Internal Localization of Modular Robot Ensembles

Stanislav Funiak, Padmanabhan Pillai, Jason Campbell, and Seth Copen Goldstein

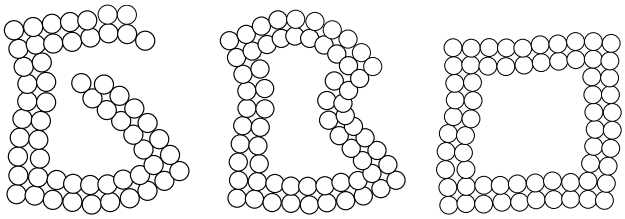
Abstract— **The determination of the relative position and pose of every robot in a modular robotic ensemble is a necessary preliminary step for most modular robotic tasks. Localization is particularly important when the modules make local noisy observations and are not significantly constrained by inter-robot latches. In this paper, we propose a robust hierarchical approach to the *internal localization* problem that uses normalized cut to identify subproblems with small localization error. A key component of our algorithm is a simple method to reduce the cost of normalized cut computations. The result is a robust algorithm that scales to large, non-homogeneous ensembles. We evaluate our algorithm in simulation on ensembles of up to 10,000 modules.**

I. INTRODUCTION

A key challenge in scaling to very large modular robotic ensembles is *internal localization*, the establishment of relative pose amongst the robot's many individual components. In systems such as Claytronics [1], internal localization must be performed on ensembles consisting of many thousands to millions of tiny modules, using only local sensing information between neighboring modules. The scale of the system, the unconstrained manner in which spheres can pack together, and uncertain intermodule sensing make developing a robust and scalable algorithm for internal localization a significant challenge.

Existing work on internal localization falls into two general categories. Constraint-based approaches [2], [3], [4] rely on strong assumptions about the ensemble structure or exact observations to scale up to large ensembles. Typically, these approaches resolve uncertainty locally, by using exact observations and geometric constraints and then propagate the solution to the rest of the ensemble. While scalable, they are neither robust to noise nor irregular, non-lattice structure. Local probabilistic approaches [5], employed in systems such as PolyBot [6], address the robustness aspects of the internal localization problem. By combining a forward kinematic model with local sensing, these approaches can eliminate as much as 90% of the positioning error. The positioning error can be further reduced using the system's mechanical latching.

While local probabilistic approaches work very well at a small scale, they tend to quickly accumulate error as the size of the ensemble increases, especially in the absence of mechanical latching. In order to obtain accurate position estimates, a scalable solution must combine observations from several modules and obtain the most probable position globally; thus, we use a global probabilistic approach to internal localization.



(a) incremental (b) incremental (c) minimum cut

Fig. 1. Comparison of a naive incremental approach and the proposed method. The total number of iterations is the same in both solutions. (a) Intermediate incremental solution before the loop closure. (b) Final solution, obtained by the incremental approach. (c) Solution, obtained with the proposed approach.

Global probabilistic localization is a challenging goal: the dense, mesh-like structure makes it difficult to apply sparse graph approaches, such as junction tree thinning [7], and tight local constraints slow down the convergence of iterative solutions, such as generalized belief propagation [8]. A simple, but effective approach is to compute the most likely positions for progressively larger sets of modules, for example, by incrementally adding modules and their observations in a breadth-first manner from a given anchor node. Similar strategies have been used to speed up Euclidean embedding approaches in wireless sensor network localization [9].

Despite the fact that the global probabilistic inference incorporates multiple observations per node, a naive implementation will still accumulate error when the ensemble is not homogeneous. For example, if the ensemble has a region with only a few inter-module observations (which we call a *weak region*), substantial rotational uncertainty will be introduced in the partial solution, and will be magnified by subsequent additions. However, if we incorporate the observations in the densely connected regions first, then the partial solution will be constrained and the error will be substantially reduced. We use this intuition to formulate a hierarchical algorithm, where we recursively split the ensemble connectivity graph into well-connected components along weak boundaries. We formulate the splitting procedure as a normalized cut problem [10] which minimizes the number of connections between components, while maximizing the connectivity within each component. Unfortunately, finding the exact normalized cut in a large graph is computationally expensive. We address this issue by computing normalized cut on an abstraction of the connectivity graph of the

This work was supported in part by Intel Corporation and NSF grant CNS-0428738.

