Topological Global Localization for Subterranean Voids

David Silver, Joseph Carsten and Scott Thayer

Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA
{dsilver,jcarsten,sthayer}@ri.cmu.edu

Summary. The need for reliable maps of subterranean spaces too hazardous for humans to occupy has motivated the development of robotic mapping tools. For such systems to be fully autonomous, they must be able to deal with all varieties of subterranean environments, including those containing loops. This paper presents an approach for an autonomous mobile robot to determine if the area currently being explored has been previously visited. Combined with other techniques in topological mapping, this approach will allow for the fully autonomous general exploration of subterranean spaces. Data collected from a research coal mine is used to experimentally verify our approach.

1 Introduction

In many parts of the world, abandoned mines present a significant environmental hazard. Toxic runoff, landslides, and subsidence are just some of the dangers presented by these structures. In the U.S. alone, there are tens of thousands of abandoned mines [3] that threaten nearby surface and subterranean operations. The first step towards combating this problem is to obtain an accurate metric survey of the mine structure. Unfortunately, in most cases an accurate survey of the mine has either been lost or never existed. Taking a new survey of the structure is often limited to inspections via boreholes, as abandoned mines are usually too dangerous for people to enter. For this reason, robots have been proposed as a method for mapping abandoned mines.

The Carnegie Mellon Subterranean Robotics group has undertaken the task of developing robotic systems that can autonomously explore abandoned mines or other hazardous subterranean voids. The initial effort led to the development of a system that can autonomously navigate and explore long stretches of a single mine portal [2]. More recent work has focussed on expanding mission profiles to include general exploration of multiple intersecting corridors. This led to a system which can detect and traverse multiple corridors [13], but can not determine when it has returned to a previously

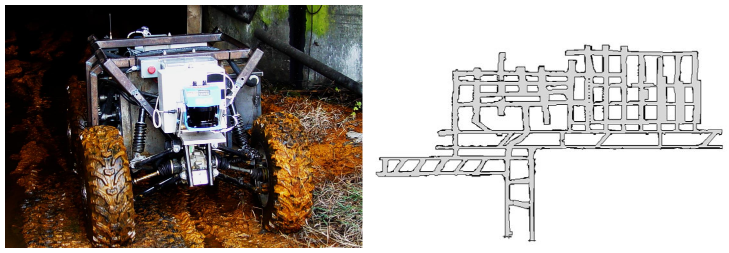


Fig. 1. Left: Groundhog, the current robotic platform of the mine mapping project. **Right:**This map was generated from data acquired during experimentation and utilizes offline globally consistent mapping techniques. It shows the highly cyclic nature of room-and-pillar mines.

visited corridor intersection from a different direction. This constraint limited the environments explored in [13] to those which did not contain loops.

This paper presents a method by which an autonomous mobile robot can identify correspondences between intersections in subterranean environments, allowing for autonomous loop closure and more general exploration. Our approach for matching intersections is based on comparisons of both 2D and 3D range data local to each intersection. The results of these comparisons are then fed to a binary classifier, which produces the probability of a match. Such a classifier can then be integrated into a complete system designed to track multiple topological map hypotheses.

The remainder of this paper discusses the relevant details of our approach. Section 2 provides background into subterranean topological exploration. Section 3 describes our technique, with experimental results presented in Section 4. We conclude with a discussion and directions for future work.

2 Subterranean Topological Exploration

2.1 Robotic Platform

Our current mine mapping platform is **Groundhog** (Figure 1), a 700 kg custom-built ATV-type robot that is physically tailored for operation in the harsh conditions of abandoned mines. Groundhog's primary sensing consists of 2 SICK LMS-200 laser range finders mounted in front and back. Each has a 180° field of view, and is mounted on a tilt mechanism with a 60° range. Tilting each laser allows for the acquisition of 3D range data. Groundhog has been used extensively in both test and abandoned mine environments, accruing hundreds of hours of mine navigation, including 8 successful portal entry experiments in the abandoned Mathies mine outside of Pittsburgh, PA. Offline techniques have been used to generate globally consistent, large-scale maps based on log data from these experiments. For a thorough overview of the Groundhog system, see [2].

