2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

Sensor-Based Dynamic Assignment in Distributed Motion Planning

Michael M. Zavlanos and George J. Pappas

Abstract— **Distributed motion planning of multiple agents raises fundamental and novel problems in control theory and robotics. Recently, one such great challenge has been the development of motion planning algorithms that dynamically assign targets or destinations to multiple homogeneous agents, not relying on any a priori assignment of agents to destinations. In this paper, we address this challenge using two novel ideas. First, we develop distributed multi-destination potential fields able to drive every agent to any available destination for almost all initial conditions. Second, we propose sensor-based coordination protocols that ensure that distinct agents are assigned to distinct destinations. Integration of the overall system results in a distributed, multi-agent, hybrid system for which we show that the mutual exclusion property of the final assignment is guaranteed for almost all initial conditions. Moreover, we show that our dynamic assignment algorithm converges after exploring at most a polynomial number of assignments, dramatically reducing the combinatorial nature of purely discrete assignment problems. Our scalable approach is illustrated with nontrivial computer simulations.**

I. Introduction

Given any multi-agent motion planning task, where the agents have to reach a desired configuration from any initial configuration [1], the *assignment problem* consists of determining an assignment between the agents and the destinations. If no a priori assignment information is provided, then the assignment has to be determined on-line. Moreover, if only local sensing information is available, then the resulting control framework is fully distributed.

Assignment problems are fundamental in combinatorial optimization and, roughly, consist of finding a minimum weight matching in a weighted bipartite graph. They arise frequently in various research areas, as diverse as operations research, computer vision and distributed robotics − formation stabilization and consensus seeking − where graphs are recently emerging as a natural mathematical description for capturing interconnection topology [2] − [12]. Depending on the form of the cost function, assignment problems can be classified as linear or quadratic. Optimal solutions to the linear assignment problem can be computed in polynomial time using the Hungarian algorithm [13], while the quadratic assignment problem is NP-hard [14] and suboptimal solutions are achieved by means of various relaxations [15], [16].

In distributed robotics, approaches to the assignment problem can be either on-line, if the assignment is determined dynamically, or off-line, if an assignment is computed a priori. An on-line approach to the assignment problem is proposed in [17], where the space of permutation invariant multi-robot formations is represented using complex polynomials whose roots correspond to the unassigned configurations of the robots in the formation. Since, the polynomial coefficients are invariant under permutation of the roots, the representation of the formation is invariant with respect to different robot-configuration assignments. The proposed approach is open loop and centralized, since it requires global knowledge of the environment. On the other hand, an off-line approach is proposed in [18], where the authors develop a polynomial time algorithm that computes a suboptimal assignment between agents and destinations based on a "minimum distance to the goal" policy. Any navigation strategy can then be used to drive the agents to their destinations.

In this paper we propose a fully distributed control framework that provides an on-line solution to the multi-agent assignment problem. Assuming that all agents have initially knowledge of all available destinations, the main idea behind our approach is to let every agent explore a sequence of destinations and eventually be assigned to the first one that is available. No communication among the agents is assumed, hence the requirement that no two agents may be assigned to the same destination is ensured by sensor-based coordination protocols. On the other hand, the sequence of destinations to be explored by each agent is determined dynamically through provably correct distributed multi-destination potential fields that can drive every agent from almost all initial configurations to any destination in the configuration space. Integration of the overall system results in a distributed multi-agent hybrid system that is shown to almost always converge to an assignment having the mutual exclusion property and have at most polynomial complexity, despite the exponential growth of the number of assignments with respect to the number of agents. The efficiency of our scalable approach is finally illustrated through nontrivial computer simulations.

The rest of this paper is organized as follows. In Section II we develop a general framework for the dynamic assignment problem. In Section III, we state the modeling assumptions and define the hybrid automata that consist the agents' models. In Section IV, we construct the aforementioned multi-destination potential fields and study their convergence properties. The integrated system is studied in Section V, where results about its complexity and equilibrium modes are also discussed. Finally, in Section VI, we state and verify through computer simulations, nontrivial assignment tasks that illustrate the efficiency of our approach.

This work is partially supported by ARO MURI SWARMS Grant W911NF-05-1-0219 and the NSF ITR Grant 0324977.

Michael M. Zavlanos and George J. Pappas are with GRASP Laboratory, Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104, USA {zavlanos,pappasg}@grasp.upenn.edu

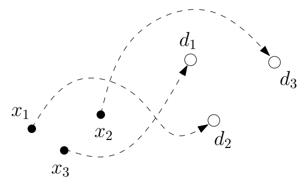


Fig. 1. Multi-agent motion planning for $n = 3$ agents and $m = 3$ destinations resulting in one out of 6 possible assignments.

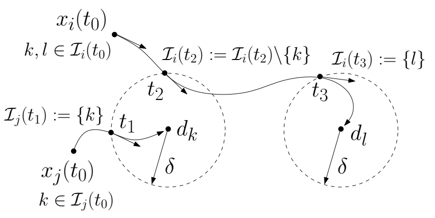

$x_i(t_0)$
$k, l \in \mathcal{I}_i(t_0)$
$\mathcal{I}_i(t_2) := \mathcal{I}_i(t_2) \backslash \{k\}$
$\mathcal{I}_i(t_3) := \{l\}$
t_2
t_3
$\mathcal{I}_j(t_1) := \{k\}$
t_1
d_k
d_l
δ
δ
$x_j(t_0)$
$k \in \mathcal{I}_j(t_0)$

Fig. 2. Example of sensor-based coordination.

