Using sketch-maps for robot navigation: interpretation and matching

Malcolm Mielle, Martin Magnusson, Achim J. Lilienthal

***Abstract*— We present a study on sketch-map interpretation and sketch to robot map matching, where maps have nonuniform scale, different shapes or can be incomplete. For humans, sketch-maps are an intuitive way to communicate navigation information, which makes it interesting to use sketch-maps for human robot interaction; e.g., in emergency scenarios.**

To interpret the sketch-map, we propose to use a Voronoi diagram that is obtained from the distance image on which a thinning parameter is used to remove spurious branches. The diagram is extracted as a graph and an efficient error-tolerant graph matching algorithm is used to find correspondences, while keeping time and memory complexity low.

A comparison against common algorithms for graph extraction shows that our method leads to twice as many good matches. For simple maps, our method gives 95% good matches even for heavily distorted sketches, and for a more complex real-world map, up to 58%. This paper is a first step toward using unconstrained sketch-maps in robot navigation.

I. Introduction

In search and rescue scenarios, a robot needs to perform navigation in unknown environments. An example is the SmokeBot project, which focuses on civil robots supporting fire brigades and aims to enhance the performance of robots under conditions of smoke, dust or fog. In communication with fire departments and other relevant end users, we identified that it is often possible to obtain prior information about the place to explore. Fire fighters frequently have maps of buildings which could be used as prior data. Good prior maps of the place to explore can be obtained through emergency maps or CAD drawings. However, the fire fighters themselves usually have not been at the site. Therefore, those maps can be hard to understand, especially in situations of stress, leading to a complicated teleoperation of the robot. This paper addresses this problem by introducing and evaluating approaches that allow using sketch-maps as an interface on top of the prior map.

Sketch-maps are very effective at conveying spatial configurations and using them to direct a robot allows for intuitive human-robot interactions [1]. The robot could use the building map as ground truth while sketches could be obtained from people who are familiar with the place, enabling non-expert people to help the robot in its navigation and exploration mission.

Assuming the robot can localize itself in the prior map, which is a topic of our ongoing work, robot navigation commands could be sent by pointing at places in the sketch. An example of such an interface can be seen in Fig 1.

Center of Applied Autonomous Sensor Systems (AASS), Örebro University, Sweden. firstname.lastname@oru.se

This work was funded in part by the EU H2020 project SmokeBot (ICT-23-2014 645101) and by the Swedish Knowledge Foundation under contract number 20140220 (AIR).

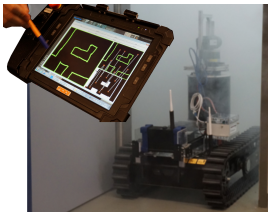


Fig. 1. Interface using a sketch-map and robot navigating in smoke.

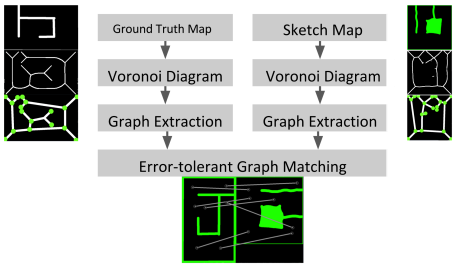


Fig. 2. Flowchart of the method.

II. Hypothesis and Contributions

Sketch-maps are inherently inaccurate. They may be incomplete and have large uncertainties in both shape and scale: walls are not straight and the scale may not be uniform. For an overview of sketch-maps and their utilities and drawbacks, see Wang and Li [2]. To deal with these inaccuracies, and to find the correspondences between the sketch and the prior map, are the main focuses of this paper.

To use a sketch-map for navigation, one has to interpret it and match it to a metric environment map: the real world map or ground truth map. For our problem, matching means that we need to be able to find elements of the sketch for which we can find a correspondence in the real world map. We define a ground truth map as the equivalent to a noise-free occupancy grid map and the sketch-map as a free hand drawing of the environment, as remembered by the user.

We separate the problem into two distinct parts :

- How to interpret the sketch and ground truth map.
- How to match the interpreted maps.

