Proceedings of the 2007 American Control Conference
Marriott Marquis Hotel at Times Square
New York City, USA, July 11-13, 2007

Dynamic Assignment in Distributed Motion Planning with Limited Information

Michael M. Zavlanos and George J. Pappas

Abstract— **Distributed motion planning of multiple agents raises fundamental and novel problems in control theory and robotics. In this paper, we consider the problem of designing distributed motion algorithms that dynamically assign targets or destinations to multiple homogeneous agents. We achieve this goal using a novel control decomposition. In particular, navigation of every agent to any available destination is due to distributed multi-destination potential fields, while the mutual exclusion property of the final assignment is guaranteed by local coordination protocols among the agents. Integration of the proposed controllers results in a hybrid model for every agent, while the overall system is shown to always converge to a valid assignment and have at most polynomial complexity, dramatically reducing the combinatorial nature of purely discrete assignment problems. We conclude by illustrating our approach through nontrivial computer simulations.**

I. Introduction

Given any multi-agent motion planning task, where a group of agents has to reach a desired destination configuration, the *assignment problem* consists of determining a permutation of the agents in the final destination set. If no such permutation of the agents is provided a priori, then it has to be determined on-line. Moreover, if only local information from neighbors 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 operations research, computer vision as well as distributed robotics, where graphs are recently emerging as a natural mathematical description for capturing interconnection topology [1] $-$ [7]. 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 [8]. The quadratic assignment problem, however, is NP-hard [9] and suboptimal solutions are achieved by means of various relaxations. Approaches are either purely discrete [10], [11] or continuous [12], based on the solution of differential equations that always converge to a discrete assignment.

In distributed robotics, the assignment problem naturally arises in tasks involving destination or target allocation. Depending on whether the discrete assignment is addressed simultaneously with the continuous navigation strategies or is solved independently in advance, approaches can be either on-line or off-line. An on-line approach is proposed in [13], 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-destination assignments. The proposed approach is open loop and centralized, since it requires global knowledge of the environment. On the other hand, in [14] a polynomial time algorithm is developed that computes off-line a suboptimal assignment between agents and destinations based on a "minimum distance to the goal" policy.

In this paper we propose a distributed *feedback* control framework that simultaneously addresses the continuous navigation strategies as well as the discrete assignment of agents to destinations. Under the assumption that every agent has knowledge of all available destinations, we build our approach based on two novel ideas. First, provably correct multi-destination potential fields, used to drive every agent from almost all initial configurations to any available destination, determine dynamically a sequence of destinations to be explored by each agent [15]. Second, local coordination protocols ensure that assignments are established only among agents and free destinations. Unlike our sensor-based approach [15], where the presence of singularities due to *ties* over available destinations could not be handled, here the mutual exclusion property of the final assignment is always guaranteed. Integration of the proposed controllers results in a hybrid model for every agent, while the overall system is shown to always converge to a valid assignment 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 algorithm is illustrated through nontrivial computer simulations.

The rest of this paper is organized as follows. In Section II we define the dynamic assignment problem, while in Section III we develop the multi-destination potential fields and discuss their convergence properties. In Section IV we discuss local coordination protocols and define the hybrid automata that consist the agents' models. The overall system is studied in Section V, where results about its complexity and equilibrium modes are also presented. 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

II. Problem Formulation

Consider n point agents in $\mathbb{R}^2$ 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,

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

where $u_i(t)$ is a sought control vector taking values in $\mathbb{R}^2$. Consider, further, $m \geq n$ destinations in $\mathbb{R}^2$ that the agents have to reach and let $\mathcal{I}_0 = \{1, \dots, m\}$ denote the index set corresponding to a fixed labeling of these destinations. We assume that every 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 system (1) should be driven to. Given that the agents are "identical" we consider any assignment equally desirable. 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 [12], [14], and then design controllers that drive each agent to its destination. 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 fully *distributed* solution to the aforementioned problem. In particular, we assume that every agent has only knowledge of its available destinations, while the assignment decision is embedded in its controller and relies on inter-agent communication. 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. The sought sequence of destinations is determined dynamically by means of multi-destination potential fields designed to drive every agent to any available destination. Then, the *mutual exclusion* property of the final assignment is guaranteed by local coordination protocols, developed in Section IV.

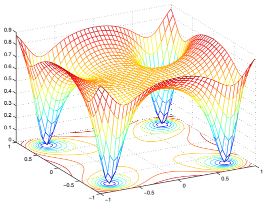


Fig. 1. Plot of the 4-destination potential function $\varphi_4(x_i, \mathcal{I}_i^a)$ for $dest(\mathcal{I}_i^a) = \{\ [.75\ .75],\ [-.75\ .75],\ [-.75\ -.75],\ [.75\ -.75]\ \}$.

III. Multi-Destination Potential Fields

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

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

The following proposition enables us to characterize the critical points of $\varphi_v(x_i, \mathcal{I}_i^a)$ by examining the simpler function $\gamma_v(x_i, \mathcal{I}_i^a)$.