S. Funiak is with the Robotics Institute, Carnegie Mellon University; S. C. Goldstein is with the Computer Science Department, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA, USA

P. Pillai and J. Campbell are with Intel Research Pittsburgh, 4720 Forbes Avenue, Pittsburgh, PA USA

ensemble. This abstraction preserves the high-level information about the connectivity of the graph and substantially decreases the computational complexity of the approach, with only a negligible decrease in accuracy.

We evaluate out algorithm on realistic 2D problems with up to 10,000 modules that accurately model unreliable observations and physical interactions among the modules. We show that the computational complexity of the approach is nearly linear in the size of the ensemble for a fixed ensemble structure. In our experiments, the proposed approach outperforms methods from wireless sensor network localization based on classical multidimensional scaling [11] and semidefinite programming (SDP) relaxations [12], [13], as well as simpler incremental heuristics. These results suggest that dense structures in internal localization may impose strong geometric constraints that are perhaps more effectively approximated with a single vector of most likely positions than the hop-count and SDP relaxations in Euclidean embedding methods.

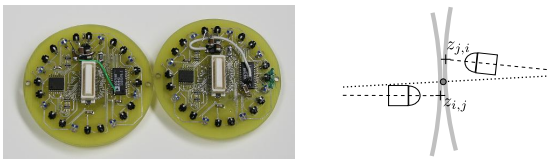
II. LOCALIZATION OF MODULAR ENSEMBLES

We assume that the location of each module can be described by a small number of parameters, such as the coordinates of its center and orientation in space. In this paper, we focus primarily on circular and spherical modules in 2D and 3D space, respectively. Each module is equipped with sensors, such as infrared transmitters/receivers, that allow a pair of modules to detect when they are in close proximity. Such observations are inherently uncertain: two modules may be in sensing range, but not in physical contact, or a measurement can be made when sensors are not aligned. We do, however, assume that (i) the observations are symmetric (that is, whenever module i observes module j then module j also observes module i), and (ii) the modules know the identity of modules they sense (that is, we do not need to address the data association problem).

The problem considered in this paper is motivated by localization in Claytronics [1]. Claytronics envisions very large ensembles of small modules, named catoms, which cooperatively form material that can change its shape under software control. Figure 2(a) shows a current working prototype of the sensing subsystems from two modules. Each module has 8 IR transmitters and 16 IR receivers, spaced evenly around the module, oriented radially outwards. Multipath interference, scattering, shadowing, and small dimensions preclude techniques such as acoustic or radio time-of-flight-based localization. Also, due to the lack of mechanical latching, modules cannot rely on strong mechanical constraints for accurate alignment and orientation.

III. LOCALIZATION AS PROBABILISTIC INFERENCE

In this section we define the probabilistic model that underlies our algorithm. In order to represent the uncertainty of a set of module locations, we use a probabilistic model that describes the probability of a joint assignment of module locations $\mathbf{X} = (X_1, \ldots, X_N)$, given observations $\mathbf{Z}$ made by



(a) module prototypes (b) sensor model

Fig. 2. (a) Sensor board from module prototype. (b) Sensor model, used in the paper. Each observation $z_{i,j}$ is represented as the location of the sensor, projected to the perimeter of the module. The circle indicates the midpoint of the two modules centers. The model penalizes the module locations x_i and x_j, based on the distance between the midpoint and the observations $z_{i,j}$ and $z_{j,i}$.

all modules in the ensemble. The location of each module i is represented by a vector, $X_i \triangleq (C_i, R_i)$, where C_i is the center of the module and R_i is its orientation (represented either as an angle θ_i in two dimensions, or a quaternion in three dimensions). We assume a uniform prior distribution on module position and orientation. This model does not explicitly represent the constraint that the modules must not overlap; instead, we have chosen to rely on the observations to obtain a non-overlapping solution.

