Preprints of the 3rd IFAC Workshop on
Distributed Estimation and Control in Networked Systems
September 14-15, 2012. Santa Barbara, CA, USA

Decentralized Multirobot Control in Partially Known Environments with Dynamic Task Reassignment

Nora Ayanian * **Daniela Rus** * **Vijay Kumar** †

* *Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: {nayanian, rus}@mit.edu).*
† *University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail: kumar@grasp.upenn.edu).*

Abstract: This paper provides a decentralized solution to multirobot coordination in partially-known environments where the problems of task assignment, trajectory planning and safe control are concurrently solved in the presence of communication constraints. We assume that the robots are homogeneous (any robot is capable of completing any assigned task), can localize themselves and have access to a known map of the environment, except for obstacles and hazards that cannot be circumvented and must be detected through local sensing. Second, we assume that only robots that are within a specified, known communication range can communicate. A group of communicating robots is able to reassign tasks within the group to refine and improve the resulting solution in terms of total time or energy required for traversal. The key contribution of this paper is an approach that concurrently solves the three typically separated problems of task assignment, path planning, and control under constraints with proofs of completeness and convergence. We illustrate the application of these ideas through simulations and experiments with applications to surveillance of multiple destinations.

Keywords: Multirobot systems, multirobot coordination, robotics.

1. INTRODUCTION

In many applications, groups of robots must navigate complex uncertain environments to accomplish a task with only a partial map. In search and rescue or surveillance, groups of robots may be deployed to different parts of a building to search for victims or for intruders. They may encounter blocked passages and may have to replan their paths or negotiate with other robots to successfully accomplish their tasks. In many cases, the specific task or destination assigned to each robot may not be important, and robots can have the freedom to negotiate and reassign tasks/destinations as long as every task is accomplished.

In this paper, we present a decentralized solution to task reassignment, planning, and control for a team of homogeneous robots in a partially-known environment that may contain unknown hazards which fundamentally affect the paths available to robots. Only robots within a specified communication range can share information about discovered hazards and can exchange destinations/tasks if they enable more efficient solutions (see later for precisely what this means). These decisions are made with only a map of the environment and locally available information, which is limited to the current positions and current task assignments of robots in the connected group, and any information they have collected about the environment.

Each of the three typically separate subproblems of task allocation, path planning, and feedback control are challenging in their own right, with vast fields of research. In general, centralized solutions to these subproblems for large teams of robots are expensive, even when addressed separately. Finding a solution for all three subproblems simultaneously, especially in the presence of uncertainty, is considerably more challenging.

⋆ Work supported by ONR MURI N00014-09-1-1031 and N00014-09-1051.

(a) (b)

Fig. 1. Benefits of task permutation. Filled shapes are robots, unfilled shapes are goal configurations, and the black shape represents the boundary. While both are optimal task assignments, (a) is infeasible but (b) is feasible.

It is not always possible to allocate tasks optimally without considering the robots' path to their goals. In Fig. 1, the number of cells traveled by all agents in both examples is 4, but the task on the left is infeasible: the circle robot cannot maneuver around the triangle robot to reach its goal. Concurrent task reassignment and path planning allows solution of otherwise infeasible problems, *e.g.* permuting the tasks of Fig. 1a to those of Fig. 1b. Even concurrent task allocation and path planning can result in failure without feedback control: wheel slip, model approximations, and noisy sensor errors can accumulate, preventing convergence. Teams of high-dimensional mobile robots, such as quadrotors, are modeled approximately to limit computational complexity. Feedback control compensates for these inaccuracies. Thus, task allocation and path planning are not sufficient; feedback control is critical for multirobot coordination.

In multirobot task allocation, market-based approaches are an attractive compromise between centralized and distributed approaches (Dias et al. (2006)). Representative task allocation algorithms can be seen in Gerkey and Mataric (2003); McLurkin and Yamins (2005); Golfarelli et al. (1997); Batalin and Sukhatme (2004). None of these address feedback control.

Once tasks are allocated, determining optimal collision free paths for multiple robots is exponential in the number of degrees of freedom [Hart et al. (1968)]. Approaches which plan

Table 1. Notation of primary importance.

Symbol	Meaning
a^i, a^j	Robots (also referred to as agents)
π_b	Permutation of task assignments
$\mathcal{P}_N$	Set of all permutations (Symmetric group on {1,2,...,N})
G	Graph
$\mathcal{G}_q$	Group
p^k, p^m	Polytope in free space
P_q^K, P_q^M	Polytope in group configuration space
c	Polytope in relative space of agents
$\mathcal{H}$	Set of free space polytopes containing hazards
$\mathcal{H}_q$	Set of hazardous polytopes known to group q

trajectories or discrete paths such as Carpin and Pagello (2002), Clark et al. (2003), and Wagner and Choset (2011) suffer a disadvantage against feedback controllers on real-world systems, since they cannot account for drift, disturbances, or robot dynamics. Additionally, Carpin and Pagello (2002) and Wagner and Choset (2011) are neither decentralized nor online.

Multirobot control has been addressed in many domains, among them flocking (Tanner et al. (2007)); formation control (Leonard and Fiorelli (2001); Olfati-Saber and Murray (2002)); and group navigation (Michael and Kumar (2008); Yang et al. (2008)); none of these address controller synthesis. Dimarogonas et al. (2006) address coordinating dynamic robots without robots knowing others' goal configurations, concurrently solving path planning and control, but this does not scale to large teams.

