

# Topological Map Merging

Wesley H. Huang      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

## Abstract

When multiple robots cooperatively explore an environment, maps from individual robots must be merged to produce a single globally consistent map. This is a difficult problem when the robots do not have a common reference frame or global positioning. In this paper, we describe an algorithm for merging embedded topological maps. Topological maps provide a concise description of the navigability of an environment, and, with measurements easily collected during exploration, the vertices of the map can be embedded in a metric space. Our algorithm uses both the structure and the geometry of topological maps to determine the best correspondence between maps with single or multiple overlapping regions. Experiments with simulated and real-world data demonstrate the efficacy of our algorithm.

## 1 Introduction

Many applications of multiple mobile robot systems require the ability to explore and map an environment. Some of these applications are time-critical, and maps must be created as quickly as possible. Examples include urban reconnaissance, search and rescue operations, and security monitoring. Even in less extreme circumstances such as house cleaning, it may be desirable to map an area efficiently to minimize power consumption or cleaning time. When a team of several robots is assigned to the task, each robot can explore only part of the environment. The individual robot maps must then be merged to form a complete map.

In this paper, we describe an algorithm for merging topological maps. Topological maps (Kuipers, 1978) use a graph to represent possibilities for navigation through an environment. Vertices represent “places” in the environment, and edges represent paths (or classes of paths) between these places. Often, vertices are junctions of hallways, and edges are paths through those hallways. Topological maps provide a concise description of the structure (or “topology”) of the environment. Since our focus is on indoor environments, the use of topological maps is appropriate.

Topological maps, however, typically represent more than just the structure of the environment. Additional information, such as the degree of vertices, the orientation of edges at vertices, and other attributes, is typically recorded and stored in annotations of the graph. With a minimal amount of metric information (e.g., orientation of edges at each vertex and path length for each edge), there is enough information to embed the vertices in a metric space and thus recover the geometry of the map.

Whereas previous approaches to topological map merging and related problems have focused on using either map structure (e.g., Dudek et al., 1998) or map geometry (e.g., Dedeoglu and Sukhatme, 2000), our algorithm takes advantage of both. Our use of map structure allows quick identification of potential matches (and rejection of mismatches), while the use of map geometry enables our algorithm to directly merge maps with multiple (disconnected) overlapping regions. Our experiments have demonstrated that this algorithm can successfully merge many different types of topological maps.

Our algorithm proceeds in two phases and is inspired by methods from graph matching and image registration.

The first phase of our algorithm identifies hypothesized matches, i.e., sets of vertex and edge pairs, between the two maps. These correspond to common connected subgraphs of the maps and reflect areas of the environment with identical structure. Exactly known attributes of paired vertices or edges, such as the degree of vertices in a static world, should match perfectly. However, attributes 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. Thus, a hypothesized match is a common area with compatible annotations and the same “topological” structure and local geometry.

The second phase of our algorithm considers the global geometry of the matches. For each hypothesis, we can estimate a geometric transformation on one map to bring paired vertices into alignment. Geometrically compatible hypotheses give rise to similar transforms, so the hypotheses can be clustered in the transformation space. The best cluster (based on size and error) is returned as the algorithm’s result, and the vertex and edge correspondences from that cluster are used to merge the maps.

In the remainder of this section, we describe related work and our assumptions. Section 2 discusses the details of forming hypothesized (structural) matches between the maps, and Section 3 describes the transform estimation and clustering. We present experimental results in Section 4 and discuss a number of extensions in Section 5.

## 1.1 Related work

In recent years, there has been increased interest in map-making with multiple robots. While most multi-robot mapping work has focused on the creation of occupancy grid maps (Thrun, 2001; Grabowski et al., 1999; Ko et al., 2003), some work has been done with feature-based or topological maps (Dedeoglu and Sukhatme, 2000; Fenwick, Newman, and Leonard, 2002; Stroupe, Martin, and Balch, 2001). Jennings, Kirkwood-Watts, and Tanis (1998) use individual robots to create maps from Voronoi paths, and then use a simple distance metric to merge the maps. Dudek et al. (1998) create maps of a graph-like world under the assumption that all robots start at the same vertex in the graph. Konolige et al. (2003) pose the map merging problem in a decision-making framework, and show the efficacy of taking a feature-based approach to merging instead of attempting to match occupancy data.

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. (2003) in which robots exchange occupancy maps and attempt to localize themselves in each others' maps using a version of particle filtering.

The work most related to ours is that of Dedeoglu and Sukhatme (2000), who present 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 pair other vertices that are close together under this transformation. In contrast, our algorithm creates vertex and edge pairings using the structure of the maps and then estimates a transformation using this match. By comparing the map structure, we can discard mismatches earlier in the algorithm, and our transformations can be computed using multiple-vertex matches instead of single-vertex matches. An earlier version of this work appeared in (Huang and Beevers, 2004a).

Our approach to this problem draws from the areas of graph matching and image registration, though there are some significant differences in our problem domain. We discuss both of these areas in turn.

### 1.1.1 Graph matching

Graph matching (Bunke, 2000) is an area that has seen an explosion of research over the past three decades, particularly in the pattern recognition literature (Conte et al., 2004). There

are many different problems in this area; the one most relevant to topological map merging is the maximal common subgraph problem (McGregor, 1982; Bunke, 1997; Durand et al., 1999). In this problem, the two graphs may have annotations on the edges and vertices, and the objective is to identify the largest set of compatible vertex and edge pairings.

