Retrieving Topological Information for Mobile Robots Provided with Grid Maps

David Portugal and Rui P. Rocha

Institute of Systems and Robotics, Department of Electrical and Computer Engineering
University of Coimbra, 3030-290 Coimbra, Portugal
{davidbsp,rprocha}@isr.uc.pt

Abstract. In the context of mobile robotics, it is crucial for the robot to have a consistent representation of the surrounding area. However, common grid maps used in robotics do not provide any evidence as to connectivity, making it harder to find appropriate paths to particular points on the site. Therefore, abstracting the environment where mobile robots carry out some mission can be of a great benefit.

Topological maps have been increasingly used in robotics, because they are fairly simple and an extremely intuitive representation for tasks that involve path planning. In this article, a method for retrieving a topological map from an *a priori* generic grid map of the environment is presented. Beyond extracting a 2D diagram which portrays the topology of the infra-structure, the focus is placed on obtaining graph-like data related to the connectivity of important points in the area, that can be passed on to robots or to a centralized planner, in order to assist the navigation task. The proposed method is further elaborated in detail and its results prove the simplicity, accuracy and efficiency of the approach.

Keywords: Robot navigation, Graphs, Topological maps, Voronoi diagrams.

1 Introduction

In robotics, it is extremely important for autonomous mobile robots to learn and preserve models of the environment. Without an internal description of the environment and information of their position and orientation with respect to this map, most mobile robotic tasks, like driving while avoiding collisions and navigating, would become impractical.

In navigation tasks, it is normally assumed that the environment is known *a priori*. On the other hand, exploration is generally related to completely or partially unknown environments. For both cases, maintaining or building internal representations of the infra-structure is one of the key issues to the successful completion of the task.

The two major distinct paradigms for mapping indoor environments are grid-based maps and topological maps [1]. In the present work, we focus on the latter one.

Grid-based methods produce accurate metric maps, are easy to build, represent and maintain, but often suffer from memory space and time complexity, because of its fine resolution restrictions which results in memory problems and also harder efficient planning and navigating in large-scale infrastructures. Topological maps, on the other hand, produce graph-like maps that can be used much more efficiently. They are simpler, permit efficient planning and do not require accurate determination of the robot's position. In these maps, vertices correspond to important places or landmarks, which are connected by edges that represent paths between them. However, its inaccuracy makes it harder to maintain consistency in large scale environments, which results in difficulties in recognizing similar places that look alike. There are some other constraints related to navigation using topological maps. Some limitations previously identified [2] were: handling inaccurate position and orientation information and detecting neighbor vertices in the topological map by traversing them, as opposed to sensing or recognizing them. Despite these constraints, the path planning method that was used produced adequate results.

In addition, map validation and self-localization in topological maps is also an important issue for correct robot navigation, which has been extensively studied [3]. In the work, the robot is given an input graph-like map and its current position and orientation with respect to the map. The robot has no instrument for measuring distances or orientations. By exploring the world systematically and using distinct markers, which can be detected, dropped and picked up, the robot can recognize places and determine whether the map is correct. The same authors later proposed [4] a multi-robot topological mapping technique. Robots simultaneously explore different regions of the environment and they meet periodically to merge their individual maps in order to create a shared merged map of the complete environment.

As seen before, topological maps can be a very important tool for most tasks using mobile robots. Many works exploit this simple representation to employ correct robot navigation. In Ferreira *et al.* [5], a robot drives around the environment and self-localizes, while using a place recognition technique to build 3D point cloud sets for monitoring changes that might take place in the environment.

In this work, the focus is mainly on obtaining a global topological abstraction from a preexisting metric representation of an environment. For example, for surveillance, monitoring and patrolling tasks with multiple robots, it is common to rely on topological maps for navigation issues. In a previous work [6], we presented a novel multi-robot patrolling algorithm based on topological maps.

In this article, we describe in detail how complete topological information is extracted from an existing 2D grid map representation of the area to be patrolled, which in turn can be obtained with a state-of-the-art robotic mapping technique, *e.g.* [7].

The next section presents a survey of previous techniques for extracting topological representations like diagrams and graphs from metric representations given *a priori*. Typically, these metric representations are occupancy grids, which are probabilistic maps wherein each cell of the grid contains a probability value that indicates whether the related location is free space or part of an obstacle [8]. These grids are usually obtained in a preceding exploration phase. In Section 3, we state the problem to be addressed and Section 4 presents the algorithm proposed to solve it. Later on, results are presented and the article ends with conclusions and future work.