The contributions of this paper are :

- An interpretation method for sketch-maps.
- A graph matching method to find the correspondences between sketch-maps and ground truth maps.

Our hypothesis is that a Voronoi diagram, extracted as in Section IV, is a useful representation of the topology of a sketch-map and that an error-tolerant graph matching can

match the graphs built from the Voronoi diagrams of the sketch and the ground truth map, while accounting for the sketch-map inaccuracies.

A flowchart of the method can be seen in Fig 2.

III. Previous Work

A. Sketch-map and map matching

Existing map matching techniques require structurally consistent maps. In Pyo, Shin, et al. [3], multiple hypothesis techniques track a robot in a cluttered environment and match the map created from GPS measurements to a model map. Probabilistic frameworks, as developed by Saeedi, Paull, et al. [4], are useful when trying to find the relative transformation between maps and fuse them. Some methods account for partial deformations and cluttering by imposing a threshold as in the work of Huang and Beevers [5]. However, a general shortcoming of existing approaches is that they do not afford matching maps with large structural differences.

In Skubic et al. [1], [6], [7] sketch-map interfaces for robot control have been implemented on a PDA, where objects are closed polygons drawn by the user. The descriptors, used to match the drawn objects to the ones seen by the robot, are histograms of forces. Since we focus on indoor navigation for emergency situations, we cannot use objects as landmarks. Indeed, objects may not stay at the same place and are not consistent landmarks. Furthermore, a map of isolated objects matched using histograms of forces is not applicable for dense, metric, indoor maps.

Another idea is to consider sketch-maps as a topological environment representation. Several authors [8]–[10] use such a topological map extracted from a sketch for navigation. Shah and Campbell [10] use a topological sketch-map representing distinct landmarks to create waypoints using the Voronoi–Delaunay graph. Their landmarks are independent, uniquely identifiable buildings. However, such landmarks are not relevant in an indoor emergency scenario. Setalaphruk, Ueno, et al. [8] use the crossings and dead-ends of the Voronoi diagram of corridors as landmarks. Although correct distances for the corridors are not needed, the walls have to be straight and the existence and connectivity of corridors have to be correct. These conditions do not hold for hand-drawn sketch-maps, and therefore sketch-map matching requires a more robust matching technique.

B. Graph matching and map matching

Graph matching is an important problem in graph theory. The most relevant sub-problem for this work is the *error-tolerant maximal common subgraph* (ETMCS) which is NP-hard. There is a large body of literature about graph matching, but here we chose to only relate to papers with a direct connection to our present work. For a survey on the graph matching techniques and learning in pattern recognition in the last 10 years, see Foggia, Percannella, et al. [11].

Neuhaus and Bunke [12] presented an algorithm to match planar graphs. The algorithm grows a seed, neighborhood by neighborhood, into a full graph-match. We have implemented their algorithm and adapted it for sketch-map matching by adding an explicit initialization step and incorporating salient attributes from sketch-maps.

Huang and Beevers [5] merged topological maps by finding all common connected subgraphs of two maps and clustering the compatible subgraphs. Some attributes should match perfectly (number of edges out of a vertex) and others, must only match approximately (edge length or the angle between edges). Saeedi, Paull, et al. [4] developed a probabilistic SLAM algorithm for multiple robots. It uses the probabilistic generalized Voronoi diagram to fuse maps and account for uncertainties. Map fusion is done using edge distances and shapes and is thus not directly applicable for sketch-maps. Both those works assume there are no missing vertices or edges in the graphs and are dependent on the geometric consistency of the maps which is not an assumption compatible with sketch-maps.

Wallgrün [13] proposes to match maps using graph matching and an annotated generalized Voronoi graph, solving a data association problem between observations and the robot's internal map. The stability of vertices and edges, and their relevance for navigation, determines the cost of adding or deleting branches. Spatial constraints are heavily used for better efficiency. The main difference as opposed to our work lies in the heavy use of such constraints that are not applicable when matching maps with nonuniform scale.

There is also a class of graph matching algorithms that require large graphs, to gather statistical information for matching, which cannot be applied to small graphs.