When two modules i and j are in the immediate neighborhood of each other, a pair of observations $(z_{i,j}, z_{j,i})$ is generated which represent the sensors at module i and j, respectively, that made the observation. We use a discrete model that captures whether two modules observe each other (and with which sensors) but not the intensity of the readings (the methods presented in this paper generalize to more accurate models). Also, for simplicity of notation, we assume that there is at most one pair of observations for every pair of modules, and we take $z_{i,j}$ to be the location of the sensor at module i, in module i's local reference frame (see Figure 2(b)). The probability of making an observation $z_{i,j}$ depends on the relative locations and orientations of modules i and j, and is highest when the sensors are roughly aligned and the modules centers c_i and c_j are a unit distance apart. These properties are captured in a model that penalizes the observation $z_{i,j}$, based on how well they predict the midpoint of the two module centers.

$$p(z_{i,j}|x_i, x_j) \quad \propto \quad \exp\left\{-\frac{1}{2}\left\|R_i z_{i,j} - \frac{c_j - c_i}{2}\right\|_2^2\right\} \quad (1)$$

Alternatively, we could use a more accurate model that captures properties of IR transmitters and receivers, such as quadratic decay and multi-modal response.

Equation 1 is the conditional density that defines an *observation model* that specifies, given a pair of module locations x_i, x_j, how likely the nodes are to make the observations $z_{i,j}$ (and $z_{j,i}$). Combining the observation model (1) for each pair of neighboring modules i, j and instantiating the observations $z_{i,j}$ gives the likelihood of the joint state $\mathbf{x}$:

$$p(\mathbf{z}|\mathbf{x}) = \prod_{i,j} p(z_{i,j}|x_i, x_j). \quad (2)$$

For internal localization, we wish to compute the maximum likelihood estimate (MLE) of the location of all the modules,

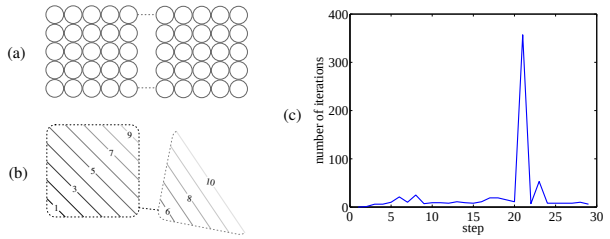


Fig. 3. (a) Ensemble, consisting of two tightly connected clusters. The clusters are connected by two pairs of observations. (b) Intermediate result, obtained when incrementally conditioning on observations, starting from the lower left corner. The numbers indicate the order of conditioning. The solution accumulates substantial error; this error takes several iterations to resolve with conjugate gradient descent, as shown in (c).

given all observations $\mathbf{z}$:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmax}}\, p(\mathbf{z}|\mathbf{x}), \tag{3}$$

up to some global translation and rotation.

IV. OPTIMIZED ORDERING OF OBSERVATIONS

It is not easy to maximize the likelihood (2) directly, since the likelihood function is non-convex and high-dimensional. One approach is to compute the solution (3) incrementally, that is, compute the maximum likelihood estimate

$$\mathbf{x}_A^* = \arg\max p(\mathbf{z}_A|\mathbf{x}_A) = \arg\max \prod_{i,j\in A} p(z_{i,j}|x_i, x_j)$$

for progressively larger sets of modules A. Here, $\mathbf{x}_A$ denotes the locations of the modules in A and $\mathbf{z}_A$ denotes all observations among the modules in A. At each step, we use the current estimate $\mathbf{x}_A^*$ to initialize the module locations for the next set of modules around the perimeter of A. A similar approach has been employed in wireless sensor network localization [9] and is analogous to Simultaneous Localization and Mapping (SLAM), where observations among modules (landmarks) are made over time.