II. Problem Formulation

Consider n point agents in a 2-dimensional space $\mathbb{R}^2$ and denote by $x_i(t) \in \mathbb{R}^2$ the coordinates of agent i at time t. We assume kinematic models for the agents and so,

$$\dot{x}_i(t) = u_i(t) \quad \forall\, i = 1, \ldots, n \tag{1}$$

where $u_i(t)$ is the control vector taking values in $\mathbb{R}^2$. Consider, further, $m \geq n$ destinations in $\mathbb{R}^2$ that the agents have to occupy and let $\mathcal{I}_0 = \{1, \ldots, m\}$ denote the index set corresponding to a fixed labeling of these destinations. We assume that each destination $k \in \mathcal{I}_0$ is uniquely associated to a coordinate vector $d_k \in \mathbb{R}^2$ through the injective map,

$$dest : \mathcal{I}_0 \to \mathbb{R}^2 \quad \text{with} \quad dest(k) := d_k, \ \forall\, k \in \mathcal{I}_0 \tag{2}$$

To simplify notation, we hereafter write d_k to refer to the injection $dest(k)$. The system of agents and destinations described above, gives rise to the *multi-agent motion planning* problem, which we define as follows.

Definition 2.1 (Multi-Agent Motion Planning): Given a set of n identical agents and $m \geq n$ destinations, derive control laws that drive each agent to a distinct destination.

Implicit in the motion planning problem defined above, is the *assignment problem*, namely, which of the $\binom{m}{n}n!$ possible assignments between agents and destinations should system (1) be driven to. Given that the agents are "identical" we consider any assignment equally desirable (Figure 1). A popular approach is to decouple the assignment and navigation subproblems in Definition 2.1, i.e., determine first an assignment between agents and destinations, which can be either random or optimal, based on a "minimum distance to the goal" policy [16], [18], and then design controllers that drive each agent to its destination [1]. Such approaches result in *centralized* and *off-line* control frameworks since, although navigation can be decentralized, an off-line centralized assignment decision needs to be made first. In this paper we propose a *dynamic* and completely *distributed* solution to the aforementioned problem. In particular, we assume that each agent has only knowledge of its available destinations, while the assignment decision is embedded in its controller and relies on local sensing information. We therefore, address the following motion planning problem.

Problem 1 (Dynamic Assignment): Given a set of n identical agents, $m \geq n$ destinations and no a priori assignment information, derive distributed control laws that drive every agent i, from any initial configuration $x_i(t_0)$, to a distinct destination $k \in \mathcal{I}_0$.

The main idea behind our approach to Problem 1 is to let every agent explore a sequence of destinations and eventually be assigned to the first one that is available. Implicit in this task is the *mutual exclusion* property of the final assignment, i.e., that no two agents can occupy the same destination. This property is guaranteed by sensor-based coordination protocols, that are developed in the following section, and enable the agents to sense whether a destination is taken or not. Then, in Section IV, multi-destination potential fields are developed that dynamically determine a sequence of destinations to be explored, while navigating the agents to each one of these destinations.

III. Distributed Coordination

Let $\mathcal{I}(t)$ and $\mathcal{I}^c(t)$ denote the index sets of available and taken destinations at time $t \geq t_0$, respectively. Clearly, $\mathcal{I}(t_0) = \mathcal{I}_0$, $\mathcal{I}^c(t_0) = \emptyset$ and $\mathcal{I}(t) \cap \mathcal{I}^c(t) = \emptyset$, $\mathcal{I}(t) \cup \mathcal{I}^c(t) = \mathcal{I}_0$ for all $t \geq t_0$. In a distributed setting, however, where the agents do not have access to the system's global variables, we require that every agent i is equipped with its own set of available destinations, denoted by $\mathcal{I}_i(t)$, such that initially $\mathcal{I}_i(t_0) = \mathcal{I}_0$. Moreover, we assume that no agent i has knowledge of the positions or the destination sets $\mathcal{I}_j(t)$, $j \neq i$ of the other agents. With the above notation, we now state the assumptions for our model.

Assumptions 3.1: Let $\mathcal{B}_r(x) = \{y \in \mathbb{R}^2 \mid \|y - x\|_2 < r\}$ denote an open ball of radius $r > 0$ centered at $x \in \mathbb{R}^2$. For every agent $i = 1, \ldots, n$ we assume that, for all time t,

(a) it can claim an available destination $k \in \mathcal{I}(t)$, if and only if $k \in \mathcal{I}_i(t)$, $|\mathcal{I}_i(t)| > 1$ and $x_i(t) \in \mathcal{B}_\delta(d_k)$,[1]
(b) there is a controller $u_i(x_i(t), \mathcal{I}_i)$ that, for any *fixed* index set $\mathcal{I}_i$, can drive it to any destination in $\mathcal{I}_i$.
(c) $\delta > 0$ is such that $\mathcal{B}_\delta(d_k) \cap \mathcal{B}_\delta(d_l) = \emptyset$ for all $k, l \in \mathcal{I}_0$.

We call $2\delta > 0$ the *sensing range* of the agents. Assumption 3.1(a) implies that a *necessary* and *sufficient* condition for agent i to be assigned to destination k is that $x_i(t) \in \mathcal{B}_\delta(d_k)$ and destination k is free, i.e., $k \in \mathcal{I}(t)$. Clearly, a sensing range 2δ or larger allows agent i to know whether there exists another agent $j \neq i$ with $x_j(t) \in \mathcal{B}_\delta(d_k)$, i.e., whether destination k is taken or free. Assumption 3.1(b), on the other hand, says that every agent is capable of navigating to any of its available destinations. Finally, Assumption 3.1(c) combined with Assumption 3.1(a) guarantees that each agent can only claim one destination at a time. The following example describes the main idea of our approach (Figure 2).