IV. Sketch-map interpretation

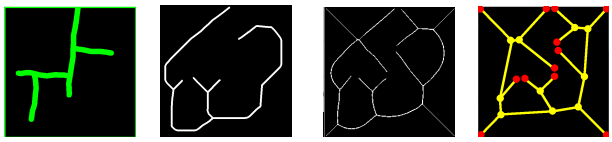
Robot maps for navigation can be either metric, topological or hybrids of both types.

We consider sketch-maps of buildings representing rooms and corridors. Although they are metric maps, they do not respect real world scales and proportions. They can miss features, sometime even "on purpose". E.g., a corridor could be missing if the user judged it not important for navigation.

We chose to use topological maps for map matching. While scale and proportion are inherent to metric maps, a topological map allows us to compare maps of the same environment, even when they have large structural differences, by choosing the graph attributes as described in Section V.

To reduce an occupancy grid to a topological map, the Voronoi Diagram is often used [4], [5], [8], [14]–[16]. Schwertfeger and Birk [17] used the Voronoi diagram to evaluate map quality by comparing graphs extracted from the Voronoi diagrams. We hypothesize that the Voronoi diagram of the environment can give us consistent information between the sketch-map and the real world map. Indeed, keypoints such as junctions and dead-ends should correspond to equivalent places in both the sketch and the ground truth. Using the Voronoi lines as edges between keypoints enable us to create a consistent topological map.

To construct a Voronoi diagram, we calculate the Euclidean distance image of the map, from which we extract local minima with a Laplacian filter. A "thinning parameter" is used to remove spurious branches. The thinning parameter



Sketch map Too simple EVG Voronoi diagram Voronoi graph

Fig. 3. Graph extraction from a grid map using the Voronoi diagram. In the final graph dead-ends are in red, and junctions are in yellow.

is a fixed percentage of the minimum of the Laplacian image. We experimentally found that keeping all pixels whose value is less than 30% of the image's strongest minimum's value, lead to clear Voronoi graphs.

Methods like EVG-thin [15], [18] are commonly used in robotics to extract the Voronoi diagram of a map. EVG-thin [15] uses Zhang and Suen [18] method to iteratively remove the contour of the free space in the map. While relatively stable in the presence of walls with artifacts from sensor noise, for sketch-maps the uncertainty is rather on the drawing interpretation. Graphs extracted by EVG-thin ignore important details by not considering them as part of the skeleton and tuning ECG-thin's parameters for every sketch is costly and inefficient (Fig 3). In our method, we remove the weakest part of the information thanks to a threshold depending on the value of the *strongest* part of the Voronoi diagram, which makes it more efficient at representing the sketch the way the user intended us to understand it, as can be seen in the results presented in Fig 5.

Using a line follower, we obtain our graph from the Voronoi diagram. Graph vertices are placed at junctions and dead-ends of the diagram, and graph edges at its lines. Fig 3 illustrates how a graph can be extracted from a grid map via a Voronoi diagram.

V. Topological Graph Matching

Having extracted a graph representation for both maps we aim at matching them; i. e., finding the best set of corresponding vertex pairs between the two undirected graphs. Working with sketch-maps, we need to cleverly select which graph attributes to use for matching.

Firstly, we consider a *vertex type* that can be of two sorts. Dead-end: every vertex that has only one edge. Junction: every vertex that has two or more edges.

Secondly, we use the ordered list of edges. We assume that, for any given sketch-map, the absolute position of landmarks and the absolute distance between them are unreliable. But their ordering, i. e. their relative position, is useful information. This assumption means that, even though the drawing has unknown proportion and nonuniform scale, the ordering and succession of corridors and rooms, will most of the time be valid. This is not a strong assumption considering the nature of sketch-maps [2]. Having the order of the edges simplifies the problem from matching two sets of vertices, to matching two ordered sets of vertices, making the matching time linear in the degree of the vertex, while it's polynomial for a general graph.