Figure 3 illustrates the behavior of the incremental approach on a small ensemble with 200 modules that consists of two dense components. Within each component, modules make observations with all of their immediate neighbors, whereas the two components share only two observations, one at each side. Suppose that we incorporate observations in breadth-first order, starting from the lower left corner. Figure 3(c) shows the running time of the algorithm at each step, expressed as the number of iterations of conjugate gradient descent until convergence. We see that while the number of iterations is typically small, it increases dramatically midway through the experiment when the observations close a loop, formed by the two square components. Intuitively, the computed solution accumulates error that takes a long time to resolve once the algorithm closes the loop.

The experiment in Figure 3 points to an important drawback of the incremental approach. A simple maximum likelihood estimate representation may not be accurate throughout the execution of the algorithm: when the relative locations of modules are uncertain, a single observation that closes the loop may significantly shift the estimate. Nevertheless if we first incorporate the observations within each dense cluster and defer the observations that join the two clusters until the very end, then the relative locations of modules within each cluster would be certain, and we can combine the two clusters in a single step, using rigid body alignment. In short, the order in which observations are incorporated has a substantial effect on performance of the incremental approach. The differences are even more pronounced for larger ensembles with more complicated, non-uniform structure. It is therefore desirable to seek orderings that minimize the running time and the approximation error.

In large-scale localization, the main sources of uncertainty are the weak regions of the ensemble, i.e., regions where only a few observations are made. Weak regions can lead to substantial rotational error. An effective heuristic is to incorporate the observations among a subset of modules that are tightly correlated, and defer the observations in the weak regions until later. This heuristic can be naturally formulated as a graph cut problem: starting from the connectivity graph G of the complete ensemble (that is, a graph, whose edges correspond to observations between modules), we seek to partition G, such that each component is well-connected and the inter-component observations are as few as possible. This intuition is similar to normalized cut [10]:

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)}, \tag{4}$$

where $Ncut(A, B)$ is the cut value, $cut(A, B)$ is the number of observations between module sets A and B, and $assoc(A, V)$ is the number of observations between the modules A and all other modules in the graph. Minimizing (4) yields a partition of G into two components A, B. Our heuristic will first incorporate observations within the clusters A and B, and then the observations between A and B.

Intuitively, normalized cut prefers partitions such that the number of observations between A and B is small, compared to all observations made by A and B. For example, in Figure 3a, the vertical cut that separates the two well-connected components has value $Ncut = O(\frac{1}{N})$, where N is the number of modules, whereas the value of the horizontal cut is $Ncut = O(\frac{1}{\sqrt{N}})$. Indeed, we see that normalized cut discriminates these two orderings very well and yields the correct ordering.

V. INTERNAL LOCALIZATION AS NORMALIZED CUT

As suggested in the previous section, normalized cut provides an intuitive heuristic for ordering conditioning operations: given a normalized cut of the connectivity graph G for an ensemble, we first condition on observations within the individual components and then condition on observations across the cut. By applying this rule recursively, we obtain an algorithm that hierarchically partitions the ensemble into progressively smaller components which are then merged bottom-up.

A key component of our algorithm is a method to reduce the cost of normalized cut computations, as described in Section V-B. As the size of the ensemble increases, the algorithm becomes impractical, since the cost of computing normalized cut increases super-linearly with ensemble size and quickly dominates other operations. We propose a heuristic that speeds up the computation of the normalized cut, by performing the computation on a smaller graph that abstracts the structure of the original ensemble.