[1]We denote by $|\mathcal{A}|$ the cardinality of the set $\mathcal{A}$.

Example 3.2: Consider agents i and j initially located at $x_i(t_0)$ and $x_j(t_0)$ with available destination sets $\mathcal{I}_i(t_0)$ and $\mathcal{I}_j(t_0)$ respectively. Consider further destination points d_k and d_l such that, $k, l \in \mathcal{I}_i(t_0)$ and $k \in \mathcal{I}_j(t_0)$. Let time $t_1 > t_0$ be such that $x_j(t_1) \in \mathcal{B}_\delta(d_k)$ and assume that destination k is *free*. Then, agent j is assigned to destination k by setting $\mathcal{I}_j(t_1) := \{k\}$ and destination k is labeled *taken*. Let time $t_2 > t_1$ be such that $x_i(t_2) \in \mathcal{B}_\delta(d_k)$. Since destination k is already taken, agent i removes it from its available destinations and so $\mathcal{I}_i(t_2) := \mathcal{I}_i(t_2)\backslash\{k\}$. Under the control $u_i(x_i(t), \mathcal{I}_i(t_2))$ it navigates towards destination l, which is assumed to be free.

Having motivated our approach with Example 3.2, we can now formally define the distributed coordination scheme for the *dynamic assignment* problem. The proposed coordination scheme consists of hybrid models for both the agents and the destinations, which interact with each other and result in the desired behavior of the overall system, which is shown in Section V to have the desired *liveness* and *safety* properties.

A. Modeling the Agents and Destinations

In view of Example 3.2, we model every agent by an *agent automaton* that continuously updates its set of available destinations $\mathcal{I}_i(t)$ and determines dynamically a sequence of available destinations in $\mathcal{I}_i(t)$ to be explored. Destinations, on the other hand, are modeled by a *destination automaton* that consists of two modes corresponding to the destination states, *free* or *taken*. The following notion of a *predicate* enables us to formally define the aforementioned automata.

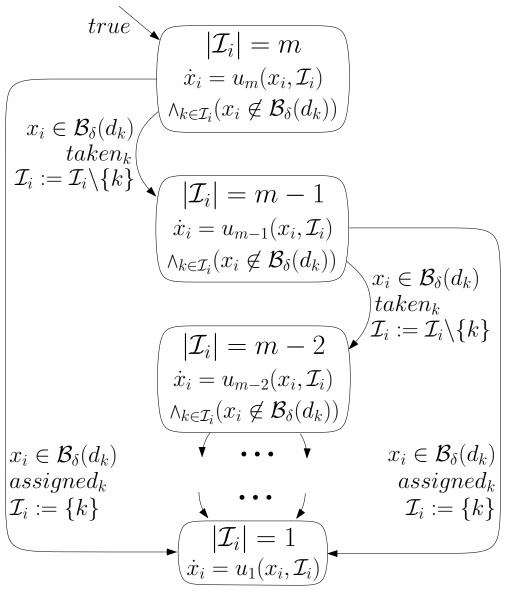
Definition 3.3 (Predicate): Let $X = \{x_1, \ldots, x_n\}$ be a finite set of variables. We define a predicate $\psi(X)$ over X to be a finite conjunction of strict or non-strict inequalities over X. We denote the set of all predicates over X by $Pred(X)$.

In other words, a predicate is a logical formula. For example, the predicate $\psi(X) = \big(\|x - x_0\|_2 < r\big)$ over the set of variables $X \in \mathbb{R}^N$ returns 1 if x belongs in the open ball $\|x - x_0\|_2 < r$ and 0 otherwise. Hence, the agent automaton of agent i can be defined as follows.[2]

Definition 3.4 (Agent Hybrid Automaton): We define the hybrid automaton of agent i to be the tuple $A_i = (X_{A_i}, V_{A_i}, E_{A_i}, \Sigma_{A_i}, sync, inv, init, guard, reset, flow)$, where,

- $X_{A_i} = \{x_i, \mathcal{I}_i\}$ denotes the set of state variables with $x_i \in \mathbb{R}^2$ and $\mathcal{I}_i \in 2^{\mathcal{I}_0}$.
- $V_{A_i} = \{1, \ldots, m\}$ denotes the finite set of control modes.
- $E_{A_i} = \{(v, v-1), (v, 1) \mid v \in V_{A_i}\backslash\{1\}\}$ denotes the set of control switches.
- $\Sigma_{A_i} = \{taken_k, assigned_k \mid k \in \mathcal{I}_0\}$ denotes the set of synchronization labels.
- $sync : E_{A_i} \to \Sigma_{A_i}$ with $sync\big((v, v-1)\big) = taken_k$ for $v \in V_{A_i}\backslash\{1, 2\}$ and $sync\big((v, 1)\big) = assigned_k$ for $v \in V_{A_i}\backslash\{1\}$ and all $k \in \mathcal{I}_0$ denotes the synchronization map mapping each control switch to a synchronization label.