The maximal common subgraph (MCS) problem is a well-known NP-hard problem; however, there are some significant differences in the topological map merging problem that make the structural phase of our algorithm require only polynomial time. First, the edges incident to a vertex in a topological map have spatial interrelationships. In the planar case, this could be represented as the counterclockwise ordering of the edges about the vertex. This results in a number of different edge pairings at a vertex that is linear in the degree of the vertex, whereas it is exponential for the MCS problem. Second, while the matched subgraph need not be connected for the MCS problem, we are only interested in connected subgraphs because each connected subgraph can result in a different geometric transformation between the two maps. The geometric compatibility of disconnected matches is evaluated in the second phase of our algorithm.

Another relevant problem in graph matching is error-tolerant subgraph isomorphism (Bunke and Allermann, 1983). This problem is to identify a common subgraph when there may be missing vertices and edges. Solutions to this problem are generally evaluated in terms of the “edit distance,” the smallest sequence of elementary graph operations (i.e., substitution, deletion, insertion) that transforms one subgraph into the other. When a robot cannot reliably detect “places” or when the environment is dynamic, the resulting topological map may have missing vertices and edges. For now, however, we assume that the maps must match directly.

### 1.1.2 Image registration

If the vertices of the topological maps are embedded in a metric space and the edges discarded, then the map merging problem becomes the same as image registration. A thorough (but aging) survey of image registration techniques was written by Brown (1992). Most image registration algorithms are iterative in nature: they attempt to find transformations between image coordinate frames by repeatedly minimizing the distances of closest-point correspondences of features.

The iterative closest point (ICP) algorithm (Besl and McKay, 1992; Chen and Medioni, 1992) is the basis of the most widely used class of image registration techniques. This algorithm takes an initial matching between feature points and computes a transformation between the two feature sets by minimizing the (weighted) squared error between corresponding features. Next, the 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.

Since topological maps contain structure that is relevant for merging maps, it is not appropriate for us to simply use the ICP algorithm. From the first phase of our algorithm, we have matches that take this structure (and local geometric information) into account, so we do not expand the matches: they are already maximal matches! In the second phase of our algorithm, a transform is estimated using a simple point-based rigid transformation algorithm as described by Fitzpatrick, Hill, and Maurer (2000). This is further described in Section 3.

## 1.2 Assumptions

We assume that the individual topological maps have been constructed by reasonably capable robots and are therefore consistent, i.e., no two vertices represent the same “place.” This implies that the robots can infer when they have revisited a place and thus are able to “close the loop.” The robots must be able to reliably recognize “places” that correspond to vertices, but we do not assume that any sensed attribute of a vertex or edge is globally unique. An environment or sensor that permits this renders the map merging problem trivial.

Sufficient odometric information must be available so that the map vertices can be embedded in a metric space. In the planar case, the path transformations (distances for straight-line paths) and the angles between edges at vertices are sufficient. Such information is generally necessary anyway for loop-closing in individual maps. We assume that there is a known error model for these measurements and that, errors notwithstanding, only a translation and rotation (but no scaling) are needed to merge the maps.

Unlike most other map merging work, we do not assume that the robots have sufficient sensing or global knowledge to maintain a common reference frame. This could arise in an application when robots with no global positioning sensor are inserted into a building at different locations.

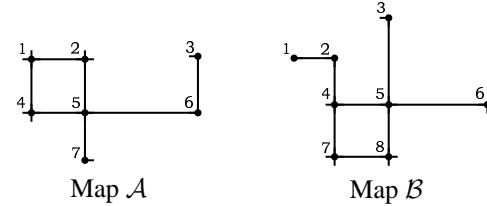
Note that though our examples are from planar environments, our algorithm will work with topological maps embeddable in any metric space.

## 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 the hypothesis building process by creating all compatible vertex pairings. Vertices are tested for compatibility by examining their attributes: exactly known attributes (e.g., vertex type) must match perfectly; in exactly known attributes



Hyp.	Rot.	Vertex correspondences
$H_1$	$0^\circ$	a3–b6
$H_2$	$180^\circ$	a7–b2, a1–b8, a2–b7, a4–b5, a5–b4
$H_3$	$180^\circ$	a7–b6
$H_4$	$0^\circ$	a1–b4, a2–b5, a4–b7, a5–b8
$H_5$	$0^\circ$	a1–b7, a2–b8
$H_6$	$0^\circ$	a1–b8
$H_7$	$0^\circ$	a2–b7
$H_8$	$90^\circ$	a2–b8
$H_9$	$180^\circ$	a1–b7, a4–b4
$H_{10}$	$180^\circ$	a4–b7
$H_{11}$	$270^\circ$	a1–b4, a2–b7
$H_{12}$	$270^\circ$	a1–b5, a2–b8, a4–b4, a5–b5
$H_{13}$	$270^\circ$	a1–b7
$H_{14}$	$270^\circ$	a1–b8, a4–b7
$H_{15}$	$270^\circ$	a4–b8

Figure 1: Hypothesis generation example for two maps from a rectilinear world. Since there are only four possible orientations in a rectilinear world, hypotheses consist of a rotation (of Map  $\mathcal{A}$ ) and a list of vertex correspondences. There are 15 hypotheses that result from these two maps.

(i.e., those subject to measurement error) must be compared with a similarity test. In our implementation, we used a test of statistical significance; see Section 4.1 for details.

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.

