continues moving is obtained by

$$\frac{\big(\tau_i(\mathbf{q}) + TC^i_{exit}\big)c_i(\mathbf{q})}{t + T^i_{exit}} + \frac{\big(\tau_j(\mathbf{q}) + TC^j_{enter} + I_j(T^i_{exit} - T^j_{enter})\big)c_j(\mathbf{q})}{t + T^i_{exit}}, \quad (4)$$

where $I_j = 1$ if robot j's footprint covers $\mathbf{q}$ while it is stopped, and zero otherwise.

- The average consumption on point $\mathbf{q}$ from robots i and j at the time the collision is avoided, assuming robot j continues moving is obtained by

$$\frac{\big(\tau_i(\mathbf{q}) + I_i T^j_{exit}\big)c_i(\mathbf{q}) + \big(\tau_j(\mathbf{q}) + TC^j_{exit}\big)c_j(\mathbf{q})}{t + T^j_{exit}}, \quad (5)$$

where $I_i = 1$ if robot i's footprint covers $\mathbf{q}$ while it's stopped, and zero otherwise.

C. Time Window Policy

This policy is similar to the *All-Time* policy, but instead of considering all past information, it only considers information in a time window of constant length T_w. This is done by exchanging t for T_w in Algorithm 4. Also, $\tau_r(\mathbf{q})$ becomes the coverage time of point $\mathbf{q}$ within the time window. We will refer to this stopping policy as the *Time-Window* policy.

VI. PERFORMANCE BOUND

Equation (2) is not useful for persistent monitoring with intersecting paths since this problem is not periodic. However, it can be used to prove performance bounds on the system. We define $V(\mathbf{q})$ to be the stability margin for point $\mathbf{q}$. That is,

$$V(\mathbf{q}) = \sum_{r=1}^{n} \frac{\tau_r(\mathbf{q})}{T_r} c_r(\mathbf{q}) - p(\mathbf{q}). \quad (6)$$

Then the stability constraint for a periodic system is $V(\mathbf{q}) > 0$, $\forall \mathbf{q}$, and the following result holds.

Theorem VI.1. *Consider a persistent task, and a set of speed controllers that result in a stability margin of $V(\mathbf{q})$ for each point of interest $\mathbf{q} \in Q$. There exists a known $\alpha > 0$ such that if $V(\mathbf{q}) > \alpha p(\mathbf{q})$ for each $\mathbf{q}$, then the robot system will remain stable under Algorithm 2. The value α is a function of the trajectories and safety disks of all robots.*

Proof. In the worst-case, each robot will have to stop at every collision zone on each cycle of its path. In this worst-case, each robot takes the same amount of time to complete each cycle, and thus we can use Equation (6). Then the stability constraint for all points $\mathbf{q}$ becomes

$$\sum_{r=1}^{n} \frac{\tau_r(\mathbf{q})}{a_r T_r} c_r(\mathbf{q}) > p(\mathbf{q}),$$

where $a_r = (T_r + T^s_r)/T_r \geq 1$, and T^s_r is the maximum stopping time for robot r in one cycle along its path, and is defined as $T^s_r := \sum_m (n'_m - 1)(\int_{\mathrm{CZ}^m_r} \frac{1}{v_r(\theta_r)} d\theta_r)$, where n'_m is the number of robots whose CZ^m_r is nonempty. Let $a = \max_r a_r$. Then,

$$\sum_{r=1}^{n} \frac{\tau_r(\mathbf{q})}{a_r T_r} c_r(\mathbf{q}) \geq \frac{1}{a} \sum_{r=1}^{n} \frac{\tau_r(\mathbf{q})}{T_r} c_r(\mathbf{q}).$$

Therefore, for the system to be stable in the worst case,

$$\frac{1}{a} \sum_{r=1}^{n} \frac{\tau_r(\mathbf{q})}{T_r} c_r(\mathbf{q}) > p(\mathbf{q}). \quad (7)$$

Substituting Equation (6) into Equation (7), we get

$$V(\mathbf{q}) > (a-1)p(\mathbf{q}). \quad (8)$$

Setting $\alpha = (a-1)$ we get the desired result. □

If a system satisfies Equation (8) $\forall \mathbf{q}$, then it is guaranteed to be stable, no matter how many collision avoidance steps are needed. Although this bound is not tight, it captures a sufficient amount of slack in the system that would result in the system being stable, collision-free and deadlock-free.