[2]To simplify notation, we hereafter drop the dependence of the state variables x_i and $\mathcal{I}_i$ on time t.



true
$|\mathcal{I}_i| = m$
$\dot{x}_i = u_m(x_i, \mathcal{I}_i)$
$\wedge_{k\in\mathcal{I}_i}(x_i \notin \mathcal{B}_\delta(d_k))$
$x_i \in \mathcal{B}_\delta(d_k)$
$taken_k$
$\mathcal{I}_i := \mathcal{I}_i\backslash\{k\}$
$|\mathcal{I}_i| = m-1$
$\dot{x}_i = u_{m-1}(x_i, \mathcal{I}_i)$
$\wedge_{k\in\mathcal{I}_i}(x_i \notin \mathcal{B}_\delta(d_k))$
$x_i \in \mathcal{B}_\delta(d_k)$
$taken_k$
$\mathcal{I}_i := \mathcal{I}_i\backslash\{k\}$
$|\mathcal{I}_i| = m-2$
$\dot{x}_i = u_{m-2}(x_i, \mathcal{I}_i)$
$\wedge_{k\in\mathcal{I}_i}(x_i \notin \mathcal{B}_\delta(d_k))$
$x_i \in \mathcal{B}_\delta(d_k)$
$assigned_k$
$\mathcal{I}_i := \{k\}$
$x_i \in \mathcal{B}_\delta(d_k)$
$assigned_k$
$\mathcal{I}_i := \{k\}$
$|\mathcal{I}_i| = 1$
$\dot{x}_i = u_1(x_i, \mathcal{I}_i)$

Fig. 3. Hybrid Automaton for Agent i.

- $inv : V_{A_i} \to Pred(X_{A_i})$ with $inv(v) = \big(\bigwedge_{k\in\mathcal{I}_i}\big(x_i \notin \mathcal{B}_\delta(d_k)\big)\big)$ for all $v \in V_{A_i}$, denotes the invariant conditions of the hybrid automaton.
- $init : V_{A_i} \to Pred(X_{A_i})$ with $init(m) = true$ denotes the set of initial conditions.
- $guard : E_{A_i} \to Pred(X_{A_i})$ with $guard(e) = \big(x_i \in \mathcal{B}_\delta(d_k)\big)$ for all $e \in E_{A_i}$ and all $k \in \mathcal{I}_0$, denotes the set of guards (or transitions) of the hybrid automaton.
- $reset : E_{A_i} \to X_{A_i}$ with $\mathcal{I}_i := reset(e) = \mathcal{I}_i\backslash\{k\}$ if $sync(e) = taken_k$ and $\mathcal{I}_i := reset(e) = \{k\}$ if $sync(e) = assigned_k$ for all $k \in \mathcal{I}_0$, denotes the set of resets associated with the guards of the hybrid automaton.
- $flow : V_{A_i} \to \dot{X}_{A_i}$ with $[\dot{\mathcal{I}}_i \ \dot{x}_i] := flow(v) = 0 \ u_v(x_i, \mathcal{I}_i)]$ for $v \in V_{A_i}$, denotes the flow conditions of the hybrid automaton that constrain the first time derivatives of the system variables in mode $v \in V_{A_i}$.

By Definition 3.4, for any automaton A_i, we see that $|\mathcal{I}_i| = v$ for all $v \in V_{A_i}$. Hence, every mode of A_i corresponds to a distinct number v of available destinations for agent i. While automaton A_i is in mode $|\mathcal{I}_i| = v$, the flow guarantees to drive agent i to a destination $k \in \mathcal{I}_i$. When the guard $x_i \in \mathcal{B}_\delta(d_k)$ is enabled, A_i transitions either to mode $v' = v - 1$ if destination k is taken, or to mode $v' = 1$ if destination k is free. Note, however, that every such transition $v \xrightarrow{e} v'$ results in $v' < v$ and so, eventually $v = 1$ which indicates an assignment for agent i. Note also that these transitions are synchronized with transitions of the destination automaton due to synchronization labels $sync(e) = taken_k$ or $sync(e) = assigned_k$. Figure 3 shows the graph representation of hybrid automaton A_i.

Having defined the agent hybrid automata, we now proceed with the destination automata. As discussed above, each destination can be in one of two states, i.e., *free* or *taken*. This gives rise to the following definition.

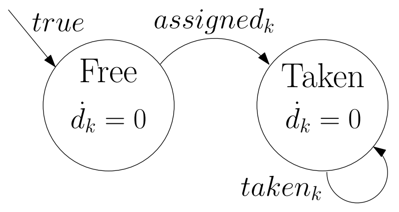


Fig. 4. Hybrid Automaton for Destination k.

Definition 3.5 (Destination Hybrid Automaton): We define the hybrid automaton of destination $k \in \mathcal{I}_0$ to be the tuple $D_k = (X_{D_k}, V_{D_k}, E_{D_k}, \Sigma_{D_k}, sync, inv, init, guard, reset, flow)$, where,

