

Multi-Robot Mapping using Manifold Representations

Andrew Howard

Gaurav S. Sukhatme

Maja J. Matarić

Robotics Research Laboratory, Computer Science Department,

University of Southern California, Los Angeles, California

Email: ahoward@usc.edu

Abstract— This paper introduces a new method for representing two-dimensional maps, and shows how this representation may be applied to concurrent localization and mapping problems involving multiple robots. We introduce the notion of a *manifold map*; this representation takes maps out of the plane and onto a two-dimensional surface embedded in a higher-dimensional space. Compared with standard planar maps, the key advantage of the manifold representation is self-consistency: manifold maps do not suffer from the ‘cross over’ problem that planar maps commonly exhibit in environments containing loops. This self-consistency facilitates a number of important autonomous capabilities, including robust retro-traverse, lazy loop closure, active loop closure using robot rendezvous, and, ultimately, autonomous exploration.

This paper introduces the basic concepts of the manifold representation and shows how it may be used to solve multi-robot mapping problems. By way of validation, we include experimental results obtained using teams of two to four robots in environments ranging in size from 400 m² to 900 m².

I. INTRODUCTION

This paper introduces a new method for representing two-dimensional maps, using *manifolds* in the place of two-dimensional *planes*. Our motivation for creating this representation flows from the desire to perform *autonomous* tasks, such as exploration and retro-traverse, in an environment that is only partially mapped; moreover, these tasks must be carried out concurrently with the simultaneous localization and mapping process. We therefore require a map representation that is at all times *self-consistent* (for autonomous behaviors, we are primarily concerned with self-consistency with respect to path-planning). Standard planar maps are ill-suited for this purpose, due their tendency to become confused in environments containing loops. Consider, for example, the situation shown in Figure 1: as the robot traverses a partial loop, the path of the robot crosses over itself. This inconsistency may be eventually be resolved when the robot closes the loop; in the interim, however, a planar map cannot be used for path-planning. In contrast, under the same conditions, the manifold representation remains entirely self-consistent: robots can always construct paths, so long as those paths are embedded in the manifold. Furthermore, when the robot finally closes the loop, it may be possible to collapse the manifold, and recover a self-consistent planar map.

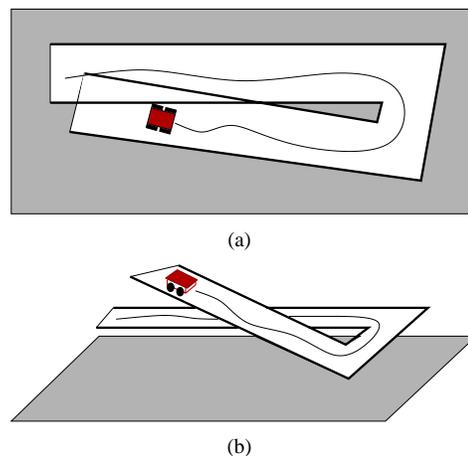


Fig. 1. Illustration of a partially closed loop. (a) Planar representation. (b) Manifold representation.

The manifold representation facilitates a number of interesting capabilities. For example, using incremental mapping alone (i.e., no loop closures), a robot can always retro-traverse to any previously visited location (or, more precisely, to any point on the manifold). In this case, the same location in the world may be represented more than once in the manifold: if the robot traverses a loop in one direction, for example, the manifold will develop a spiral structure, with the same locations being repeated over-and-over again. In spite of this ambiguity, however, the robot can always retro-traverse by traveling back along the spiral structure.

The many-to-one relationship between points on the manifold and points in the world gives rise to a second interesting capability: *lazy loop closure*. Loop closure is the most difficult part of the simultaneous localization and mapping process: in order to close a loop, one must decide that two points in the map correspond to the same point in the world (this is the data-association problem). In the manifold representation, such decisions can be indefinitely delayed, without risking map consistency; thus, one may wait until robots acquire more information to conclusively establish the correspondence (or lack thereof) between two points. In the multi-robot context, one may also take *active* steps to discover correspondence

points, using pairs of robots acting in concert. Thus, for example, a pair of robots can arrange a rendezvous at two points on the manifold that may or may not represent the same location in the world: if the robots meet, the points match and the loop is closed; if they fail to meet, the points are distinct and there is no loop.

This paper makes no attempt to cover all aspects of the manifold representation outlined above. Instead, we restrict ourselves to introducing the basic methodology and applying it to the specific problem of multi-robot mapping. We take maximum likelihood estimation techniques that have previously been applied to simultaneous localization and mapping [1], [2], and adapt those techniques for the manifold representation. For validation, we present experimental mapping results from two different (large) environments, using teams of up to four robots, under both manual and autonomous control.