Our goal is to develop a decentralized approach to solving task allocation, path planning, and feedback control simultaneously. The solution is guaranteed to be correct, though suboptimal since the approach is decentralized. We start with an initial (even random) task assignment. Initially, all robots localize in the same nominal map of the environment; as they navigate, they update the map as they sense hazards. Robots within communication range are said to be connected and form a *group*. For all practical purposes, the control of a group can be centralized since robots within a group can exchange limited information: their individual positions and destinations, and obstacles and hazards not part of the original map to allow reassignment when beneficial. Groups locally solve the allocation, planning, and control problems concurrently, guaranteeing completion of each task. This approach is most effective at reducing computation for large teams of robots in environments much larger than the communication range; the larger the communication range, the more centralized the approach becomes. Key in this method is the development of a cost-to-goal function (which can be interpreted as a utility function) that incorporates information about path planning in the free space, local robot coordination, dynamics, and control.

2. PROBLEM FORMULATION

Consider a team of N agents $\mathcal{V}_A = \{a^i | i = 1, \dots, N\}$ in a shared constrained, bounded, partially known environment with N distinct tasks. The team has the configuration or state

$$\begin{aligned} \mathbf{x} &= [\mathbf{x}_1^{\mathrm{T}}\ \mathbf{x}_2^{\mathrm{T}}\ \cdots\ \mathbf{x}_N^{\mathrm{T}}]^{\mathrm{T}} \\ \mathbf{x}_i &= [x_i\ y_i\ z_i\ \cdots]^{\mathrm{T}} \in \mathbb{R}^d \end{aligned} \tag{1}$$

with the dynamics

$$\dot{\mathbf{x}}_i = \mathbf{u}_i,\ \mathbf{x}_i \in \mathbf{X}_i \subset \mathbb{R}^d,\ i = 1, \dots, N, \tag{2}$$

if the agents are kinematic or

$$\ddot{\mathbf{x}}_i = \boldsymbol{\tau}_i,\ \mathbf{x}_i \in \mathbf{X}_i \subset \mathbb{R}^d,\ i = 1, \dots, N, \tag{3}$$

if the agents are dynamic. Initially, each agent is assigned a task (tasks correspond to achieving and maintaining a goal configuration). We denote the task assignment as a permutation $\pi \in \mathcal{P}_N$, where $\mathcal{P}_N$ is the symmetric group, or the set of all bijections of $\{1, \dots, N\}$ to itself. We allow all permutations of the tasks on the team of robots (i.e. any robot can complete any task), therefore the switched system dynamics are

$$\dot{\mathbf{x}} = f_\pi(\mathbf{x}), \tag{4}$$

for a kinematic system and

$$\ddot{\mathbf{x}} = h_\pi(\mathbf{x}, \dot{\mathbf{x}}), \tag{5}$$

for a dynamic system, where f_π (resp. h_π) refers to a distinct piecewise smooth control input for each permutation [1]. Each permutation has a distinct equilibrium $\mathbf{x}_{\pi,des}$ and the set of these equilibria is denoted

$$\mathcal{X}_{des} = \{\mathbf{x}_{\pi,des} \mid \pi \in \mathcal{P}_N\}.$$

We seek a finite sequence of switches that will guarantee that the system will achieve a configuration in $\mathcal{X}_{des}$.

Problem 1. For some initial state $\mathbf{x}_0$ and task permutation π_0, consider system (4) (respectively (5) on a dynamic system) on $\mathbb{R}^{Nd}$, with permutations $\mathcal{P}_N$. Find a switching sequence

$$S = \pi_0; \quad (\pi_0, t_0), (\pi_1, t_1), (\pi_2, t_2), \cdots, (\pi_F, t_F). \tag{6}$$

where (π_b, t_b) means that the system evolves according to $\dot{\mathbf{x}} = f_{\pi_b}(\mathbf{x})$ (resp. $\ddot{\mathbf{x}} = h_{\pi_b}(\mathbf{x}, \dot{\mathbf{x}})$ on a dynamic system), for $t_b \le t < t_{b+1}$ and control inputs

$$\dot{\mathbf{x}} = f_{\pi_b}(\mathbf{x}),\ \ b = 0, \dots, F \tag{7}$$

for a kinematic system or

$$\ddot{\mathbf{x}} = h_{\pi_b}(\mathbf{x}, \dot{\mathbf{x}}),\ \ b = 0, \dots, F \tag{8}$$

for a dynamic system, such that $\mathbf{x}|_{t=t_{F+1}} \in \mathcal{X}_{des}$ for some t_{F+1}.

3. PRELIMINARIES

Definition 2. The *configuration space* $\mathcal{C}_i$ of each agent a_i is the set of all transformations of the agent. The *free space* $\mathcal{C}_i^{free}$ of a_i is the set of all transformations of a_i which do not intersect with obstacles in the configuration space.