We do not extract any other information from the edges since other features are not expected to be reliably detectable in both the sketch and the ground truth map.

```
   Data: seeds
   Result: hypotheses_all
1  to_explore = {};
2  while seeds is not empty do
3      add first element of seeds to to_explore;
4      remove first element of seeds;
5      hypothesis = {};
6      while to_explore is not empty do
7          pair ⇐ first element of to_explore;
8          remove first element of to_explore;
9          add pair to hypothesis;
10         get the set of vertex matches from the neighbor of
           the two vertices in pair;
11         for every match in the set of matches do
12             if both vertices in match are not already in
               hypothesis then
13                 add match to to_explore;
14             if match is in seeds then
15                 remove match from seeds;
16     edit_distance ⇐ get edit distance from hypothesis;
17     add hypothesis and edit_distance to hypothesis_all;
18 return hypotheses_all;
```

Algorithm 1: Graph-Edit-Distance matching algorithm.

Having extracted the graphs, as outlined in Section IV, and their attributes as shown above, the following section describes the method we use to match them.

To determine the ETMCS, one uses the edit-distance between two graphs: the minimum number of basic modifications needed to transform one graph into the other. In our work, we use the normalized Levenshtein edit-distance (LED) [19] which considers insertions, deletions and substitutions. Operations are prioritized by cost: no-operations are prioritized over all other operations and substitutions are prioritized over deletions and additions.

The method (see Algorithm 1) is inspired by Neuhaus and Bunke's algorithm for matching planar graphs [12] and our main contributions are the initialization step and the usage of the attributes defined above.

We define a hypothesis as *a set of matched vertices between two graphs defining a possible map-to-map match*. The key elements of Algorithm 1, explained in the remainder of this section, are: 1) find all initialization seeds, 2) expand the matching algorithm for every seed, 3) find the best neighborhood matching for all matched vertices, 4) update the edit distance of every possible matching found.

1) Initialization. We start by finding all pair-wise matches between vertices of the same type in the two maps. We call those matches seeds and store them in the set named *pairwise*. In the worst case scenario (all vertices of both graphs are of just one type), we will have $n * m$ pair-wise matches, with n and m the number of vertices in each graph.

The algorithm is launched for every seed, creating a new hypothesis for each of them. Each hypothesis is associated with its total edit distance and is a valid solution whose edit-distance is an upper bound of a subset of all graph matching solutions. In the worst case, the algorithm would run $O(n * m)$ times and return $n * m$ hypotheses.

The hypothesis with the lowest edit distance is chosen as the best possible matching between the maps.

2) Matching expansion. Every seed needs to be grown into a final hypothesis. With *hypothesis* the current hypothesis being grown, *to_explore* the set of vertices left to explore as possibly being part of *hypothesis* and *hypotheses_all* the set of all explored hypotheses and their edit distances, every seed in *pairwise* is expanded, by iteratively matching the neighborhoods of the vertices that are added to *to_explore*. All new vertex matches found in this step are added to *to_explore*. This happens at line 10 in Algorithm 1 and is explained in step 3 below.

At line 14, to speed up the algorithm, every time a new match is the same as one of the seed, the seed is removed from *pairwise*. This is to avoid expanding similar graph matches from different seeds.

We iterate over all pairs in *to_explore* until the set is empty. Once no more matches remain to explore, at line 16, we calculate the final edit distance of *hypothesis* (see step 4 below), add *hypothesis* to *hypotheses_all* and start growing the next seed. We repeat the process until there are no more seeds left to explore.

3) Neighborhood matching. The neighborhood matching (line 10) returns a list of vertex-to-vertex matches, corresponding to the least expensive matching between the two neighborhoods. The cost is defined by the LED and we define a neighborhood the same way as Neuhaus and Bunke [12]: a subgraph consisting of a vertex *u* plus all vertices connected to *u* and all edges between those vertices.

This step uses string matching while taking into account the information from the vertex matches already found. The out-vertex' type determines the character representing the edge in the string and edges are ordered counter-clockwise.

