Distributed Localization of Modular Robot Ensembles

Stanislav Funiak Michael P. Ashley-Rollman
and Seth Copen Goldstein
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{sfuniak, mpa, seth}@cs.cmu.edu

Padmanabhan Pillai
and Jason D. Campbell
Intel Research Pittsburgh
Pittsburgh, PA 15213, USA
{padmanabhan.s.pillai, jason.d.campbell}@intel.com

***Abstract*— Internal localization, the problem of estimating relative pose for each module (part) of a modular robot is a prerequisite for many shape control, locomotion, and actuation algorithms. In this paper, we propose a robust hierarchical approach that uses normalized cut to identify dense subregions with small mutual localization error, then progressively merges those subregions to localize the entire ensemble. Our method works well in both 2D and 3D, and requires neither exact measurements nor rigid inter-module connectors. Most of the computations in our method can be effectively distributed. The result is a robust algorithm that scales to large, non-homogeneous ensembles. We evaluate our algorithm in accurate 2D and 3D simulations of scenarios with up to 10,000 modules.**

I. Introduction

Large self-reconfigurable modular robots have received a growing interest from the robotics community. A self-reconfigurable modular robot (SRMR) is comprised of many discrete, physically connected modules which can be rearranged to adapt the robot's shape or capabilities to the task at hand. These robot ensembles have been proposed for various applications, such as product design and visualization [1], emergency search and rescue, and rapid prototyping [2, 3]. A fundamental task in such robot ensembles is *internal localization*, the establishment of relative pose amongst the robot's many individual components. Accurate internal localization is required for many tasks, including motion planning, mechanical stability, and control.

Internal localization for large robot ensembles presents a number of challenges. As systems scale to larger ensembles of smaller, finer grain modules, one can expect only limited capabilities at individual modules. In particular, modules only make noisy observations of their immediate neighbors, and do not have access to long distance measurements, such as global time-of-flight measurements, or external beacons. A lack of strong mechanical latches in small modules precludes mechanical constraints for accurate alignment and orientation.

This work is supported in part by the NSF under grants CNS 0428738 (ITR: Synthetic Reality) and NeTS-NOSS CNS-0625518, by the ONR under grant MURI N000140710747, by Intel Corporation, and by Carnegie Mellon University.

Although localization algorithms have been well studied in robotics, many of the existing approaches do not directly apply to large-scale internal localization. Constraint-based approaches [4, 3, 5] rely on strong prior assumptions about ensemble structure (e.g., lattices) or require exact observations to scale up to large ensembles. They are neither robust to noise nor well suited to irregular, non-lattice structures, common in some SRMRs. Local probabilistic approaches have been shown to be effective in localization of relatively small modular robots, such as PolyBot[6], but require assumptions of strong sensing, or robust mechanical latching to reduce errors in larger systems. Sparse approximation techniques [7, 8] used in simultaneous localization and mapping (SLAM), are effective in dealing with large amounts of noisy observations, but are difficult to apply to SRMRs, where modules are densely packed, forming grids and loops.

A more closely related problem is localization of wireless sensor networks. Here the nodes need to combine distance information about other nodes in order to accurately triangulate their positions. A standard formulation is to treat the distance information as weights of edges in a graph and obtain a Euclidean embedding, using methods such as regularized semidefinite programming relaxations [9, 10]. Internal localization can also be viewed as Euclidean embedding; however, only distances to immediate neighbors are known. As indicated by our experiments in Section VI, this restriction appears to impair the performance Euclidean embedding methods. Therefore, it is necessary to develop new techniques that are effective in this domain.