A. Summary of the algorithm

The proposed approached is summarized in Algorithm 1. The algorithm starts by computing the normalized cut (A, B) for the connectivity graph G. By applying the localization procedure recursively, the algorithm computes the partial solution for the modules A, conditioned on all observations among A (in the algorithm description, G_A denotes the subgraph induced by A) and similarly for the modules B. We then use the partial solutions $\mathbf{x}_A^*$ and $\mathbf{x}_B^*$ to initialize the search for the optimum solution for the entire graph: we transform the observations between A and B, $\mathbf{z}_{A,B} \triangleq \{z_{i,j} : i \in A, j \in B\}$, into the global coordinate frame, using the module locations given by the partial solution $\mathbf{x}_A^*$; similarly for modules B. This procedure yields two sets of points $p = \{p_i\}$ and $q = \{q_i\}$, such that p_i and q_i are locations of matching observations in the global coordinate frame. Recall that the likelihood is maximal when sensors are in close proximity; thus, an effective initialization is to hold the relative locations of modules fixed within each cluster A, B, and compute the optimal rigid body transform between the clusters:

$$\arg \min_{R \in SO(d), t \in \mathbb{R}^d} \sum_k \|p_k - (Rq_k + t)\|_2^2 , \qquad (5)$$

where R is the rotation matrix (in 2D or 3D) and t is the translation vector. The optimal rigid body alignment (5) can be computed with closed-form solution in time linear in the number of observations between A and B [14]. This procedure yields an initial estimate of the locations of all modules, $\mathbf{x}_V^0$. The initial estimate is then refined using iterative methods, such as conjugate gradient descent or a quasi-Newton method.

B. Scaling up the solution

While the normalized cut formulation yields an effective order to incorporate observations, computing the exact normalized cut is too costly and dominates other operations. Specifically, the cost of the rigid alignment is linear in the number of observations and, as described in Section VI-B, often yields accurate initialization that is quickly refined with local methods. On the other hand, the complexity of computing a *single* normalized cut is $O(|V|^{3/2})$, where $|V|$ is the number of nodes in the graph [10], and the computation is difficult to distribute. Nevertheless, in large-scale internal localization, modules often form large clusters, and offsetting the cut by a few nodes does not substantially decrease the quality of the solution. This observation suggests that we may be able to construct an abstraction of the original connectivity graph G, whose best normalized cut is similar to that of G.[1]

Algorithm 1 $NormCutLocalize(G, V)$

1: Compute the normalized cut $(A, B) = NormCut(G)$
2: $\mathbf{x}_A^* \Leftarrow NormCutLocalize(G_A)$
3: $\mathbf{x}_B^* \Leftarrow NormCutLocalize(G_B)$
4: $p \Leftarrow$ transform the observations $\mathbf{z}_{A,B}$ into the coordinate frame, given by $\mathbf{x}_A^*$.
5: $q \Leftarrow$ transform the observations $\mathbf{z}_{B,A}$ into the coordinate frame, given by $\mathbf{x}_B^*$.
6: Compute the optimal rigid alignment R, t:

$$\arg \min_{R \in SO(d), t \in \mathbb{R}^d} \sum_k \|p_k - (Rq_k + t)\|_2^2 ,$$

7: Let $\mathbf{x}_V^0 = (\mathbf{x}_A^*, R\mathbf{x}_B^* + t)$.
8: $\mathbf{x}_V^* \Leftarrow \arg\max \log p(\mathbf{x}_V | \mathbf{z}_V)$, starting from $\mathbf{x}_V^0$

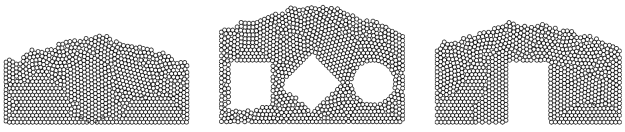
Let G be the connectivity graph on which we wish to perform normalized cut computations. Suppose that we partition G into K components $(V_1, V_2, \ldots, V_K)$, and create an abstracted graph G' with K vertices, such that two nodes i and j in G' are adjacent if and only if the corresponding components V_i and V_j are adjacent in G. Furthermore, suppose that the weight $w'_{i,j}$ of an edge in G' is equal to the total number of edges between V_i and V_j in G. Then any cut on G' corresponds to a cut on G, with the same cost $Ncut$. By performing a normalized cut on G', we are effectively restricting the solution to the cuts, induced by the partition $(V_1, \ldots, V_K)$. Since the connectivity graphs in modular robot ensembles are embedded in 2D or 3D space, offsetting a cut increases its value at most linearly (in 3D, quadratically) in the number of hops. Therefore performing normalized cut on the abstracted graph G' is often as good as performing normalized cut on the original connectivity graph.