Proposition 3.1 ([16]): 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 3.1 implies that $\varphi_v(x_i, \mathcal{I}_i^a)$ and $\gamma_v(x_i, \mathcal{I}_i^a)$ share identical critical points. In order to characterize the critical points of $\gamma_v(x_i, \mathcal{I}_i^a)$ we make use of *harmonic functions* [17]. In particular, by Proposition 3.1 $\gamma_v(x_i, \mathcal{I}_i^a)$ and $\log(\gamma_v(x_i, \mathcal{I}_i^a))$ share identical critical points too. But $\log(\gamma_v(x_i, \mathcal{I}_i^a))$ is harmonic (completely free of local minima) and so almost global convergence of our potential field $\varphi_v(x_i, \mathcal{I}_i^a)$ is guaranteed. We, thus, have the following result, which we state without proof due to space limitations.

Theorem 3.2: For any fixed destination set $\mathcal{I}_i^a$ with $|\mathcal{I}_i^a| = v$, the multi-destination control system,

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

with $K > 0$ a positive constant, is globally asymptotically stable almost everywhere (except for a set of measure zero).

According to Theorem 3.2, system (4) guarantees that agent i will eventually reach any destination in $\mathcal{I}_i^a$. Whether an assignment will be established, depends on whether the particular destination is available or not, and relies on distributed coordination among neighboring agents.

[1]We denote by $|\mathcal{A}|$ the cardinality of the set $\mathcal{A}$.
[2]The sets $\mathcal{I}_i^a$ will be formally defined in Section IV.

IV. Distributed Coordination

Let $\mathcal{I}(t)$ denote the index set of available destinations at time $t \geq t_0$ and denote by $\mathcal{I}^c(t) = \mathcal{I}_0 \backslash \mathcal{I}(t)$ its complement, where initially, $\mathcal{I}(t_0) = \mathcal{I}_0$ and $\mathcal{I}^c(t_0) = \emptyset$. Similarly, let $\mathcal{I}_i^a(t)$ denote the index set of *available* destinations from the perspective of agent i and define the set of *taken* destinations of agent i by $\mathcal{I}_i^t(t) = \mathcal{I}_0 \backslash \mathcal{I}_i^a(t)$. Since $\mathcal{I}_i^a(t) \cap \mathcal{I}_i^t(t) = \emptyset$, no destination can be considered both available and taken, while $\mathcal{I}_i^a(t) \cup \mathcal{I}_i^t(t) = \mathcal{I}_0$ implies that any destination that is not available, has to be taken. The sets $\mathcal{I}_i^a(t)$ and $\mathcal{I}_i^t(t)$ are initialized such that every agent has knowledge of all available destinations in $\mathcal{I}_0$, i.e., $\mathcal{I}_i^a(t_0) = \mathcal{I}_0$ and $\mathcal{I}_i^t(t_0) = \emptyset$, while we also require that $\mathcal{I}_i^a(t) = \{k\}$ if and only if agent i is assigned to destination $k \in \mathcal{I}_0$.

To achieve local coordination among the agents, we further define the set of neighbors of agent i at time t by $\mathcal{N}_i^\epsilon(t) = \{j \mid x_j(t) \in \mathcal{B}_\epsilon(x_i(t))\}$, where $\epsilon > 0$ indicates the *coordination radius* of agent i and $\mathcal{B}_r(x) = \{y \in \mathbb{R}^2 \mid \|y-x\|_2 < r\}$ denotes an open ball of radius $r > 0$ centered at $x \in \mathbb{R}^2$. On the other hand, to efficiently handle *ties* over the destinations, i.e., situations where multiple agents simultaneously claim the same destination, we require that every agent can identify the set of candidate agents $\mathcal{C}_i(t)$ requesting to be assigned to the same destination at time t, and can also break the tie if necessary. To achieve this specification, we introduce a *tie breaking* function, $tb : 2^{\mathbb{N}} \backslash \{\emptyset\} \to \mathbb{N}$ such that,

$$tb(A) := i \in A$$

where $i \in A$ can be chosen according to any policy, deterministic or not, and assume that every agent is equipped with such a function. Then, the action $tb(\mathcal{C}_i)$, taken by any of the agents in $\mathcal{C}_i$, can break a tie for any destination, while the outcome can be transmitted to the other neighbors.

Under the assumption that all agents are equipped with local coordination mechanisms, we now state our problem specifications.

Assumptions 4.1: For every agent $i = 1, \ldots, n$ we assume that, for all time $t \geq t_0$,

(a) it can be assigned to an available destination $k \in \mathcal{I}(t)$, if $k \in \mathcal{I}_i^a(t)$, $|\mathcal{I}_i^a(t)| > 1$ and $x_i(t) \in \mathcal{B}_\delta(d_k)$,
(b) there is a controller $u_v(x_i(t), \mathcal{I}_i^a)$, that for any fixed index set $\mathcal{I}_i^a$ can drive it to any destination in $\mathcal{I}_i^a$,
(c) $\delta, \epsilon > 0$ are such that $\mathcal{B}_\delta(d_k) \cap \mathcal{B}_\delta(d_l) = \emptyset$ for all $k, l \in \mathcal{I}_0$ and $\epsilon > 2\delta$.

Assumption 4.1(a) implies that agent i can only be assigned to an available destination in $\mathcal{I}_i^a(t)$ if it is sufficiently close to that destination, while Assumption 4.1(b) says that every agent is able to navigate to any of its available destinations, unless it has already been assigned to a destination, whence it should always remain in a neighborhood of that destination. Note that system (4) satisfies Assumption 4.1(b). On the other hand, Assumption 4.1(c) combined with Assumption 4.1(a) guarantees that every agent can only claim one destination at a time, while combined with Assumption 4.1(b) implies that any agent sufficiently close to a destination knows whether this destination is taken or not.

