J Intell Robot Syst (2014) 75:457–474
DOI 10.1007/s10846-013-9995-3

Construction of Minimized Topological Graphs on Occupancy Grid Maps Based on GVD and Sensor Coverage Information

E. G. Tsardoulias · A. T. Serafi · M. N. Panourgia · A. Papazoglou · L. Petrou

Received: 1 May 2013 / Accepted: 15 October 2013 / Published online: 21 December 2013

Abstract One of the tasks to be carried out during the robot exploration of an unknown environment, is the construction of a complete map of the environment at bounded time interval. In order for the exploration to be efficient, a smart planning method must be implemented so that the robot can cover the space as fast as possible. One of the most important information that an intelligent agent can have, is a representation of the environment, not necessarily in the form of a map, but of a topological graph of the plane, which can be used to perform efficient planning. This work proposes a method to produce a topological graph of an Occupancy Grid Map (OGM) by using a Manhattan distance function to create the Approximate Generalized Voronoi Diagram (AGVD). Several improvements in the AGVD are made, in order to produce a crisp representation of the spaces skeleton, but in the same time to avoid the complex results of other methods. To smooth the final AGVD, morphological operations are performed. A topological graph is constructed from the AGVD, which is minimized by using sensor coverage information, aiming at planning complexity reduction.

Keywords Approximate Generalized Voronoi Diagram (AGVD) · Coverage · Planning · Rescue robot · Topological graph

E. G. Tsardoulias · A. T. Serafi · M. N. Panourgia · L. Petrou (✉)
Faculty of Engineering,
Department of Electrical and Computer Engineering,
Division of Electronics and Computer Engineering,
Aristotle University of Thessaloniki,
54124 Thessaloniki, Greece
e-mail: loukas@eng.auth.gr

E. G. Tsardoulias
e-mail: etsardou@gmail.com

A. T. Serafi
e-mail: an.serafi@gmail.com

M. N. Panourgia
e-mail: maria_panourgia@hotmail.com

A. Papazoglou
School of Informatics, University of Edinburgh,
Old College, South Bridge, Edinburgh, EH8 9YL, UK
e-mail: papazoglou.anestis@gmail.com

1 Introduction

Autonomous robotics is a constantly evolving scientific area, which begins to influence in a significant degree our everyday lives. From automated industrial robotic workers to state of the art electric vacuums, autonomous agents provide speed, efficiency and accuracy to various tasks. A subset of autonomous agents is the various robot vehicles, such as a self-driving car, an autonomous exploration robot sent to extra-terrestrial environments or a fully-autonomous rescue vehicle.

An autonomous agent must have a global or local positioning system, ways to gather as much environmental information as possible and a decision module that processes the incoming information, based on both the robots and the environments state, to perform actions in order to accomplish its tasks. The current work deals with the tools needed by the decision module, and specifically with the ability to extract information from the positioning system (commonly a map of the environment), as well as minimizing the planning process, based on past gathered information.

Specifically, our work is focused on the tools needed by an autonomous agent in order to perform complete exploration of an unknown environment, as a part of a rescue robots functionality, under low exploration time and safe traversing. It must be noted that the current work proposes novel methods for creating the tools needed for an efficient exploration, but does not propose the means to utilize them, i.e. the exploration itself. It can be assumed that a map is provided (or constructed) by our system, as well as coverage information, that is the areas of the map the robot has already covered by its victim sensors, such as thermal, sound and CO_2 sensors as well as cameras.

The proposed work performs a skeletonization of the environment, producing an AGVD, efficiently implemented using the BrushFire method. In addition, a topological graph is constructed from the AGVD, such that it can be used to perform planning, e.g. in the selection of a long term strategy or just the production of the next optimal target. Finally the constructed topological graph is minimized, given the coverage information regarding the sensors that can perceive victims in the environment, and aims at reducing the planning time. The above procedures are batch and not incremental, so they must be executed every time the navigation/planning module produces a target. The proposed methods were also used in the autonomous robot of the team P.A.N.D.O.R.A. which participates in the RoboCup-RoboRescue competitions. The paper is organized as follows.