2 Related Work

In the literature, there are a few existing techniques to extract topological representations using metric maps. In particular, Voronoi Diagrams [9] have been extensively used to plan a path that stays away from obstacles as far as possible. In addition, the Generalized Voronoi Diagram (GVD) [10] was described as a "retraction of the free space of the working space onto a network of one-dimensional curves reflecting the connectivity of free space" and a hierarchically organized Voronoi-based route graph representation for robot navigation in indoor scenarios was proposed.

Furthermore, the Extended Voronoi Graph (EVG) [11] has been presented more recently. In this new approach, the diagram (or skeleton) computation overcomes limitations in the sensory horizon of robots by clearly defining a transition from corridor-following to wall-following in large rooms. Although defined as a graph, beyond the actual diagram no information about vertices or edges is available after the final computation.

Additionally, Kolling and Carpin [12] proposed a method to extract graph representations from occupancy grids for surveillance tasks, based on the GVD. Graph modifications, called contractors, are promoted to simplify the surveillance problem towards solving it using fewer robots.

Another commonly used technique when extracting graphs is the Delaunay triangulation (DT) [13]. In fact, the DT of a discrete point set generally corresponds to the dual graph of the Voronoi diagram for the same point set.

Katsilieris *et al.* [14] extract the traversability graph from the original metric map in the context of searching intruders with a team of robots. Firstly, the obstacle-free area is triangulated and then triangles are merged to form large convex regions where the vertices are extracted. The set of edges consists of all pairs of regions that share a side.

In addition, a technique called Reduced Constrained Delaunay Triangulation to build graphs is also becoming popular. This technique is based on a DT computation together with a mechanism to reduce the graph by minimizing the distance each robot has to traverse. It was applied by Fazli *et al.* [15] in the context of a multi-robot area coverage task in a known and static 2D environment.

Another method presented in the literature to extract the graph of an environment is the visibility graph [16], which is composed of straight lines joining a sequence of vertices that are normally placed near the limits of obstacles in the environment (considering the dimension and pose of the robot). The graph is created connecting all vertices that can "see" each other, hence the name visibility graph.

Among the most used techniques are also thinning [17] and skeletonization [18] methods. These are operations used to remove the foreground pixels from binary images, analogously to peeling an onion. They provide a skeleton of the image, reducing all lines to a single pixel thickness. The output of such techniques is similar to the GVD. Szabó [18] also refers in his work some other methods for extracting topological maps from occupancy grids, namely sparse pixel approaches and matching opposite contours. Moreover, another technique called C-cells can be found in the literature [19]. It has the principle to divide the space not occupied by obstacles into convex polygons and the vertices of the graph are positioned in the center of those polygons, having each vertex connected in a straight line to the closest visible vertex.

Most previous works mentioned above present methods to visually extract the topological representation. However, many of these do not correctly characterize vertices and edges on the graph (like Voronoi Diagrams, EVG, thinning or skeletonization). As for those that do, the methods which are based on Delaunay Triangulation need a manual definition of the vertices in the environment to compute the topology and usually generate non-elegant and unbalanced representations. In the case of visibility graphs, highly inadequate topologies are produced since the vertices are positioned very close to obstacles, which is unnatural for robot navigation. The Generalized Voronoi Graph (GVG), which consists of a vertex for every meet and end point of the GVD and edges reflecting the connectivity of the GVD, results in very complex graphs with a large number of vertices and edges that are mostly redundant, especially when considering navigation tasks for robots. In addition, all of these methods assume that the edges connecting pairs of neighboring vertices must be a straight line, which is not necessarily true for robots with the ability to locally plan paths between two vertices.

In this work, we intend to go further ahead, by firstly extracting an elegant topological representation and then focusing on identifying vertices, edges and their length (*i.e.*, their weight) and obtaining relevant data concerning the vertices and the connectivity between areas of the map so as to assist robots navigation in tasks like patrolling, coverage, monitoring, surveillance, pursuit-evasion and others.

3 Problem Statement

In this work, the environment is assumed to be known *a priori* and a metric representation, typically with the form of an occupancy grid, is available. The goal is to abstract the environment through a topological representation, *i.e.*, a graph, and obtain all information about the graph's connectivity.

The techniques described in the previous section will lead to different graph representations, like placing vertices close to obstacles, inserting them halfway between obstacles or positioning them in a uniform way along the terrain. Having this in mind, a careful choice should be made, considering the applications in which the extraction is applied.

We start with a skeleton representation and then focus on extracting topological information to characterize the connectivity of the environment. There are several techniques to compute the initial skeleton as seen previously in section 2. In this work, it is obtained via the EVG computation, mainly because of its results in terms of producing visually attractive path representations that stay away from obstacles and that consider the sensing range of the robots as an input of the method. Additionally, it is a relatively recent technique, which is open source and is easily adaptable to the code that was developed. Having the representation of the skeleton, the graph is obtained through image processing techniques by correct identification of vertices and edges. The topological map is modeled as an undirected graph. Vertices represent places and edges represent the connectivity (in both directions) between those places.

