Accepted to the 7th Intl. Symp. on Distributed Autonomous Robotic Systems (DARS 2004)

Topological Map Merging

Wesley H. Huang and Kristopher R. Beevers

Rensselaer Polytechnic Institute, Department of Computer Science
110 8th Street, Troy, New York 12180, U.S.A.
{whuang,beevek}@cs.rpi.edu

Summary. A key capability for teams of mobile robots is to cooperatively explore and map an environment. Maps created by one robot must be merged with those from another robot — a difficult problem when the robots do not have a common reference frame. This problem is greatly simplified when topological maps are used because they provide a concise description of the navigability of a space. In this paper, we formulate an algorithm for merging two topological maps that uses aspects of maximal subgraph matching and image registration methods. Simulated and real-world experiments demonstrate the efficacy of our algorithm.

1 Introduction

Systems of multiple mobile robots must be able to cooperatively explore and map an environment for applications such as urban reconnaissance, search & rescue operations, security monitoring, and even house cleaning. In order to create a map quickly, each robot can only explore part of the environment, and the robots' individual maps must be merged to form a complete map.

In this paper, we address the problem of map merging for topological maps. Topological maps use a graph to represent possibilities for navigation through an environment; vertices represent certain "places" in the environment and edges represent paths (or classes of paths) between these places. Often vertices and edges are annotated with certain metric information, such as path length (for edges) or relative orientations of incident paths (for vertices). Typically, vertices are junctions of hallways, and edges represent a path down a hallway from one junction to another. Topological maps provide a concise description of the environment specifically geared towards navigation. Our focus is on indoor environments, so the use of topological maps is appropriate.

Two robots that have explored overlapping regions of an environment should have topological maps that have common subgraphs with identical structure. Solving the map merging problem is thus analogous to identifying a matching between the two graphs. In general, we would expect exactly

known attributes of matched vertices, such as the degree of the vertices in a static world, to match perfectly. However, attributes of vertices and edges that are subject to measurement error, such as the length of an edge or the angles between edges leaving a vertex, must only match closely.

If a robot's topological map contains geometric information about edges (e.g. path shape and the orientations of edges at vertices), there is enough information to estimate vertex locations with respect to the robot's world frame. This suggests that the map merging problem could also be solved using image registration techniques. The most widely used algorithms for image registration are iterative closest point (ICP) algorithms. An initial matching between feature points must be provided; the algorithm first estimates a transformation between the two feature sets by minimizing the (weighted) squared error between corresponding features. Next, the feature matching is expanded by finding features that are close together under this transformation, and the transformation is re-estimated. This process continues until the change in the transformation estimate between iterations is small.

In this paper, we describe an algorithm that uses the structural aspect of subgraph matching and the geometric aspect of image registration. We first create hypothesized matches by pairing compatible vertices in the two maps, and then locally grow each match using only the graph structure and vertex and edge annotations. Many hypotheses can be eliminated in this phase because of incompatibilities in the structure of the maps, or because of incompatible annotations (such as edge lengths that are too different). After growing the consistent matches, we estimate geometric transformations for each hypothesis and perform clustering in the transformation space. The best cluster (based on size and error) is returned as the algorithm's result, and the transformation from that cluster is used to merge the maps.

1.1 Related work

In recent years, there has been increased interest in map-making with multiple robots. Most multi-robot mapping work has focused on the creation of occupancy maps, e.g. [13, 10], though some work has been done on multi-robot feature-based or topological mapping [7, 6, 4]. In particular, Konolige *et al.* [11] have shown the benefits of feature-based approaches to map merging, as opposed to matching the raw sensor data of occupancy maps.

Much of the multi-robot map making and merging work assumes that all robots in the team begin the mapping process with a common reference frame — an assumption we do not make in this paper. One notable exception is the approach taken by Ko *et al.* [10] in which robots exchange occupancy maps and attempt to localize themselves in each others' maps.