In Section 2, the state of the art is presented. Section 3 describes the construction of the AGVD, as well as the extraction of a connected topological graph based on it. In Section 4 the topological graph manipulation and minimization is described. Experiments are presented in Section 5. Conclusions and future work are discussed in Section 6.

2 Related Work

Many researchers have used the Generalized Voronoi Diagram (GVD) of an OGM, in order to implement smart and efficient methods of exploration, as the GVDs provide topological information of an environment by extracting a topological graph from it. The most common method of constructing a GVD is by performing morphological skeletonization to the plane, which simplifies and isolates the basic structure of a binary image. This method is usually time-consuming and prone to the obstacle irregularities, which often produces undesired sets of unconnected point subsets. Many methods have been proposed in order to overcome these drawbacks.

A geometric transformation for constructing the GVD of an environment was introduced by Fortune using a sweep-line algorithm [1]. The transformation is used to obtain the critical sites of the GVD rather than computing the original one. Mohammadi and Hazar introduced a method for the determination of a finite number of points belonging to the GVD [2]. In this method, random points are selected in the plane and their position is iteratively changed till they are equidistant from the two closest obstacles. The construction of a GVG (Generalized Voronoi Graph) is proposed from Ahn et al. in [3] by finding points belonging to the GVG, as a robot explores the environment. To detect points equidistant to two closest obstacles, distance sensors are solely used. In contrast to the above methods, many approaches were introduced in order to construct the full GVD of a plane, and not only a subset of it. Telea and van Wijk proposed an augmented Fast Marching Method (FMM) for computing skeletons [4]. In order to keep track of evolving boundaries, the FMM algorithm is applied, as the GVD points can be found in the collision of two or more boundaries. This technique is similar to ours, with the difference that FMM is not used in order to identify boundary surfaces. Another approach was followed by Rumpf and Telea, in which the GVD

is constructed by identifying the singularities by using higher order moments of the Distance function [5]. Lihong et al. proposed a skeletonization method by using Euclidian distance maps and Hit-and-Miss morphological operators [6].

An incremental Voronoi construction method was proposed by Kalra et al. using the dynamic BrushFire algorithm (analogously to the D* algorithm in planning search functions), in order to avoid recalculation of the GVD when the environment changes, or new areas are explored [7]. This algorithm only propagates the new information to portions of the map that could be affected. Thus, it avoids unnecessary reprocessing of the entire state space. Similar to this work, is the one proposed in [8] by Boris et al. where dynamic variances of BrushFire and skeletonization techniques are presented. Finally, a method that uses the Voronoi Diagram (VD) directly to perform exploration, without extracting a morphological graph is proposed by Garrido et al. [9]. The robot follows the potential field produced from the Extended Voronoi Graph (EVG) and the FMM algorithm.

As far as the graph construction from the GVG is concerned, Cheong et al. proposed a two-stage method in [10]. The nodes are extracted from image-based skeletonization and node point decision. The geometric data are transformed into a skeleton image and the node points are determined among the branch points in the image.

In [11] a solution to reduce the significant drawbacks that are usually introduced at geometric-based maps, such as the strong domain-specific dependencies and the inadequate mechanisms for handling sensor uncertainties and errors, is proposed by Poncela et al. Using a metric-topological representation of the environment a map is constructed, which allows explicit representation of non-explored areas and the relationships among them at topological level. In [12] an incremental construction method for GVD is presented by Choset et al., using a mobile robot with a ring of sonar sensors. The robot navigates and simultaneously constructs the GVD incrementally, based on line of sight range information. The terminating conditions for edge tracing are: a meet point, where three GVD edges join and a boundary point, where the GVD edge intersects the boundary of the environment. Meet points are detected by sensing for abrupt changes in the direction of the gradients to the closest obstacles. Kim et al. developed the boundary expansion algorithm, according to which a robotic vehicle equipped with range sensors, constructs the VD of an unknown environment by traversing all Voronoi edges [13]. They have introduced a control law that employs range measurements to make the robot track the Voronoi edges. The exploration algorithm makes decisions relying on information gathered at each intersection that the vehicle senses. Thus, the VD expands continuously to the explored area until a complete VD is constructed. In [14] a method that overcomes the GVD failure in large open environments by constructing the Saturated Generalized Voronoi Graph (S-GVG), is proposed by Tao et al. Finally in [15], Ko et al., aim to incrementally construct a hybrid map of the environment by adopting the compactness of a thinning-based topological graph and the accuracy of a metric map. The missing boundary information of the thinning-based topological graph is substituted by the added feature-related nodes. The additional nodes provide metric information related with coordinates and the orientation for corners.