A key problem with applying incremental approaches like the ones seen in SLAM is that they can accumulate error, and this error takes a long time to resolve. In the case of a modular ensemble, the greatest error will tend to accumulate in a region with only a few inter-module observations, which we call a *weak region*. A substantial rotational uncertainty will be introduced in the partial solution, and will be magnified by subsequent additions. If we selectively incorporate the observations in the densely connected regions first, the partial solution will be constrained and the error will be substantially reduced. We use this intuition to formulate a hierarchical algorithm,

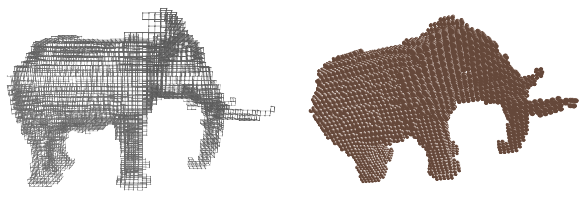
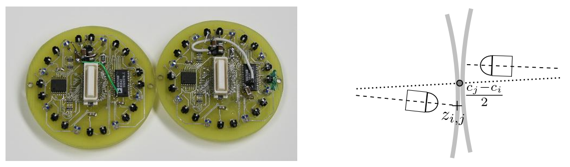


Fig. 1. Connectivity graph of ensemble with 8008 nodes, and resulting estimate of module positions; the results are accurate, subject to a rotation and translation of the coordinate space.



(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 $z_{j,i}$.

where we recursively split the ensemble connectivity graph into well-connected components using normalized cut [11]. In order to keep the normalized cut computations tractable, we perform graph abstraction, analogous to over-segmentation in image segmentation [11].

A key challenge in internal localization is that observations are not stored centrally, and it is not feasible to collect the observations to a single node. This calls for a distributed approach, but the recursive nature of our algorithm and top-down partitioning make a distributed implementation difficult to achieve. We used a declarative, logical programming language called Meld to help address these issues and create an efficient distributed implementation of our algorithm.

We evaluate our algorithm on realistic 2D and 3D problems with up to 10,000 modules that accurately model unreliable observations and physical interactions among the modules. We demonstrate that the computational complexity of the approach is nearly linear in the size of the ensemble for a fixed ensemble structure, and outperforms methods from wireless sensor network localization based on classical multidimensional scaling [12] and semidefinite programming (SDP) relaxations [9, 10], as well as simpler incremental heuristics.

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).

Figure 2(a) shows a current working prototype of a sensing subsystem fitting the properties described above. Each module shown has 8 IR transmitters and 16 IR receivers, oriented radially and spaced evenly around the circular perimeter. Note that for these modules, multipath interference, scattering, shadowing, and small dimensions effectively preclude techniques such as acoustic or radio time-of-flight-based localization.

III. Localization as Probabilistic Inference

In this section we define the probabilistic model that underlies our algorithm. We then discuss a simple incremental optimization method that motivates our approach.

A. Probabilistic Model With Attractive Potentials

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 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 in 2D as an angle, and in 3D as a quaternion).

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. 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 model penalizes an observation $z_{i,j}$, based on how well it predicts the displacement between the two module centers.

$$\phi(x_i, x_j; z_{i,j}) \propto \exp\left\{ -\frac{1}{2} \left\| r_i(z_{i,j}) - \frac{c_j - c_i}{2} \right\|_2^2 \right\}. \tag{1}$$

Note that this model does not explicitly represent the constraint that the modules must not overlap; instead, we

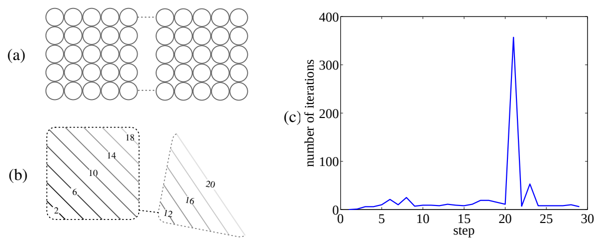


Fig. 3. (a) Ensemble, consisting of two tightly connected clusters. The clusters are connected by two pairs of observations. Within each component, modules make observations with all of their immediate neighbors, whereas the two components share only two observations, one at each side. (b) Intermediate 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 is not detected until loop closure, at step 21, and takes many iterations to resolve with conjugate gradient descent. (c) The number of iterations at each step to reach convergence.

have chosen to rely on the observations to obtain a non-overlapping solution. Alternatively, we could use a more accurate mode that captures properties of IR transmitters and receivers, such as quadratic decay and multi-modal response, but such a refinement is not key to the methods presented in this paper.