- $X_{D_k} = \{d_k\}$ denotes the set of state variables with $d_k \in \mathbb{R}^2$.
- $V_{D_k} = \{Free, Taken\}$ denotes the finite set of control modes.
- $E_{D_k} = \{(Free, Taken), (Taken, Taken)\}$ denotes the set of control switches.
- $\Sigma_{D_k} = \{taken_k, assigned_k\}$ denotes the set of synchronization labels.
- $sync : E_{D_k} \to \Sigma_{D_k}$ with $sync\big((Free, Taken)\big) = assigned_k$ and $sync\big((Taken, Taken)\big) = taken_k$ denotes the synchronization map mapping each control switch to a synchronization label.
- $inv : V_{D_k} \to Pred(X_{D_k})$ with $inv(v) = true$ for all $v \in V_{D_k}$ denotes the invariant conditions of the hybrid automaton.
- $init : V_{D_k} \to Pred(X_{D_k})$ with $init(Free) = true$ denotes the set of initial conditions.
- $guard : E_{D_k} \to Pred(X_{D_k})$ with $guard(e) = true$ for all $e \in E_{D_k}$, denotes the set of guards (or transitions) of the hybrid automaton.
- $reset : E_{D_k} \to X_{D_k}$ with $d_k := reset(e) = d_k$ for all $e \in E_{D_k}$, denotes the set of resets associated with the guards of the hybrid automaton.
- $flow : V_{D_k} \to \dot{X}_{D_k}$ with $\dot{d}_k := flow(v) = 0$, for all $v \in V_{D_k}$ denotes the flow conditions of the hybrid automaton that constrain the first time derivatives of the system variables in mode $v \in V_{D_k}$.

By Definition 3.5, we see that the synchronization labels of automaton D_k are always enabled. Hence, the transitions of D_k are always synchronized with the transitions of automaton A_i and take place when the guards of A_i are also enabled. This synchronization models the communication between automata A_i and D_k. Figure 4 shows the graph representation of hybrid automaton D_k.

IV. Multi-Destination Potential Fields

So far we have assumed that a control law that satisfies Assumption 3.1(b) exists. In this section we propose such a control law, based on multi-destination potential fields. As discussed in Sections II and III, this control law dynamically determines the sequence of destinations to be explored, while driving every agent to each one of these destinations.

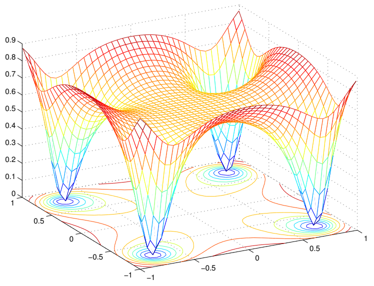


Fig. 5. Plot of the 4-destination potential function $\varphi_4(x_i, \mathcal{I}_i)$ for $dest(\mathcal{I}_i) = \{\ [.75\ .75],\ [-.75\ .75],\ [-.75\ -.75],\ [.75\ -.75]\ \}$.

Consider a single agent, say agent i, in $\mathbb{R}^2$ and as before, denote by $x_i(t) \in \mathbb{R}^2$ its coordinates at time t. Let $\mathcal{I}_0$ denote the set of all available destinations and $\mathcal{I}_i \subseteq \mathcal{I}_0$, with $|\mathcal{I}_i| = v \leq m$, the set of available destinations of agent i.[3] Denote by $\gamma_{dk}(x_i) = \|x_i - d_k\|_2^2$ the distance of agent i to destination $k \in \mathcal{I}_i$. Then, the function, $\gamma_v(x_i, \mathcal{I}_i) = \prod_{k \in \mathcal{I}_i} \gamma_{dk}(x_i)$ is a measure of the distance of agent i to the set of v destinations $\mathcal{I}_i$ since clearly, $\gamma_v(x_i, \mathcal{I}_i) > 0$ for all $x_i \notin dest(\mathcal{I}_i)$ and $\gamma_v(x_i, \mathcal{I}_i) = 0$ only if $x_i \in dest(\mathcal{I}_i)$. Consider, further, the monotone increasing functions in $[0, \infty)$, $\sigma(y) = \frac{y}{1+y}$ and $\tau_\kappa(y) = y^\kappa$ with $\kappa > 0$ and define the v-destination potential function $\varphi_v : \mathbb{R}^2 \to [0, 1]$ by the composition (Figure 5),

$$\varphi_v(x_i, \mathcal{I}_i) = \tau_{1/\kappa} \circ \sigma \circ \tau_\kappa \circ \gamma_v(x_i, \mathcal{I}_i) \tag{3}$$

The rest of this section is devoted in showing that $\varphi_v(x_i, \mathcal{I}_i)$ is free of local minima and hence, the resulting potential field globally converges to the destination set $dest(\mathcal{I}_i)$. The following proposition enables us to characterize the critical points of $\varphi_v(x_i, \mathcal{I}_i)$ by examining the simpler function $\gamma_v(x_i, \mathcal{I}_i)$.

Proposition 4.1 ([19]): Let $I_1, I_2 \subseteq \mathbb{R}$ be intervals, $\gamma : \mathcal{F} \to I_1$ and $\sigma : I_1 \to I_2$ be analytic. Define the composition $\varphi : \mathcal{F} \to I_2$ to be $\varphi = \sigma \circ \gamma$. If σ is monotonically increasing on I_1, then the sets of critical points of φ and γ coincide, i.e., $\mathcal{C}_\varphi = \mathcal{C}_\gamma$, and the index of each point is identical, i.e., $index(\varphi)\big|_{\mathcal{C}_\varphi} = index(\gamma)\big|_{\mathcal{C}_\gamma}$.

Proposition 4.1 implies that $\varphi_v(x_i, \mathcal{I}_i)$ and $\gamma_v(x_i, \mathcal{I}_i)$ share identical critical points. In order to characterize the critical points of $\gamma_v(x_i, \mathcal{I}_i)$ we make use of *harmonic functions* [20]. In particular, by Proposition 4.1 $\gamma_v(x_i, \mathcal{I}_i)$ and $\log(\gamma_v(x_i, \mathcal{I}_i))$ share identical critical points too. But $\log(\gamma_v(x_i, \mathcal{I}_i))$ is harmonic (completely free of local minima) and so almost global convergence of our potential field $\varphi_v(x_i, \mathcal{I}_i)$ is guaranteed. We, thus, have the following result, which we state without proof due to space limitations.