II. RELATED WORK

TODO

III. MAPPING ON A MANIFOLD: CORE CONCEPTS

The key conceptual difficulty with manifold mapping is the representation of the manifold itself. In principle, the manifold is an arbitrarily complex structure with varying local curvature; in practice, the representation must be discrete, and hence some degree of approximation and linearization is inevitable. In this section, we develop the basic concepts, definitions and notation used in our approximated representation.

A. Patches

The manifold is discretized by dividing it into a set of overlapping *patches*, each of which has finite extent and defines a local (planar) coordinate system. Let Π denote the set of such patches; we make the following definition:

$$\Pi = \{\pi\} : \pi = (\theta, s) \quad (1)$$

where an individual patch π consists of a *free-space polygon* s describing the extent of the patch¹, and a *projected planar pose* θ that defines the patch-local coordinate system (θ is obtained by projecting the origin of the patch onto a canonical plane).

Given these definitions, the pose of any object on the manifold can subsequently be described by a tuple $\rho = (\pi, r)$ specifying a particular patch π and the pose r of the object with respect to that patch (r must lie inside the patch polygon s). Importantly, since patches may overlap, the tuple ρ need not be unique: one can also write down the same object's pose as $\bar{\rho} = (\bar{\pi}, \bar{r})$ where \bar{r} is the pose relative to some overlapping patch $\bar{\pi}$. Two questions naturally arise from this apparent ambiguity: how does one establish that the tuples ρ and $\bar{\rho}$ represent the same pose on the manifold, and how does

¹Strictly speaking, we use *polysolids* rather than polygons for representing free space, since polysolids form a group under the operations of union and intersection (polysolids can have holes). The term 'polysolid' appears to have been coined by Hugh Maynard and Lucio Tavernini at the University of Texas at San Antonio; their work was never published, but is similar in concept, if not detail, the *polygons sets* described in [3].

one transform a pose specified with respect to patch π into a pose specified with respect to an overlapping patch $\bar{\pi}$? Consider the manifold illustrated in Figure 2: each point on the manifold will project onto exactly one point on an imaginary horizontal plane, and, conversely, some points on the plane will project onto multiple points on the manifold. This observation leads to the following condition: the tuples ρ and $\bar{\rho}$ represent the same point on the manifold if and only if the projections of the polygons s and \bar{s} overlap, and the projections of r and \bar{r} are identical. Mathematically, this condition is stated as follows:

$$(s \oplus \theta) \cap (\bar{s} \oplus \bar{\theta}) \neq \emptyset \text{ and } r \oplus \theta - \bar{r} \oplus \bar{\theta} = 0 \quad (2)$$

where \oplus is a *coordinate transform operator*. Given a projected pose θ and patch-relative pose r , the expression $r \oplus \theta$ yields the corresponding projected pose q . One can also define the inverse operator \ominus : given two projected poses θ and q , the inverse expression $q \ominus \theta$ recovers the patch-relative pose r . These operators obey the normal rules for algebraic associativity, but do not commute.

From the identity expressed in Equation 2, one can trivially derive the coordinate transform equations for overlapping patches:

$$\bar{r} = r \oplus \theta \ominus \bar{\theta} \text{ and } r = \bar{r} \oplus \bar{\theta} \ominus \theta \quad (3)$$

Collectively, Equations 2 and 3 provide the necessary tools for working with manifold poses and their planar projections. Importantly, one can use these equations to construct paths on the manifold.

B. Relations

For concurrent localization and mapping, the projected poses of the patches Π are not known a priori; instead we have a set of *relations* that constrain the patches' *relative* pose (a scan-matching algorithm, for example, may establish point-to-point correspondences). Let Φ denote the set of pairwise relations between patches; we write:

$$\Phi = \{\phi\} : \phi = (\pi, \bar{\pi}; x, \bar{x}, \sigma) \quad (4)$$

where the relation ϕ implies that point x on patch π corresponds to point \bar{x} on patch $\bar{\pi}$; σ is the uncertainty associated with the correspondence. One can write down similar definitions for point-to-line, line-to-line, relative range and relative bearing relations.