2.2 Topological Representations

Topological representations coincide nicely with the inherent structure of room-and-pillar mines, which consist almost exclusively of narrow corridors and corridor intersections (see Figure 1). A topological map is a graph representation of an environment. The nodes of the graph correspond to distinct locations in the environment, and the edges correspond to direct paths between two such locations. For mines, nodes and edges correspond to intersections and corridors, respectively. This approach was used in [10] to allow a robot to traverse known mine environments. Topological maps have also proven useful in robotic exploration tasks of unknown environments [9]. Unexplored edges in a topological map correspond to unexplored regions of the environment, thus providing a mechanism for determining which region of the environment to explore next.

The key components of a system designed for autonomous topological exploration are:

- A method for traversing an edge in the environment until a node is reached.
- A method for detecting a node and its associated edges in the environment.
- A method for determining whether the currently sensed node has been visited before, and if so which previously visited node it corresponds to (this is the problem our current work strives to solve).
- A representation of the topological map and its associated uncertainty.

The first two components have been previously developed and tested in subterranean environments, as described in the following sections.

2.3 Edge Traversal

Edge traversal is the first necessary component for autonomous topological exploration. While traversing a single corridor, Groundhog utilizes the Sense-Plan-Act (SPA) framework. While stationary, Groundhog tilts one of its lasers to accumulate 3D range data from the space in front of it. This 3D point cloud is used to generate a 2.5D cost map. Next, a goal pose is chosen that will further Groundhog's progress down the corridor (or turn it into a new corridor). A path is planned to the goal pose by feeding the cost map into a nonholonomic motion planner described in [13]. The planned path is then traversed by Groundhog, and the whole process repeated. For a more detailed description, see [2, 13].

2.4 Node Detection

A method for node detection is also critical to topological exploration. Groundhog detects intersections in its environment by searching for nodes of the *generalized Voronoi diagram* (GVD) [6]. Edges of the GVD represent sets of points equidistant from 2 objects. Nodes of the GVD represent points

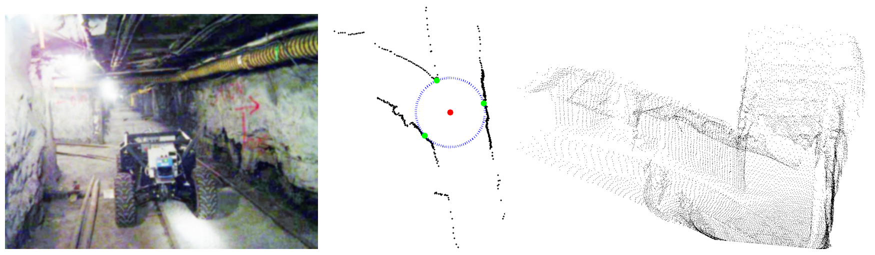


Fig. 2. The data collected at each node. **Left:** Groundhog approaching an intersection. **Center:** the 2D range data collected, as well as the detected node location and radius. **Right:** the 3D range data collected.

equidistant from 3 objects. While traversing an edge, potential GVD nodes are detected using a procedure described in [15]. Each potential node is then tracked until Groundhog drives through the intersection to which the node corresponds. The purpose of this extra traverse is to obtain a 2D map of the environment around the node with a full 360° coverage, as opposed to the 180° field of view of Groundhog's lasers. Such coverage is achieved by combining multiple laser scans from different vantage points. This 360° coverage is necessary to determine whether the intersection just traversed is worth exploring; if the end of a corridor is already within sensor range from the intersection itself, it may not be worth further exploration. This procedure also eliminates large concavities that can appear as intersections when first detected. After a node has been detected and verified, a 3D scan of the intersection is taken, and Groundhog continues its exploration. The Voronoi radius (equidistance value between the node and the objects that formed it), 2D map, and 3D scan (Figure 2) are all stored for later use.

2.5 Framework for Topological Uncertainty

For successful topological exploration, a robot must be able to determine if a given node has been previously visited. This determination can be made based purely on the local topology [7], or by combining topological information with range data or data on nearby features. The techniques described in this paper follow the latter approach.