In this work we assume:

(1) all agents are identical, thus they share the same configuration and free spaces, so that $\mathcal{C} \equiv \mathcal{C}_i,\ \mathcal{C}^{free} \equiv \mathcal{C}_i^{free}\ \forall i$;
(2) $\mathcal{C}^{free}$ has been tessellated into convex, non-overlapping finite number of polytopes p^k with matching facets [2];
(3) static hazards are contained in a finite, *a priori* unknown set of polytopes $\mathcal{H}$, in which the robots cannot initialize.
(4) $\mathcal{C}^{free}$ is connected and hazards do not cause the environment to be disconnected.

The key step in defining a *group* of agents is constructing the *communication graph* on $\mathcal{C}^{free}$. Recall that a *graph* is a pair of sets $G = (\mathcal{V}, \mathcal{E})$, with nodes $\mathcal{V} = \{v_1, v_2, \dots\}$ and edges $\mathcal{E} \subseteq [\mathcal{V}]^2$. Pairs of nodes $(v_i, v_j) \in \mathcal{E}$ are called adjacent.

Communication is defined based on polytopes in $\mathcal{C}^{free}$. A pair of robots in a pair of polytopes (p^k, p^m) can communicate only if a robot at any position inside p^k can communicate with a robot at any position inside p^m. To impose a specific communication range, or vision-based communication, $\mathcal{C}^{free}$ can be easily subdivided. To ensure collision avoidance, the distance between pairs of polytopes in which communication is not allowed must be at least the diameter of the robot.

[1] In our notation for switched systems, we closely follow Branicky (1998).
[2] A hyperplane supporting adjacent polytopes shares the same vertices in both.

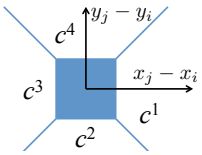


Fig. 2. Collision constraints for 2-dimensional agents a_i and a_j.

Definition 3. The *communication graph* on the team of agents is the dynamic graph $G_N = (\mathcal{V}_A, \mathcal{E}_N)$ where $\mathcal{E}_N = \{(a^i, a^j) | \mathbf{x}_i \in p^k, \mathbf{x}_j \in p^m, (p^k, p^m)$ allows communication$\}$. We call pairs of agents $(i,j) \in \mathcal{E}_N$ *neighbors*.

Robots can also relay information from their neighbors.

Definition 4. The *information graph* on the team of agents is the dynamic graph $G_I = (\mathcal{V}_A, \mathcal{E}_I)$ where $\mathcal{E}_I$ are pairs of agents connected on the communication graph [3] .

Definition 5. A *group* of agents $\mathcal{G}_q$ is a subset of the team which forms a complete subgraph on the information graph, i.e. if $(a_i, a_j) \in \mathcal{E}_I$ and $a_i \in \mathcal{G}_q$, then $a_j \in \mathcal{G}_q$.

As the team evolves, groups form and dissolve according to the agents' locations. Each group acts in a centralized manner independently of other groups (thus, larger communication distances correspond to a higher degree of centralization). When the information graph changes, the new group(s) share their task locations and known hazards, then evaluate if permuting tasks is beneficial. This occurs in the group configuration space.

Definition 6. A *group configuration space* is the Cartesian product of the free spaces of each agent in a group,

$$\begin{aligned} \mathcal{C}_q &= \prod_{1}^{|\mathcal{G}_q|} (\mathcal{C}^{free} \backslash \mathcal{H}_q), \;\; \mathcal{H}_q \subset \mathcal{H} \\ \mathbf{x}_q &= [\mathbf{x}_{q_1}^{\mathrm{T}} \; \mathbf{x}_{q_2}^{\mathrm{T}} \; \cdots \; \mathbf{x}_{q_{|\mathcal{G}_q|}}^{\mathrm{T}}]^{\mathrm{T}} \in \mathcal{C}_{\mathcal{G}_q}, \end{aligned} \tag{9}$$

where $\mathcal{H}_q$ is the dynamic set of polytopes known to the group to contain hazards, $|\mathcal{G}_q|$ the number of agents in the group $\mathcal{G}_q$, and $q_1, q_2, \ldots, q_{|\mathcal{G}_q|}$ the indices of the agents in $\mathcal{G}_q$.

We impose a collision constraint on pairs of agents in a group:

$$\lambda(\mathbf{x}_i, \mathbf{x}_j) = |\mathbf{x}_i - \mathbf{x}_j|_\infty - \delta \geq 0 \;\; \forall (a_i, a_j) \in \mathcal{E}_I, \tag{10}$$

which corresponds to an infinite annulus which we tesselate into unbounded convex polytopes $\{c^1, c^2, \ldots\}$ as in Fig. 2.

We guarantee collision avoidance for robots within a single group by the control algorithm we present in the following section. For robots not in a single group, by the tessellation and communication rules on $\mathcal{C}^{free}$, they cannot collide.

Definition 7. The *group task configuration space* $\mathcal{C}_q^T$ is the set

$$\begin{aligned} \mathcal{C}_q^T &= \mathcal{C}_q \cap \mathcal{L} \\ \mathcal{L} &\equiv \{\mathbf{x} | \mathbf{x} \in \mathcal{C}_q, \lambda(\mathbf{x}_i, \mathbf{x}_j) \geq 0 \, \forall (a_i, a_j) \in \mathcal{E}_I\}. \end{aligned} \tag{11}$$

$\mathcal{C}_q^T$ is a space composed of polytopes P_q^K in which the group cannot collide with each other or hazards known to the group.