C. Fitting Patches

Given the above definitions, one can apply maximum likelihood estimation (MLE) techniques to find the set of projected poses $\Theta = \{\theta\}$ that is *most likely* to generate the observed set of relations $\Phi = \{\phi\}$. That is, MLE searches for the estimate $\hat{\Theta}$ that maximizes the conditional probability $P(\Phi | \Theta)$:

$$\begin{aligned} \hat{\Theta} &= \arg \max_{\Theta} P(\Phi | \Theta) \\ &= \arg \max_{\Theta} \prod_{\phi \in \Phi} P(\phi | \Theta) \end{aligned} \quad (5)$$

where we make the additional assumption that the relations in Φ represent statistically independent observations. Applying

the standard log-likelihood transformation to these equations, one can equivalently search for the Θ that *minimizes* the (negative) conditional log-likelihood $L(\Phi | \Theta)$:

$$\hat{\Theta} = \arg \min_{\Theta} \sum_{\phi \in \Phi} L(\phi | \Theta) \quad (6)$$

This latter form is more convenient for most practical purposes. For point-to-point relations with Gaussian uncertainty, the log-likelihood for a single relation ϕ is given by:

$$L(\phi | \Theta) = \frac{1}{2\sigma^2} (x \oplus \theta - \bar{x} \oplus \bar{\theta})^2 \quad (7)$$

where $x \oplus \theta$ denotes the projected pose of a point on patch π and $\bar{x} \oplus \bar{\theta}$ denotes the projected pose of the corresponding point on patch $\bar{\pi}$. Intuitively, one can visualize the two points as being pulled together by a simple spring.

In principle, the maximum likelihood estimate $\hat{\Theta}$ can be found by solving:

$$0 = \sum_{\phi \in \Phi} \nabla L(\phi | \Theta) \quad (8)$$

In the case of 2D-mapping, however, the gradient terms on the right-hand side of Equation 8 generally contain transcendental components, and hence there exists no closed-form solution. Fortunately, a range of numeric optimization techniques can be used to find the minimum, including simple gradient descent and its more refined brethren, such as the Levenburg-Marquardt and Fletcher-Reeves algorithms [4].

The *confidence* in the estimate $\hat{\Theta}$ can be determined by inspecting the local curvature of $\nabla L(\Phi | \Theta)$ around $\hat{\Theta}$. Following the standard practice in the MLE literature [5], we write down the *stochastic Fisher information matrix* as:

$$J(\Phi | \hat{\Theta}) = \sum_{\phi \in \Phi} \nabla^2 L(\phi | \hat{\Theta}) \quad (9)$$

The confidence interval $\hat{\sigma}_i$ on any component i of $\hat{\Theta}$ is then given by:

$$\hat{\sigma}_i = c \sqrt{\left(J^{-1}(\Phi | \hat{\Theta}) \right)_{ii}} \quad (10)$$

where c is an appropriate critical value (e.g., 1.96 for a 95% confidence interval), and the ii subscript denotes a particular component of the inverted Fisher information matrix. In practice, we are usually less interested in the absolute fit confidence than we are in the relative confidence between a given pair of patches; i.e., how well is the pose of patch π determined with respect to patch $\bar{\pi}$? This quantity can be determined using a slightly modified version of the above procedure: we treat the components of $\hat{\Theta}$ that correspond to patch π as constants, and eliminate the corresponding rows and columns from the Fisher information matrix. Inserting this reduced matrix into Equation 10 we obtain the component-wise confidence intervals for every patch, relative to the “fixed” patch π . For the remainder of this paper, we will use $J(\Phi | \hat{\Theta})$ to denote this reduced matrix.

IV. MULTI-ROBOT MAPPING

We turn now to the specific problem of multi-robot mapping, using the mathematical tools described in the previous section. This problem can be broken into three sub-problems: incremental localization and mapping, loop closure and island merging.

Incremental localization and mapping is the basic mode of operation for the mapping algorithm: as each robot moves through the environment, odometry and laser data are used to update the robot’s current pose estimate, and, under certain circumstances, to make incremental additions to the map. The basic process is illustrated in Figure 2, and described in detail in the next section. Note that robots extend the map at the *edges* of the manifold only; a robot that is retro-traversing to a previously visited location will not add to the map.

This process punctuated by two events that require *global* changes to the map: loop closure and island merging. Loop closure is the process whereby two widely separated regions of the map are brought together (see Figure 3). In Section III-A, we showed that multiple points on the manifold may project onto the same point on the plane; in the context of mapping, this implies that two widely separated points on the manifold may in fact represent the same location in the world. Indeed, if one uses incremental mapping alone, any loops in the environment will be “unrolled” to form spiral structures, with the same series of locations repeating over and over again in the manifold. Loop closure, then, is the process whereby such repeated locations are identified, and the topology of the manifold is modified accordingly.

In a similar vein, island merging is the process whereby two unconnected regions of the manifold are combined into a single representation (see Figure 4). In the context of multi-robot mapping, there are two basic scenarios that give rise to such islands: robots enter the environment from separate locations, or robots enter the environment from the same location, but at different times. In either case, we proceed by building a separate island for each robot, and merging those islands only when a suitable correspondence point has been established.