Regardless of the specifics of the node matching approach, its output will be uncertain. There may be multiple previous nodes which match the current node closely enough to be considered a possible match, and the fact that the node may never have been previously visited adds additional uncertainty. A framework is necessary for dealing with this uncertainty until the ambiguity can be removed. A widely adopted approach is to maintain multiple hypotheses as to the correct topology of the environment [8, 11, 16]. The robot can then either take actions designed to explicity remove the ambiguity, or maintain multiple hypotheses until the natural exploration behavior of the robot produces enough additional information. In either case, the correct framework can add additional robustness on top of the chosen node matching scheme.

3 Subterranean Node Matching

We approach node matching as a topological global localization problem. When a robot arrives at a node N_i along edge E_i, it can localize itself to a discrete subset of all possible states in the world (the set of states located at a node, oriented along an edge). If the robot can properly match N_i and E_i to a previously visited N_j and E_j, then it will have relocalized itself. If the robot can properly determine that N_i has not been visited before, it will still have localized itself to the correct state, albeit a state that has not previously been visited.

To determine whether the current node N_i matches a previous node N_j, we use a hybrid approach based on both local topology and range data (Figure 3). Local topological data is rarely descriptive enough to determine explicitly whether two nodes match. However, it requires essentially no preprocessing: it is computationally inexpensive to determine whether N_i and N_j are of the same degree. For this reason, local topological data is used to pare down the number of prospective matches.

For similar reasons, 2D as well as 3D range data is used. While 2D range data is usually not descriptive enough to make an explicit determination, it is much cheaper to process than the full 3D point cloud, and can further pare down the number of prospective matches. 2D data has another advantage under our current setup: as described in Section 2.4, 2D information is collected a full 360° around the intersection. The additional coverage offered by 2D data often proves quite useful in determining final matches.

A common approach for determining whether a robot is revisiting a location is to explicitly search for features in the local environment, and try to match these features to those that have been previously detected. However, subterranean spaces provide a unique challenge for feature extraction. While such spaces are often feature rich, it is hard to characterize the features exhibited. Features can very greatly in both type and scale, and so a more robust approach is needed. For this reason, our approach compares nodes in a manner which does not require explicit extraction of predetermined features.

3.1 Comparison of Topological Properties

The first step of our node matching scheme is to use the topological properties of the detected node N_i to eliminate as many nonmatching nodes N_j as possible. These topological properties are the degree of the node and its associated Voronoi radius. Another property we explored was the relative orientations of the edges associated with the node. Previous work [12] has shown these relative orientations to be quite susceptible to noise. This lack of robustness

```
CompareNodes($N_i, N_j$):
if $N_i$.degree $\neq N_j$.degree then return 0
$d \leftarrow N_i$.degree
if $|N_i$.vRadius $- N_j$.vRadius $| > T_r^d$ then return 0
$P_2 \leftarrow$ PositionOffsetBetweenNodes($N_i, N_j$)
$R_2 \leftarrow$ MinimumErrorRotation($N_i, N_j, P_2$)
$(MSE_{2D}, P_2, R_2) \leftarrow$ TrICP2D($N_i.2D, N_j.2D, P_2, R_2$)
if $MSE_{2D} > T_e^d$ then return 0
$(MSE_{3D}, P_3, R_3) \leftarrow$TrICP3D($N_i.3D, N_j.3D, P_2, R_2$)
$E \leftarrow$ FormErrorVector($N_i, N_j, P_3, R_3$)
return LogisiticRegression($E, d$)
```

Fig. 3: Pseudocode for our node matching procedure

was also observed in our own experiments, and therefore this property was not used. Instead, if N_j has a different degree than N_i, or the difference in observed radii is more than a threshold T_r, N_j is eliminated as a candidate match. T_r is set relatively high, so as to ensure that no correct matches are ever thrown out, while eliminating as many incorrect matches as possible in a computationally inexpensive manner.