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 $\mathbf{x}$:

$$p(\mathbf{z}|\mathbf{x}) \propto \prod_{i,j} \phi(x_i, x_j; z_{i,j}). \tag{2}$$

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

B. Computing the MLE Solution Incrementally

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

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

for progressively larger connected 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, the set $\mathbf{z}_A$ is expanded, incorporating modules connected to its perimeter, and then iteratively refining the position estimates with the correspondingly expanded set of observations.

Figure 3 illustrates the behavior such an incremental approach on a small ensemble with 200 modules that consists of two dense components. The observations are 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. The computed solution accumulates error that takes a long time to resolve once the algorithm closes the loop.

IV. Guiding Localization With Normalized Cut

The experiment in Figure 3 points to an important drawback of an incremental maximum likelihood estimate (MLE) solution. Highly uncertain observations may be incorporated early, and errors magnified by subsequent module additions. With a simple MLE representation, this will remain undetected until a single observation closes the loop, at which point significant iterative computations are needed to shift the estimates back into low error bounds. However, if we were to first incorporate the observations within each dense cluster in Figure 3 and defer the observations that join the two clusters until later, the intermediate results would be more accurate, and would serve as a better starting solution for adding new observations. This suggests a hierarchical solution (Algorithm 1) that partitions the ensemble using clustering, recursively computes the estimates for each cluster, and uses the partial solutions to compute the globally optimal solution.

A. Determining an Effective Partition

Sparsely connected regions where only a few observations are made (*weak regions*) are one of the main sources of error and uncertainty as localization progresses. As illustrated in Figure 3, certain occurrences of weak regions can introduce a substantial rotational error – those that do not occur in pairs in 2D or triplets in 3D. An effective heuristic for identifying these weak regions is a cut on the connectivity graph of the ensemble: starting from a graph G, 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 criterion is effectively the one optimized in normalized cut [11]:

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

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 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.[1]

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 strongly discriminates these two orderings and yields the correct ordering.

B. Summary of the Algorithm

Our proposed approach 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 a 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 $\mathbf{p} = \{p_i\}$ and $\mathbf{q} = \{q_i\}$, 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 [13]. 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.

[1]We selected binary partition, since as discussed below, the solutions between two clusters can be merged very efficiently.

Algorithm 1 $NormCutLocalize(G, V)$

```
1: if $V$ is sufficiently small then
2:     compute $\arg\max p(\mathbf{x}_V|\mathbf{z}_V)$ using local heuristics
3: else
4:     Compute the normalized cut $(A, B) = NormCut(G)$
5:     $\mathbf{x}_A^* \Leftarrow NormCutLocalize(G_A, A)$
6:     $\mathbf{x}_B^* \Leftarrow NormCutLocalize(G_B, B)$
7:     $\mathbf{p} \Leftarrow$ transform the observations $\mathbf{z}_{A,B}$ into the coordinate frame, given by $\mathbf{x}_A^*$.
8:     $\mathbf{q} \Leftarrow$ transform the observations $\mathbf{z}_{B,A}$ into the coordinate frame, given by $\mathbf{x}_B^*$.
9:     Compute the optimal rigid alignment $R, t$:
           $\arg \min_{R \in SO(d), t \in \mathbb{R}^d} \sum_k \|p_k - (Rq_k + t)\|_2^2,$
10:    Let $\mathbf{x}_V^0 = (\mathbf{x}_A^*, R\mathbf{x}_B^* + t)$.
11:    $\mathbf{x}_V^* \Leftarrow \arg\max p(\mathbf{x}_V|\mathbf{z}_V)$, starting from $\mathbf{x}_V^0$
```

C. Scaling Up the Solution

While the normalized cut formulation yields an effective sequence in which observations should be incorporated, computing the exact normalized cut is costly and dominates other operations. Specifically, the cost of the rigid alignment is linear in the total number of observations, whereas the complexity of computing a *single* normalized cut is $O(|V|^{3/2})$, where $|V|$ is the number of nodes [11]. A standard method to decrease the computational complexity is to compute an abstraction of the graph, using a simpler clustering algorithm, such as k-means [11]. In particular, in image segmentation, this procedure amounts to computing an over-segmentation of the image. The normalized cut is then computed on a smaller graph G', where each node of G' corresponds to a cluster of nodes in the original graph G.