The loop closure and island merging processes depend on our ability to uniquely identify a particular location in the world (the traditional data association problem). In the case of single-robot mapping, there are two basic methods for making this identification: recognizing a unique feature associated with that location (including pre-placed fiducials) or making plausible guesses based on patterns of non-unique features. These two methods have been well treated in the single-robot mapping literature [2], [6]–[8], and will not be covered here. Instead, we focus on a third method that is unique to the multi-robot mapping domain: using the robots themselves as unambiguous landmarks. Whenever two robots sight one another – a process we refer to as *mutual observation* – we establish a correspondence between two points on the manifold; mutual observations can therefore be used to close loops and merge islands.

In the following sections, we describe the incremental mapping, loop closure and island merging processes in detail. Note that, throughout this presentation, we assume that the mapping algorithm is *centralized*; i.e., data from all of the robots is communicated to a common location, where it is assembled to form a map.

A. Incremental Localization and Mapping

Incremental localization and mapping is performed independently and concurrently for each robot on the team. Two pieces of sensor data are used in this process: odometry data (which measures changes in the robot’s pose) and laser scan data (which measures the range and bearing of nearby features).

Let $o_{tt'}$ be the measured (odometric) change in pose between times t and t' , and let $s_{t'}$ be the laser scan that is subsequently recorded. If $\rho_t = (\pi_t, r_t)$ is the robot pose estimate at some time t , the updated robot pose estimate $\rho_{t'} = (\pi_{t'}, r_{t'})$ can be determined as follows.

- 1) Create a new patch $\pi^* = (\theta^*, s^*)$ such that:

$$\theta^* \leftarrow o_{tt'} \oplus r_t \oplus \theta_t \text{ and } s^* \leftarrow s_{t'} \quad (11)$$

i.e., the projected pose θ^* is computed by combining the measured (odometric) change in pose $o_{tt'}$ with the robot’s current pose estimate ρ_t .

- 2) Create a *local map* around the current patch π_t ; the local map is the set of patches Π^* that are both nearby (in the planar projection) and *well fitted* with respect to π_t ; that is, Π^* contains all patches π that satisfy the condition:

$$1.96 \sqrt{\left(J^{-1}(\Phi | \hat{\Theta}) \right)_{\theta\theta}} < \epsilon \quad (12)$$

The notation here requires a little explanation: $J(\Phi | \hat{\Theta})$ is the Fisher information matrix computed for patch π_t (see Section III-C); the patch π is included in the local map only if the confidence interval on every component of θ is less than some threshold ϵ .

- 3) Match features in the new patch π^* against features in the local map Π^* . We omit the details of the scan matching algorithm, and assume only that it produces some set of relations Φ^* .
- 4) Use MLE to fit the new patch against the local map; i.e., find the projected pose θ^* that is most likely to give rise to the observed relations Φ^* :

$$\theta^* \leftarrow \arg \min_{\theta} L(\Phi^* | \theta) \quad (13)$$

The minimum value is found using numeric optimization.

- 5) Compute the new robot pose estimate $\rho_{t'} = (\pi_{t'}, r_{t'})$ by projecting the new patch back into the manifold.

$$\begin{aligned} \pi_{t'} &\leftarrow \arg \min_{\pi \in \Pi^\dagger} \|\theta^* \ominus \theta\| \\ r_{t'} &\leftarrow \theta^* \ominus \theta_{t'} \end{aligned} \quad (14)$$

where Π^\dagger is a subset of Π^* containing only those patches whose scan polygons overlap with π_t ; the potential

ambiguity in the projection is resolved by selecting the nearest patch from this set.

Steps 3 and 4 of the algorithm may be applied iteratively (EM-style) to improve the quality of the fit.

The key step in this algorithm lies in the creation of the local map. In effect, that part of the manifold that is well localized with respect to the robot is projected onto a plane; the robot is then localized by fitting its laser scan against this planar projection. In this context, the choice of the threshold ϵ becomes crucial, since this parameter implicitly controls the number of patches included in the local map; if ϵ is too small, there may not be enough patches to adequately constrain the robot pose; if ϵ is too large, the local map may contain gross inconsistencies that lead to widely inaccurate pose estimates.

Having localized the robot, we may need to extend the map. There are a number of conditions that can trigger this process: e.g., the new patch is far from any of the existing patches in the local map, or the new patch ‘covers’ a significant area of the manifold that is not covered by the current local map Π^* . If none of these conditions are true, the patch π^* is discarded; otherwise, the patch and its relations are appended to the map. In this case, the robot pose estimate $\rho_{t'} = (\pi_{t'}, r_{t'})$ must also reset such that the robot lies at the origin of the new patch; i.e.:

$$\pi_{t'} \leftarrow \pi^* \text{ and } r_{t'} \leftarrow 0 \quad (15)$$

B. Loop Closure

The loop closure algorithm is triggered by events (such as mutual observation) that generate new relationships between previously unrelated patches. The key challenge for this algorithm lies not in the integration of this singular relation into the map; rather, it lies in the integration of any *additional* relations that may be induced as the change is propagated through the map. Consider, for example, a pair of robots traveling in opposite directions around a circular environment. If the robots should fail to observe each other on the first few passes (and thus fail to close the loop), the manifold will develop a double spiral structure (one spiral for each robot). The loop closure algorithm must be such that a single subsequent mutual observation will the collapse the *entirety* of both spirals into a single loop.