In the next section, we present the algorithm to extract the complete topological information and its simplicity and efficiency becomes clear.

Table 1. EVG-THIN parameters [11]

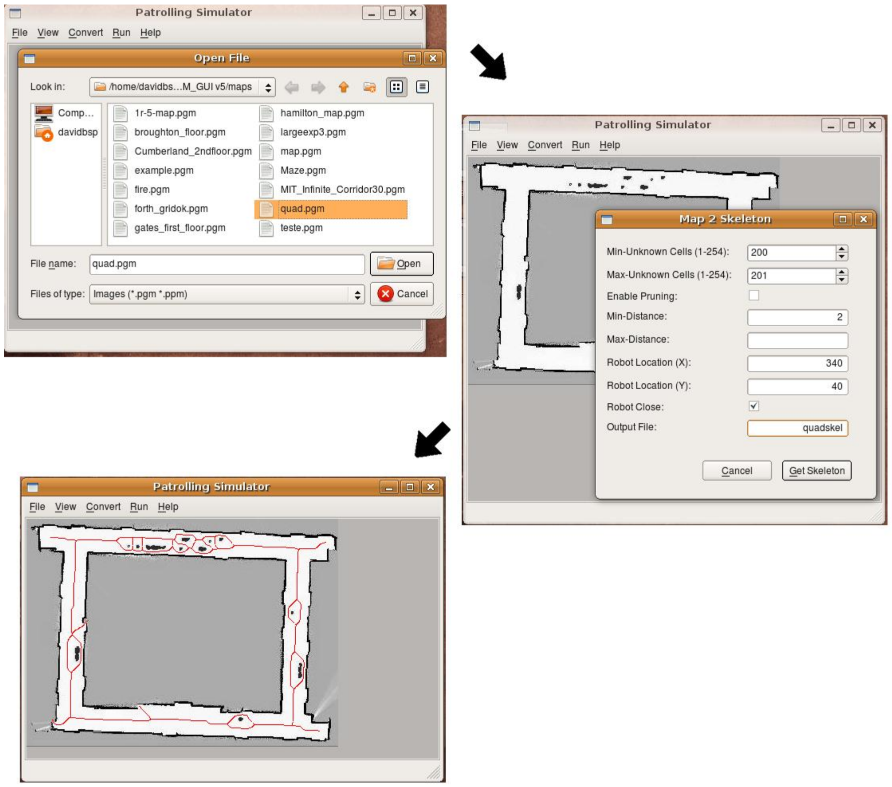
Command	Description
min-unknown [N]	The minimum greyscale value (1-254) of unknown cells. Occupied cells are 0-(N-1).
max-unknown [M]	The maximum greyscale value (1-254) of unknown cells. Free cells are (M+1)-255.
pruning [0|1]	Turns pruning on or off. Pruning removes all "branches" of the skeleton except those that meet one of these conditions: 1) The branch touches the edge of the grid. 2) The branch touches unknown cells.
min-distance [R]	Bleeds obstacles by R cells before calculating skeleton. This removes branches that come too close to obstacles.
max-distance [S]	If the skeleton exceeds the S cells from the nearest occupied cells, it switches to following the occupied cells S away.
robot loc [X Y]	This location is used to select which skeleton is valid, given complex images with multiple, disjoint skeletons. By default, the "robot" is located at the centre of the image.
robot-close [0|1]	The robot location (see above) is used to choose which skeleton is valid (if multiple exist). This is done by Euclidean distance between the robot's location at the skeletal points. This option turns off the checking mechanism except for points where the robot's distance is within the distance of the skeletal point to its closest obstacle.

4 The Algorithm

In a previous work [6] a patrolling simulator was used to validate a multi-robot patrolling approach based on balanced graph partitioning. Here, we show how the pre-assumed graph is provided to the patrolling simulator. The algorithm is comprised of 4 steps which are detailed below.

4.1 Acquiring the Skeleton

A tool named EVG-THIN developed by Patrick Beeson [11] was used to acquire skeletons from occupancy grids in this project. "EVG" stands for the Extended Voronoi Graph, which although being called a graph, does not present output related to information about vertices and edges on the graph, giving instead an enhanced skeleton on top of a grid map, when compared to the GVD. "THIN" accounts for the pixel-based thinning algorithm that finds skeletons of bitmaps, which is a fast approximation of the Voronoi diagram. Its code was written to be applied in real-time to occupancy grids, where cells are either occupied, free, or unknown, but it also works on any grayscale bitmap image in other domains. This command-line application runs in any Linux console upon the definition of some parameters, such as the pixels' threshold for considering them free, unknown or occupied; or the robot's minimum distance to an obstacle to account for the