As far as topological information is concerned, in [16], a semi-supervised place classification over a GVG is proposed, aiming at classifying the nodes of the GVG based on their topological information (corridor nodes, office/room nodes etc). The above consists of three different techiques, Support Vector Machines (SVMs), Conditional Random Fields (CRFs) and the Generalized Voronoi Graph (GVG). Finally, a different graph tool, created for efficient coverage planning are the Beam Graphs [17]. There, the topological information is coded in a sequence of virtual sensor beam crossings. In that way, goals can be selected, such that the paths leading to them cross as much beams as possible, increasing the space coverage.

Our work proposes an efficient method of constructing an AGVD, in comparison with the Thinning algorithm and Evg-thin, a tool for constructing AGVD, hosted in OpenSlam web page. In addition, a topological graph is exported from the AGVD, which is a satisfactory representation of the environments structure. Furthermore, a novel representation of the robots covered space

is introduced, based on which the topological graph is minimized. This approach aims at reducing the total execution time of planning, which uses the nodes of the topological graph as potential goals, in order to achieve full space coverage.

3 Construction of AGDV and Topological Graph

In mathematics, the VD is a special case of a metric space decomposition, which is based on distances between objects that are members of a set in that space. Specifically, let S be a set of two dimensional points, placed in a Euclidean plane X. For each point x of the space X, there is a member $s_i \in S$, such that s_i is closer to x_i than any other $s_j, i \neq j$. This applies for every point $x_i \in X$, except for those that are equidistant to two elements of S. The set of all points $x_i \in X$, which is closest to a specific element $s_j \in S$, consists the Voronoi Region, or Dirichlet Cell of element s_j.

Generally, the tuple of Voronoi Regions, associated with each $s_j \in S$, form the VD. In formal definition, let $(P_k), k \in K$, where K is a set of indices, be a tuple of non-empty subsets in space X. A Voronoi cell R_k, associated with the site P_k is the set of all points in X whose distance to P_k is not greater than their distance to other sites P_j, $j \neq k$. If $d(x, A) = inf\{d(x, a) : a \in A\}$, $d(x, A)$ denotes the distance between the point x and the subset A, then $R_k = \{x \in X : d(x, P_k) \leq d(x, P_j), \forall k \neq j\}$.

The difference between GVD and the classic VD is the set S. While in VD set S consists of discrete points in the space X, in GVD S consists of convex surfaces of polygons or other shapes that usually denote the obstacles of the space, while the other points $x \in X, x \notin S$ represent the free space. As in VD, the GVD consists of Voronoi Cells R_k so that $R_k = \{x \in X : d(x, P_k) \leq d(x, P_j), \forall k \neq j\}$, although now P_k is not a point but an entire convex surface.

Usually in robotics, the Euclidean space X, is the map that the robot constructs using a SLAM algorithm and in many applications this map is an occupancy grid (OGM). By OGM is denoted a grid representing the ground plan of an environment, each pixel of which holds the probability value that the cell is occupied. Thus, if a pixel has a value of 1 is certainly an obstacle, if it has value of 0 belongs certainly in free space and if it has 0.5, its occupancy is unknown or in other words it is unexplored.

In order to create the GVD of an OGM, each convex surface of the obstacles is declared as P_j. As the identification of these surfaces can be a very complicated problem, especially in unstructured environments, the most common method employed is the skeletonization of the free space. Skeletonization algorithm comes from the Image Processing field. Its goal is to construct the topological skeleton of a two-featured plane. Each pixel has a value of either 0 (background pixels) or 1 (foreground pixels). The obvious advantage of the skeletonization is that it is equally efficient in both structured and unstructured environments. The most common method of performing skeletonization is Thinning [18], which is a combination of morphological operators, based on which the foreground pixels turn to background, leaving the skeleton of the environment. A morphological operator consists of a kernel, which is applied to each pixel of an image and either changes its value, or leaves it intact, depending on the properties of the kernel. The drawback of performing Thinning is that the algorithm is iterative and it executes a big number of plane sweeps till the final result is achieved, leading to a time-consuming process.

