A Scalable Distributed Algorithm for Shape Transformation in Multi-Robot Systems

Ramprasad Ravichandran[1], Geoffrey Gordon[2], Seth Copen Goldstein[3]
[1]Robotics Institute, [2] Machine Learning Department, [3]Computer Science Department
Carnegie Mellon University, Pittsburgh, PA 15213
{rravicha,ggordon,seth}@cs.cmu.edu

***Abstract*— Distributed reconfiguration is an important problem in multi-robot systems such as mobile sensor nets and metamorphic robot systems. In this work, we present a scalable distributed reconfiguration algorithm, Hierarchical Median Decomposition, to achieve arbitrary target configurations. Our algorithm is built on top of a novel distributed median consensus estimator. The algorithms presented are fully distributed and do not require global communication. We show results from simulations in an open source multi-robot simulator.**

I. Introduction

In this work we look at the geometric-formation problem where a group of robots is given a specific target configuration. This problem finds many useful applications in modular robot systems and mobile sensor networks. In modular robot systems (e.g., [11]), often, the main task is to reconfigure into different shapes. In mobile sensor networks, a scenario involves an ad-hoc deployment of mobile sensors which may then be required to rearrange into specific (possibly disconnected) formations to perform their assigned tasks (Figure 1).

Each robot in the system is usually equipped with a minimal set of sensors, actuators, processor and memory. Typically, each of these individual robots are very cheap to manufacture, and their homogeneity makes the system robust because no single robot is critical to the mission. The simplicity of the modules come at the cost of increased control complexity. Control in such robotic systems is decentralized, and communication is generally restricted to immediate neighbors. Apart from the above general challenges, geometric-formation has the additional challenge that the robots know only their own positions and have no information about the locations of other robots in the ensemble. In sensor networks, depending on the type of deployment (aerial or manual), the robots may not even know the initial shape of the ensemble.

The problem of deploying agents to form arbitrary target configurations with distributed decision-making and limited communication is still open [10]. In this paper, we present an initial solution to this problem using a novel distributed median consensus estimator to establish a bijection between the initial positions of the set of agents and the set of target positions. Each robot requires $O(\log(n))$ memory to achieve this bijection. (The focus of this work is in establishing this bijection, and not in the motion plan of the modules. [14] is a good reference for solutions to the latter problem.)

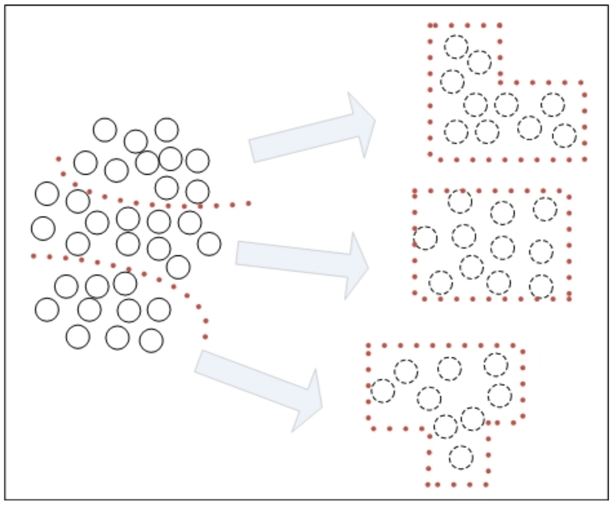


Fig. 1. An example scenario for disconnected deployment of mobile sensor nets.

II. Related Work

Reconfiguration of formations has been an active topic in the past decade. We highlight some of the related work in the sensor networks literature. Most of the past work has been on network reconfiguration in response to some stimulus or performance objective such as event monitoring [4], cooperative control [1], controlling node density [19], network coverage [5], or formation control [20]. Works such as [2], [7] looked at behavior-based approaches to maintain robot formations. In [9], the authors use an energy minimization technique to achieve target configurations. While this works for connected target configurations, it doesn't lend itself to problems, such as Figure 1, where the destination configuration is disconnected.

The current work is most related to [6], where the authors use a small number of leader nodes to reposition the agents. Unlike their approach, our algorithm requires neither leaders nor the knowledge of the initial network shape.