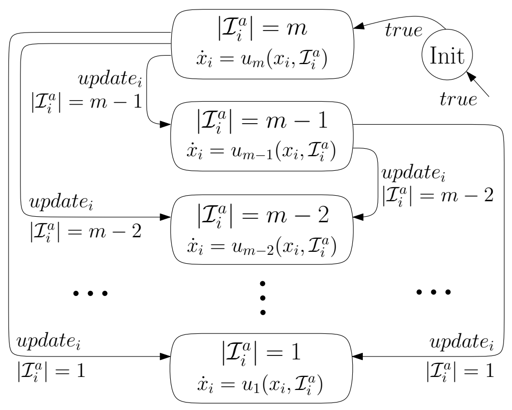

$|\mathcal{I}_i^a| = m$
$\dot{x}_i = u_m(x_i, \mathcal{I}_i^a)$
$true$
Init
$true$
$update_i$
$|\mathcal{I}_i^a| = m-1$
$|\mathcal{I}_i^a| = m-1$
$\dot{x}_i = u_{m-1}(x_i, \mathcal{I}_i^a)$
$update_i$
$|\mathcal{I}_i^a| = m-2$
$update_i$
$|\mathcal{I}_i^a| = m-2$
$|\mathcal{I}_i^a| = m-2$
$\dot{x}_i = u_{m-2}(x_i, \mathcal{I}_i^a)$
$update_i$
$|\mathcal{I}_i^a| = 1$
$|\mathcal{I}_i^a| = 1$
$\dot{x}_i = u_1(x_i, \mathcal{I}_i^a)$
$update_i$
$|\mathcal{I}_i^a| = 1$

Fig. 2. Navigation Automaton for Agent i.

A. Modeling the Agents

Developing discrete in nature coordination protocols and integrating them with the continuous multi-destination potential fields of Section III, gives rise to a hybrid model for every agent [18]. In particular, every agent consists of a *navigation automaton* generating a path to any destination in $\mathcal{I}_i^a(t)$, and a *coordination automaton* responsible for identifying its neighbors $\mathcal{N}_i^\epsilon(t)$ for all time t and exchanging information with them. The following notion of a *predicate* enables us to formally define the aforementioned automata.

Definition 4.2 (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 navigation automaton of agent i can be defined as follows.[3]

Definition 4.3 (Navigation Hybrid Automaton): We define the navigation hybrid automaton of agent i to be the tuple $N_i = (X_{N_i}, V_{N_i}, E_{N_i}, \Sigma_{N_i}, sync, inv, init, guard, reset, flow)$, where,

- $X_{N_i} = \{x_i\}$ denotes the set of owned state variables with $x_i \in \mathbb{R}^2$.
- $V_{N_i} = \{1, \ldots, m, Init\}$ denotes the finite set of control modes.
- $E_{N_i} = \{(Init, m), (v, v-p),\ \forall\ 0 < p \leq v-1 \mid v \in V_{N_i} \backslash \{1, Init\}\}$ denotes the set of control switches.
- $\Sigma_{N_i} = \{update_i\}$ denotes the set of synchronization labels.
- $sync : E_{N_i} \to \Sigma_{N_i}$ with $sync(e) = update_i$ for all $e \in E_{N_i} \backslash \{(Init, m)\}$, denotes the synchronization map mapping each control switch to a synchronization label.
- $inv : V_{N_i} \to Pred(X_{N_i})$ with $inv(v) = true$ for all $v \in V_{N_i}$, denotes the invariant conditions of the hybrid automaton.

[3]To simplify notation, we hereafter drop the dependence of the state variables on time.

- $init : V_{N_i} \to Pred(X_{N_i})$ with $init(v) = true$ for $v = Init$ denotes the set of initial conditions.
- $guard : E_{N_i} \to Pred(X_{N_i})$ with $guard\big((Init, m)\big) = true$ and $guard\big((v,\, v-p)\big) = \big(|\mathcal{I}_i^a| = v - p\big)$ for all $v \in V_{N_i}\backslash\{1, Init\}$ and all $0 < p \leq v - 1$, denotes the set of guards of the hybrid automaton.
- $reset : E_{N_i} \to X_{N_i}$ with $x_i := reset(e) = x_i$ for all $e \in E_{N_i}$, denotes the set of resets associated with the guards of the hybrid automaton.
- $flow : V_{N_i} \to \dot{X}_{N_i}$ with $\dot{x}_i := flow(v) = u_v(x_i, \mathcal{I}_i^a)$ for $v \in V_{N_i}\backslash\{Init\}$ and $\dot{x}_i := flow(Init) = 0$, denotes the flow conditions of the hybrid automaton that constrain the first time derivatives of the system variables in mode $v \in V_{N_i}$.