A vertex pairing must also specify which incident edges are to be paired together. Two vertices can be matched with several different edge correspondences, each of which is considered as a separate vertex pairing. The spatial relationship between incident edges should be used to determine which (if any) edge pairings are acceptable. In the planar case with Gaussian error, the squared angular orientation error between paired edges should be minimized by adjusting the relative orientation of the vertices; if this error is too large, the edge pairing should be rejected.

These compatible vertex pairs form the initial set of matches between the two maps. They should be stored in a hash table for efficient access during the growth step.

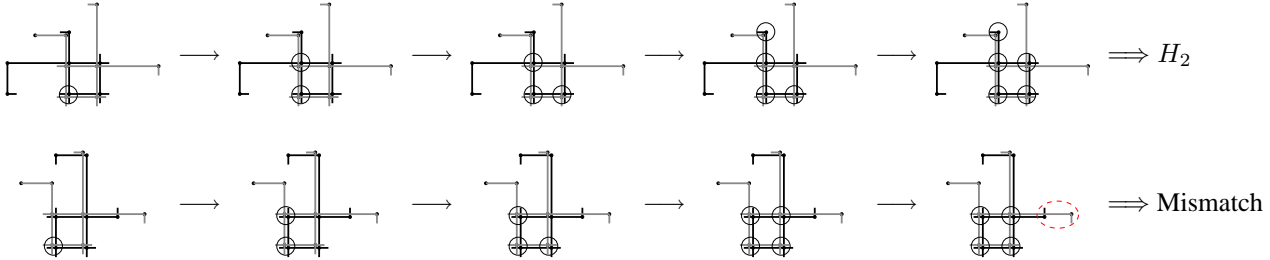


Figure 2: Growing matches for the example maps from Figure 1. Map  $\mathcal{A}$  (solid) is transformed to align the first paired vertex with its match in Map  $\mathcal{B}$  (gray); circles indicate paired vertices. The match is expanded by comparing corresponding pairs of edges and adding vertex pairs until the match is maximal (top example, which results in Hypothesis 2) or until the match is shown to be inconsistent (bottom example). Note, for the bottom example, that once the a7–b6 mismatch is discovered, the four circled vertex pairs will be discarded. The topmost pair of vertices (a6–b3) is still a valid pairing, but will be rejected upon expansion because the a5–b5 pair has already been discarded.

## 2.2 Growing matches

The single-vertex matches must next be expanded using the structure of the maps. We pick one of the matches and test corresponding pairs of edges leaving the paired vertices. Like vertices, edges may have exactly known attributes that can be compared directly and inexact attributes (such as path length) that must be compared using a similarity test.

If the edges are compatible and the vertices at the ends are also compatible, then they are added to the match. Note that all compatible vertex pairs have already been computed, so this simply requires looking up the vertex pair in the hash table. If it is found, the vertices are compatible and the pair is removed from the table so that a duplicate match will not be re-grown from that single-vertex pairing. If the edges are incompatible or the vertices are not found in the hash table, the entire match is rejected.

Note that an edge may have been explored in one map but not the other. This does not constitute an incompatibility — only edges explored in both maps are cause for rejection or expansion of the match.

This process is repeated until the match cannot be expanded further; the match then becomes a hypothesis, and we attempt to grow any remaining single-vertex matches. Note that a subgraph in one map can be matched to multiple subgraphs in the other (under separate hypotheses), but any pair of vertices (with a given edge correspondence) can appear in only one hypothesis.

## 2.3 An example

For an example of hypothesis generation, we use two maps from a simple 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 this example, we also assume a static world, so vertex degrees must match exactly. Degree two vertices (which correspond to a corner) can match only for a single edge pair-

ing; degree four vertices can match under four different edge pairings.

We generate hypotheses as discussed previously. Valid hypotheses that remain after the growth process are shown in Figure 1. Figure 2 depicts the growth process for a valid matching between the maps of Figure 1, as well as for an invalid matching.

## 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 to group consistent hypotheses together into a single match.

### 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 and Saffiotti (2000), Lu and Milios (1997), and Golfarelli, Maio, and Rizzi (1998). 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 our planar examples, we estimate transforms using a two-dimensional version of the point-based rigid registration algorithm described by Fitzpatrick, Hill, and Maurer (2000). This algorithm constructs a two-by-two covariance matrix using the displacements of pairs of matched vertices from the centroids of their respective point sets. It then computes the singular value decomposition (SVD) of this matrix. The unitary matrices from the SVD are used to find the best

rotation between the two point sets; the optimal translation is then found from the vertex centroids in each map. This algorithm has the effect of finding the transformation between maps that minimizes squared error between matched vertices.

### 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 the total metric error between matched vertices under the hypothesis transform. Hypotheses within the threshold distance in transformation space are clustered together. There are a variety of techniques for clustering; we used a simple hierarchical agglomerative clustering method in our implementation, as follows:

1. Start with each hypothesis in its own cluster.
2. Compare all pairs of clusters and mark the pair that is closest.
3. Compare the distance between this pair to the threshold distance:
  - If the distance is less than the threshold, group the pair into a single cluster and return to Step 2.
  - If the distance is greater than the threshold, clustering is complete.

A naïve implementation of this algorithm requires  $O(n^3)$  time, where  $n$  is the number of hypotheses. A more efficient implementation that builds a similarity matrix and updates it at each cluster joining needs only  $O(n^2 \log n)$  time but requires  $O(n^2)$  memory (Day and Edelsbrunner, 1984).