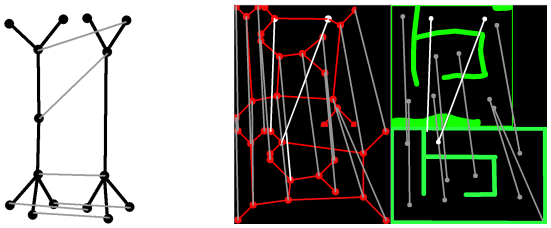
If no vertices in the neighborhoods are already matched, which is only the case for the seed, we use a cyclical string matching algorithm. The worst case complexity, when considering graph of bounded valence (maximum number of edges connected to a vertex) v is $O(v^2)$.

Otherwise, we start by determining all the already-matched vertices in each neighborhood. For n vertices already matched per neighborhood, we cut the neighborhoods, between each of the n vertices, in n associated sub-strings. By matching those sub-strings together, we obtain the best matching between the neighborhoods while conserving the information given by previous vertex matches. In the worst case scenario, matching two strings is $O(k * l)$ where k and l are the sizes of the strings.

4) Edit distance update. At line 16, to update the final edit distance value between both graphs, all unmatched vertices are removed. Each vertex removal has a cost of 1. Once every unmatched vertex has been removed, we also remove every edge that does not link two vertices anymore. Each edge removal also costs 1.

VI. Experiments and results

We conducted two types of experiment to validate our algorithm: random graph matching to validate the graph matching method and to study its robustness with regard to errors in the graphs, and sketch-map matching to validate the sketch-map interpretation.



(a) Matching fails when adding a vertex "in the middle".

(b) Example of errors induced by differences in the graph.

Fig. 4. Example of a matching failure. In Fig 4b the incorrectly matched links are white. All gray links are linking to the same places. Most places are matched correctly but the added vertex on top right leads to a partly incorrect place matching.

A. Graph matching validation

To quantify robustness to errors in the graphs, we constructed synthetic graphs with increasing amounts of errors. We created graphs of 10 to 50 vertices, all linked together in a line. We added a fixed number of edges between random vertices of the graph to create loops. Finally, for each such graph we created a copy with a fixed amount of corruption added, and tried to match the corrupted copy to the original. This is to simulate a corrupted graph from the sketch-map and a perfect graph from the ground-truth map. A vertex match was considered correctly matched only if the corrupt graph's vertex corresponded to its exact counterpart in the original. The goal is to determine at which amount of corruption the original graph is not recognizable anymore.

One shortcoming of the graph matching algorithm lies in its neighborhood-by-neighborhood matching: as in Fig 4a, one wrong vertex in the middle of the graph can break the iterative matching process. The graphs must not be broken in multiple matching parts and a big enough succession of correct nodes, that we call the "skeleton" of the graph, must be correctly extracted. For an example of a matching failure due to a broken skeleton in a real sketch-map, see Fig 4b. Additional graph preprocessing could potentially detect and remove those problematic vertices.

We have run the test procedure outlined above for different amount of corruption, using 100 random graphs per graph size and per corruption amount. We have not included tests with more than 50 vertices since a graph of that size corresponds to a map with more rooms/places than what can be easily remembered and drawn.

To compare our matching algorithm to state of the art algorithms, we also ran the procedure against the VF2 (sub)graph matching algorithms [20]. VF2 is one the fastest graph matching algorithm thanks to its linear memory complexity.

The results are summarized in Fig 5. As expected, the algorithm is able to perfectly match a graph to itself: no matching errors when the graphs are not corrupted. The percentage of correct matches is always above 50%, even when the corrupted graphs has equal amounts of corrupted and correct information. E. g., the algorithm still extracts 56% good matches in graphs with 20 vertices and 100%

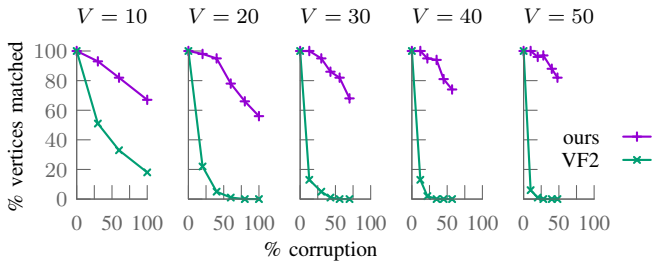


Fig. 5. Comparing graph matching algorithms w.r.t. noise. V is the number of vertices in the original graph, corruption is the percentage of added vertices and edges. Our method significantly outperforms VF2.