Look in: /home/davidbs...M_GUI v5/maps

Comp...
davidbsp

1r-5-map.pgm
broughton_floor.pgm
Cumberland_2ndfloor.pgm
example.pgm
fire.pgm
forth_gridok.pgm
gates_first_floor.pgm
hamilton_map.pgm
largeexp3.pgm
map.pgm
Maze.pgm
MIT_Infinite_Corridor30.pgm
quad.pgm
teste.pgm

File name: quad.pgm — Open

Files of type: Images (*.pgm *.ppm) — Cancel

Min-Unknown Cells (1-254):	200
Max-Unknown Cells (1-254):	201
Enable Pruning:	☐
Min-Distance:	2
Max-Distance:	
Robot Location (X):	340
Robot Location (Y):	40
Robot Close:	✓
Output File:	quadskel

Fig. 1. EVG-THIN incorporated in the simulator

robot's geometric dimension and sensing range when navigating in the map, as seen in Table 1. Setting these parameters correctly is important, because the computed skeleton and final topology depends substantially on those parameters.

The Patrolling Simulator interface displays a window to set all the input parameters reusing and evolving the EVG-THIN's code, which is run in background[1], obtaining the 1 pixel-thin skeleton of the environment, as seen on Figure 1 (right).

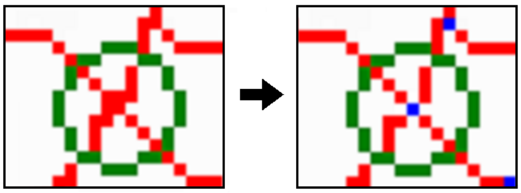


Fig. 2. Cluster detection and correction. The blue dots correspond to vertices subsequently identified.

[1] Note that this program is released under the GNU General Public License (GPL), which is intended to guarantee the freedom to share and change free software.

4.2 Cluster Image Removal

Due to aliasing, sometimes the skeleton representation generated in the previous section presents clusters, *i.e.*, sets of 4 pixel squares, as shown on Figure 2. These clusters are problematic, in the sense that they interfere with the subsequent detection of vertices in the graph, due to the 1 pixel-thin skeleton assumption. Therefore, an intermediate phase to avoid future data corruption when detecting the vertices and edges is necessary: Cluster Removal.

A protective filter was developed to detect and remove such clusters. Basically, it sweeps the image and when a cluster is detected, two actions are possible:

• Pixel Removal: Eliminate unnecessary pixels from the cluster;
• Pixel Shifting: Move pixels to the cluster's neighborhood.

These actions are conducted without affecting the connectivity of the graph. In Figure 3, on the left side, two examples of their application are shown. On top, the 2 pixels marked by a cross are removed, because they are unnecessary to guarantee the connection between all branches.

When a cluster is identified, the filter checks the implications of removing pixels; more specifically it guarantees that no disconnections resulting from erasing pixels will exist, by inspecting all the neighbor pixels around the cluster.

Erasing pixels is not always possible, as it can be observed in the bottom left of Figure 3, where removing red pixels would create gaps in the skeleton underneath. Hence, a pixel shifting strategy was created to remedy such situations.

4.3 Detecting Vertices

Having the cluster-free skeleton, the next step is extracting all vertices of the graph by image processing, which is done on top of the previously generated skeleton. The process is fairly simple, due to the 1-pixel thin assumption. Vertices correspond to single pixels that are placed in two specific locations:

• Dead-ends: Pixels that typically only have one neighbor pixel that also belongs to the skeleton. They originate vertices with degree[2] one.

• Branch intersections: Crucial pixels that correspond to crossings of the branches of the skeleton. They usually have three or more neighbor pixels that also belong to the skeleton, within the eight pixels around them. They originate higher degree vertices.

The process shown in pseudo-code by Algorithm 1 is conducted to identify all vertices in the graph. Basically, the intent is to inspect the 9-pixel window centered on the analyzed pixel of the skeleton and verify if it is a dead-end or a branch intersection. When the vertex is identified, a single blue pixel is drawn in the correct spot. Note that this algorithm relies on the fact that no clusters exist in the graph.

After running the algorithm, the visual representation of the graph on top of the grid is complete, as it will be shown on the results section, where the vertices were highlighted for better identification.

[2] "The degree (or valency) $d_G(v) = d(v)$ of a vertex v is the number $|E(v)|$ of edges at v (...), this is equal to the number of neighbours of v" [20].