III. Theory

In this section we discuss the theory behind our distributed median consensus estimator. First, we talk about the distributed mean consensus algorithm, followed by a discussion of Lyapunov-based proofs. We then discuss the distributed median consensus estimator and provide mathematical intuition suggesting that it converges.

A. Background

1) Distributed Mean Consensus: In distributed consensus, a set of networked agents with possibly different initial estimates of a global variable try to reach an agreement on the value of the variable, using limited storage and computation at each agent. In distributed mean consensus, the desired consensus is the mean: each agent knows a number z_i, and the problem is to compute $\frac{1}{n}\sum_i z_i$.

Consider a system of agents whose interaction topology is represented by the undirected graph $G(V,E)$, where $V = \{1,2,\ldots,N\}$ is the set of agents and $E \subseteq V \times V$ is the set of communication links between agents. Let $x_i(t)$ denote agent i's estimate of the mean at time t. It has been shown (e.g. [16]) that the following simple linear system achieves mean consensus:

$$\mu \dot{x}_i(t) = \sum_{\{i,j\}\in E} (x_j(t) - x_i(t)) \tag{1}$$

Here $\frac{1}{\mu}$ is a gain parameter. In (1), we initialize $x_i(0) = z_i$ for each agent; the limiting value of x_i is the same for all agents, and is the desired mean consensus. More recent work has shown that similar systems can work in networks with dynamic topology and communication delays [17].

Eq. 1 can be written using a *Graph Laplacian Operator*:

$$\mu \dot{x} = -Lx \tag{2}$$

where L denotes the *Laplacian* of the connectivity graph G. If d_i is the degree of node i, the Laplacian L is defined as

$$L_{ij} = \begin{cases} d_i & \text{if} \quad i=j \\ -1 & \text{if} \quad \{i,j\}\in E \\ 0 & \text{else} \end{cases} \tag{3}$$

Eq. 2 works only if the z_i values are static. If the agents have dynamic values, say $z_i(t)$, then we can implement forgetting:

$$\mu \dot{x} = -Lx + \gamma(z - x) \tag{4}$$

Large values of the forgetting factor γ mean that we get rid of old information quickly, but unfortunately also introduce a bias so that the limiting values of x_i are no longer exactly the same for all agents. To get rid of the bias, we can add integrator variables α_i [8]:

$$\mu \dot{x} = -Lx + \gamma(z-x) - L\alpha \tag{5}$$
$$\mu \dot{\alpha} = Lx \tag{6}$$

2) Lyapunov Stability: We use Lyapunov functions [15] to prove convergence. A continuous scalar function $V(x)$: $\Re^n \to \Re_+$ is a Lyapunov function in an open region $U \subset \Re^n$ containing 0 if and only if:

- $V(x) > 0$ everywhere in U except $V(0) = 0$.
- $\dot{V}(x) < 0$ everywhere in U except at $x = 0$.

(We could equally well pick the minimum of V to occur somewhere other than $x = 0$, and let the minimum value of V be an arbitrary finite number.) One way to interpret Lyapunov stability is by analogy to a physical system with state x that loses energy $V(x)$ over time. Such a system must finally come to rest at a state of minimum energy.

B. Distributed Median Consensus

We now look at the central problem which will allow us to build our hierarchical decomposition: finding the *median* of the agents' variables.

Let $x_i(t)$ denote agent i's estimate of the median of $z_i(t)$; $g_i(t) \in [-1,1]$ denote agent i's estimate of whether $z_i(t)$ is above or below the median; and $f_i(t)$ denote agent i's estimate of the mean of the $g_i(t)$ votes. The sign of f_i tells us the side of the median where agent i believes a majority of the z_i values lie. So, if $f_i(t) = 0$, agent i believes exactly half of the agents lie on each side of the median estimate—the desired outcome. We update these estimates according to the following nonlinear system:

$$\dot{g} = -\delta(g - \mathrm{sgn}(z-x)) \tag{7}$$
$$\mu_1 \dot{f} = -(L+\gamma I)f - L\alpha + \gamma g \tag{8}$$
$$\mu_1 \dot{\alpha} = Lf \tag{9}$$
$$\dot{x} = \underbrace{-\mu_0 Lx}_{\text{fast}} + \underbrace{\mu_2 f}_{\text{slow}} \tag{10}$$

initialized with $f(0) = g(0) = \alpha(0) = 0$ and $x(0) = z(0)$.