4. SWITCHING POLICY

When a new group is formed, the group evaluates their task assignments to determine if permuting tasks would result in a lower cost-to-goal. Note that agents in a group can only permute tasks within the group. We define the cost-to-goal of a task permutation as the sum of the costs of the paths each agent must take to their goal on the *local polytope graph* which is a dynamic local version of the *lower polytope graph*.

Definition 8. The *lower polytope graph* is a weighted graph $G^p = (\mathcal{V}^p, \mathcal{E}^p)$ on the polytopes $\mathcal{V}_p = \{p^1, p^2, \ldots\}$ in $\mathcal{C}^{free}$, where $\mathcal{E}^p = \{(k,m) | p^k$ shares a facet with $p^m, \; \forall k, m\}$. Each edge $(k,m) \in \mathcal{E}^p$ is associated with a weight $w^{k,m} > 0$.

All agents initialize with identical maps of the lower polytope graph, but each agent maintains its own dynamic local map of the environment. Robots sense hazards in adjacent polytopes and within a specified distance; when an agent discovers a hazard, each agent in that group updates its local list of hazards $\mathcal{H}_i^{loc} \subseteq \mathcal{H}$, removing the associated edges from its *local polytope graph*. When groups form, they synchronize graphs.

Definition 9. The *local polytope graph* for robot a_i is a weighted dynamic graph $G_i^p = (\mathcal{V}^p, \mathcal{E}_i^p)$ on the polytopes $\mathcal{V}_p$. $\mathcal{E}_i^p = \{(k,m) | p^k$ is adjacent to $p^m, k, m \notin \mathcal{H}_i^{loc}, \; \forall k, m\}$ is the set of transitions in $\mathcal{C}^{free}$ which are perceived to be allowed. Each edge $(k,m) \in \mathcal{E}^p$ is associated with the weight $w^{k,m} > 0$.

We estimate the cost-to-go as the sum of the costs of the shortest paths for each agent without considering other robots, $\sum_{i=1}^N \psi_i$, where ψ_i is a minimal cost path from $\mathbf{x}_i$ to $\mathbf{x}_{g_i}$ on G^p. The actual cost can be larger, e.g. if an agent must exit a polytope to allow another agent through then return, and can only be determined by searching the joint configuration space of the group. To this end, we build a discrete representation of $\mathcal{C}_q^T$.

Definition 10. The *group polytope graph* is a weighted graph $G_q^P = (\mathcal{V}_q^P, \mathcal{E}_q^P)$ on the polytopes in $\mathcal{C}_q^T$, where $\mathcal{V}_q^P = \{P_q^1, P_q^2, \ldots\}$, P_q^K is the K-th polytope in $\mathcal{C}_q^T$, and $\mathcal{E}_q^P = \{(K,M) | P_q^K$ is adjacent to $P_q^M, \; \forall K, M\}$, the set of all pairs of polytopes which share a (matching) facet. Each edge $(K,M) \in E_q^P$ is associated with a weight $w_q^{K,M} > 0$.

Each group computes a path in the group configuration space for all members, but the actual cost-to-goal depends on polytopes visited in $\mathcal{C}^{free}$. To guarantee the path chosen in the group configuration space is no worse than the path in the physical space corresponding to the current permutation, we penalize changes in the relative position space (robots must not run circles around each other) but make the penalty small enough that the optimal solution in the joint space corresponds to the optimal solution in the physical space. We set weights for edges corresponding to changes in relative space such that

$$0 < w_q^{K,M} < \frac{\min_{k,m} w^{k,m}}{2^{d\binom{|\mathcal{G}_q|}{2}}}, \;\; \forall (K,M) \in [\mathcal{E}_q^P]_{relative}, \tag{12}$$

where $2^{d\binom{|\mathcal{G}_q|}{2}}$ is the maximum number of relative space polytopes for $|\mathcal{G}_q|$ agents in $\mathbb{R}^d$. In Fig. 3c, the weight of the edge connecting P_q^1 and P_q^2, corresponding to change in relative space constraints, would have a much smaller weight than that connecting P_q^1 and P_q^3, corresponding to change in free space polytopes. Space constraints limit further discussing this result.

Problem 11. For group $\mathcal{G}_q$ in polytope $P_q^K \in \mathcal{V}_q^P$, find a task permutation π_b and corresponding path $\mathcal{K} = (P_q^K, \ldots, P_q^{\pi_b})$ on the group polytope graph G_q^P, where $P_q^{\pi_b}$ is the goal node corresponding to π_b, such that

$$\pi_b = \arg\min_{\pi} \sum_{\mathcal{K}} w_q^{K,K+1}.$$

Each time a new group $\mathcal{G}_q$ is formed, Problem 11 is solved using the A* algorithm presented in Ayanian et al. (2011) with the

[3] Recall agents a_i and a_j are connected if G_N contains a path from a_i to a_j.