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

- Total number of vertices is a good primary indicator of cluster quality.
- The amount of error (i.e., total squared error between matched vertices 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. All other things being equal, a single large hypothesis is preferable to a cluster of small hypotheses since it is also known to be structurally consistent.

Note that 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. In general, clusters containing only one or two pairs of vertices — especially clusters with only a small number of single-match hypotheses — are not meaningful unless the vertices are somehow unique. If the best cluster is very small or has large error, the merging algorithm should treat this as failure to find a match.

### 3.3 Example continued

To continue our example, we first embed the maps using a version of the algorithm by Duckett and Saffiotti (2000) adapted to planar rectilinear maps.

Figure 3 shows the hypothesis clusters for our example. Because this example is in a rectilinear world, the transformation 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.

## 4 Results

We have implemented the map merging algorithm for arbitrary two dimensional topological maps, and have tested this implementation with a variety of maps. Our results indicate that the algorithm is very effective, even for large maps with small overlap. Experimentally, we have found that the algorithm even performs well when the hypothesis formation considers only the structure of the maps (and not local geometry or other attributes).

### 4.1 Implementation notes

Our experiments assume the robot is able to determine the degree of a map vertex after visiting it. In our figures, a “stub” edge is drawn if the robot detected a path but has not explored it yet. Two vertices are considered potential initial matches if they have the same degree and the incident edges are reasonably aligned. The number of possible correspondences between a pair of vertices of the same degree is thus at most equal to that degree.

In our implementation, similarity of edge lengths was tested based on a Gaussian odometry error model. A statistical significance test was used to determine the likelihood that two length measurements were taken from the same edge. This

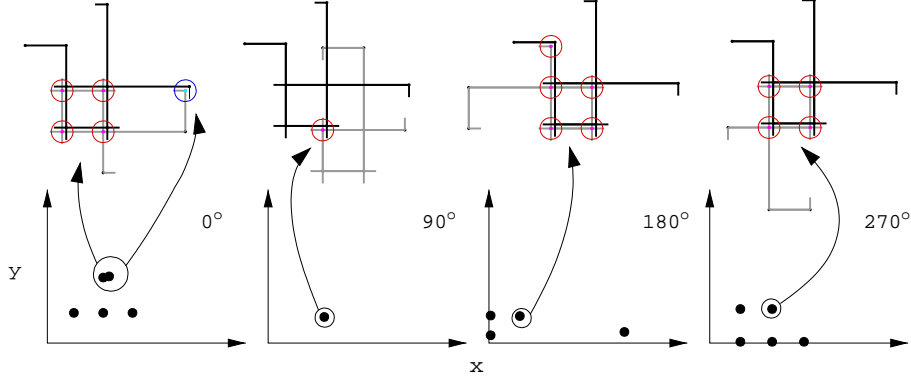


Figure 3: The hypotheses from our example clustered in the transformation space. Since these maps are from a rectilinear world, the transformation space consists of a translation space for each of the four rotations. One cluster for each rotation is illustrated; map  $\mathcal{A}$  is shown in black, map  $\mathcal{B}$  is shown in gray, and matched vertices are circled. The  $180^\circ$ , 5-vertex cluster (from the single hypothesis  $H_2$ ) is the best match; the  $0^\circ$ , 5-vertex cluster consisting of two hypotheses ( $H_1$  and  $H_4$ ) is the second best match.

was done by computing the  $z$ -score of the measurements:

$$z^2 = \sum_i \frac{(l_i - \hat{\ell})^2}{\sigma^2} \quad (1)$$

where the  $l_i$  are the edge measurements,  $\hat{\ell}$  is the estimate of the edge length, and  $\sigma^2$  is the variance from the error model. The  $z$ -score follows a  $\chi^2$  distribution, which can be used in computing the desired likelihood.

Edge pairings were rejected if the computed likelihood was small. In our implementation, we used a very conservative threshold, rejecting edge pairings only if it was more than 99% certain that the measurements were not from the same edge.

Thresholds on intra-cluster translational distance in transformation space were set to 3 times the root of the largest mean squared (translational) error among the individual hypotheses; the threshold on rotational distance was fixed at  $\pi/8$ .

## 4.2 Map generation and merging

Map merging was performed on maps created by simulated exploration of maze like worlds, maps generated using random vertex placement, hand-made maps, and real-world data.

### 4.2.1 Maze maps

The first map generation method creates large random maze-like worlds (with loops) on a grid. It then generates a partial map of the world for each robot by picking a random start cell and performing a breadth-first search to a specified depth, using a simple “hall-following behavior.” Although the mazes are rectilinear in nature, the maps are not because of error. (No rectilinearity assumption is made by the robot as it makes its map.)

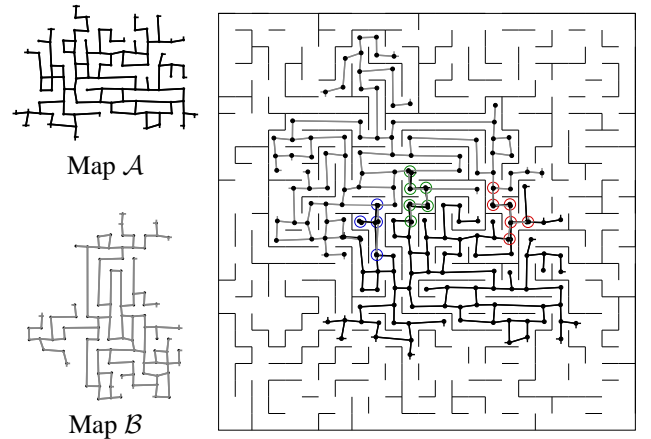


Figure 4: Merged maps from a simulated maze environment shown overlaid on the maze with paired vertices circled. This match corresponds to a cluster of three hypotheses; the associated transform rotates map  $\mathcal{B}$  by approximately  $-90^\circ$ .