The work most related to ours is that of Dedeoglu and Sukhatme [4], who presented a method for merging landmark-based maps without a common reference frame. They estimate a transformation between two maps using a single-vertex match found with simple heuristics, and match other vertices

using this transformation. In contrast, we estimate geometric transformations between the two maps, but rather than use these transformations to generate vertex correspondences, we compute the transformations *using* correspondences that we find from the structure of the maps. Our use of the maps' structure results in quicker and more effective discovery of potential matches. Additionally, our transformations are computed from multiple-vertex matches using well-known image registration techniques [2], rather than from single-vertex matches.

Our work draws on ideas from the graph matching literature. In particular, the topological matching problem can be viewed as an instance of the maximal common subgraph problem [3], and is closely related to the problems of structural matching and error-tolerant subgraph isomorphism [14].

1.2 Assumptions

The maps to be merged must be consistent — no two vertices may represent the same "place." This means that the robots can either recognize or infer when they have revisited a place and thus are able to "close the loop." However, we do not assume that the vertices have "distinguishing" attributes, which would make the map merging problem significantly simpler.

The robots must record enough information that map vertices can be embedded in a metric space; path shapes and the angles between edges leaving vertices are sufficient. We assume there is a known error model for measurements and that, errors notwithstanding, only translation and rotation (but no scaling) are needed to merge the maps. Our examples are from rectilinear worlds, but the algorithm we develop will work with any topological maps.

2 Hypothesis building

The first phase of our algorithm creates hypotheses by locally growing single-vertex matches. A hypothesis is a list of vertex and edge correspondences between two maps; finding them is, in essence, the problem of finding all maximal common connected subgraph matchings between the two maps.

2.1 Vertex matching

We start with a pair of compatible vertices, one from each map. Vertices are tested for compatibility by examining their attributes: exactly known attributes (e.g. vertex type) must match perfectly; inexactly known attributes (e.g. edge lengths or orientations) must be compared with a similarity test.

It often makes sense to assume that the robots will know the degree of vertices exactly; robots with sufficiently powerful sensing should easily be able to determine the number of paths leading from a vertex. In dynamic worlds,

where the degree of vertices may change — e.g., due to opened or closed doors — vertex degree cannot be treated as an exact attribute.

Relative edge orientations at a vertex are typically known with some uncertainty, so they must be compared using a similarity test which determines how similar two vertices must be in order to be paired.

2.2 Growing matches

We now grow the match by testing corresponding pairs of edges leaving the paired vertices. If the edges are compatible and the vertices at the ends are also compatible, then they are added to the match. If the edges or vertices are incompatible, the entire match is rejected.

The vertices are tested with the same criteria and similarity tests used to form the initial pair. Edges may also have both exactly and inexactly known attributes. Typically they have a path length, compared with a similarity test.

Our initial hypotheses are the unique matches that survive the growing process. We avoid generating duplicate hypotheses by keeping a table of vertex pairings. When vertices are paired during the growth phase, the corresponding entry is marked in the table. This entry is then ineligible as an initial pairing of vertices. A subgraph in one map can be matched to multiple subgraphs in the other (under separate hypotheses), but a pair of matched vertices (with a given edge correspondence) can appear in only one hypothesis. The matching/growing process is repeated until all valid vertex pairings are examined.

2.3 An example

For an example of hypothesis generation, we use two maps from a rectilinear world, shown in Figure 1. The rectilinear world assumption implies that we know exact orientations of paths leaving vertices (relative to the robot's coordinate frame). In the example, we also assume a static world, so vertex degrees must match exactly. Degree two and three vertices can match only for a single edge pairing; degree four vertices can match for four edge pairings.

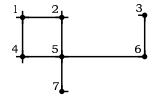
To construct all unique hypotheses, we create tables as described previously. The matches are grown by adding compatible incident edges and vertices. When a match is found to be inconsistent (usually due to vertex degree mismatch), we still grow it as much as possible so that those vertex pairs can all be marked as incompatible. This avoids regrowing the same match from a different initial vertex pair.