VII. SIMULATION RESULTS AND IMPLEMENTATION

A. Stopping Policy Simulation Results

We implemented in simulation the collision and deadlock avoidance strategies, as well as the three stopping policies in Section V. The *Time-Window* policy was implemented with two different time windows: $T_w = \max_r(T_r)$ and $T_w = 3\max_r(T_r)$. We refer to the former as *Time-Window*$_1$ and the later as *Time-Window*$_3$. We also implemented a *Greedy* stopping policy, which allows the first robot to enter a collision zone, and queues subsequently arriving robots.

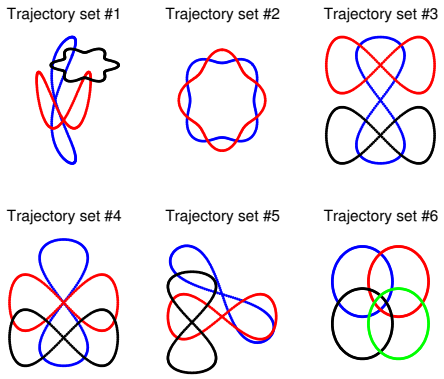


Fig. 5: Six different sets of trajectories used to obtain results on the performance of the tested stopping policies. These trajectories range from using only two robots to using four robots. Each robot has a different colored trajectory.

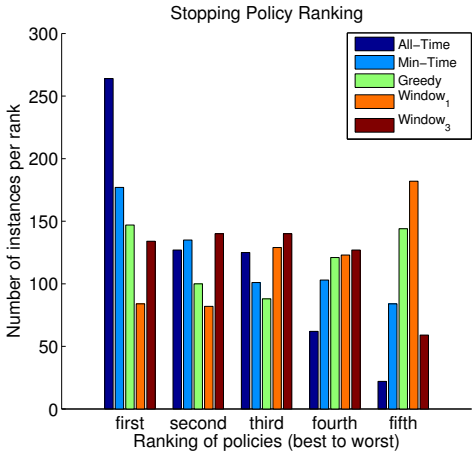


Fig. 6: Results from stopping policy simulations. The horizontal axis corresponds to the ranking of the policies in the simulation instances, from first place (corresponding to the best policy) to fifth place (corresponding to the policy with worst results). The best policy refers to the policy that generated the largest empirical stability margin. The vertical axis corresponds to the number of instances that the policy achieved a particular ranking.

We generated six sets of test trajectories shown in Figure 5, for systems ranging from two robots to four robots. We simulated each trajectory set 100 times. Each one of these simulations is called a *test case*, and contained 10 randomly located points with random production rates, and a speed controller obtained from [2] that stabilized the environment. Each test case was simulated five times, one for each stopping policy. In each simulation instance, the robots had a safety disk of ρ_r equal to 1.25 times the radius of the robot, and they were initialized in collision-free starting locations. Each simulation instance ran for 10,000 iterations and, after finishing, the empirical stability margin was recorded.

The aggregated results from all the simulations can be seen in Figure 6. Figure 6 shows the ranking of the policies versus the number of instances that the policy achieved a ranking. A first place ranking corresponds to the policy producing the largest empirical stability margin at the end of the simulation instance. The simulation data shows that the most effective