Eq. 7 makes $g_i(t)$ tends toward $+1$ when agent i's input z_i is greater than its median estimate x_i, and -1 when it is less. Eqs. 8–9 are similar to Eqs. 5–6: f is a consensus estimate of the mean of g, and α is an integrator variable. In Eq. 10, f provides feedback to servo the median consensus estimate to settle at the true median of the system.

We use multiple time scales here to encourage the coupled estimates to converge [12]. We choose scales so that μ_0 is the fastest dynamics, and μ_2 is the slowest; μ_1 is intermediate.

C. Convergence of the Median Estimator

In this subsection, we give an intuitive argument for the convergence of our median consensus estimate (Eqs. 7, 8, 9 and 10). We do not yet have a proof of convergence.

We first show the convergence of Eqs. 8 and 9 for fixed g. (See [8] for an alternate proof which covers varying g.) Rewriting Eqs. 8–9 in matrix form, we have:

$$\begin{bmatrix} \dot{f} \\ \dot{\alpha} \end{bmatrix} = \underbrace{\begin{bmatrix} -(L+\gamma I) & -L \\ L & 0 \end{bmatrix}}_{\mathbf{M}} \begin{bmatrix} f \\ \alpha \end{bmatrix} + \begin{bmatrix} \gamma I \\ 0 \end{bmatrix} g \tag{11}$$

The gain matrix (M) can be written as a sum of a negative semi-definite matrix (N) and a skew-symmetric matrix (S):

$$M = \underbrace{\begin{bmatrix} -(L+\gamma I) & 0 \\ 0 & 0 \end{bmatrix}}_{\mathbf{N}} + \underbrace{\begin{bmatrix} 0 & -L \\ L & 0 \end{bmatrix}}_{\mathbf{S}} \tag{12}$$

So, M is negative semi-definite, which means that the system 8–9 converges to the set $(\dot{f}, \dot{\alpha}) = 0$. Hence, at convergence:

$$-(L+\gamma I)f - L\alpha + \gamma g = 0 \tag{13}$$

$$Lf = 0 \tag{14}$$

So long as the graph G is connected, Eq. 14 implies that f is a multiple of $\widehat{1}$, the vector of all ones. Substituting Eq. 14 into Eq. 13 and dividing through by γ we get:

$$f + \frac{L\alpha}{\gamma} = g \tag{15}$$

Since $L\alpha$ has no component along $\widehat{1}$, f must be the projection of g onto $\widehat{1}$. Hence each element of f is equal to the mean of g. (Note that α may be arbitrary, which is not an issue as we are only concerned about the convergence of f.)

Now return to the full system (7–10). Since μ_0 is the fastest dynamics in the system, as far as the rest of the system is concerned we are effectively operating under the constraint $Lx = 0$, so that x is a multiple of $\widehat{1}$. And, since μ_2 is the slowest dynamics, x is effectively constant in Eq. 7 (after an initial transient while Lx converges to 0).

Taking these two assumptions into account, Eq. 7 is effectively just a proportional controller acting on a constant input $x = \widehat{1}\langle x\rangle$. (Here $\langle\cdot\rangle$ denotes the average of a vector's elements.) So, g_i converges to the sign of $z_i - \langle x\rangle$ for all i. Once the g_i variables converge, Eqs. 8–9 are a mean consensus, and so f_i converges to $\langle g\rangle$ for all i.

Now, since μ_1 is adjusted to be faster than μ_2, we can assume that $\dot{f} = 0$ while analyzing Eq. 10. Substituting this assumption into Eq. 10 yields

$$\dot{x} = -\mu_0 Lx + \mu_2\langle \mathrm{sgn}(z - \langle x\rangle)\rangle\widehat{1} \tag{16}$$

Below, we argue that Eq. 16 converges. And, if Eq. 16 converges, we must have

$$\mu_0 Lx = \mu_2\langle \mathrm{sgn}(z - \langle x\rangle)\rangle\widehat{1}$$

which implies that $\langle \mathrm{sgn}(z - \langle x\rangle)\rangle = 0$ and $Lx = 0$, since Lx has no component along $\widehat{1}$. These two properties, in turn, imply that each element of x is the median of the z_is, as desired.