3 Transform estimation & clustering

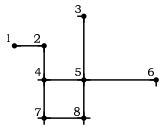
Our hypotheses now consist of maximal matched connected subgraphs. These matchings represent a single overlapping area, but the two maps may have several (separate) overlapping areas. In this phase of the algorithm, we consider the geometric relationships of hypotheses.

	b2	b3	b6
a3	×	×	H_1
a6	×	×	×
a7	H_2	×	H_3

Degree-2 Vertices



Map $\mathcal{A}$



Map $\mathcal{B}$

Map $\mathcal{A}$ rotation	b4	b5	b7	b8
90° / 180° + 0° / 270°				
a1	× / × + H_4 / H_{11}	× / × + × / H_{12}	× / H_9 + H_5 / H_{13}	× / H_2 + H_6 / H_{14}
a2	× / × + × / ×	× / × + H_4 / ×	× / H_2 + H_7 / H_{11}	H_8 / × + H_5 / H_{12}
a4	× / H_9 + × / H_{12}	× / H_2 + × / ×	× / H_{10} + H_4 / H_{14}	× / × + × / H_{15}
a5	× / H_2 + × / ×	× / × + × / ×	× / × + × / H_{12}	× / × + H_4 / ×

Degree-4 Vertices

Fig. 1. Hypothesis generation example for two maps from a rectilinear world. All possible single vertex matches are represented in the two tables; an "×" indicates that there is no valid hypothesis with that vertex match (and orientation). For example, hypothesis H_1 is generated from a matching with no rotation between vertex 3 in Map $\mathcal{A}$ and vertex 6 in Map $\mathcal{B}$ (a3–b6). There are no common edges from these vertices, so it cannot be grown further. For a 180° rotation of Map $\mathcal{A}$, hypothesis H_2 matches a1–b8, a2–b7, a4–b5, a5–b4, and a7–b2; this is the extent to which it can be grown. For a 90° rotation of Map $\mathcal{A}$, we can match a1–b7, a2–b4, a5–b5, a4–b8, and a6–b3. Though a7–b6 should also match, the edges a5–a7 and b5–b6 have significantly different lengths, so all these matches are marked invalid. There are 15 hypotheses that result from these two maps.

3.1 Transform estimation

In order to estimate a geometric transform, we must first embed the vertices of the maps in the plane. The problem of generating a consistent geometric map from the local distance and angular measurements added to a topological map has been addressed by several researchers, including Duckett *et al.* [5], Lu and Milios [12], and Golfarelli *et al.* [9]. Any of these methods would suffice; the reference frame for the vertices can be placed arbitrarily.

We can now estimate a transformation (translation and rotation) for each hypothesis using the vertex correspondences of the hypothesis. In image registration, this would typically be done using iteratively reweighted least-squares, where the weights help make the method robust to outliers. It is appropriate for us to use an unweighted least squares estimation because corresponding vertices should be close together; large error indicates that a hypothesis is geometrically bad (despite being structurally good).

3.2 Clustering

We group hypotheses into clusters to determine which hypotheses are consistent with each other. The clustering is done in the hypothesis transformation space, for which an appropriate distance function must be used.

Clustering requires some threshold distance to determine when two transforms are close enough to be compatible. This distance could be defined in terms of the map, e.g., a fraction of the minimum edge length in the hypothesis, or in terms of some aspect of the hypotheses themselves, such as metric error. Hypotheses within the threshold distance in transformation space are clustered together. There are a variety of techniques for clustering; we used a simple agglomerative clustering method in our implementation.