Compared to other clustering tasks, the clustering task in localization is simpler in two ways. First, unlike in applications, such as image segmentation, where shifting the cut can adversely affect the visual quality of the segmentation, the clustering here is only used as a heuristic, and offsetting the cut does not substantially decrease the quality of location estimates. Furthermore, since the connectivity graph G has unit edge weights, the cut value itself increases at most linearly (in 2D) or quadratically (in 3D) in the number of hops away from the optimal cut. Therefore, we have found that it is often sufficient to partition the graph greedily into a fixed number of components. As discussed in Section VI, using as few as twenty components yields accurate solutions (the actual number of needed components will depend on the amount of uncertainty in the ensemble).

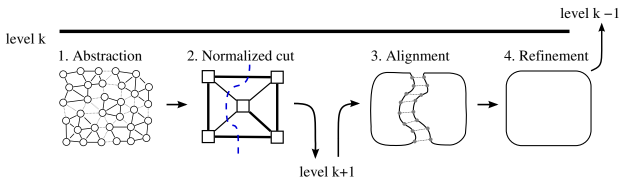


Fig. 4. Control flow for level k of the distributed implementation

V. Distributed Localization

While centralized localization in a self-reconfigurable modular robot is useful, a distributed localization is much more appealing, since it can significantly reduce the communication cost, enables online control, and avoids a centralized point of failure. In this section, we propose a distributed version of Algorithm 1 that uses a combination of data aggregation techniques and local refinement steps to compute each module's own location. In combination with a declarative programming language [14], we obtain a fully executable distributed solution.

A. Localization through Aggregation and Dissemination

Our distributed solution mirrors the operations of the centralized algorithm. Figure 4 summarizes the control flow for one level of the algorithm execution. The first three steps – graph abstraction, normalized cut, and rigid body alignment – use data aggregation techniques and perform key steps of the computation at the group leader. Note that the gradient of the log-likelihood (2) decomposes linearly over the nodes of the cluster and their neighbors. Therefore, the iterative refinement step can be performed locally, without global coordination.

In order to compute the graph abstraction, we use a random sampling strategy that partitions the graph into a set of connected subgraphs, centered around randomly chosen leaders. Each node elects itself as a leader with a small probability, and the nodes greedily join the nearest leader (as measured by the hop-count). The description of the abstracted graph is then aggregated to a single node that computes the normalized cut using a standard centralized implementation. Since we need to perform normalized cut only on small graph abstractions, a centralized implementation is sufficient. Alternatively, one could use a distributed algorithm based on decentralized power iteration. [15]

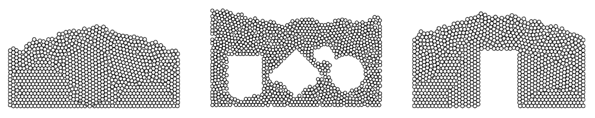
A key step in Algorithm 1 is computing the optimal rigid body alignment between two sides of the partition. While, at first glance, it is not clear how to distribute this step, a closer look at the method in [13] reveals that the method only depends on the first- and second-order statistics of the points $\{(p_i, q_i)\}$ in Equation 5. These statistics can be aggregated from the boundary towards the group leader. The leader then computes the optimal transform and disseminates the result. Since the aggregated information depends only on the dimensionality of the aligned points (2 or 3), rather than their count, the communication cost of aggregating and disseminating the optimal transform is small.