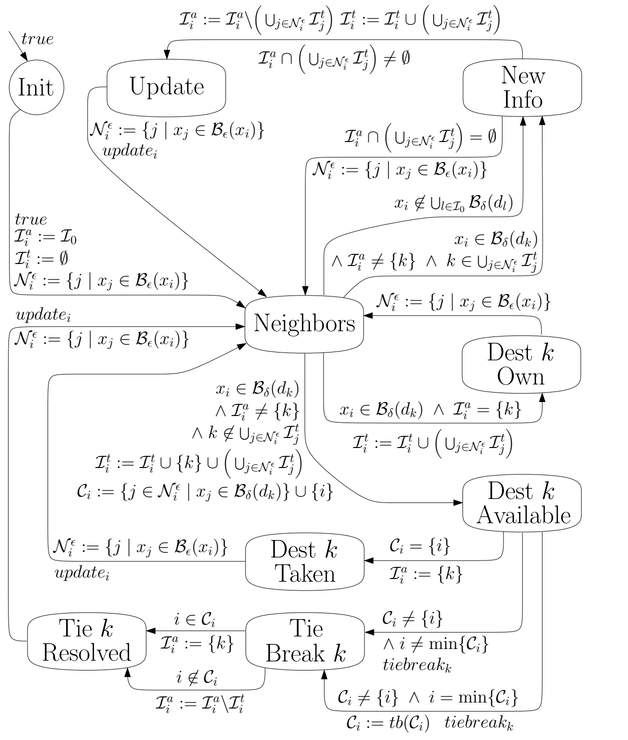
By Definition 4.3, for any automaton N_i, we see that $|\mathcal{I}_i^a| = v$ for all $v \in V_{N_i}$. Hence, every mode of N_i corresponds to a distinct number v of available destinations for agent i. While automaton N_i is in mode $|\mathcal{I}_i^a| = v$, control law (4) guarantees to drive agent i to one of the destinations in $\mathcal{I}_i^a$. On the other hand, transitions in N_i are triggered whenever the set of available destinations $\mathcal{I}_i^a$ is updated. Such updates can either take place because a *free* destination has been discovered or because information about *taken* destinations has been received from agent i's neighbors $\mathcal{N}_i^\epsilon$. 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 coordination automaton due to synchronization labels $sync(e) = update_i$. Figure 2 shows the graph representation of hybrid automaton N_i.

In the following we define the coordination automaton for agent i. The coordination automaton is designed to continuously update agent i's neighbors $\mathcal{N}_i^\epsilon$, while the coordination mechanism uses nearest neighbor information and describes how agent i should update its state variables $\mathcal{I}_i^a$ and $\mathcal{I}_i^t$, when it is close to an available destination, when it is close to a taken destination, when it has been assigned to a destination and when it is far from any destination.

Definition 4.4 (Coordination Hybrid Automaton): We define the communication hybrid automaton of agent i to be the tuple $C_i = (X_{C_i}, V_{C_i}, E_{C_i}, \Sigma_{C_i}, sync, inv, init, guard, reset, flow)$, where,

- $X_{C_i} = \{\mathcal{I}_i^a, \mathcal{I}_i^t, \mathcal{N}_i^\epsilon, \mathcal{C}_i\}$ denotes the set of owned state variables with $\mathcal{I}_i^a, \mathcal{I}_i^t \in 2^{\mathcal{I}_0}$ and $\mathcal{N}_i^\epsilon, \mathcal{C}_i \in 2^{\{1,\ldots,n\}}$.
- $V_{C_i} = \{Init, N, I, U, O_k, A_k, T_k, B_k, R_k \mid k \in \mathcal{I}_0\}$ denotes the finite set of control modes.[4]
- $E_{C_i} = \{(Init, N), (N, I), (I, N), (I, U), (U, N), (N, O_k), (O_k, N), (N, A_k), (A_k, T_k), (A_k, B_k), (T_k, N), (B_k, R_k), (R_k, N) \mid k \in \mathcal{I}_0\}$, denotes the set of control switches.
- $\Sigma_{C_i} = \{update_i, tiebreak_k \mid k \in \mathcal{I}_0\}$ denotes the set of synchronization labels.
- $sync : E_{C_i} \to \Sigma_{C_i}$ with,
  - $sync\big((A_k, B_k)\big) = tiebreak_k$, for all $k \in \mathcal{I}_0$,

[4]The shorthand notation stands for $I := New\ Info$, $N := Neighbors$, $U := Update$, $O_k := Dest\ k\ Own$, $T_k := Dest\ k\ Taken$, $A_k := Dest\ k\ Available$, $B_k := Tie\ Break\ k$ and $R_k := Tie\ k\ Resolved$.