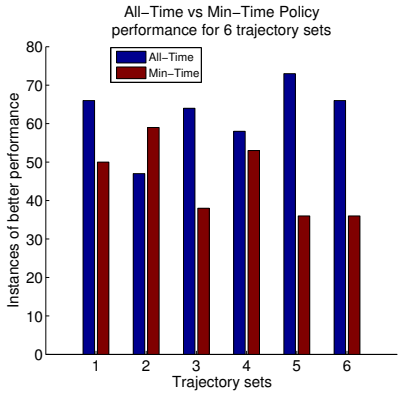


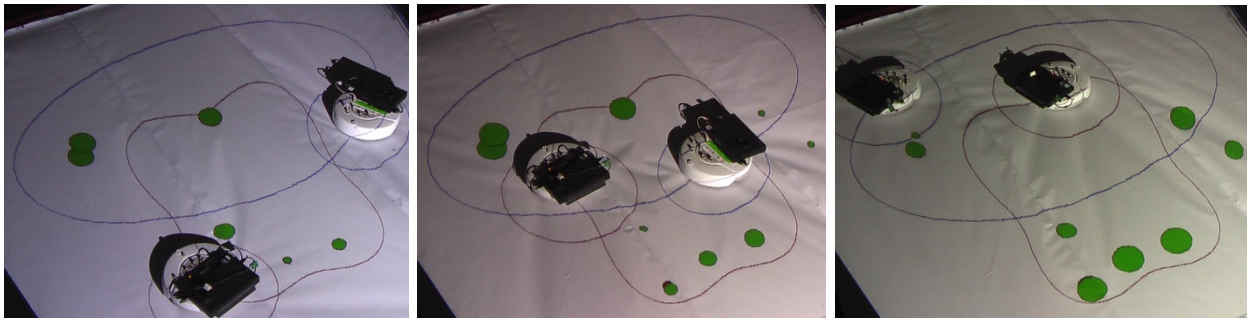
Fig. 7: Head-to-head results from the *All-Time* and *Min-Time* policies. The horizontal axis corresponds to the trajectory set number, and the vertical axis corresponds to the number of instances that the policy generated the better performance between the two.

policy is the *All-Time* policy, followed by the *Min-Time* policy. Figure 7 shows the trajectory set number (using the same reference as in Figure 5) versus the number of instances that the policy outperformed its counterpart for the *All-Time* and *Min-Time* policies. The data shows that the overall best performance is achieved by the *All-Time* policy, but there is one trajectory where the *Min-Time* policy outperforms it. This shows that the geometry of the trajectories can strongly affect the performance of the stopping policies. We plan to investigate this further in our future work.

In summary, six trajectory sets were simulated for five different stopping policies and for 100 different sets of points of interest. In total, 3,000 instances of the system were simulated, and 100% of the tested instances were free of collision and deadlock, which verifies the correctness of our collision avoidance procedure.

B. Distributed Implementation

We implemented the persistent monitoring task with collision avoidance on a multi-robot system consisting of two iRobot Create robots. Algorithm 2 used safety disks with ρ_r equal to 1.1 times the radius of the robot, and it used the *All-Time* stopping policy. Figure 8 shows three snapshots of the evolution of the system in the implementation. This implementation was executed in a distributed way. Each robot only knew information about itself, and communicated with the other robot when entering a collision zone in order to decide whether to continue its trajectory or stop to avoid collision. The robots tracked their paths with their speed profiles using a controller based on dynamic feedback linearization [23]. More than 20 experiments were executed, and all were collision-free and deadlock-free. In 20 trials of the setup shown in Figure 8, we observed that robot #1, which follows the ellipsoid trajectory, stopped an average of four times every 10 cycles. Robot #2 stopped an average of 5 times for every 1,000 cycles. Our video submission displays one of these trials.



(a) t = 5 seconds (b) t = 12 seconds (c) t = 34 seconds

Fig. 8: Snapshots at different times of a distributed implementation for the persistent monitoring task with collision avoidance for two robots. The points of interest are represented as green-filled circles, whose size is proportional to the value of the accumulation function $Z(\mathbf{q}, t)$ for each point $\mathbf{q}$. Each robot's footprint is represented by a concentric circle around the robot's location, and it is the same color as the trajectory that robot is following.

VIII. Conclusions and Future Directions

In this paper we presented a collision avoidance procedure for persistent tasks. The procedure was based on computing collision zones, and then ensuring that only one robot occupied a given collision zone at any moment in time. This was performed by stopping robots before they entered a collision zone, and resuming motion only once the zone was clear. We empirically investigated the performance of several different stopping policies, and determined that the *All-Time* policy resulted in the best stability margin. We also presented a distributed implementation on the iRobot Create platform.

For future work we are interested in analytically characterizing the performance of the different stopping policies. We are also looking into ways to tighten our analysis of the required nominal stability margin. Finally we are interested in investigating other application areas for our collision avoidance procedure, such as in traffic control, or automated material handling.

References

[1] S. L. Smith and D. Rus, "Multi-robot monitoring in dynamic environments with guaranteed currency of observations," in *Proc CDC*, Atlanta, GA, Dec. 2010, pp. 514–521.

[2] S. L. Smith, M. Schwager, and D. Rus, "Persistent robotic tasks: Monitoring and sweeping in changing environments," *IEEE Trans Robotics*, Jul. 2010, submitted. Available at `http://arxiv.org/abs/1102.0603`.