Our method for achieving this collapse is as follows: given a new relation, the algorithm propagates changes outwards from the closure point, alternating between inducing new relations and re-fitting the map. The process terminates only when no new relations can be induced. Let ϕ_{ab} denote a new relation between previously unrelated patches π_a and π_b ; the algorithm is as follows:

- 1) Add the new relation ϕ_{ab} to the map and re-fit:

$$\Theta \leftarrow \arg \min_{\Theta'} \sum_{\phi \in \Phi} L(\phi | \Theta') \quad (16)$$

- 2) Create a queue of patches; initialize the queue with the set of all patches that are related to π_a or π_b .

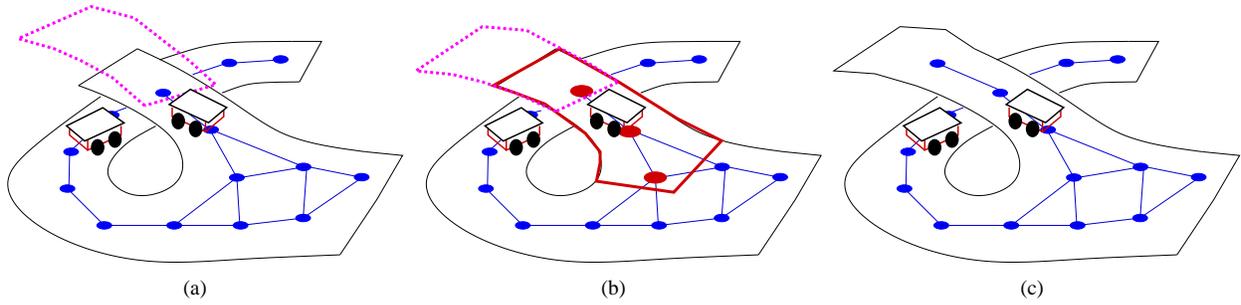


Fig. 2. Incremental localization and mapping. (a) & (b) A new laser scan (dotted polygon) is fitted against the robot's local map (solid polygon), to generate a corrected robot pose estimate. (c) If the laser scan covers unexplored regions of the manifold, a new patch is added to the map.

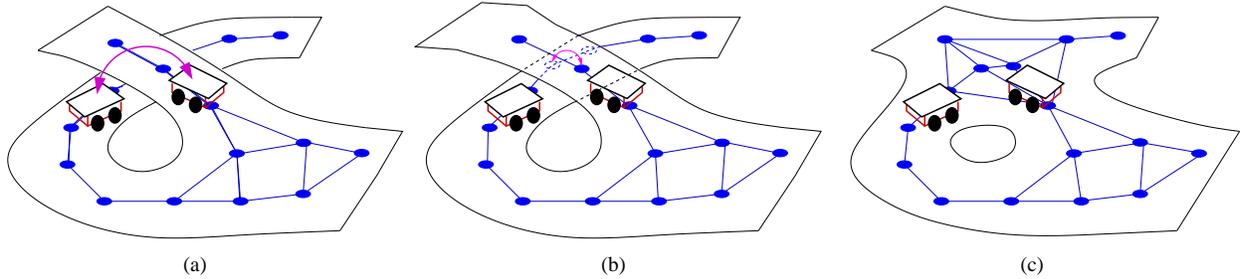


Fig. 3. A loop closure triggered by a mutual observation. (a) & (b) Two robots observe each other, generating a new relation. (c) The change in topology is propagated through the manifold, inducing new relations and forcing patches to be re-fitted.