true
Init
Update
New Info
$\mathcal{I}_i^a := \mathcal{I}_i^a\backslash\Big(\cup_{j\in\mathcal{N}_i^\epsilon}\mathcal{I}_j^t\Big)\ \mathcal{I}_i^t := \mathcal{I}_i^t \cup \Big(\cup_{j\in\mathcal{N}_i^\epsilon}\mathcal{I}_j^t\Big)$
$\mathcal{I}_i^a \cap \Big(\cup_{j\in\mathcal{N}_i^\epsilon}\mathcal{I}_j^t\Big) \neq \emptyset$
$\mathcal{N}_i^\epsilon := \{j \mid x_j \in \mathcal{B}_\epsilon(x_i)\}$
$update_i$
$\mathcal{I}_i^a \cap \Big(\cup_{j\in\mathcal{N}_i^\epsilon}\mathcal{I}_j^t\Big) = \emptyset$
$\mathcal{N}_i^\epsilon := \{j \mid x_j \in \mathcal{B}_\epsilon(x_i)\}$
$x_i \notin \cup_{l\in\mathcal{I}_0}\mathcal{B}_\delta(d_l)$
$x_i \in \mathcal{B}_\delta(d_k)$
$\wedge\ \mathcal{I}_i^a \neq \{k\}\ \wedge\ k \in \cup_{j\in\mathcal{N}_i^\epsilon}\mathcal{I}_j^t$
$true$
$\mathcal{I}_i^a := \mathcal{I}_0$
$\mathcal{I}_i^t := \emptyset$
$\mathcal{N}_i^\epsilon := \{j \mid x_j \in \mathcal{B}_\epsilon(x_i)\}$
$update_i$
$\mathcal{N}_i^\epsilon := \{j \mid x_j \in \mathcal{B}_\epsilon(x_i)\}$
Neighbors
$\mathcal{N}_i^\epsilon := \{j \mid x_j \in \mathcal{B}_\epsilon(x_i)\}$
Dest k Own
$x_i \in \mathcal{B}_\delta(d_k)$
$\wedge\ \mathcal{I}_i^a \neq \{k\}$
$\wedge\ k \notin \cup_{j\in\mathcal{N}_i^\epsilon}\mathcal{I}_j^t$
$\mathcal{I}_i^t := \mathcal{I}_i^t \cup \{k\} \cup \Big(\cup_{j\in\mathcal{N}_i^\epsilon}\mathcal{I}_j^t\Big)$
$\mathcal{C}_i := \{j \in \mathcal{N}_i^\epsilon \mid x_j \in \mathcal{B}_\delta(d_k)\} \cup \{i\}$
$x_i \in \mathcal{B}_\delta(d_k)\ \wedge\ \mathcal{I}_i^a = \{k\}$
$\mathcal{I}_i^t := \mathcal{I}_i^t \cup \Big(\cup_{j\in\mathcal{N}_i^\epsilon}\mathcal{I}_j^t\Big)$
Dest k Available
$\mathcal{N}_i^\epsilon := \{j \mid x_j \in \mathcal{B}_\epsilon(x_i)\}$
$update_i$
Dest k Taken
$\mathcal{C}_i = \{i\}$
$\mathcal{I}_i^a := \{k\}$
Tie k Resolved
$i \in \mathcal{C}_i$
$\mathcal{I}_i^a := \{k\}$
Tie Break k
$\mathcal{C}_i \neq \{i\}$
$\wedge\ i \neq \min\{\mathcal{C}_i\}$
$tiebreak_k$
$i \notin \mathcal{C}_i$
$\mathcal{I}_i^a := \mathcal{I}_i^a\backslash\mathcal{I}_i^t$
$\mathcal{C}_i \neq \{i\}\ \wedge\ i = \min\{\mathcal{C}_i\}$
$\mathcal{C}_i := tb(\mathcal{C}_i)\quad tiebreak_k$

Fig. 3. Coordination Automaton for Agent i.

  - $sync(e) = update_i$, for $e = (U, N), (T_k, N), (R_k, N)$.

  denotes the synchronization map mapping each control switch to a synchronization label.
- $inv : V_{C_i} \to Pred(X_{C_i})$ with $inv(v) = true$ for all $v \in V_{C_i}$, denotes the invariant conditions of the hybrid automaton.
- $init : V_{C_i} \to Pred(X_{C_i})$ with $init(v) = true$ for $v = Init$, denotes the set of initial conditions.
- $guard : E_{C_i} \to Pred(X_{C_i})$ with,
  - $guard\big((N, I)\big) = \big(x_i \notin \cup_{l\in\mathcal{I}_0}\mathcal{B}_\delta(d_l)\big) \bigvee \big(x_i \in \mathcal{B}_\delta(d_k)\ \wedge\ \mathcal{I}_i^a \neq \{k\}\ \wedge\ k \in \cup_{j\in\mathcal{N}_i^\epsilon}\mathcal{I}_j^t\big)$, for all $k \in \mathcal{I}_0$,
  - $guard\big((I, N)\big) = \big(\mathcal{I}_i^a \cap \big(\cup_{j\in\mathcal{N}_i^\epsilon}\mathcal{I}_j^t\big) = \emptyset\big)$,
  - $guard\big((I, U)\big) = \big(\mathcal{I}_i^a \cap \big(\cup_{j\in\mathcal{N}_i^\epsilon}\mathcal{I}_j^t\big) \neq \emptyset\big)$,
  - $guard\big((N, O_k)\big) = \big(x_i \in \mathcal{B}_\delta(d_k)\ \wedge \mathcal{I}_i^a = \{k\}\big)$, for all $k \in \mathcal{I}_0$,
  - $guard\big((N, A_k)\big) = \big(x_i \in \mathcal{B}_\delta(d_k)\ \wedge\ \mathcal{I}_i^a \neq \{k\}\ \wedge k \notin \cup_{j\in\mathcal{N}_i^\epsilon}\mathcal{I}_j^t\big)$, for all $k \in \mathcal{I}_0$,
  - $guard\big((A_k, T_k)\big) = \big(\mathcal{C}_i = \emptyset\big)$, for all $k \in \mathcal{I}_0$,
  - $guard\big((A_k, B_k)\big) = \big(\mathcal{C}_i \neq \emptyset\big)$, for all $k \in \mathcal{I}_0$,
  - $guard(e) = true$, otherwise,