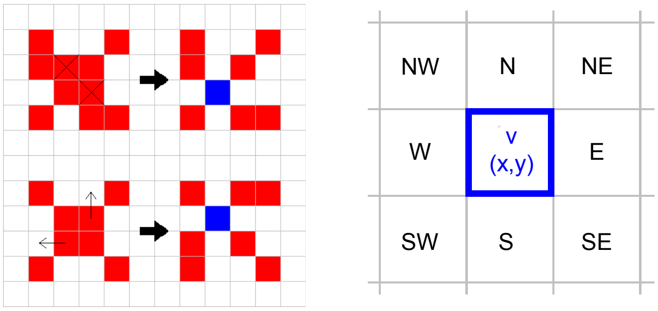
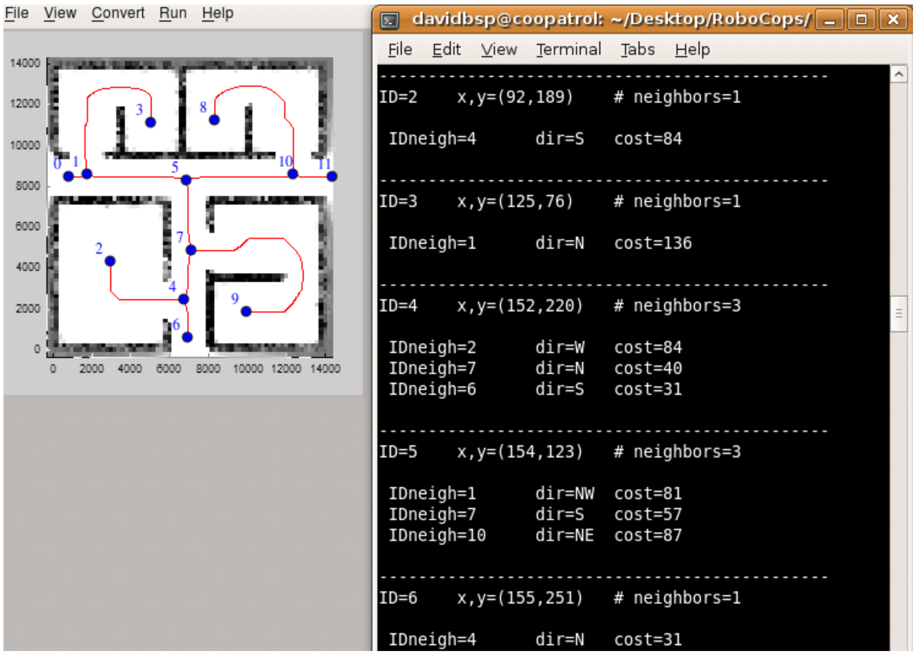


Fig. 3. On the left side: Cluster Analysis - Pixel removal on top and pixel shifting below. On the right side: Possible neighbor directions.


File View Convert Run Help
14000
12000
10000
8000
6000
4000
2000
0
0 2000 4000 6000 8000 10000 12000 14000
0 1 2 3 4 5 6 7 8 9 10 11
davidbsp@coopatrol: ~/Desktop/RoboCops/
File Edit View Terminal Tabs Help

```
--------------------------------------------
ID=2    x,y=(92,189)    # neighbors=1

 IDneigh=4      dir=S   cost=84

--------------------------------------------
ID=3    x,y=(125,76)    # neighbors=1

 IDneigh=1      dir=N   cost=136

--------------------------------------------
ID=4    x,y=(152,220)   # neighbors=3

 IDneigh=2      dir=W   cost=84
 IDneigh=7      dir=N   cost=40
 IDneigh=6      dir=S   cost=31

--------------------------------------------
ID=5    x,y=(154,123)   # neighbors=3

 IDneigh=1      dir=NW  cost=81
 IDneigh=7      dir=S   cost=57
 IDneigh=10     dir=NE  cost=87

--------------------------------------------
ID=6    x,y=(155,251)   # neighbors=1

 IDneigh=4      dir=N   cost=31
```

Fig. 4. Extracting the graph and relevant topological information. Example of a simple tree-like graph with 12 vertices.

4.4 Extracting Relevant Topological Information

The last step of the algorithm is to compute the rest of the topological information to completely characterize the graph (edges, connectivity, etc.). In this work, each vertex data structure will have the following fields associated to it:

- ID
- Coordinates (x, y)
- Degree
- Neighbors (Vertex Degree)={ID, direction, edge weight}

Algorithm 1. Pseudo-Code of the vertex detection process

```
foreach pixel p of the image do
    if p is red then
        count $\leftarrow$ number of red pixels inside 9-pixel window centered on p;
    else
        count $= 0$;
    end
    if count $> 3$ then          // Exclude pixels with 3 or more neighbors that do not correspond to branch intersections.
        foreach two adjacent red pixels around p do
            count = count $-1$;
        end
        foreach three adjacent red pixels around p do
            count = count $-2$;
        end
    else if count $== 3$ then          // Dead-end (with more than one neighbor pixel)
        if two adjacent red pixels around p then
            count = count $+1$;
        end
    end
    if count $> 3$ || count $= 2$ then
        p is a vertex $\Rightarrow$ blue pixel in p;
    end
end
```

In this step, the main concern is to convert visual information into graph data that can be used by robots to abstract the environment.