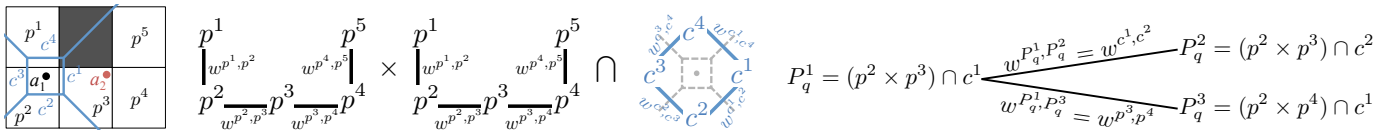


Fig. 3. Constructing a 2-agent group polytope graph. (a) $\mathcal{C}^{free}$ with two robots and collision constraints. (b) Cartesian product of lower polytope graphs intersected with collision constraints. (c) Using (b), we construct polytopes and determine adjacency.

key difference of having multiple possible goal states. The algorithm simultaneously constructs and searches the group task configuration space $\mathcal{C}_q^T$ and polytope graph G_q^P to concurrently find the task permutation with the lowest cost-to-goal and the path to the corresponding goal configuration, while guaranteeing the existence of a feedback controller. The solution to Problem 11 may be a set of permutations $\mathcal{P}$; the switching policy addresses which permutation is chosen.

Algorithm 12. (Switching Policy). Let a group $\mathcal{G}_q$ be formed at time t_b with current task permutation π_{b-1}, and the solution set to Problem 11 be $\mathcal{P}$. There are two classes of outcome.

(1) If $\pi_{b-1} \in \mathcal{P}$, do nothing.
(2) If $\pi_{b-1} \notin \mathcal{P}$, choose the first optimal permutation π_b.

5. NAVIGATION FUNCTIONS AND CONVERGENCE

The solution to Problem 11 in conjunction with Algorithm 12 define a path in G_q^P. This path is then translated into local navigation functions on each pair of sequential polytopes on the path using the approach presented in Ayanian et al. (2011). These functions, used to drive the system to the goal, also determine the Lyapunov function of each task permutation.

Definition 13. (Rimon and Koditschek (1992)). Let $\mathcal{Q}$ be an n-dimensional compact, simply connected manifold with boundary and let $\mathbf{x}_{q,g} \in \mathcal{Q}$ be a unique point. A function $\varphi : \mathcal{Q} \mapsto [0,1]$ is a *navigation function* if is twice differentiable on $\mathcal{Q}$, achieves a unique minimum of 0 at $\mathbf{q}_g \in \mathcal{Q}$, is uniformly maximal (i.e. evaluates to 1) on the boundary, and is Morse.

We use the navigation function to generate a feedback controller to drive the system to the goal while staying inside manifold $\mathcal{Q}$. The control law for the kinematic system (2) is

$$\mathbf{u} = -\nabla\varphi(\mathbf{x})$$

and convergence can be shown using the Lyapunov function

$$V(\mathbf{x}) = \varphi(\mathbf{x}). \tag{13}$$

For the dynamic system (3) the control law is given by $\boldsymbol{\tau}$ and convergence shown by $V(\mathbf{x}, \dot{\mathbf{x}})$, where

$$\boldsymbol{\tau} = -\nabla\varphi(\mathbf{x}) - \Gamma\dot{\mathbf{x}}, \;\; \Gamma \in \mathbb{R}^{Nd\times Nd} \succ 0 \tag{14}$$

$$V(\mathbf{x}, \dot{\mathbf{x}}) = \varphi(\mathbf{x}) + 0.5\,\dot{\mathbf{x}}^{\mathrm{T}}\dot{\mathbf{x}}. \tag{15}$$

At time t, let the task permutation of the group be given by π_b with corresponding path $\mathcal{K}_b = (P_q^1, P_q^2, \ldots)$. The navigation function constructed in Ayanian et al. (2011) drives any state in P_q^1 to a local goal in P_q^2 where the function achieves the minimum 0. This continues through P_q^3, etc, until a robot enters or exits the group, or the goal is reached. Since navigation functions are synthesized in obstacle-free polytopes, the resulting field is free of local minima and saddle points. For each permutation π_b, the system converges to the corresponding equilibrium $\mathbf{x}_{\pi_b,des}$, satisfying all tasks.

To show convergence of the system, it is meaningful to define a *Lyapunov-like* function in the sense of Branicky (1998).

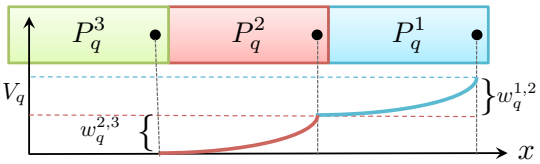


Fig. 4. Illustration of Lyapunov-like function for a group with path (P_q^1, P_q^2, P_q^3). Black dots represent local goals.

Definition 14. Given a strictly increasing sequence of switching times $T = t_0, t_1, t_2, \ldots$, we say V_{π_b} is *Lyapunov-like* for (7) (or (8)) over T if:

- $\dot{V}_{\pi_b}(\mathbf{x}) \leq 0$ for $\bigcup_{j\in\mathbb{Z}^+}[t_{2j}, t_{2j+1}]$;
- V_{π_b} is monotonically non increasing on the even sequence of $T : t_0, t_2, t_4, \ldots$.