So, if we can choose our gains μ_0, μ_1, μ_2 so that $Lx \approx 0$, x is effectively constant in Eq. 7, and f is effectively constant in Eq. 10, we will get convergence. We have not been able to show that such gains always exist; however, our experiments suggest that we may be able to find good gains in practical examples.

It remains to argue that Eq. 16 converges. Let $\max\{x_i(t)\}$ denote the maximum median estimate among all agents at time t, and $\min\{x_i(t)\}$ denote the minimum. Then we will claim that

$$\begin{aligned} V(x) &= V_1(x) + V_2(x) \\ &= [\max\{x_i(t)\} - \min\{x_i(t)\}] + \sum_i \big|\langle x\rangle - z_i\big| \end{aligned}$$

is a Lyapunov function for Eq. 16.

First, $V_1(x)$ is positive except when x is a multiple of $\widehat{1}$. And, the second term in Eq. 16 does not alter $V_1(x)$, so $\dot{V}_1 = -\mu_0 \frac{dV_1}{dx} \cdot Lx$.

Now, $\frac{dV_1}{dx}$ is $+1$ if $x_i(t) = \max\{x_i(t)\}$, -1 if $x_i(t) = \min\{x_i(t)\}$, and 0 otherwise. If $x_i(t) = \max\{x_i(t)\}$, then the ith component of Lx is positive, since $x_i(t) \geq x_j(t)$ for all neighbors j of i; similarly, if $x_i(t) = \min\{x_i(t)\}$, then the ith component of Lx is negative. So, every term in $\frac{dV_1}{dx} \cdot Lx$ is nonnegative, which means $\dot{V}_1 \leq 0$. In fact, $\dot{V}_1 = 0$ iff $Lx = 0$ iff $V_1 = 0$.

$V_2(x)$ achieves its minimum exactly when $\langle x\rangle$ is equal to the true median z_{med}. So, V_1 and V_2 are simultaneously minimized (i.e., x is a multiple of $\widehat{1}$ and $\langle x\rangle = z_{\mathrm{med}}$) exactly when $x = z_{\mathrm{med}}\widehat{1}$. So, V is a Lyapunov function. We also have

$$\begin{aligned} \dot{V}_2 &= \sum_i \mathrm{sgn}(\langle x\rangle - z_i)\frac{1}{n}\widehat{1} \cdot \dot{x} \\ &= \sum_i \mathrm{sgn}(\langle x\rangle - z_i)(\mu_2\langle \mathrm{sgn}(z - \langle x\rangle)\rangle) \end{aligned}$$

since $\widehat{1} \cdot Lx = 0$ and $\widehat{1} \cdot \widehat{1} = n$. So,

$$\dot{V}_2 = \mu_2 n\langle \mathrm{sgn}(\langle x\rangle - z)\rangle\langle \mathrm{sgn}(z - \langle x\rangle)\rangle$$

which is negative unless $z_{\mathrm{med}} = \langle x\rangle$ already. So, V is a Lyapunov function for Eq. 16.

IV. The HMD Algorithm

The Hierarchical Median Decomposition Algorithm (HMD) uses the above median estimator iteratively to establish a bijection between the agents and the target locations. We discuss the algorithm after listing some assumptions.

A. Assumptions

Initial configuration is connected. Any agent can communicate with any other agent (though it may take more than one hop). It is important to note that the agents need not know the initial configuration of the ensemble.

Agents have a consistent coordinate system. We require the agents to have a consistent coordinate system and know their location in the coordinate system. This can be achieved in many ways. For example, they can be equipped with GPS, or they can be equipped with a simple compass and a relative coordinate system can be established algorithmically [18].

Agents have at least a coarse representation of the target configuration. In smaller systems with 10–100 sensors, it may be easy to store all the exact target locations in each sensor. But in metamorphic robots or other large systems where the number of modules can easily reach tens of thousands, it might be more efficient for each module to store only a coarse representation of the target configuration, e.g., an image. In case a coarse representation is used, the final positions of the agents will be uniformly distributed in the target region.

B. Algorithm

The HMD algorithm consists of two phases: 1) Every agent ascertains their position relative to other agents in a distributed manner, and 2) Every agent then uses this relative position to establish its position in the target shape.