added corruption, and 68% good matches with graphs of 30 vertices and 70% added corruption. With a corruption amount of 50%, for every graph size the algorithm correctly matches more than 80%. To match two graphs of 50 vertices, the algorithm takes 2.5 s. It takes less than a second for graphs under 40 vertices and around 1.1 s for graphs of 40 vertices when ran on a core i7 with 16Gb of RAM. When comparing the method to VF2, although not as fast, our algorithm leads to vastly better matches than VF2. VF2 cannot handle heavy corruption, with results falling quickly under 20%. The only case when VF2's speed is relevant is for graphs with no corruption, where, for graphs of 50 vertices it took around 0.17 ms.

Having confirmed that the graph matching algorithm is stable under noise, we can test the sketch-map interpretation.

B. Virtual environment

To build a sketch-map dataset, users were invited to explore two places in a virtual environment in a web browser. The first was custom-made and the second corresponded to the ground truth of the KTH dataset for SLAM[1] (see Fig 8 and Fig 7). The custom environment was designed to be simple so that users could easily memorize it, the KTH map was selected to validate the algorithm on real-world data. Anyone can test the virtual environment on the online website[2]. The users were all anonymous.

We ran the method of Schwertfeger and Birk [17] on these sketch-maps. Similarly to our approach, their method extracts topological graphs from maps using Voronoi diagrams. To achieve a good match between a sketch and a ground-truth map, a certain number of parameters need to be selected. However, even when the sketches represent the same environment, the same parameter set cannot be applied for all. For each sketch's parameter set, their method failed to match other sketch-maps on the ground truth maps. We hypothesize that their method for extracting graphs is not designed for inaccurate sketches, but rather for robot maps with relatively small conceptual errors.

The matching evaluation for every sketch-map was made by comparing the algorithm's result to 4 ground truth sketch-to-model matches (GTM), made by 4 independent users, for each sketch-map. The GTM is the set of "best links", according to the user, between the features of the sketch-map

[1]http://www.nada.kth.se/~johnf/kthdata/dataset.html
[2]http://smokebot.eu/sketchmap-web/

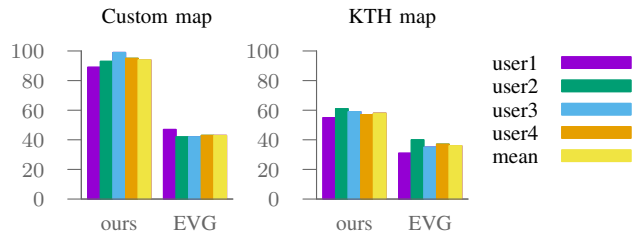


Fig. 6. Ground truth match (GTM) results for the two environments, comparing results on a small custom map and on a more complex (KTH) environment, of our map interpretation to that of EVG-thin.

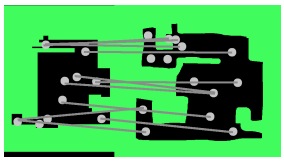


Fig. 7. GTM of a sketch-map for the KTH dataset. Several vertices in the sketch (right) are linked to one point in the model map (left).

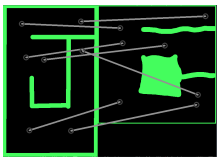


Fig. 8. Custom data set: Correct matching of a very distorted sketch-map (right) to the ground truth (left).

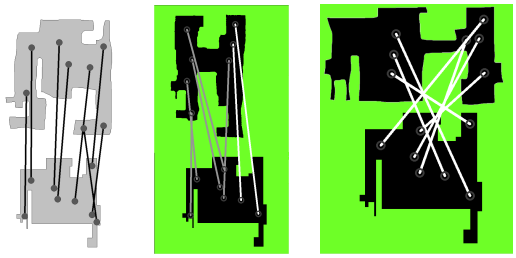
and the model map. The features were the ones extracted by the sketch-interpretation method (Section IV). To build the GTMs, users were given an image of the sketch-map and the equivalent ground truth map with all junction nodes of their respective graphs printed, but without the edges or the dead-end nodes. They were asked to draw links between the junctions of the sketch and the ground truth map, fitting the way they interpreted the drawing. They were free to draw more than one link per junction, in case they thought that one given junction could be matched to multiple junctions. Fig 7 shows an example GTM.