Theorem 4.2: For any fixed destination set $\mathcal{I}_i$ with $|\mathcal{I}_i| = v$, the multi-destination control system,

$$\dot{x}_i = u_v(x_i, \mathcal{I}_i) := -K \nabla_{x_i} \varphi_v(x_i, \mathcal{I}_i) \tag{4}$$

with $K > 0$ a positive constant, is globally asymptotically stable almost everywhere.

[3]Note that, for the purpose of deriving multi-destination potential fields, $\mathcal{I}_i$ is considered constant.

V. Integration of the Overall System

Having defined the agent and destination automata, we now proceed with their composition and study the properties of the overall integrated system.

Definition 5.1 (Product System): We define the product of the hybrid automata $A_1, \dots, A_n, D_1, \dots, D_m$ by the tuple $S = (X_S, V_S, E_S, \Sigma_S, sync, inv, init, guard, \ reset, flow)$, where,

- $X_S = X_{A_1} \cup \dots \cup X_{A_n} \cup X_{D_1} \cup \dots \cup X_{D_m}$ denotes the set of state variables.
- $V_S = V_{A_1} \times \dots \times V_{A_n} \times V_{D_1} \times \dots \times V_{D_m}$ denotes the finite set of control modes.
- $E_S = \{(\|_{j \in \mathcal{J}} e_{A_j}) \| e_{D_k} \mid \forall \ k, \mathcal{J}\}$ with $\mathcal{J} \subseteq \{1, \dots, n\}$, $\mathcal{J} \neq \emptyset$ denotes the set of control switches, where $e_S = (\|_{j \in \mathcal{J}} e_{A_j}) \| e_{D_k} \in E_S$ is defined as the control switch of S corresponding to control switches $e_{A_j} \in E_{A_j}$, $j \in \mathcal{J}$, and $e_{D_k} \in E_{D_k}$, with $sync(e_{A_j}) = sync(e_{D_k})$. Thus, the variables owned by automata A_j for $j \notin \mathcal{J}$ do not change when control switch $e_S = (\|_{j \in \mathcal{J}} e_{A_j}) \| e_{D_k}$ is taken.
- $\Sigma_S = \Sigma_{A_1} \cup \dots \cup \Sigma_{A_n} \cup \Sigma_{D_1} \cup \dots \cup \Sigma_{D_m}$ denotes the set of synchronization labels.
- $sync : E_S \to \Sigma_S$ denotes the synchronization map mapping each control switch to a synchronization label.
- $inv : V_S \to Pred(X_S)$ with $inv(v_S) = inv(v_{A_1}) \wedge \dots \wedge inv(v_{A_n}) \wedge inv(v_{D_1}) \wedge \dots \wedge inv(v_{D_m})$ for all $v_S \in V_S$, denotes the invariant conditions of the product automaton.
- $init : V_S \to Pred(X_S)$ with $init(v_S) = init(v_{A_1}) \wedge \dots \wedge init(v_{A_n}) \wedge init(v_{D_1}) \wedge \dots \wedge init(v_{D_m})$ for all $v_S \in V_S$, denotes the set of initial conditions.
- $guard : E_S \to Pred(X_S)$ with $guard(e_S) = \left(\bigwedge_{j \in \mathcal{J}} guard(e_{A_j})\right) \wedge guard(e_{D_k})$ for all $e_S \in E_S$, denotes the set of guards (or transitions) of the hybrid automaton.
- $reset : E_S \to X_S$ with $reset(e_S) = \left(\bigcup_{j \in \mathcal{J}} reset(e_{A_j})\right) \cup reset(e_{D_k})$ for all $e_S \in E_S$, denotes the set of resets associated with the guards of the hybrid automaton.
- $flow : V_S \to \dot{X}_S$ with $flow(v_S) = flow(v_{A_1}) \cup \dots \cup flow(v_{A_n}) \cup flow(v_{D_1}) \cup \dots \cup flow(v_{D_m})$ for all $v_S \in V_S$, denotes the flow conditions of the hybrid automaton that constrain the first time derivatives of the system variables in mode v_S.

Note that Definition 5.1 of the product automaton S, in general allows transitions $v_S \xrightarrow{e_S} v'_S$ with $e_S = (\|_{j \in \mathcal{J}} e_{A_j}) \| e_{D_k}$ for $|\mathcal{J}| \geq 1$. Consider, thus, any mode v_S with $v_{A_j} > 1$ for $j \in \mathcal{J}$ and $v_{D_k} = Free$, and assume that the control switch $e_S = \left(\|_{j \in \mathcal{J}} (v, 1)_{A_j}\right) \| (Free, Taken)_{D_k} \in E_S$ is enabled with $|\mathcal{J}| > 1$. Then, $v'_{A_j} = 1$ and $\mathcal{I}_j := reset(e_S) = \{k\}$ for all $j \in \mathcal{J}$. In other words, multiple agents are assigned simultaneously to the same destination k, which is not a desired scenario. We argue, however, that such *tie breaking* scenarios are of measure zero, since they require that the agents are initially located at symmetric points with respect to a destination.[4]

Clearly, the product system S, being the composition of all elementary automata A_i and D_k, models the interconnection between them. Hence, studying S we can identify the properties of the whole multi-agent system. The following result shows that S always takes the desired action whenever a destination is explored. In other words, agent i is always assigned to an available destination if it is sufficiently close to it, while it always removes a taken destination from $\mathcal{I}_i$.[5]

Proposition 5.2: For any agent i, any destination $k \in \mathcal{I}_0$ and all time t, the product system S has the following properties:

(a) If $x_i(t) \in \mathcal{B}_\delta(d_k)$ and destination k is available at time t, then $\mathcal{I}_i(t) := \{k\}$.