Establishing relative position. This part of the algorithm essentially builds a kd-tree [3] in a distributed manner. The pseudo-code for this part is given in Figure 4.

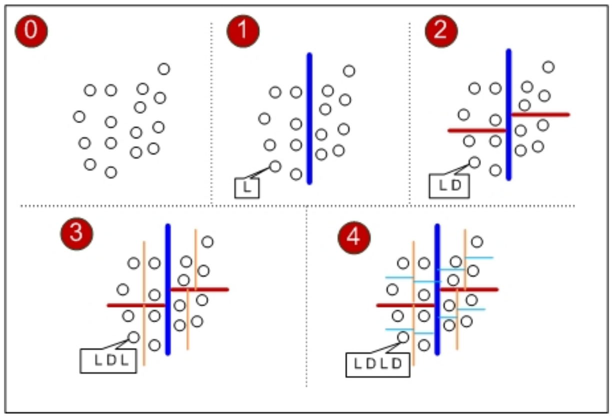


Fig. 2. Iterative Application of the Median Estimator to establish relative position. Also depicts the construction of an ID for an arbitrary node.

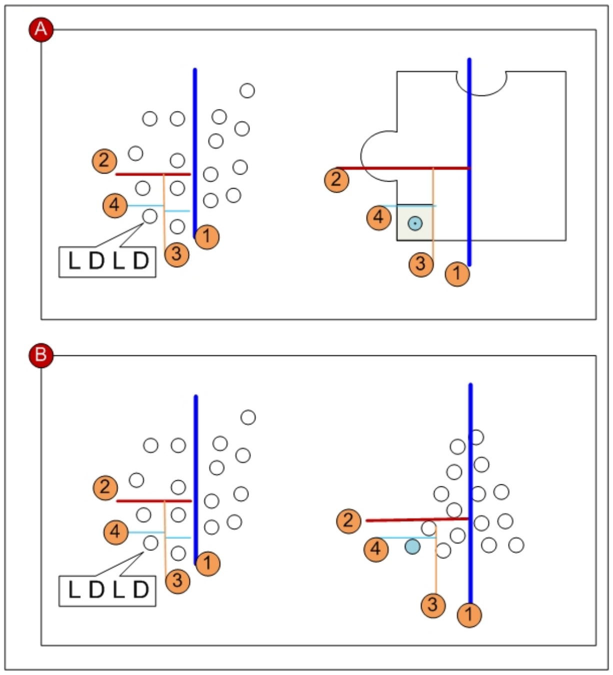


Fig. 3. Using the relative position in the initial configuration (Left-hand side) to determine the position in the final configuration (Right-hand side). Numbers indicate order of the splitting planes. (A) depicts an instance where an approximate map is known. (B) depicts an application for an exact final formation.

```
ID = [];
NbrSet = NEIGHBORS-WITH-SAME-ID(ID)
while (NbrSet is not empty)
   $x_{med}$ = GET-MEDIAN-CONS(NbrSet,ID,$Z_x$)
   if ($Z_x > x_{med}$)
     ID = APPEND-TO-ID(ID,R)
   else
     ID = APPEND-TO-ID(ID,L)
   endif
    $y_{med}$ = GET-MEDIAN-CONS(NbrSet,ID,$Z_y$)
    if ($Z_y > y_{med}$)
      ID = APPEND-TO-ID(ID,U)
    else
      ID = APPEND-TO-ID(ID,D)
    endif
 endwhile
 return ID
```

Fig. 4. Pseudo code for Phase 1 of the HMD algorithm

In the first iteration, all agents try to achieve consensus on the median value of the x-coordinate values of their position (Figure 2, Step 1). The agents first initialize their ID by a literal indicating the side of the median that they lie on (either (L)eft or (R)ight). After achieving consensus, each group of agents with the same ID, in parallel, achieves consensus on their Y-coordinate values (Figure 2, Step 2), and appends to their ID a literal ((U)p or (D)own). This median consensus takes place iteratively among the agents with the same ID, and at the end of each iteration, the number of agents with the same ID reduces until each agent has a unique ID.

Thus, when the kd-tree is completed, each agent is assigned an ID that is both unique and also determines the relative position of the agent in the system.