It is quite straightforward to develop such a function. For example, let the path to a goal for a group be (P_q^1, P_q^2, P_q^3). We can generate an every (or utility) function by scaling the navigation function in each polytope P_q^K by the weight of the corresponding edge in the group polytope graph, starting with 0 at the goal. Then the current value of the Lyapunov-like function for this goal permutation would be $V_{\pi_b} = w_q^{2,3} + \gamma w_q^{1,2}$, where $0 \leq \gamma \leq 1$ (see Fig. 4). This function meets the criteria of *Lyapunov-like* for all permutations $\pi_b \in \mathcal{P}_N$, and is sufficient to show stability and convergence. First, we use a theorem from Branicky (1998) to show stability.

Theorem 15. (Branicky (1998)). Suppose we have candidate Lyapunov functions V_{π_b}, $\pi_b \in \mathcal{P}_N$ and vector fields $\dot{\mathbf{x}} = f_{\pi_b}(\mathbf{x})$ (resp. $\ddot{\mathbf{x}} = h_{\pi_b}(\mathbf{x}, \dot{\mathbf{x}})$) with $f_{\pi_b}(0) = 0$ (resp. $h_{\pi_b}(0) = 0$) $\forall\pi$. Let $\mathcal{S}$ be the set of all switching sequences associated with the system. If for each $S \in \mathcal{S}$ we have that for all π_b, V_{π_b} is Lyapunov-like for f_{π_b} (resp. h_{π_b}) and the resulting trajectory $\mathbf{x}_S(\cdot)$ over the set of times the permutation π_b is active, then the system is stable in the sense of Lyapunov.

Proof. This is a straightforward application of Branicky (1998).

While the assumption in Branicky (1998) is that $V(\mathbf{0}) = 0$, the system can be shifted so the goal configurations are the origin.

5.1 Convergence in a Perfectly Known Environment

We have shown that the system is stable, but convergence can only occur if a goal is reachable.

Corollary 16. There exists at least one task permutation for which the goal is reachable.

Proof. The environment is connected, and hazards do not disconnect the environment. Therefore, each robot can reach each goal if there are no other robots in the space. If any robots are blocking another from reaching its goal according to the current permutation, then since we allow all permutations, the robots can permute goals to find a reachable solution.

The switching policy (Alg. 12) ensures the system converges.

Theorem 17. The system will converge to some $\mathbf{x}_{\pi,des} \in \mathcal{X}_{des}$.

Proof. Although the exact value of V_{π_b} is unknown, the Alg. 12 ensures the system only either (1) keeps the current permutation or (2) switches to one with a lower maximum Lyapunov value.

(1) *Maintain current permutation*: If the Alg. 12 maintains the current permutation, the system's Lyapunov function continues to monotonically decrease. This is not a switch.

(2) *Switch to new permutation*: Let the current cell be P_q^1 and the new permutation be π_b. If the switching policy results in a switch, then π_b has a lower maximum value in the current polytope than the previous permutation π_{b-1}. Trivially, a decrease in $V_q(\mathbf{x})$, the utility function for a group, results in a decrease of $V(\mathbf{x})$, the utility across the team. We cannot be certain that $V_q(\mathbf{x}_q|_{t_b}, \pi_b) \leq V_q(\mathbf{x}_q|_{t_b}, \pi_{b-1})$. In other words, the actual Lyapunov value may increase, although the maximum attainable value in the polytope decreases, $\max_{\mathbf{x}_q \in P_q^1} V_q(\mathbf{x}_q, \pi_b) < \max_{\mathbf{x}_q \in P_q^1} V_q(\mathbf{x}_q, \pi_{b-1})$. Since switching occurs only when an interface is crossed, we know that at the next time t_{b+1} when switching may be considered, the value V will have decreased. Therefore, should system π_{b-1} be reactivated, it will be activated at a lower value than it was activated previously. Since $V_q(\mathbf{x}_q, \pi_{b-1})$ is monotonically decreasing at switching times, the system will converge.

5.2 Stability in a Partially Known Environment

In an environment with unknown blockages the cost of traveling between two points is unknown, thus evaluating the true cost-to-goal of a permutation is impossible. The robots execute the same switching policy (Algorithm 12), using the local polytope graph as an estimate of the environment's state. Unknown hazards may cause poor decisions that increase the value of the actual Lyapunov function. However, since the environment is bounded and the number of hazards is finite, once all hazards are discovered by all robots (sufficient but not necessary), the environment is known and Theorem 17 above applies.

6. SIMULATIONS AND EXPERIMENTS

We have successfully tested our approach in simulations and experiments; here we present selected illustrative examples.

6.1 SIMULATIONS

Our first representative simulation involves 15 kinematic robots in an urban environment, computed in MATLAB. Figure 5a shows the initial configuration and task assignment, as well as unknown hazards in yellow. Each robot is depicted by a distinct filled shape and color combination; goals are shown in corresponding unfilled shape and color. Fig. 5b depicts robot trajectories after the final permutation. In all, 16 permutation events occur, at times depicted by vertical lines in Fig. 5c, including an event at initiation. Dots indicate robots involved in an event. The time between events increases as the task evolves.