  denotes the set of guards of the hybrid automaton.
- $reset : E_{C_i} \to X_{C_i}$ with, $[\mathcal{I}_i^a\ \mathcal{I}_i^t\ \mathcal{N}_i^\epsilon\ \mathcal{C}_i] := reset(e)$ such that,
  - $reset\big((Init, N)\big) = \big[\mathcal{I}_0\ \emptyset\ \{j \mid x_j \in \mathcal{B}_\epsilon(x_i)\}\ \emptyset\big]$,
  - $reset\big((N, I)\big) = \big[\mathcal{I}_i^a\ \mathcal{I}_i^t\ \mathcal{N}_i^\epsilon\ \mathcal{C}_i\big]$,
  - $reset\big((I, N)\big) = \big[\mathcal{I}_i^a\ \mathcal{I}_i^t\ \{j \mid x_j \in \mathcal{B}_\epsilon(x_i)\}\ \mathcal{C}_i\big]$,
  - $reset\big((I, U)\big) = \big[\mathcal{I}_i^a\backslash\big(\cup_{j\in\mathcal{N}_i^\epsilon}\mathcal{I}_j^t\big)\ \mathcal{I}_i^t \cup \big(\cup_{j\in\mathcal{N}_i^\epsilon}\mathcal{I}_j^t\big)\ \mathcal{N}_i^\epsilon\ \mathcal{C}_i\big]$,

  - $reset\big((U,N)\big) = \big[\mathcal{I}_i^a\ \mathcal{I}_i^t\ \{j \mid x_j \in \mathcal{B}_\epsilon(x_i)\}\ \mathcal{C}_i\big]$,
  - $reset\big((N,O_k)\big) = \big[\mathcal{I}_i^a\ \mathcal{I}_i^t \cup \big(\cup_{j\in\mathcal{N}_i^\epsilon} \mathcal{I}_j^t\big)\ \ \mathcal{N}_i^\epsilon\ \mathcal{C}_i\big]$,
  - $reset\big((O_k,N)\big) = \big[\mathcal{I}_i^a\ \mathcal{I}_i^t\ \{j \mid x_j \in \mathcal{B}_\epsilon(x_i)\}\ \ \mathcal{C}_i\big]$,
  - $reset\big((N,A_k)\big) = \big[\mathcal{I}_i^a\ \ \mathcal{I}_i^t \cup \{k\} \cup \big(\cup_{j\in\mathcal{N}_i^\epsilon} \mathcal{I}_j^t\big)\ \mathcal{N}_i^\epsilon\ \{j \in \mathcal{N}_i^\epsilon \mid x_j \in \mathcal{B}_\delta(d_k)\} \cup \{i\}\big]$,
  - $reset\big((A_k,T_k)\big) = \big[\{k\}\ \mathcal{I}_i^t\ \mathcal{N}_i^\epsilon\ \mathcal{C}_i\big]$,
  - $reset\big((A_k,B_k)\big) = \big[\mathcal{I}_i^a\ \ \mathcal{I}_i^t\ \ \mathcal{N}_i^\epsilon\ \ tb(\mathcal{C}_i)\big]$, if $i = \min\{\mathcal{C}_i\}$ and $reset\big((A_k,B_k)\big) = \big[\mathcal{I}_i^a\ \mathcal{I}_i^t\ \mathcal{N}_i^\epsilon\ \mathcal{C}_i\big]$, if $i \neq \min\{\mathcal{C}_i\}$,
  - $reset\big((T_k,N)\big) = \big[\mathcal{I}_i^a\ \mathcal{I}_i^t\ \{j \mid x_j \in \mathcal{B}_\epsilon(x_i)\}\ \ \mathcal{C}_i\big]$,
  - $reset\big((B_k,R_k)\big) = \big[\{k\}\ \mathcal{I}_i^t\ \mathcal{N}_i^\epsilon\ \mathcal{C}_i\big]$, if $i \in \mathcal{C}_i$ and $reset\big((B_k,R_k)\big) = \big[\mathcal{I}_i^a \backslash \mathcal{I}_i^t\ \mathcal{I}_i^t\ \mathcal{N}_i^\epsilon\ \mathcal{C}_i\big]$, if $i \notin \mathcal{C}_i$,
  - $reset\big((R_k,N)\big) = \big[\mathcal{I}_i^a\ \mathcal{I}_i^t\ \{j \mid x_j \in \mathcal{B}_\epsilon(x_i)\}\ \ \mathcal{C}_i\big]$,

  for all $k \in \mathcal{I}_0$, denotes the set of resets associated with the guards of the hybrid automaton.
- $flow : V_{C_i} \rightarrow \dot{X}_{C_i}$ with $[\dot{\mathcal{I}}_i^a\ \dot{\mathcal{I}}_i^t\ \dot{\mathcal{N}}_i^\epsilon\ \dot{\mathcal{C}}_i] := flow(v) = [0\ 0\ 0\ 0]$ for all $v \in V_{C_i}$, denotes the flow conditions of the hybrid automaton that constrain the first time derivatives of the system variables in mode $v \in V_{C_i}$.