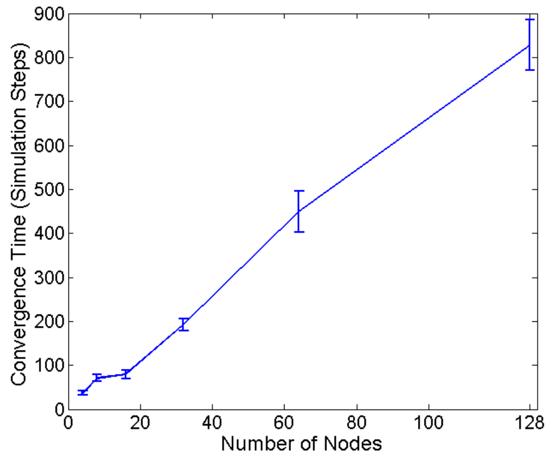
Establishing final position. In this part of the algorithm the agent tries to determine its unique final location in the target configuration. This step is basically the reverse of the previous step (see right side of Figure 3).

Each agent loops through every literal in its ID (constructed in the previous stage), and cycles through the coordinate axes to select the splitting planes in the target shape. At each step, the point selected to create the splitting plane is the median of the points being put into the kd-tree, with respect to their coordinates in the axis being used. It is possible to do this a priori as each agent knows the entire final (coarse or fine) representation.

V. Implementation

We implemented the HMD algorithm in DPRSim [21], a multi-threaded open-source simulator for testing distributed algorithms for large scale multi-robot systems. The simulator is written in C++ and every agent runs as a separate thread to support the asynchronous nature of the agents. During the actual implementation of the algorithm, we faced challenges in practical scenarios. We list some of the implementation caveats below.

Heuristic for detecting end of a median consensus stage. The HMD algorithm consists of multiple consecutive runs of the median consensus estimator. To determine when a run of the median consensus estimator terminates, we use the trailing history of the maximum and the minimum of $O(n)$ consecutive $f_i(t)$ values. If the difference between max and min remains below a small threshold, then we start the next run of the median consensus estimator. In practice however, we can get away with waiting for far fewer than $O(n)$ steps. Typically, we can start the subsequent run of the median consensus while still participating in the current run since the rare case of being initially misclassified in the subsequent run does not affect previous runs of the median consensus.



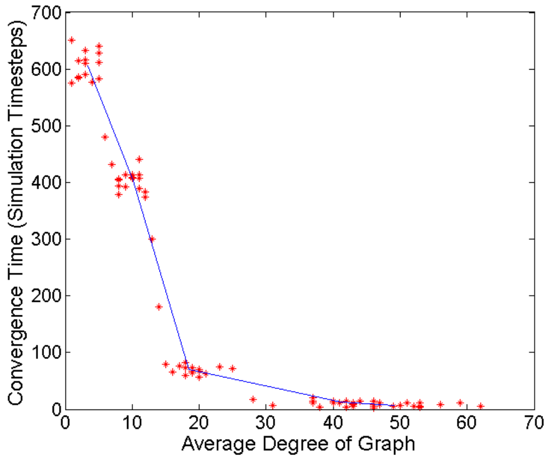


Fig. 5. Time taken for a single iteration of the Median Estimator to converge.

This is because the information flow is unidirectional from the previous run to the next run.

Tie-breaking using perturbations. In lattice-based systems such as Claytronics [11], multiple agents may share the same x or y coordinates. Since the median planes are parallel to the coordinate axes, the median plane might contain a set of such points, and classification might become more difficult. We circumvent this problem by perturbing the true position and report the perturbed value as our $z_i(0)$.

Agents that lie on the median. If there are an odd number of agents, then an agent is bound to be on the median. Hence, instead of having two values for each literal in the ID, we use three values. E.g., instead of (L)eft and (R)ight, we use (L)eft, (C)enter, and (R)ight.

Singular target configurations. We also deal with singular target configurations like straight lines by imposing constraints on the aspect ratios of each module's area in the final configuration. For instance, suppose the target configuration does not permit splitting along the Y-axis (for example, a line of 4 agents aligned with the Y-axis). In this case, we constrain all subsequent splitting to be parallel to the X-axis. Since the agents know these constraints and the target configuration beforehand, the change in the splitting pattern is deterministic. Hence, we have a unique target location assigned to each agent.