In the present work, a method is described that uses a single plane sweep in order to compute the skeletonization of the free space. The result can be described as Approximate GVD, since it employs Manhattan distance instead of the Euclidean one. In addition, it should be noted that the OGMs used in the current chapter, as well as in Section 4, are constructed by the Critical Rays Scan Match SLAM (CRSM SLAM), an improvement on SMG SLAM proposed in [19].

3.1 Construction Based on BrushFire Algorithm

BrushFire is an algorithm commonly used in robotics, the goal of which is to construct the field containing the distance function from the obstacles of a space [20]. The general idea of the BrushFire algorithm is to simulate natural waves which initiate from specific points $x_i \in X$ and

propagate through a set of points with a distinct characteristic. In the problem of skeletonization, the BrushFire algorithm initiates from the pixels that have an occupancy value higher than 0.5, i.e. the obstacles of the environment ($x_i \in X_{\text{occu}}$) and propagates through the pixels that have occupancy value lower than 0.5, i.e. the free space ($x_i \in X_{\text{free}}$). The propagation finishes when there are no more unvisited pixels in X_{free}.

The algorithm starts by initializing the X_{frontier} by inserting all $x_i \in X_{\text{occu}}$ and updating their BrushFire values to 0. Then, in each iteration each $x_i \in X_{\text{free}}$ that has not been assigned with a BrushFire value and is adjacent to an element of X_{frontier} (assuming 8-point connectivity), updates its BrushFire value to 1 and is inserted in the set $X_{\text{nextFrontier}}$. The $X_{\text{nextFrontier}}$ set denotes the pixels to be investigated in the next iteration. In iteration i, the BrushFire value of each $x_i \in X_{\text{frontier}}$ is i and the algorithm updates the values of each $x_i \in X_{\text{nextFrontier}}$ with the value of $i+1$. When all $x_i \in X_{\text{frontier}}$ are investigated, $X_{\text{nextFrontier}}$ takes the place of X_{frontier} and thus the next iteration takes place. The algorithm ends when $X_{\text{nextFrontier}} = \emptyset$.

The skeletonization method proposed, identifies the pixels that constitute the space skeletonization during the propagation of the BrushFire algorithm. If the propagation of the BrushFire values is visualized as distinct waves that start from the convex obstacle values, it can be stated that a pixel $x_i \in X_{\text{free}}$ belongs to the free space skeleton, given that it belongs to the collision frontier of two or more waves propagating from distinct surfaces. This is true since each wave propagates with the same speed as any other, so if a pixel is placed in the collision frontier of two waves, this means that it is equidistant from the two surfaces in terms of Manhattan distance

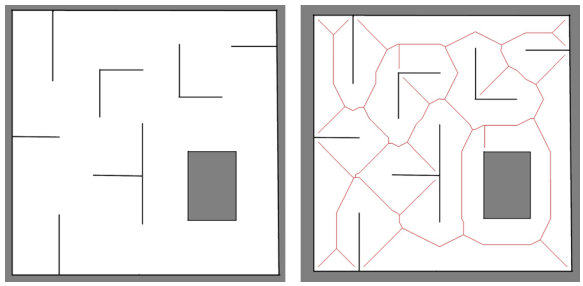
As the information of the distinct surfaces is unavailable in order to detect the different waves, the idea of parenthood in each pixel that the waves are propagated through is inserted. The point x_j is denoted as $P(x_i)$, if its expansion led to the update of the BrushFire value of x_i. The general idea is that during the propagation, the parents of the updated pixels are copied in the propagated pixels, indicating the origin of each one. Given the collision of two waves, the first indicator of different origin surfaces of the waves, may be the distance between the parents of the pixels at the point of collision. The minimum distance between two parents, for the waves that have initiated from different surfaces, is denoted as P_{MIN}.In addition $Neigh(x_i, X)$ is the set of the pixels that are neighbors to x_i and belong to set X.