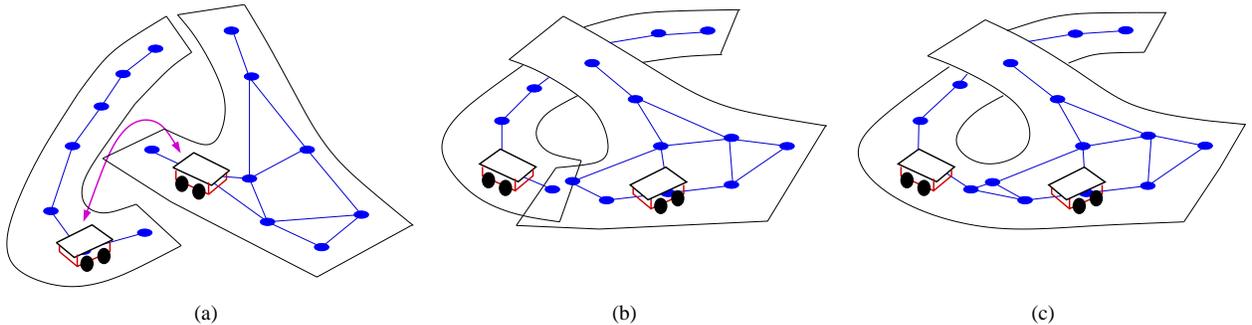


Fig. 4. Merging islands after a mutual observation. (a) Two robots observe each other, generating a new relation. (b) & (c) The two islands are roughly aligned, and the change in topology is propagated through the manifold.

- 3) Pop the first patch off the queue; call this patch π^* . Induce the local map Π^* for this patch using the procedure described in Section IV-A.
- 4) Use scan-matching to fit the patch π^* to the local map Π^* . From the set of relations Φ^* found by the scan-matching algorithm, eliminate those that are already in the map; i.e., Φ^* should represent the set of *new* relations for patch π .
- 5) If Φ^* is non-empty:
 - a) Add the new relations Φ^* to the map and re-fit as per step 1.
 - b) Add all patches that are related to π^* to the queue.

The process continues until the queue is empty.

Compared to incremental localization and mapping, the loop closure algorithm is relatively expensive: re-fitting the entire

map is non-trivial, and may be performed more than once for any given loop closure. Fortunately, closure events are relatively rare (their frequency depends on the number of loops in the environment). In addition, the loop closure algorithm is executed only *once* for each loop in the environment; subsequent traversals of a loop will incur no penalty.

C. Island Merging

The island merging algorithm is triggered by events that induce relations between patches belonging to separate islands (patches are said to belong to the same island only if and only if they are connected by some sequence of relations). In this case, as with loop closure, we must admit the possibility that there is substantial overlap between the two islands, and any changes made at the point where the islands are merged must