VI. Evaluation

A. Methodology

In this section we discuss the experiments we ran to demonstrate the feasability and scalability of the median consensus estimator and the HMD algorithm. We set up an $M \times M$ area in simulation with N nodes. The nodes' locations are sampled from a uniform distribution, with rejection sampling so that the corresponding adjacency graph has a single connected component. (Note that the uniform distribution tends to lead to well-connected communication graphs compared to some other initial distributions, so scaling performance might be different with different initial conditions.) We vary the number of nodes N from 2 to 128 to study the scalability of the algorithms. We also vary the radio range r of the nodes to study the role of the underlying graph's connectivity. We similarly generate random target configurations, and average over multiple trials to arrive at each data point.

We say that a system has converged at some t if for some chosen ϵ, $\max_{\tau \in (t,\infty)} |x(t) - x(\tau)| \leq \epsilon$. To measure the quality of HMD's matching, we use the sum of individual distances travelled by the agents between the initial and final positions, i.e.,

$$\text{perf} = \sum_{i \in V} \|pos_{initial,i} - pos_{final,i}\| \tag{17}$$

B. Evaluation of the Median Estimator

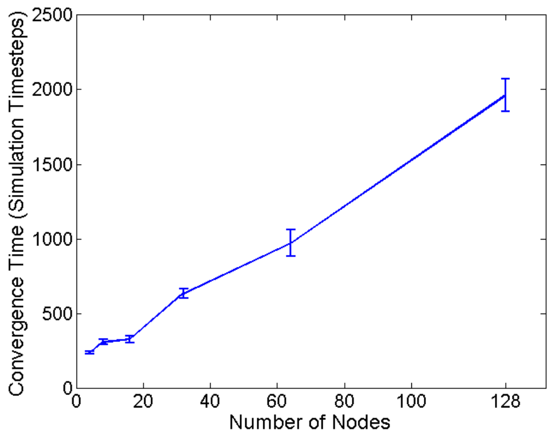
In this subsection we study the performance of the median estimator. We first varied the number of nodes in the graph from 2 to 128 nodes while measuring the convergence time of the estimator. Figure 5 shows the result of this experiment. We see that the convergence time is nearly linear in the number of nodes.

An anomaly we notice is that experiments with very few nodes exhibit a similar convergence time. We attribute this to the fact that all the experiments have an almost constant startup delay due to the simulation environment. This startup cost is comparable to the convergence time of the experiments with the lower number of nodes.

We also examined the influence of connectivity of the graph on the convergence time by keeping the number of agents fixed at 50 and varying r. We show the dependence of the convergence time on the average degree of the graph in Figure 5. As expected, the convergence time drops rapidly as the degree of the graph increases.

C. Evaluation of the HMD Algorithm

In this subsection we benchmark the running time and matching quality of the HMD algorithm. Figure 6 (top) shows that the runtime scales almost linearly with the number



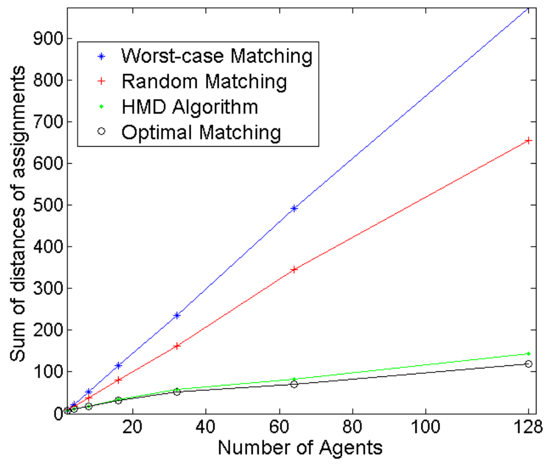


Fig. 6. Performance Comparison of the HMD algorithm.

of modules. Figure 6 (bottom) compares the matching quality of the HMD algorithm to optimal, worst- case, and random matchings. We use the centralized Hungarian algorithm to compute optimal matchings. It is important to note that this is an "unfair" comparison: to the best of our knowledge, there is no distributed version of the Hungarian algorithm that is suitable for our application.