B. Declarative Implementation using Meld

The distributed algorithm described in the previous section presents a number of implementation challenges. Unlike simple message-passing style inference algorithms found commonly in literature [16], our localization approach uses multiple aggregation and dissemination steps that require both local and non-local communication across the ensemble and operate in asynchrony. These steps rely on a number of data structures, used to represent the graph, the location, and the rigid body transform statistics. Due to the recursive nature of the algorithm, the implementation may need to maintain parallel data structures for all of the concurrently active levels. These challenges make it tedious to implement the algorithm in a standard message-passing framework. In this section, we briefly outline our implementation that uses Meld [14], a logical, declarative, high-level programming language for modular robots.

Meld is a declarative language with syntax similar to Prolog. A Meld program consists of rules that specify sufficient preconditions to derive new facts from existing ones. A key benefit of Meld is that it lets the programmer focus on the logical, information processing aspects of an algorithm, while automatically taking care of the mechanics of distributed programming, such as communication. For example, a simple distributed spanning tree algorithm can be specified in two rules: a rule that determines the root of the tree, and a rule that lets a node join a tree that extends to one of its neighbors. In a similar manner, Meld simplifies implementation of other distributed data structures.

We found that many features of Meld fit well with the needs of our algorithm, but also exposed some drawbacks of our approach. The language let us naturally represent the graph abstraction process and aggregate and disseminate sufficient statistics for the rigid alignment. The results of different phases were easily chained together. Furthermore, when intermodule connections were lost or network layout changed, Meld was able to automatically recover, recomputing the relevant portions of these distributed data structures and rerunning parts of the localization algorithm. On the downside, Meld's declarative programming model made it more difficult to express certain imperative sequences and loops. More fundamentally, a change in or removal of one fact (for example, the origin of the coordinate system) may trigger the removal and subsequent rederivation of a large number of other facts. This drawback is inherent to our localization algorithm and is subject to ongoing research.



(a) solid (b) sparse (c) open

Fig. 5. Scenarios used in our experiments. The scenarios were generated by settling randomly inserted modules in a gravitational field.

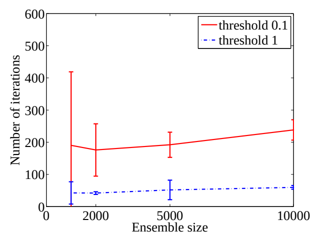


Fig. 6. The average number of iterations per module for the triple scenario as the number of modules grows. At different scales, the ensemble retains its shape and the proportions. With a threshold of 1.0, the average location RMS error was 1.26; with a threshold of 0.1, the average location RMS error was 0.80.

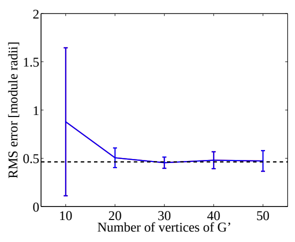


Fig. 7. The location RMS error for the triple scenario with 2000 modules, when using the normalized cut approximation in Section IV-C. The horizontal dashed line indicates the fidelity of the solution, obtained with exact normalized cut.

VI. Experimental Results

In this section, we present experimental results that illustrate both the centralized and the distributed aspects of our solution. We generated input scenarios with a C++ simulator [17] that models IR sensing and physical interactions between the modules. Each module in the simulation had 12 IR transceivers (colocated emitter/detector pairs), whose IR response was modeled according to an inverse-square law, similar to the model in [18]. The threshold for detecting observations between a pair of neighboring nodes was set to 20 per cent of the peak intensity. At this setting, a sensor can report a connection even if the modules are not in a physical contact and if the transmitter and the receiver are not perfectly aligned.

A. Scenarios

We constructed both 2D and 3D test ensembles. The 2D ensembles were generated by randomly settling simulated spheres under a simulated gravity field into a fixed container of the desired overall shape. The results were configurations with realistic, irregular, largely amorphous structures. Several of these configurations, illustrated in Figure 5 mimic planar slices of a 3D shape capture scenario [4]. Each shape in Figure 5 was instantiated ten times, with different initial velocities and locations of the modules, allowing us to average results across repeated runs using configurations very similar in overall shape but where module connectivity and spacing varies. The 3D test ensembles were generated by rasterizing 3D outlines (defined by OBJ files from various shape libraries) into a designated target lattice, either hexagonal-close-packed or cubic.