The pseudo code that describes the creation of the AGVD based on the BrushFire algorithm is presented in (Algorithm 1). Let X_{GVD} be the set

Algorithm 1 BrushFire-based AGVD calculation

Input: X_{occu}, X_{free}, P_{MIN}, B_{MIN}
Output: X_{GVD}, $Bf(x_i)$

```
for all x_i ∈ X_occu do
    Insert x_i in X_frontier
    Set P(x_i) ← x_i
    Bf(x_i) ← 0
end for
Step ← 0
while X_frontier ≠ Ø do
    Step ← Step + 1
    for all x_i ∈ X_frontier do
        for all x_j ∈ Neigh(x_i, X_free) do
            if Bf(x_j) not initialized then
                Bf(x_j) ← Step
                P(x_j) ← P(x_i)
                Insert x_j in X_nextFrontier
            else if Bf(x_i) = Step then
                if Dist(P(x_i), P(x_j)) > P_MIN
                and Bf(x_i) > B_MIN then
                    Insert x_j in X_GVD
                end if
            else if Bf(x_j) = Step − 1
            then
                if Dist(P(x_i), P(x_j)) > P_MIN
                and Bf(x_i) > B_MIN and
                x_j ∉ X_GVD then
                    Insert
                    x_i in X_GVD
                end if
            end if
        end for
    end for
    X_frontier ← X_nextFrontier
end while
```



(a) OGM Sample (b) AGVD of the environment (thin line)

Fig. 1 Demonstration of OGM-produced AGVD

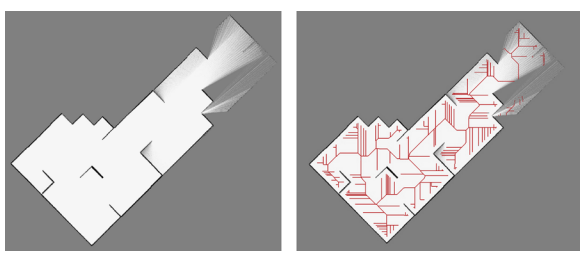
that holds the AGVD points. In addition, since the AGVD will be used to create a topological graph, it is rational to contain elements (pixels) in a minimum distance from the obstacles. This distance is denoted as B_{MIN}. Finally $Bf(x_i)$ holds the arithmetic value of the BrushFire expansion of the pixel x_i.

The result of the above algorithm is depicted in Fig. 1a and b. The drawback of the above method is that while it performs well in simulated OGMs, i.e. maps that are produced by an image processing software (as the map in Fig. 1a), the results in a real OGM produced by a Laser Range Finder, are disappointing (Fig. 2a and b).

The reason of the algorithm failure is that the obstacles in real OGMs are not smooth, so the irregularities produce a large number of parasitic edges in the AGVD. It is interesting that even in the Fig. 1b two parasitic lines can be detected, which exist due to the irregularities inserted by the drawing procedure of the map in the image manipulation software. Therefore a method must be implemented to eliminate those outliers and keep the real space skeletonization. The proposed solution is based on the idea of contamination. Next a contamination is applied in all the terminal elements of the AGVD, i.e. the elements $x_i \in X_{\mathrm{GVD}}$ for which $SizeOf(Neigh(x_i, X_{\mathrm{GVD}})) \leq 2$, forming the set $X_{\mathrm{contaminated}}$. The next step is to eliminate $x_i \in X_{\mathrm{contaminated}}$ from X_{GVD} and spread the contamination in their neighbours that now are terminal elements. The above procedure is iteratively executed since no more eliminations from X_{GVD} occur, and the remaining pixels consist the AGVD.

Intuitively the contamination method eliminates all parasitic edges up to the point that a junction in the AGVD is approached, i.e. a pixel x_i that $SizeOf(Neigh\ (x_i, X_{\mathrm{GVD}})) > 3$. In other words, the contamination procedure simulates a BrushFire algorithm initiating from the leafs of AGVD, propagating only through it while simultaneously removing the visited elements from the AGVD set. The algorithm stops locally in the AGVD's junction elements.