Some sketch-maps were not understandable by the users and were removed from the test. Out of the 15 sketches of KTH, 4 sketches were removed. None of the simple map sketches were removed. It must be noted that the KTH sketch-maps were unconstrained hand drawings of a real world place, explored in a virtual environment, and were a lot harder to recognize, even for a human user.

For every graph match returned by the algorithm, a vertex-to-vertex match from the algorithm was considered valid if it corresponded to a match in the user GTM. The percentage of good matches was calculated as the ratio between the number of correct vertex-to-vertex matches and the maximum number of matches that could be extracted from a GTM. We compared our Voronoi graph extraction to the well established EVG-thin approach.

In Fig 6, we can see that the proposed algorithm performs well for the custom map with around 94% of correct matches. This can be explained by the simplicity of the environment, which was easy to memorize and to draw, and from which we got relatively accurate sketches. Additionally, users had no problem interpreting the sketch map, even when "far" from the ground truth. The graphs extracted from the sketches by the algorithm were clear and easy to match, for the user making the GTMs, and for the algorithm itself (Fig 8).

For the KTH map, the algorithm produces 58% good matches over the 4 users' GTMs. KTH sketches were harder to interpret for the users, the graphs extracted were messier



(a) Correct match. (b) Partially correct match. (c) Wrong match.

Fig. 9. Three different results of the matching algorithm. The sketch-maps, on top, represent the inside of the KTH map, on the bottom. The white links are the ones between vertices that should not be matched.

since the drawings were unconstrained and the shape of the room was not easily identifiable (see Fig 9 for KTH sketches with varying "qualities").

Finding correspondences between a hand-drawn building sketch-map and a ground truth is a very hard task, even for humans, and there is no previous body of research on this particular problem. We believe that a matching rate of 58% for real-world sketches with large structural errors is a highly encouraging first result.

VII. Conclusion and Future Work

We have presented an algorithm to match hand drawn sketch-maps to ground truth maps. We use a Voronoi diagram to extract features from both maps. Those features are used to construct a graph, where each vertex contains carefully chosen attributes to be able to match the sketch-map's graph and the ground-truth's graph.

The matching algorithm is an iterative process that grows possible matching hypotheses from a given set of seeds. The seeds are determined using the vertex attributes. After every seed has been expanded, the best matching, as measured by the lowest edit distance, is kept.

Our method quickly extracts features and do the matching. It can cope with a certain amount of mistakes in the drawing: errors in position, scale and mistakes in the presence or absence of rooms and corridors. We tested the graph matching method against a state-of-the-art algorithm and the sketch-map interpretation method against a similar interpretation method for robot maps. Results were promising with twice as many good matches for our proposed method.

It is notoriously difficult to extract noise-free Voronoi diagrams, especially from drawings such as sketch-maps. Nevertheless, using the procedure proposed in this paper leads to substantially more robust interpretations of sketch-maps than state-of-the-art methods [15], [17], [18]. We believe that a matching rate of 58% for real-world sketches with large structural errors is a highly encouraging result. From our experiments, it can also be concluded that it is difficult to interpret free-hand drawings such as sketch-maps, even for humans.

Future work includes studying additional preprocessing steps of the Voronoi diagrams, as well as the use of probabilistic matching techniques that have previously been employed in the pattern analysis community [21]. We are also working on alternative methods for extracting topological information from noisy maps, as well as ways of incorporating the sketch into the robot's SLAM process.

References

[1] M. Skubic, D. Anderson, S. Blisard, D. Perzanowski, and A. Schultz, "Using a hand-drawn sketch to control a team of robots," en, *Autonomous Robots*, vol. 22, no. 4, pp. 399–410, Mar. 2007.