In order to implement the above strategy, we greedily partition the graph into a set of fixed components of a fixed granularity (given by an approximate number of components created by the partitioning procedure). At each level of the hierarchy, we form a new abstracted graph G' of the given granularity. As described below, we obtain accurate solutions with as few as twenty components.

VI. EXPERIMENTAL RESULTS

We implemented the centralized version of the proposed algorithm in Matlab and generated input scenarios in a C++ simulator [15] that models IR sensing and physical interactions between the modules. Each module in the simulation has 12 IR transceivers (colocated emitter/detector pairs), whose IR response follows an inverse-square law, similar to the model in [5]. The threshold for detecting observations

[1] In other clustering applications, such as image segmentation, where the affinity matrix is dense, it is often possible to subsampled the *edges* of the graph at a minimal loss of quality. However, the connectivity graphs in internal localization are already sparse, and we need to seek a different approximation.



(a) solid (b) triple (c) open

Fig. 4. Scenarios used in our experiments. The scenarios were generated by settling randomly inserted modules in a gravitational field.

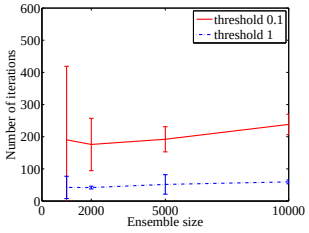


Fig. 5. The average number of iterations per module for the triple scenario as the number of modules grows. At different scales, the ensemble retains its shape and the proportions.

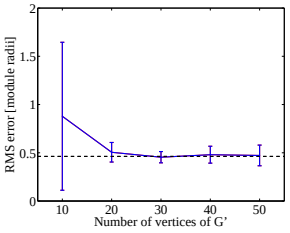
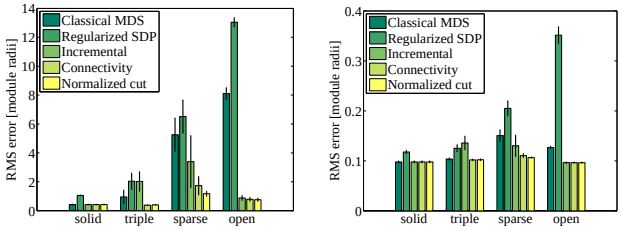


Fig. 6. The location RMS error for the triple scenario with 2000 modules, when using the normalized cut approximation in Section V-B. The horizontal dashed line indicates the fidelity of the solution, obtained with exact normalized cut.



(a) global (b) local

Fig. 7. RMS error of the location estimates. (a) Global RMS error, averaged over all modules. (b) RMS error of modules relative to their neighbors. Here, we greedily partition the ensemble into connected regions with diameter of 6 modules or less and compute the RMS error using the optimal rigid alignment for each region.

between a pair of neighboring nodes was set to 20 per cent of the peak intensity. At this setting, a sensor reports a positive reading even if the modules are not in a physical contact and if the transmitter and the receiver are not perfectly aligned.

A. Scenarios

For the experiments, we generated several simulated ensemble configurations, illustrated in Figure 4 that mimic 2D slices of a 3D shape capture scenario [2]. The configurations were generated by placing the modules in a gravitational field above a fixed container of the desired shape. The result are configurations with realistic, irregular structures. Each shape in Figure 4 was instantiated 10 times, with different initial velocities and locations of the modules. These repeated runs generated configurations that look similar, but whose module connectivity and spacing varies.