(b) If $x_i(t) \in \mathcal{B}_\delta(d_k)$ and destination k is taken at time t, then $\mathcal{I}_i(t) := \mathcal{I}_i(t) \backslash \{k\}$.

Hence, the construction of our model is consistent with the system requirements in Section III. The following proposition shows that every agent that has not yet been assigned to an available destination, has always knowledge of at least all available destinations in $\mathcal{I}(t)$. This property of system S is necessary to show that each agent will eventually be assigned to a distinct destination in $\mathcal{I}_0$.

Proposition 5.3: The product system S guarantees that $\mathcal{I}(t) \subseteq \mathcal{I}_i(t)$ for all time t and all agents i with $|\mathcal{I}_i(t)| > 1$.

Our next result concerns the running time of the hybrid system S. In particular, we show that the product system S in the worst case will only take a finite number of transitions $v_S \xrightarrow{e_S} v'_S$, which is polynomial with respect to the number of agents n. Hence, the number of assignments that can be explored is at most polynomial with n. This result is important, given that the number of assignments, and hence the space of control modes V_S of S, grows exponentially with the number of agents n.

Proposition 5.4: Let $v^\star_S = (v^\star_{A_1}, \dots, v^\star_{A_n}, v^\star_{D_1}, \dots, v^\star_{D_m})$ be such that $v^\star_{A_i} = 1$ for all i and $v^\star_{D_l} = Taken$ for exactly n indices l. Then, initialized at v^0_S, the product system S can reach $v^\star_S$ in at most $\frac{n(n+1)}{2}$ transitions $v_S \xrightarrow{e_S} v'_S$, where n is the total number of automata A_i.

Having showed that the product system S satisfies the problem specifications and has also reasonable complexity, we now use Propositions 5.3 and 5.4 to show that it also has the desired *liveness* and *safety* properties. In other words, we show that every agent will eventually be assigned to a destination in the set $\mathcal{I}_0$ and that no two agents will be assigned to the same destination. We hence, have the following theorem.

Theorem 5.5: For almost all initial conditions $x_i(t_0)$, there exists a constant $T > 0$ such that for all time $t > t_0 + T$, the product system S is in mode $v^\star_S = (v^\star_{A_1}, \dots, v^\star_{A_n}, v^\star_{D_1}, \dots, v^\star_{D_m})$ with $v^\star_{A_i} = 1$ for all i and $v^\star_{D_l} = Taken$ for exactly n indices $l = 1, \dots, m$. We call $v^\star_S$ the equilibrium mode of the system.

[4]In order to resolve *tie breaking* scenarios efficiently, communication between the agents is necessary.

[5]Due to space limitations we omit the proof of this and the subsequent results of this section.

VI. Simulation Results

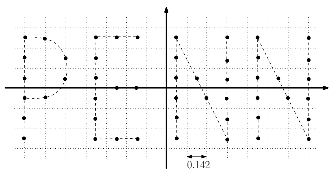
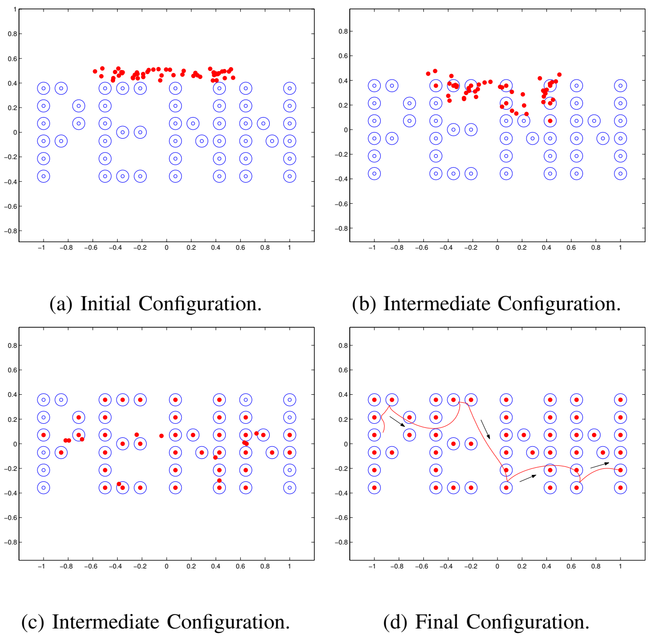


Fig. 6. Destination set $dest(\mathcal{I}_0)$. Destinations are marked with dots.

We consider a navigation task where $n = 50$ agents, starting from randomly chosen initial configurations, have to reach the destination set $dest(\mathcal{I}_0)$ consisting of $m = 50$ destinations (Figure 6). Figures 7 show the evolution of the system at 4 different time instants. The agents are denoted with dots, the destinations with small circles and the δ-neighborhoods (with $\delta = .05$) around each destination, with big circles. Observe that the hybrid system S eventually drives every agent to a distinct destination. Note also in Figure 7(d) part of the trajectory of one of the agents (one the last agents to reach its destination) until it is assigned to a free destination. The sequence of destinations explored is dynamically determined by the multi-destination potential fields which involve only unexplored destinations. On the other hand, when taken destinations are explored, they are removed from its set of available destinations $\mathcal{I}_i$.