[3] H. Choset, "Coverage for robotics – A survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1-4, pp. 113–126, 2001.

[4] Y. Elmaliach, N. Agmon, and G. A. Kaminka, "Multi-robot area patrol under frequency constraints," in *Proc ICRA*, Roma, Italy, Apr. 2007, pp. 385–390.

[5] P. F. Hokayem, D. Stipanović, and M. W. Spong, "On persistent coverage control," in *Proc CDC*, New Orleans, LA, Dec. 2007, pp. 6130–6135.

[6] N. Nigram and I. Kroo, "Persistent surveillance using multiple unmannded air vehicles," in *IEEE Aerospace Conf.*, Big Sky, MT, May 2008, pp. 1–14.

[7] K. M. Lynch, I. B. Schwartz, P. Yang, and R. A. Freeman, "Decentralized environmental modeling by mobile sensor networks," *IEEE Trans Robotics*, vol. 24, no. 3, pp. 710–724, 2008.

[8] J. Cortés, "Distributed Kriged Kalman filter for spatial estimation," *IEEE Trans Automatic Ctrl*, vol. 54, no. 12, pp. 2816–2827, 2009.

[9] F. Zhang and N. E. Leonard, "Cooperative filters and control for cooperative exploration," *IEEE Trans Automatic Ctrl*, vol. 55, no. 3, pp. 650–663, 2010.

[10] J. L. Ny and G. J. Pappas, "On trajectory optimization for active sensing in gaussian process models," in *Proc CDC*, Shanghai, China, Dec. 2009, pp. 6282–6292.

[11] F. Bourgault, A. A. Makarenko, S. B. Williams, B. Grocholsky, and H. F. Durrant-Whyte, "Information based adaptive robotic exploration," in *Proc IROS*, Lausanne, Switzerland, Oct. 2002, pp. 540–545.

[12] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int J Robotic Research*, vol. 5, no. 1, pp. 90–98, 1986.

[13] J. Snape, J. van den Berg, S. J. Guy, and D. Manocha, "Independent navigation of multiple mobile robots with hybrid reciprocal velocity obstacles," in *Proc IROS*, St. Louis, MO, Oct. 2009, pp. 5917–5922.

[14] S. M. LaValle and S. A. Hutchinson, "Optimal motion planning for multiple robots having independent goals," *IEEE Trans Robotics & Automation*, vol. 14, no. 6, pp. 912–925, 1998.

[15] S. Akella and S. A. Hutchinson, "Coordinating the motions of multiple robots with specified trajectories," in *Proc ICRA*, Washington, DC, 2002, pp. 624–631.

[16] M. Jager and B. Nebel, "Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots," in *Proc IROS*, vol. 3, Maui, HI, Oct. 2001, pp. 1213 – 1219.

[17] T. Arai, H. Ogata, and T. Suzuki, "Collision avoidance amongts multiple robots using virtual impedance," in *Proc IROS*, Tsukuba, Japan, 1989, pp. 479–485.

[18] S. Qutub, R. Alami, and F. Ingrand, "Hot to solve deadlock situations within the plan-mergin paradigm for multi-robots cooperation," in *Proc IROS*, vol. 3, Grenoble, France, 1997, pp. 1610 – 1615.

[19] R. Alami, F. Ingrand, and S. Qutub, "A scheme for coordinating multi-robot planning activities and plans execution," in *Proc ECAI*, 1998.

[20] P. A. O'Donnell and T. Lozano-Pérez, "Deadlock-free and collision-free coordination of two robot manipulators," in *Proc ICRA*, vol. 1, Scottsdale, AZ, 1989, pp. 484–489.

[21] A. L. Schuote and P. J. Bouwens, "Deadlock-free traffic control with geometrical critical sections," in *Proc Computing Science in the Netherlands*, Amsterdam, 1994, pp. 260–270.

[22] Z. Butler and D. Rus, "Distributed planning and control for modular robots with unit-compressible modules," *Int J Robotic Research*, vol. 22, no. 9, pp. 699–715, 2003.

[23] G. Oriolo, A. D. Luca, and M. Vendittelli, "WMR control via dynamic feedback linearization: design, implementation, and experimental validation," *IEEE Trans Control Systems Technology*, vol. 10, no. 6, pp. 835–852, 2002.