The ID and coordinates (x, y) of each vertex are easily extracted by looking for the blue pixels in the picture and assigning them ID's and saving their (x, y) pixel coordinates.

The other fields are obtained subsequently, also by image processing, by following all branches leaving the vertices via the red pixels' routes. When a blue pixel is found at the end of a route, a neighbor vertex is recognized, therefore the degree of the initial pixel is incremented, and a new entry to the neighbors table of the initial vertex is filled: the ID of the newly-detected neighbor vertex, its direction related to the initial vertex and the edge weight between the 2 vertices.

The ID is obtained by relating the coordinates of the newly-detected vertex to the ID list previously computed. The direction is defined in a similar way to the compass rose designation, as show on the right side of Figure 3, *e.g.*, if the direction of the route leading to the neighbor vertex starts in the pixel above the examined vertex, than the direction is defined as North (N). Also, the edge weight between vertices is defined as the cost of travelling from one vertex to another in pixel units, more specifically, the number of red pixels in the edge between both vertices.

Every vertex is identified by a single blue pixel, as shown previously in Figure 2. After obtaining all the data fields, each vertex will have a table of neighbors with the size of its degree and all the edges and the connectivity between neighbor vertices will be well characterized. This is clear in Figure 4. Having all this information, the framework for agents' navigation in the Patrolling Simulator is finally created.

5 Results and Discussion

The algorithm presented in this article was tested using a large diversity of maps that generated elegant and balanced representations that range from simple topologies to complex ones. An important aspect is that the complexity of the topological maps extracted has no relation to the dimension of the environment (in terms of Cartesian distances). The algorithm attained clear and visually attractive graph representations from metric maps of all sizes and shape, presenting no vertices with self-loops, staying away from obstacles and considering the sensing range of the robots; showing that it scales well in terms of environment, as shown in Figures 5-8. Beyond the accurate results obtained, the algorithm has shown to be computationally efficient. The C++ programming language was used and the Graphical User Interface (GUI) for the Patrolling Simulator was implemented using QT Open Source Edition 4.4.3, which is also based on C++ Language[3]. The system ran on Ubuntu 8.10 Intrepid on an AMD Athlon 64 Processor 3500+, 2.21GHz, with 1GB RAM.

Table 2 illustrates the time taken by the algorithm to extract all information from diverse metric maps, which is dependent not only on the amount of pixels in the free area of each metric map, as well as the complexity of the generated graph. Around 95% of the time values shown is spent computing the skeleton of the environment (first step of the Algorithm), which is the responsibility of EVG-THIN. The three other steps of the approach are computed in a few hundredths of second, which is extremely fast considering the application for which it was designed for.

The image pixels presented in the table do not relate in the same way, for every case, to the free pixels detected in each image. It is also clear that the computation time does not depend exclusively on this. It also depends on the graph's dimension as, for example, the fourth and fifth line of the table highlights.

These results attest to the applicability of the method proposed herein. It is a valuable method for obtaining a topological map of the environment from either an *a priori* known metric map or a metric map being built on the fly.

[3] The Patrolling simulator code is available at:
`http://isr.uc.pt/~davidbsportugal/packages`