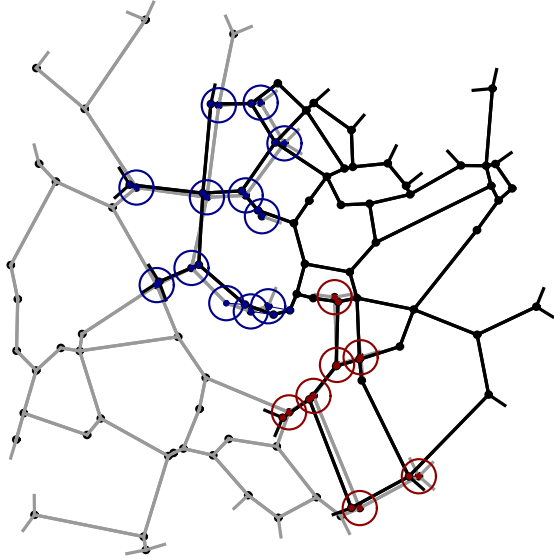


Figure 5: Two merged random planar maps. The best cluster consists of two hypotheses.

Error is introduced into the partial maps by randomly perturbing cell centers according to a Gaussian distribution with configurable standard deviation. The vertices of each partial map are perturbed independently. The resulting error approximates error in locating vertices of the topological map. Maps generated with this method are immediately embeddable in metric space since their edge length measurements are consistent.

Figure 4 shows two maps from a random maze environment and a merged map overlaid on the maze. The two maps overlap in several structurally disconnected places; the final result of the matching process is a cluster of three consistent hypotheses. Note that there is another small, two vertex match that appears to be consistent with this cluster, just to the left of the leftmost highlighted hypothesis. A conservative threshold on intra-cluster distance (in transformation space) prevented it from joining the other three hypotheses in the best cluster.

#### 4.2.2 Random planar maps

This method first creates an embedded graph by randomly placing vertices in the plane. These vertices are then connected randomly, with bias toward connecting nearby vertices, while ensuring that the map remains planar by disallowing edge intersections. Individual robot maps are created by picking a random start vertex and performing breadth-first exploration until a specified number of vertices have been visited. During this exploration, edge lengths in the robot maps are perturbed with Gaussian error. Maps produced in this fashion must be embedded prior to merging since they are not metrically consistent.

Figure 5 shows two merged random planar maps. This type

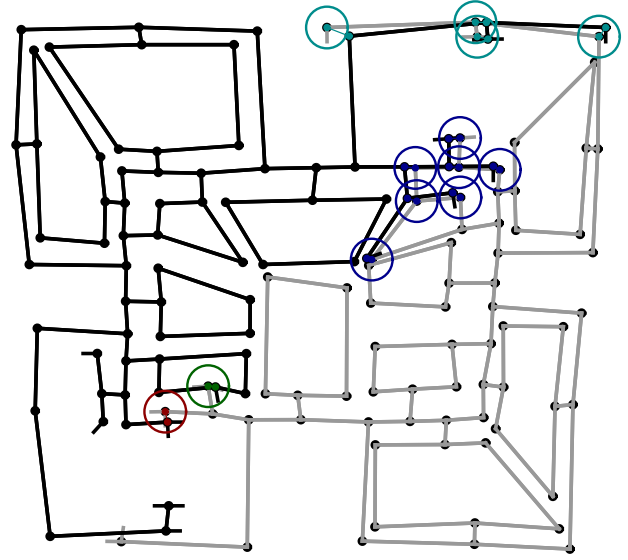


Figure 6: Merged maps from a hand-generated map of an office building-like environment.

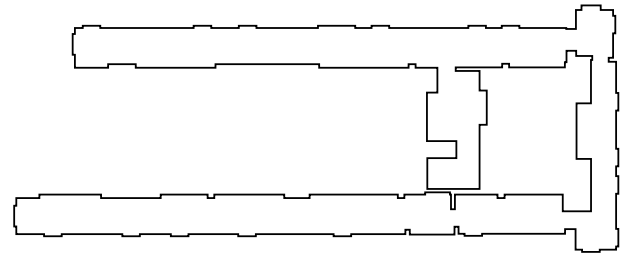


Figure 7: Plan of the first floor hallways and lounge of the Amos Eaton building at RPI. The dimensions are approximately 12 m  $\times$  30 m.

of map was used for much of the performance testing of the map merging algorithm.

#### 4.2.3 Hand-generated maps

A few hand-generated maps were created to test specific situations. These maps tended to be more structured than the random planar maps but less so than the maze maps.

Figure 6 shows a merging between two hand-generated maps. A complete map of a office building-like environment was created by hand, and the individual robot were generated maps by simulated exploration. Structurally, the environment is fairly self-similar in nature. The best cluster of hypothesized matchings between the two partial maps consists of four hypotheses, two of which are single vertex hypotheses. Note that once again, in the bottom left of the matched maps is a pair of vertices whose pairing is consistent with the cluster, but is not included because of error in the maps.