Observing Definition 4.4 we see that whenever agent i is sufficiently close to an available destination k, automaton C_i transitions to mode A_k and the set of taken destinations $\mathcal{I}_i^t$ is updated with new information from neighbors, according to $reset((N,A_k))$. If there is no need for *tie breaking* or if agent i wins the tie break, destination k is assigned to agent i, as indicated by the resets $reset((A_k,T_k))$ and $reset((B_k,R_k))$, respectively. On the other hand, if agent i loses the tie break, then $\mathcal{I}_i^a$ is updated by removing any new taken destinations, according to $reset((B_k,R_k))$. Now, if agent i is close to a taken destination or if it is far from any destination, automaton C_i transitions to mode I and exchanges information with its neighbors in order to update the sets of available and taken destinations $\mathcal{I}_i^a$ and $\mathcal{I}_i^t$, according to the resets $reset((I,N))$ and $reset((U,N))$. Note that whenever the state variable $\mathcal{I}_i^a$ is updated with new information, a transition is automatically triggered in automaton N_i due to synchronization labels "$update_i$". This synchronization models the communication between automata C_i and N_i. Similarly, in a case of a *tie* for destination k, all the involved coordination automata are synchronized according to the synchronization labels "$tiebreak_k$" to participate in a tie break where the agent with the smallest label is responsible for breaking the tie, according to $reset((A_k,B_k))$. Figure 3 shows the graph representation of hybrid automaton C_i.

V. Integrated System

Having defined the models for the agents, we now proceed with their composition in an overall product system S and study its convergence properties [18]. The following result characterizes the transition guards in S. [5]

Proposition 5.1: For any agent i and any destination $k \in \mathcal{I}_0$ such that $x_i \in \mathcal{B}_\delta(d_k)$ and $\mathcal{I}_i^a \neq \{k\}$, the product system S satisfies:

(a) $k \notin \cup_{j\in\mathcal{N}_i^\epsilon} \mathcal{I}_j^t$ if and only if destination k is available.
(b) $k \in \cup_{j\in\mathcal{N}_i^\epsilon} \mathcal{I}_j^t$ if and only if destination k is taken.

Proposition 5.1 implies that the product system S can always identify whether a destination is available or taken. The following result shows that agent i is always assigned to an available destination if it is sufficiently close to it, while it appropriately updates its sets of available and taken destinations, $\mathcal{I}_i^a$ and $\mathcal{I}_i^t$ respectively, otherwise.

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

(a) If $x_i(t) \in \mathcal{B}_\delta(d_k)$ and destination k is available at time t, then $\mathcal{I}_i^a(t) := \{k\}$ and $\mathcal{I}_i^t(t) := \mathcal{I}_i^t(t) \cup \{k\} \cup \big(\cup_{j\in\mathcal{N}_i^\epsilon(t)} \mathcal{I}_j^t(t)\big)$.
(b) If destination k is available at time t and $x_i(t) \in \mathcal{B}_\delta(d_k)$ simultaneously for multiple agents i, then S is able to break the tie.
(c) $\mathcal{I}_i^a(t) := \mathcal{I}_i^a(t) \backslash \big(\cup_{j\in\mathcal{N}_i^\epsilon(t)} \mathcal{I}_j^t(t)\big)$ and $\mathcal{I}_i^t(t) := \mathcal{I}_i^t(t) \cup \big(\cup_{j\in\mathcal{N}_i^\epsilon(t)} \mathcal{I}_j^t(t)\big)$ otherwise.

Proposition 5.2 further implies that $\mathcal{I}_i^a(t) \cap \mathcal{I}_i^t(t) = \emptyset$ and $\mathcal{I}_i^a(t) \cup \mathcal{I}_i^t(t) = \mathcal{I}_0$, whenever $|\mathcal{I}_i^a(t)| > 1$, as required in Section IV. The following proposition shows that every agent that has not yet been assigned to a destination, has always knowledge of at least all available destinations in $\mathcal{I}(t)$. This result is necessary to show that every agent will eventually be assigned to a *distinct* destination in $\mathcal{I}_0$.

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

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 can only take a finite number of transitions $v_S \xrightarrow{e_S} v'_S$ such that $sync(e_S) = update_i$ for any i, which is polynomial with respect to the number of agents n.[6] This implies that the number of assignments that can be explored is also 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 n.

Proposition 5.4: Let $v_S^\star = (v_{N_1}^\star, \ldots, v_{N_n}^\star, v_{C_1}, \ldots, v_{C_n})$ be such that $v_{N_i}^\star = 1$ and $\mathcal{I}_i^a \cap \mathcal{I}_j^a = \emptyset$ for all $j \neq i$. Then, the product system S can reach $v_S^\star$ in at most $\frac{n(n+1)}{2}$ transitions $v_S \xrightarrow{e_S} v'_S$ such that $sync(e_S) = update_i$.

Having showed that the product system S satisfies the problem specifications and has also polynomial complexity, we now 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 $x_i(t_0)$,[7] there exists a constant $T > 0$ such that for all time $t > t_0 + T$, the product system S is in mode $v_S^\star = (v_{N_1}^\star, \ldots, v_{N_n}^\star, v_{C_1}, \ldots, v_{C_n})$ with $v_{N_i}^\star = 1$ and $\mathcal{I}_i^a(t) \cap \mathcal{I}_j^a(t) = \emptyset$ for all $j \neq i$. We call $v_S^\star$ the equilibrium mode of the system.

[5] Due to space limitations, a formal definition of the product system S as well as proofs of the results in this section are omitted.