Once we have a set of hypothesis clusters, we order these clusters in terms of their "quality." The quality of a hypothesis is not straightforward to determine without information about the size, complexity, and self-similarity of the environment, which could be used in assessing the distinctiveness of the matched portions of the maps. Absent such information, we suggest the following heuristics and methods:

- Total number of vertices is a good primary indicator of cluster quality. Clusters containing only one or two pairs of vertices — particularly, clusters with only a small number of single-match hypotheses — are generally not significant unless the vertices are somehow unique.
- The amount of error (i.e., total squared error under the cluster transform) is a good secondary indicator of quality.
- The number or size of hypotheses in a cluster can be used as a secondary quality indicator. For example, a single large hypothesis is preferable to a cluster of small hypotheses.
- There is always some tradeoff between size and quality of a cluster: a single matched pair of vertices has no error, but generally does not constitute a significant match.

3.3 Example continued

For rectilinear worlds, the errors in orthogonal directions are decoupled. Embedding a map in the plane is thus reduced to two one-dimensional problems. We find the maximum likelihood embedding using a simple spring model.

We estimate a transform for each hypothesis using a two-dimensional version of the point-based rigid registration algorithm described by Fitzpatrick *et al.* [8]. This involves computing the singular value decomposition of a two-by-two covariance matrix.

Figure 2 shows the resulting clusters from our example. Because the example is in a rectilinear world, the transform space consists of a two-dimensional translation space for each of the four possible rotations. The quality of clusters was determined first by number of vertices and then by total error between vertex pairings under the cluster transform.

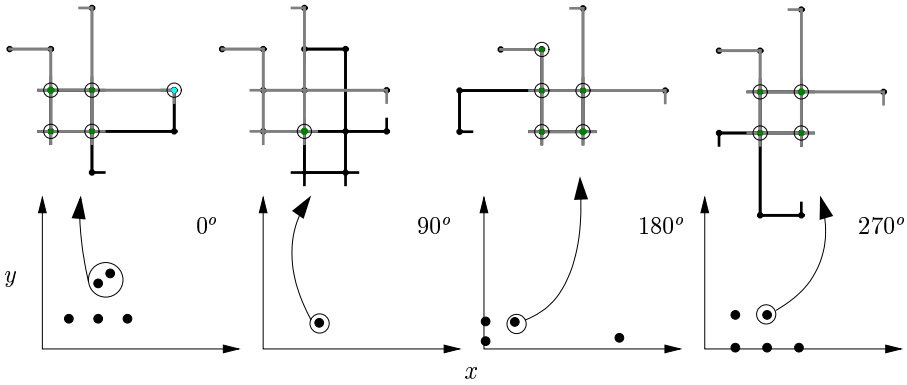


Fig. 2. Transformation spaces and example clusters for our example. The 180°, 5-vertex cluster (from a single hypothesis) is the best match; the 0°, 5-vertex cluster consisting of two hypotheses is also a good match. Map $\mathcal{A}$ is shown in black, map $\mathcal{B}$ is shown in gray, and matched vertices are circled.

4 Implementation issues

Once a hypothesis cluster has been deemed correct, it is fairly straightforward to merge the two maps. The estimates of path lengths can be updated by combining the measurements from the two maps for corresponding edges. The edge orientations at the corresponding vertices can be similarly merged. Portions of one map not present in the other should be added. Appropriate strategies for map storage and variations on the merging algorithm can simplify the implementation and improve its efficiency.

4.1 Map storage

Even the best cluster choice may later turn out to be incorrect. For example, early in the process of exploring a self-similar environment, the robots might seem to be exploring the same area when in fact they are exploring similar but distinct areas. We must consider how to merge and store maps so that incorrect hypotheses can be removed without discarding the whole map. Also, we would like to be able to merge the maps of several robots, not just two.

We can think of a robot's map as being represented in several layers. Layer 0 should be a robot's own map, recording only the measurements that robot has taken. Layer 1 is used to store other robots' Layer 0 maps that have been matched. Layer 2 contains maps that have been matched to Layer 1 maps (for which the matches have been computed either by another robot or locally using another robot's data), and so on.