VII. Conclusion and Discussion

We presented the HMD algorithm, a scalable distributed reconfiguration algorithm for multi-robot systems. HMD was built on top of a novel distributed median consensus estimator, and leverages the median consensus estimator in a hierarchical fashion to achieve a bijection between the nodes and the target locations. Our simulations in an open source multi-robot simulator demonstrated that the median estimator as well as the HMD algorithm scale well with increasing number of nodes and network connectivity. Further, our simulations demonstrate that the HMD algorithm performs nearly optimally on the examples considered: distance traveled is comparable to the centralized Hungarian method.

Currently we are analyzing the convergence rates and robustness of the median consensus. We are also incorporating other metrics in the HMD algorithm to support holonomic multi-robot systems with constraints. Lastly, we are analyzing the quality of the matching computed by the HMD algorithm, as compared to the optimal matching computed by the centralized Hungarian method.

Acknowledgement

This work was supported in part by NSF CNS-0428738, and Intel Corporation. We would also like to thank the Claytronics group at Carnegie Mellon for the helpful discussion, and Prof. James Kuffner in particular, for his valuable thoughts.

References

[1] R. Bachmayer and N. E. Leonard. Vehicle Networks for Gradient Descent in a Sampled Environment. In *Proceedings of IEEE Conference on Decision and Control*, 2002.

[2] T. Balch and M. Hybinette. Behavior-based coordination of large-scale robot formations. In *Proceedings of the Fourth International Conference on Multiagent Systems (ICMAS '00)*, pages 363 – 364, July 2000.

[3] J. L. Bentley. K-d Trees for semidynamic point sets. In *Proceedings of Symposium on Computational Geometry*, 1990.

[4] Z. Butler and D. Rus. Event-based motion control for mobile sensor networks. *IEEE Pervasive Computing*, 2(4), October 2003.

[5] J. Cortes, S. Martinez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. In *Proceedings of IEEE ICRA*, 2002.

[6] J. C. Derenick, C. R. Mansley, and J. R. Spletzer. Efficient motion planning strategies for large-scale sensor networks. In *WAFR*, 2006.

[7] J. Fredslund and M. Mataric. A general algorithm for robot formations using local sensing and minimal communications, 2002.

[8] R. A. Freeman, P. Yang, and K. M. Lynch. Distributed estimation and control of swarm formation statistics. In *Proceedings of American Control Conference*, 2006.

[9] A. Ganguli, J. Cortes, and F. Bullo. On rendezvous for visually-guided agents in a nonconvex polygon. In *Proceedings of IEEE CDC-ECC*, 2005.

[10] A. Ganguli, S. Susca, S. Martinez, F. Bullo, and J. Cortes. On collective motion in sensor networks: sample problems and distributed algorithms. In *Proceedings of IEEE CDC-ECC*, 2005.

[11] S. C. Goldstein, J. D. Campbell, and T. C. Mowry. Programmable matter. *Computer*, 38(6), 2005.

[12] P. Kokotovic, H. Khalil, and J. O'Reilly. Singular perturbation methods in control: Analysis and design. *Academic Press, pp. 344*, 1986.

[13] H. W. Kuhn. The hungarian method for the assignment problem. In *Naval Research Logistic Quarterly*, pages 83 – 97, 1955.

[14] S. M. LaValle. *Planning Algorithms*. Cambridge University Press.

[15] A. Lyapunov. *Stability of Motion*. Academic Press, New-York and London, 1966.

[16] M. Mehyar, D. Spanos, J. Pongsajapan, S. Low, and R. Murray. Asynchronous Distributed Averaging on Communication Networks. *IEEE Transactions of Networking*, August 2007.

[17] R. Olfati-Saber and R. M. Murray. Consensus Problems in Networks of Agents with Switching Topology and Time-Delays. *IEEE Transactions on Automatic Control*, 49, September 2004.

[18] G. Reshko. Synthetic reality: Communication and localization. Master's thesis, Carnegie Mellon University, August 2004.

[19] B. Zhang and G. S. Sukhatme. Controlling sensor density using mobility. In *Proceedings of IEEE Workshop on Embedded Networked Sensors*, 2005.

[20] F. Zhang, M. Goldgeier, and P. S. Krishnaprasad. Control of small formations using shape coordinates. In *Proc. of IEEE International Conf. of Robotics and Automation*, 2003.

[21] DPRSim. http://www.pittsburgh.intel-research.net/dprweb/.