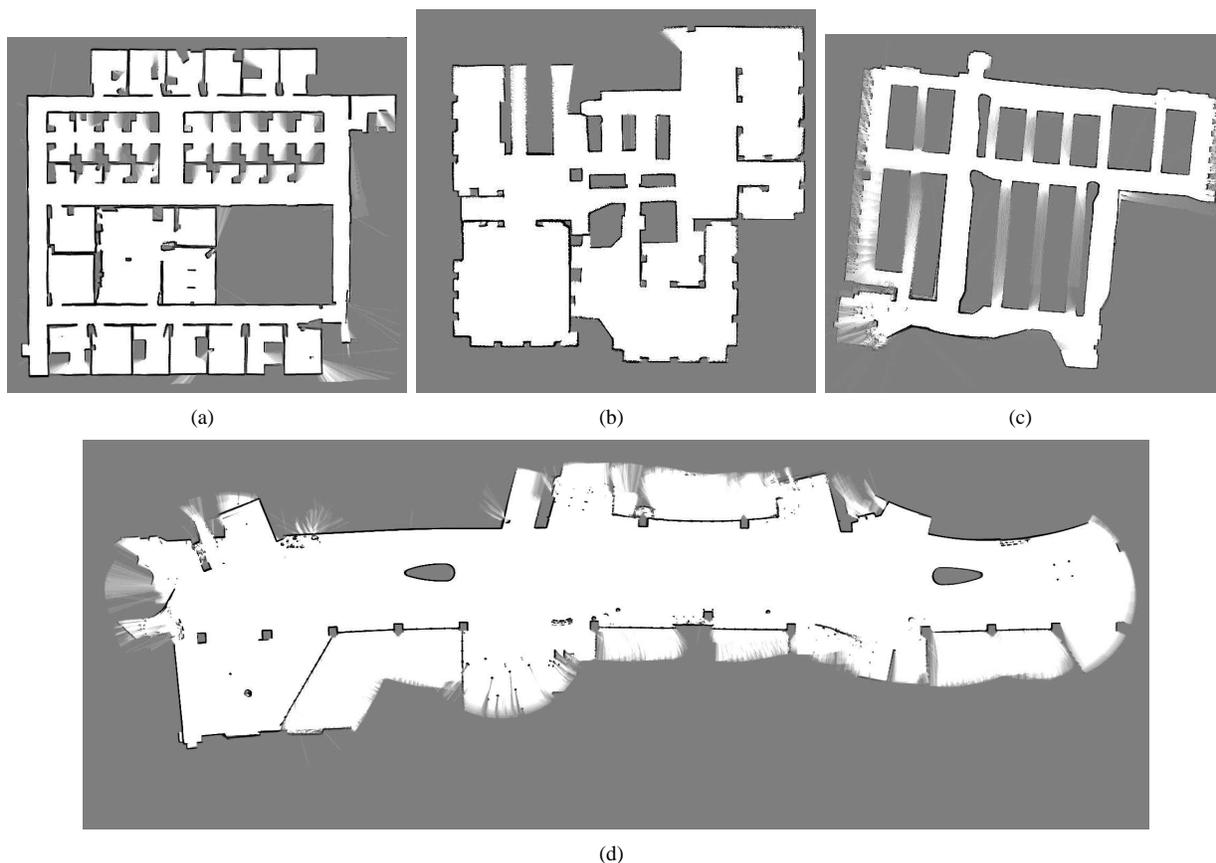


Fig. 5. Occupancy grids generated by the mapping package. (a) SAIC site A (two robots). (b) SAIC site B (four robots). (c) USC Science Library (four robots). (d) California Science Center (one robot).

be propagated throughout the map. The algorithm for island merging is therefore identical to that used for loop closure, with one exception: prior to merging, we treat the two islands as rigid bodies, and quickly bring them into rough alignment, thus saving a great deal of time in the re-fitting process.

Compared with incremental mapping, island merging is a relatively expensive process. For a team of N robots, however, the algorithm will be executed at most $N - 1$ times; moreover, since robots are likely to commence mapping from a relatively small number of initial locations, most of these mergers are trivial (i.e., involving islands with only a handful of patches).

V. EXPERIMENTS

The multi-robot mapping approach described in this paper has been applied to a wide range of environments of varying size and complexity: Figure 5 shows a selection of the final occupancy grid maps produced by the algorithm; note that all maps were generated autonomously, and in real time. Many of these maps, and the data-sets used to generate them, can be downloaded from the Radish [9] web-site.

Figure 6 shows the results for one particularly challenging experiment, conducted in a large test environment. Four robots were deployed into this environment from *two different locations* to execute an autonomous exploration algorithm. The robots were comprised of a Pioneer2 DX base, a

SICK LMS200 scanning laser range-finder, Sony PTZ camera, and a pair of fiducials (to facilitate mutual recognition); the Player robot device server [10] was used to control the robots. For this experiment, the relative pose of the two entry points was unknown; each pair of robots was therefore required to explore and map independently, giving rise to the two unconnected islands shown in Figure 6(c). After approximately 10 minutes, however, the two sets of robots encountered one another, and, using this mutual observation, the two maps were merged into one (Figure 6(d)). The combined robot team then proceeded to complete the exploration and mapping task, yielding the final occupancy grid map shown in Figure 6(b).

It should be emphasized that exploration and mapping was entirely autonomous, with the exception of a single user intervention to direct a robot into the otherwise unexplored room at the bottom left of the map. Maps were generated in real time, in an environment approximately 600 m^2 in area.

VI. CONCLUSION AND FURTHER WORK

While the manifold techniques described in this paper may be used to generate maps from robots under manual control, our key motivation in designing this representation is to support *autonomous* behaviors for incompletely mapped environments. As a result, our current research is heavily