High-fidelity dynamic simulation of KUKA youBots is done using GAZEBO, ROS, and a ROS-MATLAB bridge (Michael (2011)); real-time computations and control are done in MATLAB. In Fig. 6, four youBots in an environment with a hazard (yellow) plan their path without knowledge of the others; the hazard is initially only known to robot P (purple, dotted line). Robot B proceeds through its planned path, discovers the hazard and replans (Fig. 6b). When robots R, G, and B form a group, they permute tasks (Fig. 6c). Similarly, R and B permute tasks (Fig. 6d), driving R again towards G, which again permute (Fig. 6e), then all agents proceed to their task locations (Fig. 6f).

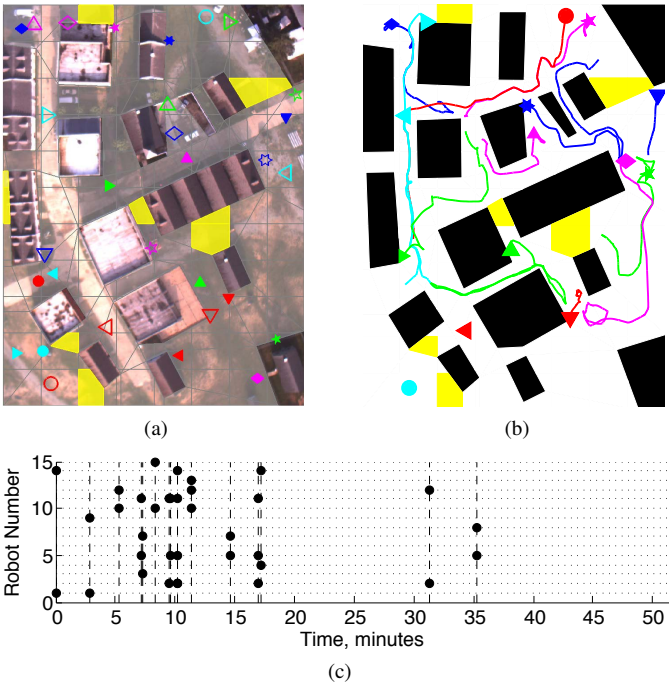


Fig. 5. A simulation with 15 robots in an urban environment. (a) Each robot is shown as a distinct filled shape and color combination at its initial position, with its initial task shown as the corresponding unfilled shape and color. Unknown hazards in yellow. (b) Robot pose elapsed since last permutation. Hazards discovered by at least one robot shown in yellow. (c) Number of switches by robot shown as dots against time, vertical lines indicate switching events. Note the early concentration of switching.

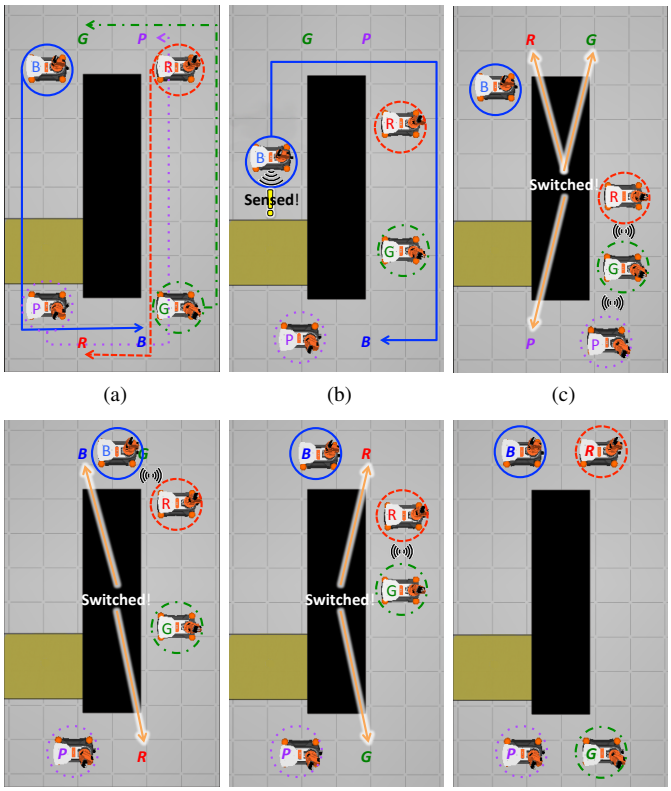
6.2 Experiment

Here we use VICON motion capture, with real-time MATLAB computation on a single laptop although the algorithm is distributed. Four youBots with goals on antipodal corners of the environment initially plan without knowledge of the other robots or the hazard shown in yellow (Fig. 7a). After multiple encounters and permutations, the initial configuration becomes the goal configuration. Note the jagged edges, especially for the red circle robot. Partial VICON failure caused poor tracking, but the controller was robust to compensate. The cyan triangle appears out of bounds, but this is a string of incorrect readings.

7. CONCLUDING REMARKS

The central contribution of this paper is a decentralized approach to coordinating a team of homogeneous robots with local communication connectivity in a partially-known environment. This approach concurrently solves task assignment, path planning, and feedback control, thus guaranteeing convergence on real robots in environments with unknown hazards.