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 5(b) and formed a set of progressively larger ensembles. At each scale, the ensemble retains the same overall 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 6 shows the average number of iterations in preconditioned conjugate gradient descent as a function of the number of modules. We see that the number of iterations, needed to attain the same accuracy (as measured by the gradient norm), increases very slowly.

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 5(b) with 2000 modules. Figure 7 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 maximum diameter of clusters, rather than their count, the displayed node count 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. 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. These results suggest that small graph abstractions provide meaningful results and can be analyzed at each leader node centrally.

D. Performance in 3D

In the third set of experiments we extended the algorithm to three dimensions, using a quaternion representation for each catom's 3D orientation rather than the scalar orientation parameter used in the 2D case. Since the quaternion may become unnormalized in the process of gradient descent computations, we implicitly

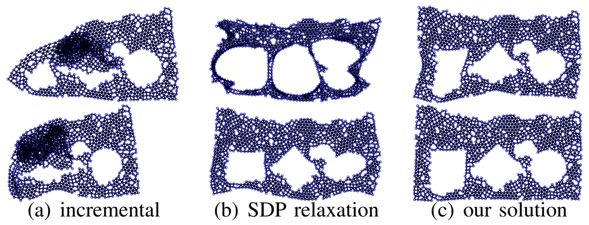

(a) incremental (b) SDP relaxation (c) our solution

Fig. 8. Example results using three algorithms on the sparse scenario. Lower images reflect results after additional iterative refinement steps.

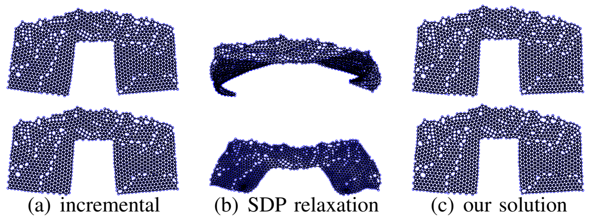

(a) incremental (b) SDP relaxation (c) our solution

Fig. 9. Example results using three algorithms on the open scenario. Lower images reflect results after additional iterative refinement steps.

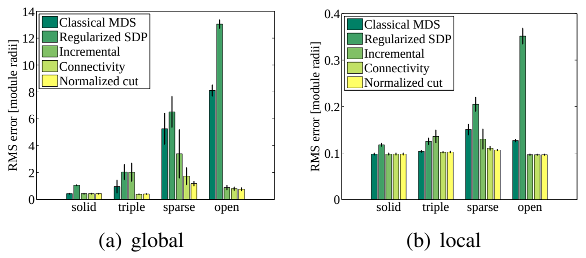


Fig. 10. RMS error of the location estimates. (a) Global RMS error, averaged over all modules. (b) RMS error of modules relative to their neighbors. Here, we greedily partition the ensemble into connected regions with diameter of 6 modules or less and compute the RMS error using the optimal rigid alignment for each region.

normalize the quaternion in the observation model (1) and in the corresponding gradient.

We generated lattice-bound test ensembles from various 3D outlines. (The lattice-bound property of these ensembles was due to limitations of our ensemble-generation code.) We simulated spherical modules each with 50 sensors scattered across their surfaces. Despite the larger number of sensors when compared to the 2D case, the available angle constraints in the 3D test cases were generally much weaker. We found that our algorithm was capable of determining positions in these 3D tests very accurately, within 1 module radius. Figure 1 shows an example of the results obtained on a large 3D scenario with 8008 modules.

E. Comparison with Prior Work

In the fourth 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 heuristics. Figures 8 and 9 show examples of how incremental and semidefinite programming approaches qualitatively perform poorly compared to our algorithm, even after applying significant number of iterative refinement steps. SDP in particular suffers from overestimation of distances, and artifacts due to projection to a 2D space from a manifold in a higher dimensional space.