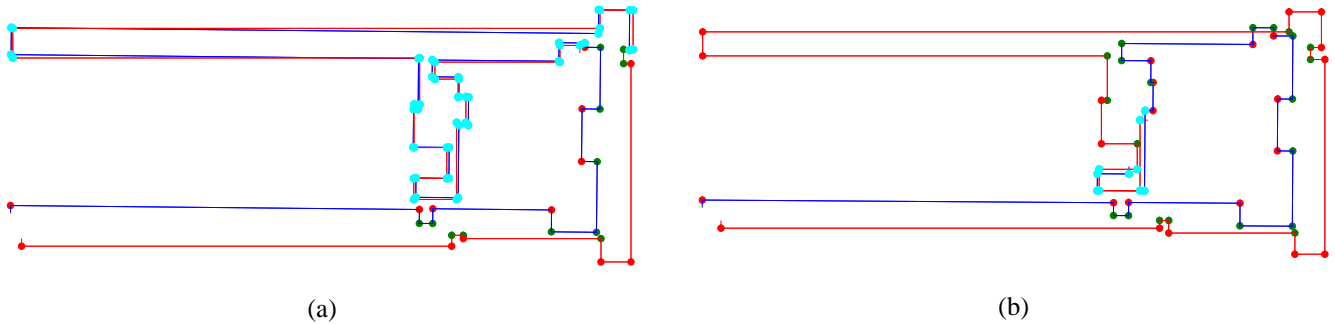


Figure 8: Matchings between partial maps of the Amos Eaton building at RPI. Matched vertices are the large light-colored dots. Figure (a) shows a matching for partial maps with relatively large overlap; Figure (b) shows a matching for maps with a smaller overlap. In both cases, the correct matching is found.

#### 4.2.4 Real-world maps

Our real-world data comes from a small mobile robot exploring the hallways of the first floor of an academic building — the Amos Eaton building at RPI, shown in Figure 7. A topological map of the building was created using a wall-following behavior, where vertices in the map are “well-defined” corners of the walls in the environment. For complete details of our robot hardware and the mapping algorithm, see (Huang and Beevers, 2004b).

The complete map was split into two separate maps, and artificial error was introduced into the overlapping parts of the map by randomly perturbing vertices, in similar fashion to the methods used in creating maze maps. Figure 8 shows merged maps from two pairs of maps with different amounts of overlap between the maps.

### 4.3 Performance

Even for very large maps (300 vertices each) with considerable overlap, the matching process was quick, taking less than 0.5 seconds on a 650 MHz Pentium 3 processor.

Table 1 reports statistics from merging the example maps shown in Figures 4–8. Note that with the real-world data (Figure 8), there are very few remaining hypotheses after the structural growth phase of the algorithm. To some extent this is due to the relatively small sizes of the maps. In part, though, the small number of hypotheses occurs because most real-world environments present few regions with structural self-similarity for maps with meaningful overlap; also, there are typically many structurally distinctive features in real-world environments.

The maze maps were the most self-similar due to the underlying rectilinear grid of the maze. Despite the self-similarity, our map merging algorithm performs well on these maps in most cases, even when the maps are large and have small overlap.

We conducted more extensive tests of our algorithm using random planar maps. Since the run-times were all reasonably

short, we only collected statistics on whether the algorithm merges the maps correctly. Table 2 shows our results. In the first set of tests, the degree of overlap between the two maps was varied; the algorithm showed very little variation in performance with different amounts of overlap, including the case of zero overlap in which the algorithm concludes that there is no meaningful match between the maps. In these tests, the algorithm was given an accurate model of the error.

A second set of tests varied the amount of noise in the maps while leaving the merging algorithm’s error model unchanged. Noise in the maps is generated by perturbing edge lengths according to a Gaussian distribution with standard deviation equal to some percentage of the true edge length, giving longer edges more error than shorter edges. In this set of tests, the map noise was generated with a standard deviation varied from 1% to 11% of the edge length (the “map  $\sigma$ ” in Table 2), while the merging algorithm assumed 5%.

The algorithm performed well when there was less error than assumed; though some incorrect edge matchings were accepted, topological comparison was generally sufficient to reject incorrect matches. In some circumstances, matches that are topologically correct but metrically incorrect might be accepted if measurement error is significantly overestimated. When there was more error in the map than assumed, the algorithm performed poorly. At 7%, the algorithm was still correct in 91.9% of the tests, but performance declined sharply with more error. Under these circumstances, a correct hypothesis is often rejected because corresponding edges or vertices appear too different under the assumed error model. For this reason, it is best to use conservative error estimates, assuming greater measurement error than normally expected.

### 4.4 Structure-only hypothesis formation

When the environment is “structurally rich,” it is possible to entirely disregard local geometric information (e.g., edge lengths) in the hypothesis formation phase of our algorithm, thus avoiding the need for an error model.