(a) Initial Configuration. (b) Intermediate Configuration.

(c) Intermediate Configuration. (d) Final Configuration.

Fig. 7. Simulation for $n = 50$ agents.

VII. Conclusions

In this paper, we considered the problem of determining an assignment between the agents and the destinations in a multi-agent motion planning task. The assignment was determined dynamically by means of exploring available destinations and using distributed sensor-based coordination to ensure that distinct destinations were assigned to distinct agents. On the other hand, the sequence of destinations to be explored, as well as navigation of the agents to any of these destinations, was guaranteed for almost all initial conditions by novel multi-destination potential fields. The overall hybrid system was shown to almost always guarantee the mutual exclusion property of the final assignment and have at most polynomial complexity, despite the exponential growth of the number of assignments with respect to the number of agents. Finally, our scalable approach was verified through non-trivial computer simulations.

References

[1] D. V. Dimarogonas, S. G. Loizou, K. J. Kyriakopoulos and M. M. Zavlanos. *A Feedback Stabilization and Collision Avoidance Scheme for Multiple Independent Non-point Agents*, Automatica, vol. 42(2), pp. 229-243, Feb. 2006.

[2] A. Jadbabaie, J. Lin and A. S. Morse. *Coordination of Groups of Mobile Autonomous Agents using Nearest Neighbor Rules*, IEEE Trans. on Aut. Control, vol. 48(6), pp. 988-1001, 2003.

[3] R. Olfati-Saber and R. M. Murray. *Consensus Problems in Networks of Agents with Switching Topology and Time-Delays*, IEEE Trans. on Aut. Control, vol. 49, pp. 1520-1533, Sep. 2004.

[4] H. Tanner, A Jadbabaie and G. Pappas. *Flocking in Fixed and Switching Networks*, IEEE Trans. on Aut. Control, April 2005. Submitted.

[5] J. Cortes, S. Martinez and F. Bullo. *Robust Rendezvous for Mobile Autonomous Agents via Proximity Graphs in Arbitrary Dimensions*, IEEE Trans. on Aut. Control, vol. 51(8), pp. 1289-1298, 2006.

[6] L. Chaimowicz, Michael, N., and V. Kumar. *Controlling Swarms of Robots Using Interpolated Implicit Functions*, Proc. IEEE Int. Conf. on Rob. and Automation, pp. 2498 - 2503, Barcelona, Spain, April 2005.

[7] R. Sepulchre, D. Paley, N. E. Leonard. *Stabilization of Planar Collective Motion: All-to-All Communication*, IEEE Trans. on Aut. Control, 2006. To appear.

[8] G. Lafferriere, A. Williams, J. Caughman and J.J.P. Veerman. *Decentralized Control of Vehicle Formations*, Systems and Control Letters, vol. 54(9), pp. 899-910, Sept. 2005.

[9] T. Balch and R.C. Arkin. *Behavior-based Formation Control for Multirobot Teams*, IEEE Trans. on Rob. and Automation, vol. 14(6), pp. 926-939, Dec. 1998.

[10] W. Ren and R. Beard. *Consensus of Information under Dynamically changing Interaction Topologies*, Proc. American Control Conf., pp. 4939-4944, June 2004.

[11] Sameera Poduri and Gaurav S. Sukhatme. *Constrained Coverage for Mobile Sensor Networks*, IEEE Int. Conf. on Rob. and Automation, pp. 165-172, New Orleans, LA, May 2004.

[12] J. Lin, A.S. Morse and B.D.O. Anderson. *The Multi-Agent Rendezvous Problem*, Proc. 42nd IEEE Conf. on Decision and Control, pp. 1508-1513, Maui, Hawaii, Dec. 2003.

[13] H. W. Kuhn. *The Hungarian Method for the Assignment Problem*, Naval Research Logistics, vol. 2, pp. 8397, 1955.

[14] M. R. Garey and D. S. Johnson. *Computers and Intractabiltiy: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, CA, 1979.

[15] H. A. Almohamad and S. O. Duffuaa. *A Linear Programming Approach for the Weighted Graph Matching Problem*, IEEE Trans. on Pattern An. and Machine Intel., vol. 15(5), pp. 522-525, May 1993.

[16] Michael M. Zavlanos and George J. Pappas. *A Dynamical Systems Approach to Weighted Graph Matching*, Proc. 45th IEEE Conf. on Decision and Control, pp. 3492-3497, San Diego, Dec. 2006.

[17] S. Kloder and S. Hutchinson. *Path Planning for Permutation-Invariant Multirobot Formations*, IEEE Trans. on Robotics, vol. 22(4), pp. 650 - 665, Aug. 2006.

[18] M. Ji, S. Azuma, and M. Egerstedt. *Role-Assignment in Multi-Agent Coordination*, Int. J. of Assistive Rob. and Mechatronics, vol. 7(1), pp. 32-40, March 2006.

[19] D. E. Koditschek and E. Rimon. 1990. *Robot Navigation Functions on Manifolds with Boundary*, Advances in Applied Mathematics, vol. 11, pp. 412 - 442, 1990.

[20] J. O. Kim and P. K. Khosla. *Real-time Obstacle Avoidance using Harmonic Potential Functions*, IEEE Trans. on Rob. and Automation, vol. 8(3), pp. 338 - 349, Jun. 1992.