[6] A mode of the product system S is defined by the tuple $v_S = (v_{N_1}, \ldots, v_{N_n}, v_{C_1}, \ldots, v_{C_n})$ and e_S indicates a transition from mode v_S to v'_S.

[7] *Almost all* is due to the measure zero saddle points in Theorem 3.2.

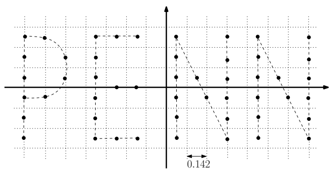
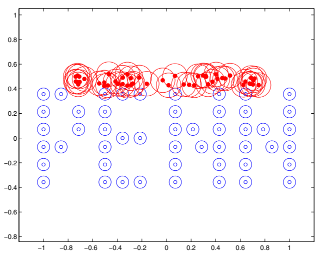


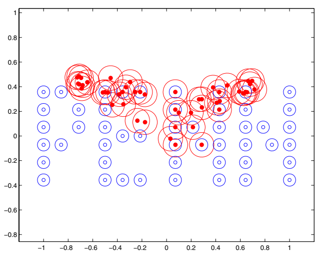
Fig. 4. Destination set $dest(\mathcal{I}_0)$. Destinations are indicated with dots.

VI. Simulation Results

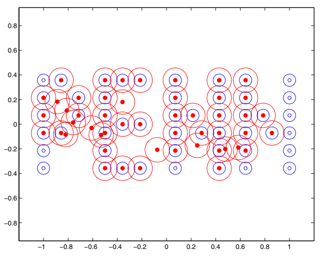
We consider a navigation task where $n = 50$ randomly initialized agents have to reach the destination set $dest(\mathcal{I}_0)$ shown in Figure 4 consisting of $m = 50$ destinations. Figures 5 show the evolution of the system at 4 different time instants. The destinations are denoted with blue small circles and the δ-neighborhoods (with $\delta = .05$) around each destination, with big blue circles. The agents, on the other hand, are denoted with red color and the ϵ-neighborhoods (with $\epsilon = .1$) of each agent, with red circles. We observe that the hybrid system S eventually drives each agent to a distinct destination. Moreover, in Figure 5(d) one can see the final paths followed by two of the agents until they reach their destinations. Note how these agents change direction of motion when they receive information about taken destinations from their neighbors, without actually visiting the taken destinations themselves.



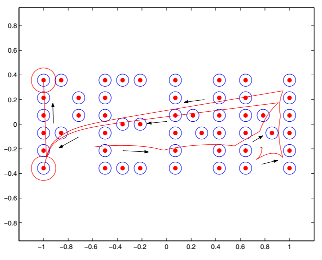
(a) Initial Configuration.



(b) Intermediate Configuration.



(c) Intermediate Configuration.



(d) Final Configuration.

Fig. 5. Simulation for $n = 50$ agents.

VII. Conclusions

In this paper, we considered the problem of determining an assignment between the agents and destinations in a multi-agent motion planning task. Navigation to any of the available destinations was achieved through provably correct multi-destination potential fields, while local coordination among the agents ensured the mutual exclusion property of the final assignment. Integration of the continuous navigation controllers with the discrete coordination protocols resulted in a hybrid model for every agent, while the overall system was shown to always converge to a valid assignment and have at most polynomial complexity, despite the exponential growth of the number of assignments with respect to the number of agents. Our approach was completely distributed, since each agent had access only to local information from its neighbors, and on-line, since the assignment was determined dynamically as the system converged to its equilibrium. Finally, the efficiency of our approach was verified through non-trivial computer simulations.

References

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

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

[3] H. Tanner, A Jadbabaie and G. Pappas. *Flocking in Fixed and Switching Networks*, IEEE Transactions on Automatic Control, April 2005. To Appear.

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

[5] R. Sepulchre, D. Paley, N. E. Leonard. *Stabilization of Planar Collective Motion: All-to-All Communication*, IEEE Transactions on Automatic Control, 2006. To Appear.

[6] S. Poduri and G. S. Sukhatme. *Constrained Coverage for Mobile Sensor Networks*, In IEEE International Conference on Robotics and Automation, pp. 165-172, New Orleans, LA, May 2004.

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

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

[9] 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.

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

[11] H. Wolkowicz. *Semidefinite Programming Approaches to the Quadratic Assignment Problem*, Nonlinear Assignment Problems: Algorithms and Applications, Combinatorial Optimization Series, 7, 143-174, Kluwer Academic Publishers, 2000.

[12] M. M. Zavlanos and G. J. Pappas. *A Dynamical Systems Approach to Weighted Graph Matching*, Proceedings of the 45th IEEE Conference on Decision and Control, pp. 3492-3497, San Diego, CA, Dec. 2006.

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

[14] M. Ji, S. Azuma, and M. Egerstedt. *Role-Assignment in Multi-Agent Coordination*, International Journal of Assistive Robotics and Mechatronics, vol. 7(1), pp. 32-40, March 2006.

[15] M. M. Zavlanos and G. J. Pappas. *Sensor-Based Dynamic Assignmnet in Distributed Motion Planning*, IEEE International Conference on Robotics and Automation, Rome, Italy, April 2007. To Appear.

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

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

[18] T. A. Henzinger. *The Theory of Hybrid Automata*, Proceedings of the 11th Annual Symposium on Logic in Computer Science (LICS), pp. 278-292, IEEE Computer Society Press, 1996.