We repeated the two sets of tests on random planar maps

Table 1: Example merging statistics

Statistic	Fig. 4	Fig. 5	Fig. 6	Fig. 8a	Fig. 8b
Size (map 1 / map 2 vertices)	97/91	53/53	70/70	39/36	28/27
Single-vertex correspondences	3918	2497	2011	708	378
Hypotheses after growth	72	85	14	3	4
Clusters	69	83	10	3	4
Best cluster (hypotheses/vertices)	3/16	2/19	4/13	1/25	1/5
Running time (seconds)	0.04	0.02	0.02	0.01	< 0.01

Table 2: Map merging statistics (1000 runs for each trial)

Overlap	% correct	map $\sigma\%$	% correct
0%	99.2%	1%	99.0%
2%	99.1%	3%	99.5%
4%	99.6%	5%	99.6%
6%	99.4%	7%	91.9%
8%	99.2%	9%	36.5%
10%	99.3%	11%	12.6%
12%	99.3%		

Table 3: Structure-only growth map merging statistics (200 runs for each trial)

Overlap	% correct	map $\sigma\%$	% correct
0%	99.0%	1%	99.5%
2%	98.5%	3%	98.5%
4%	99.0%	5%	99.5%
6%	99.5%	7%	99.0%
8%	98.0%	9%	98.5%
10%	99.0%	11%	99.5%
12%	99.0%		

with a modified version of our algorithm that does not perform a similarity test on edge lengths or relative edge orientations; the results are shown in Table 3. The performance of the algorithm is comparable to the full version of the algorithm but without the performance reduction when the error is larger than assumed.

The tradeoff for not using metric information during the hypothesis formation phase is that fewer bad hypotheses are rejected. Since clustering is the most expensive part of our algorithm, this could lead to significantly longer computation times. However, in many real-world situations, maps are small enough and different parts of the environment are structurally unique enough that this can be a viable tradeoff. For environments that have significant structural self-similarity, such as our simulated maze environments, this approach is not so effective; the use of metric information is necessary to distinguish between structurally similar parts of the environment.

## 4.5 Clustering & transformation estimation

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 detection error; as such, the matches were not added to otherwise correct clusters of hypotheses. Section 5.3 discusses a solution to this problem.

The estimated geometric transformations for correct clusters sometimes introduce significant “skew” into the merged map because of inaccuracy in the transformation estimates. The skew is largest for maps with small overlap; maps with large overlap provide more data for use in transform estimation. Note that estimated transformations can be improved as individual robots expand their own maps and compare their new data with the merged map.

## 5 Extensions

As described, our map merging algorithm works well and is reasonably efficient. In practice, there are several ways of it could be extended to be more efficient in particular circumstances. Here, we discuss techniques for map merging and storage, incremental updates, and other computational issues.

### 5.1 Map merging & storage

Once a hypothesis cluster has been deemed correct, it is fairly straightforward to merge (or “flatten”) the two maps into a single map. 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.

However, 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. To protect against such situations, it is useful to store and merge maps so that incorrect hypotheses can be removed without discarding the whole map. The simple solution is to store the map from each robot separately, either

computing a separate merged map, or computing merged measurements as needed and caching values as memory permits. Should a hypothesis later be found incorrect, the merged map can be recomputed or the cached values marked invalid.

This approach extends to the case of merging maps with multiple robots, but a more complex dependency tree structure arises when a merged map from two robots is merged with the map of a third robot. Note that another robot could send a single (merged) map or its entire dependency tree with all the raw map data. Since each merging operation takes a pair of maps and produces a combined map, the dependencies can be represented by a binary tree in which the leaf nodes are the raw maps, and the root node represents the current overall map.

At some point, however, memory limitations may preclude storing the raw maps of all robots. In this case, two matched maps can be permanently merged. In the dependency tree, this has the effect of replacing a node and its children with a new node whose children are all the grandchildren of the original node. After a permanent merging operation, the raw maps can be discarded.

## 5.2 Incremental updates

After merging maps once, two robots may later merge their maps again; an incremental update can save a substantial amount of computation. This is possible with minimal book-keeping effort: each robot must maintain timestamps so that only new and modified vertices and edges (since the last update) are exchanged.

The computational savings depends on the amount of information retained from the original map merging. At one extreme, all hypotheses and cluster information are retained; at the other, only the best cluster of hypotheses is kept. In between, we might keep the hypotheses from some number of the best clusters.

Briefly, the steps for an incremental update are as follows:

1. Discard old hypotheses invalidated by the updated information.
2. Extend existing hypotheses from new and modified edges incident to the hypothesis.
3. Create and grow hypotheses from new and modified vertices. To update only existing clusters, hypotheses could be grown only from new vertex pairings with transforms geometrically close to that of an existing cluster.
4. Recluster the hypotheses after removing modified hypotheses from their previous clusters.

In the worst case, when performing every computation described above, the incremental update is computationally equivalent to the basic matching algorithm. However, in many cases, one or two hypothesis clusters are clearly superior to all others. Expanding only hypotheses in these clusters and

not growing new hypotheses can further reduce computation; this computation is linear in the number of modified and new vertices and edges. Note that even if other hypotheses are not updated, this can be done at any time if necessary — for example, if the previous best hypotheses are rejected during an update.

## 5.3 Iterative reclustering

A problem that occasionally occurs when merging maps with multiple overlapping regions (multiple correct hypotheses) is that, because of error in the maps, correct hypotheses are sometimes not included in the best cluster. Most often, this occurs with small hypotheses of one or two vertices, where noise can lead to transformation estimates that are significantly different from the true transformation.

A solution is to take an “iterative” clustering approach, similar to the iterative closest point methods used in image registration. After the initial clustering, new hypotheses may be added to a cluster based on metric error between paired vertices under the cluster transformation, rather than on distance in transformation space. This process can be repeated — adding hypotheses and re-estimating cluster transforms — until no cluster changes occur.