This approach yields a "dependency" structure that makes it straightforward to discard hypotheses that are later determined to be incorrect, along with all other hypotheses that depend on them. Also, it does not require much extra storage; if necessary, upper layers can be compressed into a single layer.

4.2 Computational issues

When used online, robots may have additional information that can be used to merge maps more quickly. For example, if two robots exchange maps only when they are within communication range, we can start the hypothesis formation by pairing vertices near both robots.

After merging maps once, two robots may later merge their maps again. Here, an "incremental" map update can save a substantial amount of computation; this is possible with minimal bookkeeping effort. The other robot's map, stored separately, can be updated with new or changed vertices and edges. Updates can be used to verify and expand (or eliminate) hypotheses from previous mergings, and to form new hypotheses. All surviving hypotheses undergo the remainder of the merging process.

Although a map merging can be computed relatively quickly, it may be desirable to distribute the computation between robots. The hypothesis building and transform estimation steps can easily be split. The transformation clustering, however, is done most straightforwardly on a single robot.

5 Results

We have implemented the map merging algorithm for rectilinear worlds, and have tested this implementation in simulation with randomly-generated maps, and with maps generated from real-world data. Our results indicate that the algorithm is very effective, even for large maps with small overlap.

In our implementation, similarity of edge lengths was tested based on an odometry error model: when one edge was within a 95% confidence bound of the other, the edges were deemed acceptable matches. Thresholds on intra-cluster distance in transformation space were set to be equal to 3 times the largest mean squared (translational) error among the individual hypotheses; in rectilinear worlds, rotational error need not be considered.

For large maps (greater than 100 vertices), there were several thousand single-vertex correspondences; after the topological growth process, less than 100 hypotheses remained. Typically, there were only one or two large hypotheses (more than three vertices). Even for large maps, the matching process was quick, usually under one second on a 650 MHz Pentium 3 processor.

Clustering on the hypothesized matches worked well, but occasionally very small (correct) matches yielded transformations that were significantly different from the true transformation because of vertex positioning error; as such, the matches were not added to otherwise correct clusters of hypotheses. A potential solution is to take an "iterative" clustering approach, similar to the iterative closest point methods used in image registration, in which new hypotheses may be added to a cluster based on metric error under the transformation of the cluster, rather than on distance in transformation space.

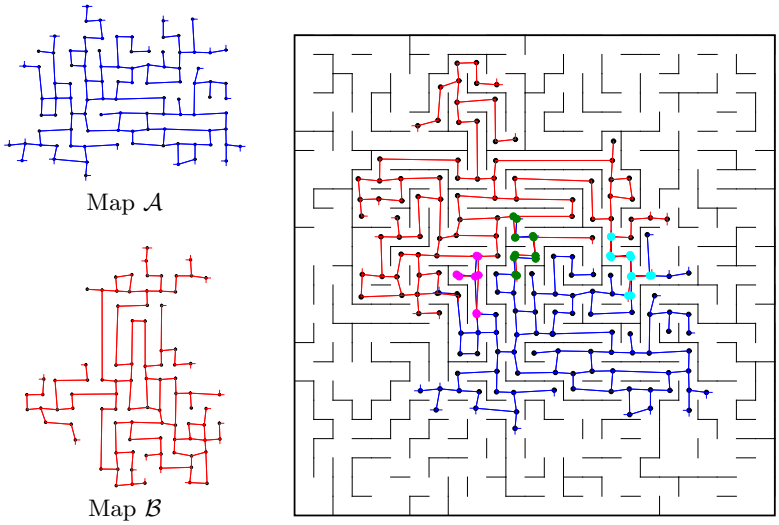


Fig. 3. Results of merging simulated rectilinear maps. Matched vertices are shown with large dots. The best matching occurs when map $\mathcal{B}$ is rotated $-90°$.