B. Scalability

In the first experiment, we evaluated the performance of the proposed method as the number of modules in an ensemble increases. We selected the structured triple scenario in Figure 4(b) and formed a set of progressively larger ensembles. At each scale, the ensemble retains the same shape and the proportions, but the number of modules that form the shape increases. We then run Algorithm 1 such that, at each level of the hierarchy, the estimate $\mathbf{x}_A^*$ reaches a fixed level of accuracy, as measured by the norm of the gradient of the likelihood function at $\mathbf{x}_A^*$. This procedure ensures that each estimate $\mathbf{x}_A^*$ is sufficiently accurate, before it is used at the higher level. Figure 5 shows the resulting number of iterations of preconditioned conjugate gradient descent per module, averaged over ten executions, for two choices of the threshold on the gradient norm. We see that the number of iterations, needed to attain the same accuracy, increases only moderately with the size of the ensemble. Note that the gradient threshold controls the fidelity of the solution indirectly. With a threshold of 1.0, the average location RMS error was 1.26; with a threshold of 0.1, the average location RMS error was 0.80. As we decrease the gradient threshold to 0, the solution attains the fidelity of the maximum likelihood estimate (3).

C. Sensitivity to abstraction

In the second experiment, we evaluated the sensitivity of the proposed localization method to errors, introduced by performing normalized cut on the abstracted, rather than the original connectivity graph. We took the structured scenario in Figure 4(b) with 2000 modules. In order to keep the execution time constant, we perform a small fixed number of iterations at the intermediate levels of hierarchy and 300 iterations of preconditioned conjugate gradient descent at the top level. Figure 6 shows the root mean square (RMS) error as we vary the number of nodes in the abstraction of the connectivity graph (since we controlled the diameter of clusters, rather than their count, the displayed number of nodes is approximate). In order to account for the overlap, introduced by the objective (1), we uniformly scale the locations of the modules, so that the average spacing equals the module diameter.[2] Then, using the ground truth locations of the modules, we compute an optimal rigid alignment and report the error for the aligned solution. We see that the performance of the proposed localization method is insensitive to abstraction errors: with 20 or more nodes in the abstracted graph, the approach yields a sufficiently small RMS error (with smaller examples, the error was even less pronounced). These results suggest that small graph abstractions provide meaningful results and can be analyzed at each leader node centrally.

[2]Similarly, we scaled the locations of the ground truth modules to account for the overlap, introduced by the physics engine. If this scaling is not desirable (e.g., because the modules can be compressed), we could use a few externally calibrated anchors to minimize the distortion.

D. Comparative evaluation

In the third set of experiments, we compared the performance of the proposed algorithm to Euclidean embedding methods, used in wireless sensor network localization, as well as simpler incremental and hierarchical heuristics. We evaluated the following methods: (i) classical multi-dimensional scaling [11], (ii) the inequality formulation of regularized semidefinite programming [12], (iii) the simpler incremental approach, discussed in Section IV, (iv) a simple hierarchical approach that merges pairs of clusters bottom-up, in the order given by their algebraic connectivity[3], and (v) the proposed method, using exact normalized cut. We perform repeated experiments on the scenarios in Figure 4 with 1000 modules. The initial solution, obtained by each method is refined with 300 iterations of preconditioned conjugate gradient descent. In addition, for the last three methods, we perform 10 steps of preconditioned gradient descent at each iteration.

Figure 7 shows the average RMS error for each scenario. We see that approaches, based on Euclidean embedding (classical MDS, regularized SDP) generally do not perform very well in this setting, especially for the sparse version of the triple scenario and the large open-loop scenario. For classical multi-dimensional scaling, the error results from approximating true distances with hop-count; for regularized SDP, the errors come either from the SDP relaxation or the underlying solver. The incremental and simple hierarchical approaches perform better, but are outperformed by our normalized cut formulation on the scenarios with non-homogeneous structure (triple, sparse). It is worth noting that the Euclidean embedding methods are substantially more computationally expensive: an optimized implementation of a state-of-the-art SDP relaxation method [13] takes 5-10 minutes to run on an input with 5000 nodes, whereas the Matlab implementation of our hierarchical algorithm runs in less than a minute.