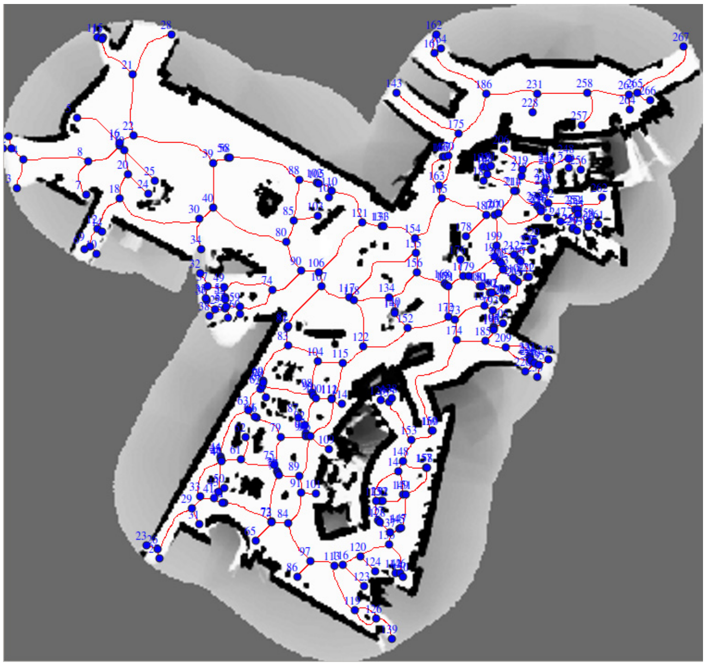


Fig. 5. Results for a complex graph (268 vertices) with high connectivity

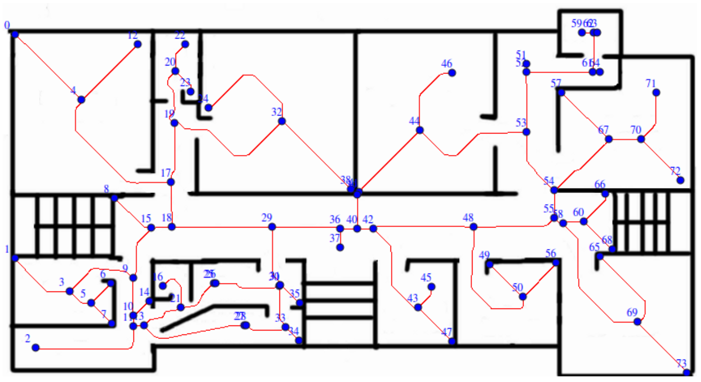


Fig. 6. Results for a medium graph (74 vertices) with defined connectivity

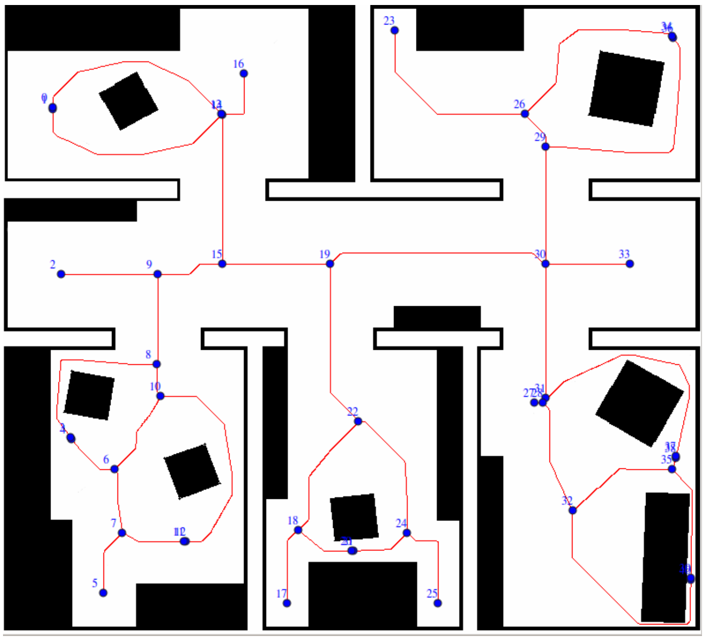


Fig. 7. Results for a medium graph (41 vertices) with defined connectivity

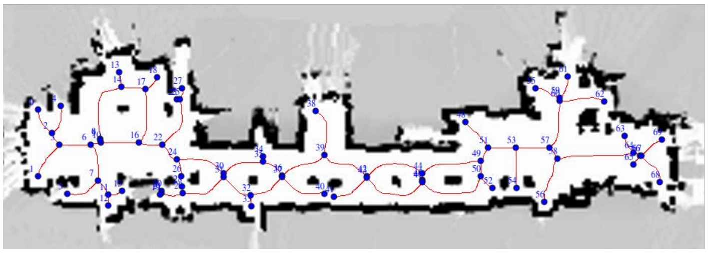


Fig. 8. Results for a graph with 70 vertices and a few connectivity constraints

Table 2. Computation time for different metric maps

Image Pixels	# Vertices	Time (s)
90K	12	0.40
341K	74	1.38
379K	20	1.42
343K	66	1.57
509K	206	1.61
544K	268	1.63
815K	135	3.62

6 Conclusions and Future Work

In this article, a novel way to extract topological maps and connectivity information from standard grid-like grayscale representations is described. The approach presented gains advantage from its simplicity, accuracy and performance. One possible disadvantage of using the EVG-THIN method to compute the skeleton of the grid map is the dependency on the correct parameterization, which is not straightforward for most cases. However, the approach presented in this paper is not limited to the use of EVG-THIN to extract the skeleton, other techniques like those mentioned in Section 2, can also be used.

Unlike most previous works in this area, here the intent is not to present solely a representation of the graph on top of the grid, but also to give one step ahead by proposing a way to convert visual information into data structures, by means of image processing techniques as described.