Figure 3 shows the matching found for two randomly generated maps in a maze-like world. The result is a cluster of three consistent hypotheses. In computing the matching, there were 3918 initial vertex pairs; after the topological growth process, there were 72 hypotheses, which were placed into 69 clusters. The entire process took 0.04 seconds. Notice that just to the left of the leftmost (magenta) hypothesis are two vertices that should be matched and are not, an example of the clustering problem with small hypotheses.

The algorithm was also tested with maps of a real-world indoor environment (an academic building at RPI). Though these maps were of a reasonably large environment (approximately 12 m $\times$ 30 m), they were relatively small, particularly in terms of complexity, when compared to our simulated tests. With the real-world data, the robot found the correct matchings between partial maps in all cases. For further details and additional experiments, see [1].

6 Conclusions

In this paper, we have presented an algorithm for merging two topological maps. The algorithm uses the structure of the maps to find a set of hypothesized matchings, and then uses the geometric transformations of hypotheses to group them into consistent clusters. In addition, we have proposed approaches to map storage that are effective for our hypothesis-based merging, and that facilitate merging maps from multiple robots. Finally, we have discussed ways to reduce the computational cost when robots re-merge updated maps.

We have demonstrated our algorithm on simulated and real-world maps. In our experiments, even maps with minimal overlap are often merged correctly; for those that are not, our algorithm returns a set of consistent mergings. Maps with meaningful overlap were always merged correctly.

Acknowledgement. We would like to thank Charlene Tsai and Chuck Stewart for discussions on the relationship between image registration and topological map merging. This work was supported by the NSF through grant IIS–9983642.

References

1. K. R. Beevers. Topological mapping and map merging with sensing-limited robots. Master's thesis, Rensselaer Polytechnic Institute, Troy, NY, April 2004.
2. P. Besl and N. McKay. A method for registration of 3-D shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992.
3. H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, 1997.
4. G. Dedeoglu and G. S. Sukhatme. Landmark-based matching algorithm for cooperative mapping by autonomous robots. In L. E. Parker, G. W. Bekey, and J. Barhen, editors, *Distributed Autonomous Robotic Systems 4*, pages 251–260. Springer-Verlag, 2000.
5. T. Duckett, S. Marsland, and J. Shapiro. Learning globally consistent maps by relaxation. In *IEEE Intl. Conf. on Robotics & Automation*, pages 3841–3846, 2000.
6. G. Dudek, M. Jenkin, E. Milos, and D. Wilkes. Topological exploration with multiple robots. In *7th Intl. Symp. on Robotics with Applications*, 1998.
7. J. Fenwick, P. Newman, and J. Leonard. Cooperative concurrent mapping and localization. In *IEEE Intl. Conf. on Robotics & Automation*, 2002.
8. J. M. Fitzpatrick, D. L. Hill, and C. R. Maurer, Jr. Image registration. In M. Sonka and J. M. Fitzpatrick, editors, *Handbook of Medical Imaging, volume 2: Medical Image Processing and Analysis*, chapter 8. SPIE, 2000.
9. M. Golfarelli, D. Maio, and S. Rizzi. Elastic correction of dead-reckoning errors in map building. In *Intl. Conf. on Intelligent Robots and Systems*, pages 905–911, 1998.
10. J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai. A practical, decision-theoretic approach to multi-robot mapping and exploration. In *Intl. Conf. on Intelligent Robots and Systems*, 2003.
11. K. Konolige, D. Fox, B. Limketkai, J. Ko, and B. Stewart. Map merging for distributed robot navigation. In *Intl. Conf. on Intelligent Robots and Systems*, pages 212–217, 2003.
12. F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4(4):333–349, 1997.
13. S. Thrun. A probabilistic online mapping algorithm for teams of mobile robots. *Intl. Journal of Robotics Research*, 20(5):335–363, 2001.
14. R. Wilson and E. Hancock. Structural matching by discrete relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6), June 1997.