[2] J. Wang and R. Li, "An empirical study on pertinent aspects of sketch maps for navigation," in *ICCI* CC*, IEEE, 2012, pp. 130–139.

[3] J.-S. Pyo, D.-H. Shin, and T.-K. Sung, "Development of a map matching method using the multiple hypothesis technique," in *Proc. IEEE Intelligent Transportation Systems*, 2001, pp. 23–27.

[4] S. Saeedi, L. Paull, M. Trentini, M. Seto, and H. Li, "Efficient map merging using a probabilistic generalized Voronoi diagram," in *IROS*, Oct. 2012, pp. 4419–4424.

[5] W. H. Huang and K. R. Beevers, "Topological Map Merging," en, *The International Journal of Robotics Research*, vol. 24, no. 8, pp. 601–613, Aug. 2005.

[6] M. Skubic, C. Bailey, and G. Chronis, "A sketch interface for mobile robots," in *Proc. Systems, Man and Cybernetics*, vol. 1, IEEE, 2003, pp. 919–924.

[7] G. Parekh, M. Skubic, O. Sjahputera, and J. M. Keller, "Scene matching between a map and a hand drawn sketch using spatial relations," in *ICRA*, IEEE, 2007, pp. 4007–4012.

[8] V. Setalaphruk, A. Ueno, I. Kume, Y. Kono, and M. Kidode, "Robot navigation in corridor environments using a sketch floor map," in *Proc. Computational Intelligence in Robotics and Automation*, vol. 2, IEEE, 2003, pp. 552–557.

[9] G. Chronis and M. Skubic, "Robot navigation using qualitative landmark states from sketched route maps," in *ICRA*, vol. 2, IEEE, 2004, pp. 1530–1535.

[10] D. C. Shah and M. E. Campbell, "A qualitative path planner for robot navigation using human-provided maps," *The International Journal of Robotics Research*, vol. 32, no. 13, pp. 1517–1535, Nov. 2013.

[11] P. Foggia, G. Percannella, and M. Vento, "Graph matching and learning in pattern recognition in the last 10 years," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 28, no. 01, Feb. 2014.

[12] M. Neuhaus and H. Bunke, "An Error-Tolerant Approximate Matching Algorithm for Attributed Planar Graphs and Its Application to Fingerprint Classification," en, in *Structural, Syntactic, and Statistical Pattern Recognition*, ser. Lecture Notes in Computer Science 3138, Springer Berlin Heidelberg, 2004, pp. 180–189.

[13] J. O. Wallgrün, "Voronoi Graph Matching for Robot Localization and Mapping," en, in *Transactions on Computational Science IX*, ser. Lecture Notes in Computer Science 6290, Springer Berlin Heidelberg, 2010, pp. 76–108.

[14] ——, "Hierarchical voronoi-based route graph representations for planning, spatial reasoning, and communication," in *CogRob-2004*, Citeseer, 2004, pp. 64–69.

[15] P. Beeson, N. K. Jong, and B. Kuipers, "Towards autonomous topological place detection using the extended voronoi graph," in *ICRA*, IEEE, 2005, pp. 4373–4379.

[16] S. Saeedi, L. Paull, M. Trentini, M. Seto, and H. Li, "Group Mapping: A Topological Approach to Map Merging for Multiple Robots," *IEEE Robotics Automation Magazine*, vol. 21, no. 2, pp. 60–72, Jun. 2014.

[17] S. Schwertfeger and A. Birk, "Evaluation of map quality by matching and scoring high-level, topological map structures," in *ICRA*, May 2013, pp. 2221–2226.

[18] T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Commun. ACM*, vol. 27, no. 3, pp. 236–239, Mar. 1984.

[19] L. Yujian and L. Bo, "A normalized levenshtein distance metric," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1091–1095, Jun. 2007.

[20] L. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub)graph isomorphism algorithm for matching large graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 10, pp. 1367–1372, Oct. 2004.

[21] R. Myers, R. Wison, and E. Hancock, "Bayesian graph edit distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 6, pp. 628–635, Jun. 2000.