Quantitatively, we evaluated the following methods: (i) classical multi-dimensional scaling [12], (ii) the inequality formulation of regularized semidefinite programming [9], (iii) the simpler incremental approach, discussed in Section III-B, (iv) a simple hierarchical approach that merges pairs of clusters bottom-up, in the order given by their algebraic connectivity, and (v) the proposed method, using exact normalized cut. We perform repeated experiments on the scenarios in Figure 5 with 1000 modules. The initial solution, obtained by each method is refined with 300 iterations of preconditioned conjugate gradient descent.

Figure 10 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 [10] 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.

F. Distributed Results

Finally, we evaluate the message complexity of the distributed implementation. Figure 11 shows the average number of messages sent by each module as a function of the ensemble size. The figure confirms that the number of messages per module required by the algorithm increases only logarithmically in the total number of modules in the ensemble. Thus, the implementation scales to large ensembles. Figure 12 shows the split of the messages among different components of the algorithm for two ensemble sizes. Interestingly, the messages sent are dominated by the gradient descent. In particular, the extra work done to determine the normalized cut is small compared to the cost of iterative refinement.

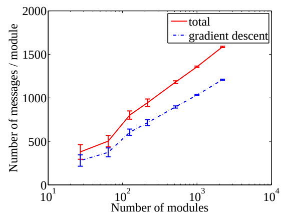


Fig. 11. The number of messages per module as a function of the ensemble size.

Procedure / Test case	$5 \times 5 \times 5$	$10 \times 10 \times 10$
Neighbor detection	0.5%	0.3%
Graph abstraction	7.7%	7.3%
Normalized cut	6.4%	6.5%
Rigid alignment	9.7%	9.5 %
Gradient descent	75.8%	76.3 %

Fig. 12. The relative number of messages sent by each component.

VII. Discussion and Future Work

In this paper, we examine large-scale localization in modular robot ensembles using uncertain, local observations. We formulate internal localization as a probabilistic inference problem and introduce a novel approach which hinges on selection of an effective ordering of observations using a normalized cut criterion. In combination with closed-form solutions for rigid alignment and simple graph abstraction scheme, this approach leads to accurate, scalable solutions. We perform an extensive evaluation of our proposed approach on a test suite of realistic 2D and 3D configurations with up to 10,000 nodes and demonstrate that our approach outperforms both recent methods using Euclidean embedding and simpler heuristics. Finally, we describe a fully distributed implementation of our algorithm that computes the results by sending only a few messages between the nodes.

While this paper goes a long way towards robust internal localization, there are some questions left to be answered. For example, it would be interesting to examine the merits of fast iterative methods developed for SLAM, such as [19]. These methods may make it possible to quickly recover from small changes in the ensemble and cope with dynamic settings. More broadly, one may hope to combine the normalized cut heuristic with additional structure in the problem to obtain a fully probabilistic solution. Such an approach would recover not only a point estimate, but also its uncertainty. Addressing both these questions would lead to truly robust solution to internal localization in modular robotics and drive research in other fields.

Acknowledgment

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 an implementation of the regularized SDP localization algorithm.

References

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

[2] K. Gilpin, K. Kotay, and D. Rus, "Miche: Modular shape formation by self-dissasembly." in *ICRA*, 2007, pp. 2241–2247.

[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] 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.

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

[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] U. Frese and L. Schroder, "Closing a million-landmarks loop," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, Oct. 2006.

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

[14] M. Ashley-Rollman, S. C. Goldstein, P. Lee, T. Mowry, and P. Pillai, "Meld: A declarative approach to programming ensembles," in *Proceedings of IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2007.

[15] P. Yang, R. A. Freeman, G. Gordon, K. Lynch, S. Srinivasa, and R. Sukthankar, "Decentralized estimation and control of graph connectivity in mobile sensor networks," in *American Control Conference*, June 2008.

[16] C. Crick and A. Pfeffer, "Loopy belief propagation as a basis for communication in sensor networks," in *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI-2003)*, C. Meek and U. Kjærulff, Eds. San Francisco: Morgan Kaufmann Publishers, Inc., 2003.

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

[18] 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.

[19] G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard, "Efficient estimation of accurate maximum likelihood maps in 3d," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, 29 2007-Nov. 2 2007.