## 5.4 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 spatially close to both robots.

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.

## 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 extensions to the algorithm that improve the efficiency of practical implementations. These extensions focus on strategies for map storage, incremental map updates, and clustering that reduce the computational cost of map merging and can improve the results.

We have demonstrated our algorithm on simulated and real-world maps. In our experiments, even maps with minimal overlap are almost always merged correctly; for those that are

not, our algorithm returns a set of consistent mergings. The algorithm is most sensitive to discrepancies between the error model used in merging and the true error in the maps; overestimating the error reduces the efficiency of the algorithm but guards against discarding correct matches. The algorithm is less sensitive to the size of overlap between maps being merged. Maps with meaningful overlap (more than one or two vertices) were merged correctly in the vast majority of cases. In our real-world tests, maps were always merged properly.

The map merging algorithm presented here is capable of merging topological maps in greater than two dimensions, provided sufficient metric information is present in the maps. The addition of a second angle for edges incident to map vertices is sufficient to extend the maps to three dimensions; this angle can be compared using a similarity test, as with the other inexact map attributes.

At present, the merging algorithm assumes maps are made in static environments. Extending the algorithm to handle quasistatic or dynamic environments will greatly improve its applicability to real-world situations. The issue of dynamic environments is also related to the problem of structural errors and inconsistencies in the maps; detecting and resolving these situations is necessary, since they are likely to occur in sufficiently complex environments.

## 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

- Besl, P. and McKay, N. 1992. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256.
- Brown, L. 1992. A survey of image registration techniques. *ACM Computing Surveys*, 24(4):325–376.
- Bunke, H. 1997. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694.
- Bunke, H. and Allermann, G. 1983. Inexact graph matching for structural pattern recognition. *Pattern Recognition Letters*, 1(4):245–253.
- Bunke, H. 2000. Graph matching: Theoretical foundations, algorithms, and applications. In *International Conference on Vision Interface*, 82–88.
- Chen, Y. and Medioni, G. 1992. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3):145–155.
- Conte, D., Foggia, P., Sansone, C., and Vento, M. 2004. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(3):265–298.
- Day, W. and Edelsbrunner, H. 1984. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification*, 1(7):1–24.
- Dedeoglu, G. and Sukhatme, G. 2000. Landmark-based matching algorithm for cooperative mapping by autonomous robots. In *Proceedings of the 2000 International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 251–260.
- Duckett, T. and Saffiotti, A. 2000. Building globally consistent gridmaps from topologies. In *Proceedings of the 6th International IFAC Symposium on Robot Control (SYROCO)*, 357–361.
- Dudek, G., Jenkin, M., Milos, E., and Wilkes, D. 1998. Topological exploration with multiple robots. In *Proceedings of the Seventh International Symposium on Robotics with Applications (ISORA '98)*.
- Durand, P., Pasari, R., Baker, J., and Tsai, C. 1999. An efficient algorithm for similarity analysis of molecules. *Internet Journal of Chemistry*, 2, article 17.
- Fenwick, J., Newman, P., and Leonard, J. 2002. Cooperative concurrent mapping and localization. In *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*, 1810–1817.
- Fitzpatrick, J., Hill, D., and Maurer, Jr., C. Image registration. In Sonka, M. and Fitzpatrick, J., editors, *Handbook of Medical Imaging, volume 2: Medical Image Processing and Analysis*, chapter 8. SPIE, 2000.
- Golfarelli, M., Maio, D., and Rizzi, S. 1998. Elastic correction of dead-reckoning errors in map building. In *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots & Systems*, 905–911.
- Grabowski, R., Navarro-Serment, L., Paredis, C., and Khosla, P. 1999. Heterogeneous teams of modular robots for mapping and exploration. *Autonomous Robots*, 8(3):293–308.
- Huang, W. H. and Beevers, K. R. 2004a. Topological map merging. In *7th International Symposium on Distributed Autonomous Robotic Systems (DARS 2004)*, 91–100.
- Huang, W. H. and Beevers, K. R. 2004b. Topological mapping with sensing-limited robots. In *6th International Workshop*

- on the Algorithmic Foundations of Robotics (WAFR 2004)*, 367–382.
- Jennings, J., Kirkwood-Watts, C., and Tanis, C. 1998. Distributed map-making and navigation in dynamic environments. In *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots & Systems*, 1695–1701.
- Ko, J., Stewart, B., Fox, D., Konolige, K., and Limketkai, B. 2003. A practical, decision-theoretic approach to multi-robot mapping and exploration. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots & Systems*, 212–217.
- Konolige, K., Fox, D., Limketkai, B., Ko, J., and Stewart, B. 2003. Map merging for distributed robot navigation. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots & Systems*, 212–217.
- Kuipers, B. 1978. Modeling spatial knowledge. *Cognitive Science*, 2:129–153.
- Lu, F. and Milios, E. 1997. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4 (4):333–349.
- McGregor, J. 1982. Backtrack search algorithms and the maximal common subgraph problem. *Software Practice and Experience*, 12(1):23–34.
- Stroupe, A., Martin, M., and Balch, T. 2001. Distributed sensor fusion for object position estimation by multi-robot systems. In *Proceedings of the 2001 IEEE International Conference on Robotics & Automation*, 1092–1098.
- Thrun, S. 2001. A probabilistic online mapping algorithm for teams of mobile robots. *International Journal of Robotics Research*, 20(5):335–363.