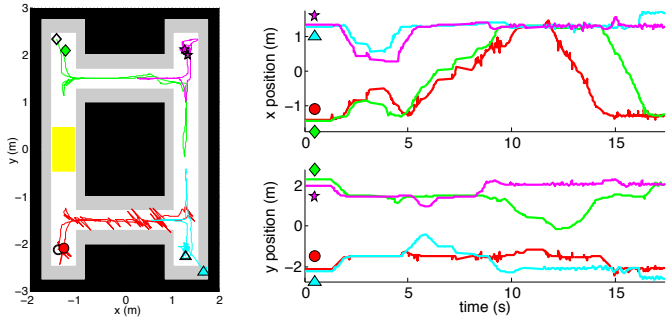
Critical to our approach is the ability to permute tasks, starting from a suboptimal or even random task assignment policy, based on estimated collective distance travelled. Task permutations enable decentralization while avoiding the deadlock and livelock scenarios that plague decentralized approaches. Furthermore, task permutations allow efficient handling of unknown obstacles in the environment. While scenarios we present here remove a polytope from the workspace if it contains a hazard, this is easily extended to more complex scenarios, where the geometry of the space has changed, or when



(a) (b) (c)

(d) (e) (f)

Fig. 6. Four robot simulation in GAZEBO. (a) Initial positions and planned paths. The hazard is only known to robot P (purple, dotted line). (b) Robot B discovers the hazard along its path and must replan. (c) Robots R, G, P interact and permute tasks. (d) Robots R, B permute tasks. (e) Robots R and G interact again and permute their task. (f) All tasks have been accomplished.



(a) (b)

Fig. 7. Experiment with 4 youBots. (a) Robot trajectories. Obstacles are padded (grey) for the robot's size. Network and sensor errors result in erroneous data, but the controller is robust enough to complete the task. (b) x-, y-pose vs. time.

there are partial blocks within a polytope, if robots are capable of remapping the affected area. In the case of a perfectly known environment, the switching policy ensures the task assignment progressively improves with each permutation. For partially-known environments, the algorithm relies on all changes in the environment being observed and communicated for completeness and convergence.

An important limitation to our approach is centralization within groups, which can be difficult in a tightly constrained environment, leading to inefficiencies due to repeated task permutation. However, the method enables distributed computation and control, and overcomes the difficulties with scaling to larger numbers that centralized approaches cannot circumvent.

REFERENCES

Ayanian, N., Kallem, V., and Kumar, V. (2011). Synthesis of feedback controllers for multiple aerial robots with geometric constraints. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 3126–3131. San Francisco, CA.

Batalin, M. and Sukhatme, G. (2004). Using a sensor network for distributed multi-robot task allocation. In *Proc. IEEE Int. Conf. Robot. Autom.*, volume 1, 158 – 164.

Branicky, M. (1998). Multiple lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Trans. Autom. Control*, 43(4), 475 –482.

Carpin, S. and Pagello, E. (2002). On parallel RRTs for multi-robot systems. In *Proc. 8th Conf. Italian Association for Artificial Intelligence*, 834–841.

Clark, C., Rock, S., and Latombe, J.C. (2003). Motion planning for multiple mobile robots using dynamic networks. In *Proc. IEEE Int. Conf. Robot. Autom.*, volume 3, 4222 – 4227.

Dias, M., Zlot, R., Kalra, N., and Stentz, A. (2006). Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7), 1257 –1270.

Dimarogonas, D.V., Loizou, S.G., Kyriakopoulos, K.J., and Zavlanos, M.M. (2006). A feedback stabilization and collision avoidance scheme for multiple independent non-point agents. *Automatica*, 42(2), 229 – 243.

Gerkey, B. and Mataric, M. (2003). Multi-robot task allocation: analyzing the complexity and optimality of key architectures. In *IEEE Intl. Conf. Robot. Autom.*, volume 3, 3862–3868.

Golfarelli, M., Maio, D., and Rizzi, S. (1997). A task-swap negotiation protocol based on the contract net paradigm. Technical Report 005-97, CSITE.

Hart, P., Nilsson, N., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.*, 4(2), 100 –107.

Leonard, N. and Fiorelli, E. (2001). Virtual leaders, artificial potentials and coordinated control of groups. In *Proc. IEEE Conf. Dec. Contr.*, volume 3, 2968–2973.

McLurkin, J. and Yamins, D. (2005). Dynamic task assignment in robot swarms. In *Robotics: Science and Systems*.

Michael, N. (2011). Ros-matlab bridge. `http://github.com/nmichael/ipc-bridge`.

Michael, N. and Kumar, V. (2008). Controlling shapes of ensembles of robots of finite size with nonholonomic constraints. In *Proc. Robotics: Science and Systems*, 157–162.

Olfati-Saber, R. and Murray, R. (2002). Distributed cooperative control of multiple vehicle formations using structural potential functions. In *Proc. of IFAC World Congress*. Barcelona.

Rimon, E. and Koditschek, D.E. (1992). Exact robot navigation using artificial potential fields. *IEEE Trans. Robot. Autom.*, 8(5), 501–518.

Tanner, H.G., Jadbabaie, A., and Pappas, G.J. (2007). Flocking in fixed and switching networks. *IEEE Trans. Automat. Contr.*, 52(5), 863–868.

Wagner, G. and Choset, H. (2011). M*: A complete multirobot path planning algorithm with performance bounds. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*

Yang, P., Freeman, R., and Lynch, K. (2008). Multi-agent coordination by decentralized estimation and control. *IEEE Trans. Automat. Contr.*, 53(11), 2480–2496.