VII. DISCUSSION AND FUTURE WORK

In this paper, we examine robust, large-scale localization in modular robot ensembles from uncertain, local observations. We formulate internal localization as a probabilistic inference problem and introduce a novel approach that selects an effective ordering of observations, by computing an approximate normalized cut on the connectivity graph of the ensemble. We propose a method to approximate the cut by performing normalized cut on a rough abstraction of the connectivity graph. This approximation provides a substantial speed-up with only a negligible loss of accuracy. We perform an extensive evaluation of the proposed approach on a test suite of realistic configurations with up to 10,000 nodes. We demonstrate that the method scales very well and outperforms recent methods in Euclidean embedding as well as simpler incremental heuristics.

[3]Algebraic connectivity is defined as the second smallest eigenvalue of the Laplacian of the connectivity matrix; this value is zero if and only if the graph is disconnected.

There are several directions to extend the work, presented in this paper. A natural extension is to implement Algorithm 1 on a distributed system with neighbor-to-neighbor communication, using a combination of leader election and data aggregation techniques. It would also be interesting to exploit the geometric information, encoded in the connectivity graph. For example, the normalized cut is indifferent to the spacing of the contact points along the cut. A heuristic that is aware of the geometry encoded in the graph may perform better for structures with large loops. Finally, it may be possible to perform splitting and alignment operations on several clusters at a time. This extension would address scenarios where several components are connected by relatively few observations; incorporating all of these observations at once can avoid introducing bias in the intermediate estimates.

VIII. ACKNOWLEDGMENTS

The authors thank Rahul Sukthankar and Dhruv Batra for their valuable feedback. We also thank Casey Helfrich and Michael Ryan for building the DPRSim simulator which was used for many of our experiments Finally, we thank Rahul Biswas for his implementation of the regularized SDP approach to sensor network localization.

REFERENCES

[1] S. C. Goldstein and T. Mowry, "Claytronics: A scalable basis for future robots," in *Robosphere*, Nov 2004.

[2] P. Pillai, J. Campbell, G. Kedia, S. Moudgal, and K. Sheth, "A 3D fax machine based on claytronics," in *IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, Oct. 2006, pp. 4728–35.

[3] P. White, V. Zykov, J. Bongard, and H. Lipson, "Three dimensional stochastic reconfiguration of modular robots," in *Proceedings of Robotics Science and Systems*, 2005.

[4] G. Reshko, "Localization techniques for synthetic reality," Master's thesis, Carnegie Mellon University, 2004.

[5] K. D. Roufas, Y. Zhang, D. G. Duff, and M. H. Yim, "Six degree of freedom sensing for docking using IR LED emitters and receivers," in *Proceedings of International Symposium on Experimental Robotics VII*, 2000.

[6] M. Yim, D. Duff, and K. Roufas, "PolyBot: a modular reconfigurable robot," in *Proceedings of IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2000.

[7] M. A. Paskin, "Thin junction tree filters for simultaneous localization and mapping," in *Proc. IJCAI*, 2003.

[8] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Generalized belief propagation," in *NIPS*, 2000, pp. 689–695.

[9] P. Biswas, T.-C. Lian, T.-C. Wang, and Y. Ye, "Semidefinite programming based algorithms for sensor network localization," *ACM Trans. Sen. Netw.*, vol. 2, no. 2, pp. 188–220, 2006.

[10] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[11] Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz, "Localization from mere connectivity," in *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*. ACM Press New York, NY, USA, 2003, pp. 201–212.

[12] P. Biswas, T.-C. Lian, T.-C. Wang, and Y. Ye, "Semidefinite programming based algorithms for sensor network localization," *ACM Transactions on Sensor Networks (TOSN)*, vol. 2, no. 2, pp. 188–220, May 2006.

[13] Z. Wang, S. Zheng, S. Boyd, and Y. Yez, "Further relaxations of the SDP approach to sensor network localization," Stanford University, Tech. Rep., 2006.

[14] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, April 1991.

[15] "Dprsim: The dynamic physical rendering simulator," http://www.pittsburgh.intel-research.net/dprweb/.