The proposed approach offers, as output, a complete characterization of the topological aspects of the environment, which has the ability to assist robot's navigation in a broad spectrum of activities, especially those that include path planning. This technique has been recently used to provide a topological map for mobile robots in cooperative patrolling misions [21]

As for future work, it would be interesting to test this approach using different methods in the literature to obtain the underlying diagram to check whether it is possible to speed up the first step of the algorithm without losing quality on the topological representation. Additionally, some questions are still left open like addressing fast update of the Voronoi Diagram given dynamic changes in the environment as well as considering 3D models and deal with topological navigation using mobile robots in real world scenarios.

Acknowledgements. This work was financially supported by a PhD grant (SFRH/BD/64426/2009) from the Portuguese Foundation for Science and Technology (FCT) and the Institute of Systems and Robotics (ISR-Coimbra) also under regular funding by FCT.

References

1. Thrun, S.: Learning maps for indoor mobile robot navigation. Artificial Intelligence 99, 21–71 (1998)
2. Zimmer, U., Fischer, C., Von Puttkamer, E.: Navigation on topologic feature-maps. In: 3rd International Conference on Fuzzy Logic, Neural Nets and Soft Computing, Fukuoka, Japan, pp. 131–132 (1994)
3. Dudek, G., Jenkin, M., Milios, E., Wilkes, D.: Map validation and robot self-location in a graph-like world. Robotics and Autonomous Systems 22(2), 159–178 (1997)
4. Dudek, G., Jenkin, M., Milios, E., Wilkes, D.: Topological exploration with multiple robots. In: 7th International Symposium on Robotics with Applications (ISORA 1998), Anchorage, Alaska, May 10-14 (1998)
5. Ferreira, F., Davim, L., Rocha, R., Dias, J., Santos, V.: Presenting a technique for registering images and range data using a topological representation of a path within an environment. J. of Automation, Mobile Robotics & Intelligent Systems 1(3), 47–55 (2007)

6. Portugal, D., Rocha, R.: Msp algorithm: Multi-robot patrolling based on territory allocation using balanced graph partitioning. In: 25th ACM Symposium on Applied Computing, SAC 2010, Sierre, Switzerland, pp. 1271–1276 (2010)
7. Thrun, S., Bugard, W., Fox, D.: A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. In: Int. Conf. on Robotics and Automation, San Francisco, pp. 321–328 (2000)
8. Elfes, A.: Using occupancy grids for mobile robot perception and navigation. Computer 22(6), 46–57 (1989)
9. Voronoi, G.: Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Journal für die Reine und Angewandte Mathematik 134, 198–287 (1908)
10. Wallgrün, J.: Hierarchical voronoi-based route graph representations for planning, spatial reasoning, and communication. In: 4th Int. Cognitive Robotics Workshop (CogRob 2004), pp. 64–69 (2004)
11. Beeson, P., Jong, N., Kuiper, B.: Towards autonomous topological place detection using the extended voronoi graph. In: Int. Conf. on Robotics and Automation (ICRA 2005), Barcelona, pp. 4373–4379 (2005)
12. Kolling, A., Carpin, S.: Extracting surveillance graphs from robot maps. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2008), Nice, France, pp. 2323–2328 (2008)
13. Delaunay, B.: Sur la sphére vide. Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk 7(6), 793–800 (1934)
14. Katsilieris, F., Lindhé, M., Dimarogonas, D., Ögren, P., Johansson, K.: Demonstration of multi-robot search and secure. In: Int. Conf. on Robotics and Automation (ICRA 2010), Anchorage, Alaska, USA (2010)
15. Fazli, P., Davoodi, A., Pasquier, P., Mackworth, A.: Fault-tolerant multi-robot area coverage with limited visibility. In: 2010 IEEE Int. Conf. on Robotics and Automation (ICRA 2010), Anchorage, Alaska, USA (2010)
16. Lozano-Pérez, T., Wesley, M.: An algorithm for planning collision-free paths among polyhedral obstacles. Communications of the ACM 22(10), 560–570 (1979)
17. Ko, B., Song, J., Lee, S.: Realtime building of thinning-based topological map. In: lnt. Conf. on Intelligent Robots and Systems, Sandal, Japan (2004)
18. Szabó, R.: Topological navigation of simulated robots using occupancy grid. Int. J. of Advanced Robotic Systems 1(3), 201–206 (2004)
19. Machado, A.: Patrulha multiagente: Uma análise empírica e sistemática. Master's thesis, Centro de Informática, Universidade Federal de Pernambuco (UFPE), Recife, Brasil (2002)
20. Reinhard, D.: Graph Theory, electronic edition. Springer, Heidelberg (2010)
21. Portugal, D., Rocha, R.P.: On the performance and scalability of multi-robot patrolling algorithms. In: 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR 2011), Kyoto, Japan, November 1-5, pp. 50–55 (2011)