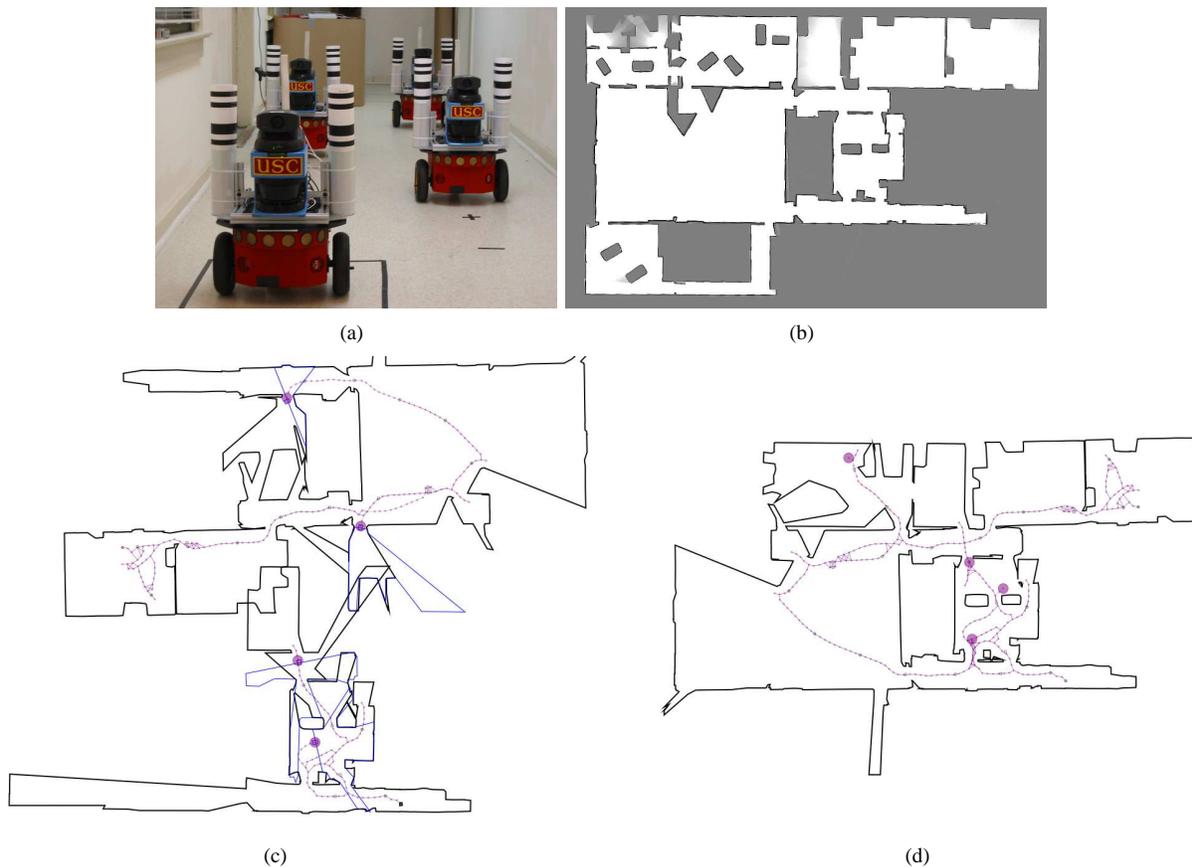


Fig. 6. (a) The mapping team: each of the four robots carries a unique laser-visual fiducial. (b) Occupancy grid map produced during a multiple-robot, multiple-entry trial: two robots entered from the door at the right top, another two robots entered from the door at the right bottom (the relative pose of the two doorways was unknown). The environment is approximately 600 m² in area. (c) A pair ‘sketch’ maps prior to merging (the sketch map is a vector representation showing the currently explored free space). Also show is the topological map used for navigation on the manifold. (d) Combined sketch map following merging; the indicated robots have made a mutual observation.

focused on this topic. To date, we have created an autonomous exploration algorithm that exploits the manifold map to direct and coordinate multiple robots; our near-term aim is to extend this algorithm to include *planned rendezvous*: i.e., the use of robots in pairs to explore regions of the manifold that appear similar.

ACKNOWLEDGMENTS

This work is sponsored in part by DARPA grants 4400057784 SDR (Software for Distributed Robotics; in conjunction with SAIC, Telcordia, and the University of Tennessee Knoxville) and 5-39509-A (Mobile Autonomous Robot Software; in conjunction with the University of Pennsylvania, Georgia Institute of Technology and BBN Technologies).

REFERENCES

- [1] F. Lu and E. Milius, “Globally consistent range scan alignment for environment mapping,” *Autonomous Robots*, vol. 4, pp. 333–349, 1997.
- [2] J. Gutmann and K. Konolige, “Incremental mapping of large cyclic environments,” in *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, 2000.
- [3] B. R. Vatti, “A generic solution to polygon clipping,” *Communications of the ACM*, vol. 35, no. 7, pp. 57–63, 1992.

- [4] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. Cambridge University Press, 1999.
- [5] L. L. Scharf, *Statistical Signal Processing: Detection, Estimation and Time Series Analysis*, ser. Electrical and Computer Engineering: Digital Signal Processing. Addison-Wesley, 1991.
- [6] S. Thrun, D. Fox, and W. Burgard, “A probabilistic approach to concurrent mapping and localisation for mobile robots,” *Machine Learning*, vol. 31, no. 5, pp. 29–55, 1998, joint issue with *Autonomous Robots*.
- [7] S. Thrun, “A probabilistic online mapping algorithm for teams of mobile robots,” *International Journal of Robotics Research*, vol. 20, no. 5, pp. 335–363, 2001.
- [8] D. Hähnel, W. Burgard, B. Wegbreit, and S. Thrun, “Towards lazy data association in slam,” in *Proceedings of the 10th International Symposium of Robotics Research (ISRR’03)*, Siena, Italy, October 2003.
- [9] A. Howard and N. Roy, “Radish: the robotics data set repository,” <http://radish.sourceforge.net>, May 2003.
- [10] B. P. Gerkey, R. T. Vaughan, and A. Howard, “The Player/Stage project: Tools for multi-robot and distributed sensor systems,” in *Proceedings of the International Conference on Advanced Robotics (ICAR 2003)*, Coimbra, Portugal, June–July 2003, pp. 317–323.