3.2 2D map matching

The next phase of node matching is to compare each node's 2D local map. Before the 2D maps can be compared, they must be properly aligned. Alignment of 2D point sets can be achieved using the Iterative Closest Point (ICP) algorithm [4]. ICP assumes that each point in the data set corresponds to the closest point in the model set. These correspondences are used to compute the transformation between the two sets that minimizes the Mean Squared Error (MSE). The correspondences are then recomputed, and the process iterates until convergence.

Due to the manner in which our 2D maps are constructed, the assumption that every point in the data set has a corresponding point in the model set is often violated to a degree that degrades performance. Therefore, the Trimmed Iterative Closest Point algorithm (TrICP) [5] is used instead. The key difference between ICP and TrICP is that TrICP assumes that only a proportion ξ of the points in the data set correspond to points in the model set. At each iteration, only ξK of the K points in the data set are used. The ξK points used are those with the smallest squared distance to their corresponding point in the model set. When unknown beforehand, ξ can be automatically set by minimizing the function

$$\psi(\xi) = MSE(\xi)\xi^{-(1+\lambda)} \tag{1}$$

where $MSE(\xi)$ is the MSE of the ξK points with the smallest squared distance to their corresponding point in the model set. The parameter λ balances the tradeoff between using more points and increasing MSE. In [5], $\lambda = 2$.

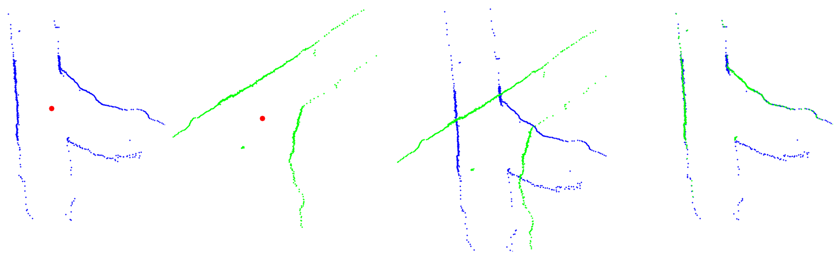
Both ICP and TrICP require a fairly accurate initial alignment in order to converge correctly. By framing node matching as a global localization problem, it is assumed that there does not exist a good long term estimate of metric position. In practice, this is usually the case, as Groundhog's online position estimation is not stable over long distances (accurate metric maps are produced offline). Since Groundhog's perceived metric position can not be used for an initial alignment, the locations of the nodes themselves are used. Since each node is embedded into the environment, if the two local maps are really of the same intersection, then the location of the Voronoi node in each map corresponds to the same point in space, represented in different coordinate frames. Setting the origin of each local map to be the corresponding Voronoi point thus produces an initial alignment in position. However, the orientation of each map relative to the node is still unknown. To fix the orientation, TrICP is run 8 times, with the initial orientation of one map relative to the other equally spaced at 45° intervals. TrICP is able to overcome such large errors in initial orientation because the error in initial position is small. The final alignment that results in the smallest MSE is selected as the correct 2D alignment (Figure 4(a)).

The MSE of the final alignment (after recomputing ξ) is compared against a threshold T_e. Just as with T_r, T_e is set to eliminate as many false matches as possible, while not eliminating any correct matches.

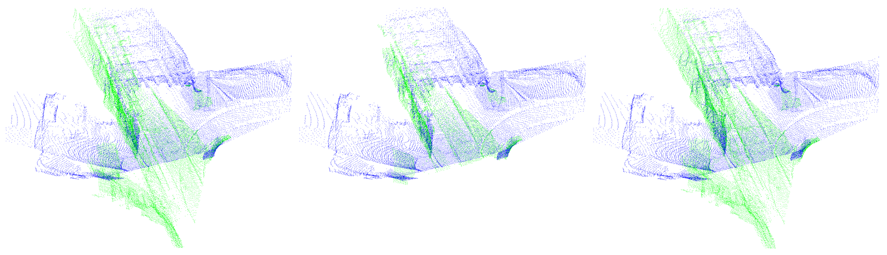
3.3 3D map matching

The last phase of node matching uses the 3D range data gathered after each node is detected. As with the 2D data, the 3D data must first be properly aligned. The 3D alignment is also achieved using TrICP. The initial 3D alignment used for TrICP is based on the final 2D alignment. Using the 2D alignment between the candidate nodes, and the known position of each node relative to the origin of the 3D scan, an initial 3D alignment is computed that is fairly accurate in x, y and yaw. Just as running TrICP with only an initial x and y allows the 2D alignment to converge to the correct orientation, running TrICP with an initial x, y and yaw allows the 3D alignment to converge to the correct z, roll, and pitch (Figure 4(b)).

In this phase, TrICP is run with one modification. Normally, ξ is computed according to (1) once during the first iteration. Thus, ξ depends heavily on the initial alignment. Since the initial alignment could have significant error in 3 of the 6 degrees of freedom, ξ must be occasionally recomputed. For this purpose, an additional loop is added around TrICP. After TrICP successfully converges, ξ is recomputed based on the final alignment. The final alignment is then fed back into TrICP as the new initial alignment. This process repeats until the value of ξ converges.



(a) 2D alignment: Each map is centered around its Voronoi node, and then one map is rotated relative to the other to find the minimum MSE alignment.



(b) 3D alignment: the 2D alignment is used as the initial 3D alignment (**left**). The ξN_d closest points (**center**) are then used to find the final alignment (**right**).

Fig. 4. 2D and 3D alignment of range data at an intersection

After each 3D alignment is complete, an error vector $E = \{e_1, ..., e_n\}$ is produced for each prospective match $N_i \leftrightarrow N_j$. The error vector consists of both 2D and 3D error measures. The 2D metrics are used despite the availability of 3D metrics, due to the 360° coverage of 2D data. In addition to MSE, additional error metrics based on the normal vectors of the 3D range data are used. This error metric is especially useful for classifying potential matches with a small ξ. The specific error vector used is described in Section 4.

3.4 Classification

After an error vector has been produced, the final task is to determine as accurately as possible whether or not N_i matches N_j. This can be viewed as a binary classification problem, with matching and non-matching classes. One approach to binary classification is *logistic regression* [1]. Under this approach, the probability of a match is computed from the error vector E as

Table 1. The results of each phase of node matching

Stage of Comparison	# of Incorrect Matches Remaining	# of Correct Matches Remaining
Original Dataset	1962	108
Degree Matching	1002	108
Radii Difference	588	108
2D MSE	173	108
Logistic Regression	23	108

$$P(N_i \leftrightarrow N_j | E = \{e_1, ..., e_n\}) = \frac{1}{1 + \exp(-z)} \tag{2}$$

$$z = w_0 + w_1 \Phi_1(e_1) + w_2 \Phi_2(e_2) + ... + w_n \Phi_n(e_n) \tag{3}$$

W is vector of weights $\{w_0, ..., w_n\}$, computed from training data using a maximum likelihood formulation. Each Φ_i is constructed as a classifier based on an individual element of the error vector. Our approach constructs each Φ_i as a Gaussian classifier of the i^{th} element of the error vector

$$\Phi_i(e_i) = \frac{\mathcal{N}(e_i, \mu_i^+, \sigma_i^+)}{\mathcal{N}(e_i, \mu_i^+, \sigma_i^+) + \mathcal{N}(e_i, \mu_i^-, \sigma_i^-)} \tag{4}$$

where μ_i^+ and σ_i^+ are the mean and standard deviation of the i^{th} element of E over matches, μ_i^- and σ_i^- are the mean and standard deviation over non-matches, and $\mathcal{N}(e, \mu, \sigma)$ is the Gaussian probability density function.

4 Experimental Results

4.1 Data Collection

To test our node matching approach, data was collected from the Bruceton research coal mine near Pittsburgh, PA. The dataset consists of the same topological, 2D, and 3D data that would be collected during autonomous exploration and intersection detection. 3D range data was downsampled to one point per 5cm voxel [14], to ensure equivalent resolution from multiple vantage points and to provide a significant decrease in computation. For each intersection, data was collected from each corridor leading into the intersection. Data was gathered from 46 different intersection/corridor combinations, resulting in 2070 possible matches. Of these, 108 are correct matches. The results of each phase of node matching are shown in Table 1.

4.2 Topological Matching

Of the 2070 possible matches, 960 (46%) can be immediately eliminated, because the degree of one node does not match the degree of the other node.

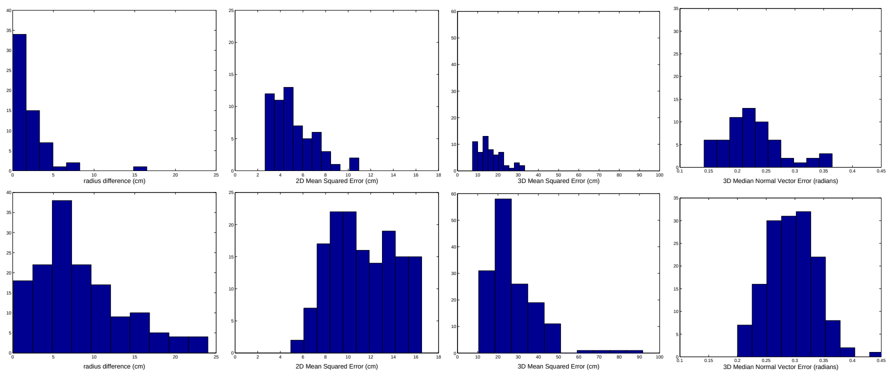


Fig. 5. The distributions of each value of the final error vector over nodes of degree 3. The distributions over matches are shown on top, and non-matches on the bottom

Next, matches are eliminated based on the Voronoi radius. To make it as unlikely as possible that any correct matches are eliminated in this phase, T_r is set at 1.5 times the maximum difference in Voronoi radii observed in a correct match. To take into account the differences in various types of intersections, a different threshold T_r^d is chosen based on the degree d of the node. Solely based on radii thresholding, 1219 (59%) of the possible matches can be immediately eliminated. Combining radii thresholding with the enforcement of degree equality eliminates 1374 (66%) of the possible matches. Thus, approximately 2/3 of prospective matches are eliminated almost immediately.

4.3 2D Matching

After thresholding on topological properties, the next phase is to align the 2D range data, and compare the MSE against a threshold T_e. For 2D TrICP, a λ value of 2 was used. As with radius thresholding, a different T_e^d is used for each node degree d, and each T_e^d is set at 1.5 the maximum observed MSE in a correct match. Of the 2070 possible matches, 1573 (76%) can be eliminated solely based on 2D MSE thresholding. By also only considering matches that passed the topological matching phase, 1789 (86%) matches are eliminated. Thus, the relatively inexpensive topological and 2D matching phases are able to quickly eliminate all but about 14% of the possible matches.

4.4 3D Matching

Next, the 3D range data associated with the remaining prospective matches is aligned. For 3D TrICP, a λ value of 1.5 was used. After 3D alignment is completed, the final error vector E is formed. An error vector consisting of the following fields has so far produced the best results:

- The difference in Voronoi Radius

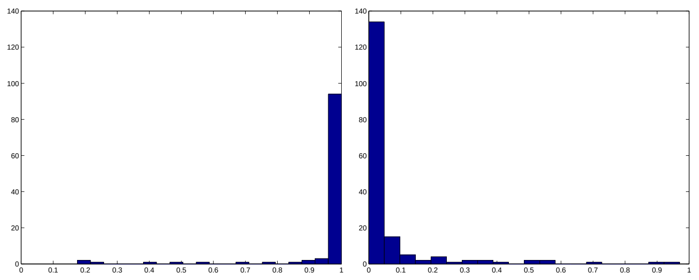


Fig. 6. Output of the final classifier on matches (**left**) and non-matches (**right**)

- The 2D MSE of the $\xi_2 K_2$ points with the smallest error
- The 3D MSE of the $\xi_3 K_3$ points with the smallest error
- The median angle between normal vectors of the $\xi_3 K_3$ points with the smallest error

Example distributions of these 4 elements over both correct and incorrect matches are shown in Figure 5.

4.5 Final Classification

After the error vector has been computed, it is fed into the classifier to compute a final match probability. For this experiment, the classifier was trained over the set of all matches that were not eliminated by thresholding tests. To help reduce the chance of a false negative, correct matches were weighted twice as heavily as incorrect matches during training. As with all other phases, a separate classifier is used for each possible node degree.

The distributions of the final probabilities over both correct and incorrect matches are shown in Figure 6. Thresholding the final probability at 0.1 results in all 108 correct matches still being considered, with only 23 remaining false positives. This accuracy is more than sufficient for use within a multi-hypotheses topological framework.

5 Conclusion

In this paper, we have presented a method for approximating the probability that two corridor intersections in a subterranean void match. Such an approach can be used by an autonomous mine mapping robot to determine when it is revisiting an intersection. This approach, in conjunction with other topological techniques, will allow for the full autonomous exploration of mine environments, including autonomous loop closure.

Future work will focus on making our node matching technique robust to the point that multi-hypotheses tracking will almost never be necessary. One method for achieving this would be to use multiple 3D scans from each visit to a node to provide the same 360° coverage that the 2D scans achieve. Also, more intelligent means of computing T_r and T_e will be explored. Further, the possibility of more descriptive 3D error metrics will be investigated.

References

1. A. Agresti. *Categorical Data Analysis.* Wiley-Interscience, 2002.
2. C. Baker, A. Morris, D. Ferguson, S. Thayer, C. Whittaker, Z. Omohundro, C. Reverte, W. Whittaker, D. Hähnel, and S. Thrun. A Campaign in Autonomous Mine Mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, New Orleans, LA, 2004.
3. J. Belwood and R. Waugh. Bats and mines: Abandoned does not always mean empty. *Bats*, 9(3), 1991.
4. P.J. Besl and N.D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, 1992.
5. D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek. The trimmed iterative closest point algorithm. In *Proc. Int. Conf. on Pattern Recognition*, 2002.
6. H. Choset and J. Burdick. Sensor based planning, part II: Incremental construction of the generalized voronoi graph. In *Proc. of IEEE Conference on Robotics and Automation*, pages 1643 – 1648, Nagoya, Japan, May 1995. IEEE Press.
7. H. Choset and K. Nagatani. Topological simultaneous localization and mapping (slam): towards exact localization without explicit localization. *IEEE Transactions on Robotics and Automation*, 17(2):125–137, Apr. 2001.
8. G. Dudek, P. Freedman, and S. Hadjres. Using local information in a non-local way for mapping graph-like worlds. In *Proc. of the 13th International Joint Conference on Artificial Intelligence*, 1993.
9. G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. Robotic exploration as graph construction. *Trans. on Robotics and Automation*, 7(6):859–865, Dec. 1991.
10. E. Duff, J. Roberts, and P. Corke. Automation of an underground mining vehicle using reactive navigation and opportunistic localization. In *IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, 2003.
11. B. Kuipers, J. Modayil, P. Beeson, M. MacMahon, and F. Savelli. Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In *IEEE International Conference on Robotics and Automation*, 2004.
12. B. Lisien, D. Morales, D. Silver, G. Kantor, I. Rekleitis, and H. Choset. Hierarchical simultaneous localization and mapping. In *IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, volume 1, pages 448–453, Oct. 2003.
13. A. Morris, D. Silver, D. Ferguson, and S. Thayer. Towards topological exploration of abandoned mines. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2005.
14. J. Rossignac and P. Borrel. *Multi-Resolution 3D Approximations for Rendering Complex Scenes.*, pages 455–465. Springer-Verlag, 1993.
15. D. Silver, D. Ferguson, A. Morris, and S. Thayer. Feature extraction for topological mine maps. In *IEEE/RSJ Conf. on Intelligent Robots and Systems*, 2004.
16. N. Tomatis, I. Nourbakhsh, and R. Siegwart. Hybrid simultaneous localization and map building: Closing the loop with multi-hypotheses tracking. In *IEEE International Conference on Robotics and Automation*, 2002.