The last flaw in the algorithms result is that the AGVD has not one-pixel width throughout all its structure, property that the result of direct free space Thinning would have. This may seem unimportant, but it is crucial to the easy extraction of the terminal and junction nodes. The action proposed to deal with this kind of problems is to perform the Thinning algorithm, which will be applied only to the pixels of the GVD, thus it will not be time-consuming. The final result is illustrated in Fig. 3.



(a) OGM created by a real LRF (b) Noisy AGVD because of LRFs irregularities

Fig. 2 Failure of AGVD due to a real OGM

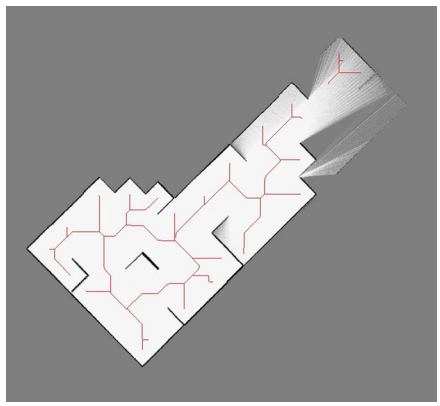


Fig. 3 AGVD after the "contamination" and Thinning procedures

In order to demonstrate the usefulness of the contamination and thinning steps, a number of experiments were executed, counting the number of nodes calculated by the algorithm that extracts the topological graph of an environment. Referring to the environment shown in Fig. 3, the initial AGVD resulted in 257 nodes, after the contamination procedure the nodes reduced to 71 and finally the morphological smoothing resulted in the number of 42 nodes, a reduction of 83.65 % related to the initial size.

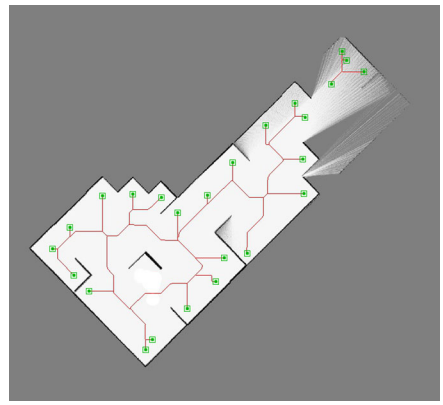
3.2 Extraction of a Connected Topological Graph Based in AGVD

Provided that the AGVD has one-pixel width throughout the entire skeleton, it is a quite simple task to extract the features of a graph, consisting of the terminal pixels and the junctions of the AGVD as vertices and the possible connections through the AGVD as edges.

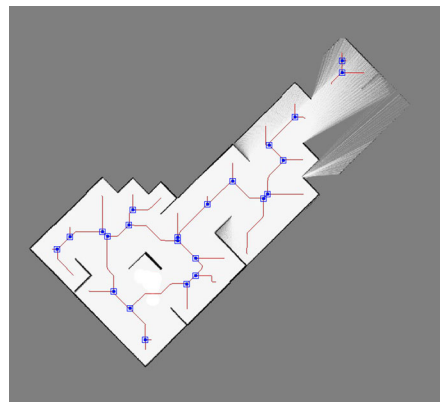
First of all, the pixels $x_i \in X_{\text{GVD}}$ that $SizeOf(Neigh(x_i, X_{\text{GVD}})) = 1$, are denoted as terminal vertices. The result is depicted in Fig. 4a with light green squares. The second step is to detect the junctions of the AGVD . Equivalently, the condition for $x_i \in X_{\text{GVD}}$ to be a junction vertex is if $SizeOf(Neigh(x_i, X_{\text{GVD}})) > 3$ (Fig. 4b—blue squares). The result is the completion of detection of the vertices extracted from AGVD.

Generally, it is possible that the AGVD consists of more than one unconnected parts. We take as an axiom that the most important part is the one closest to the robot pose, as it makes sense for the robot to plan its actions taking into consideration the nearby space, since the environment is unknown a priori, and larger parts of it are explored gradually.

In order to detect the closest connected part of AGVD, at first the pixel $x_i \in X_{\text{GVD}}$: $Argmin_{x_i} Dist(x_i, R)$, $x_i \in X_{\text{GVD}}$ is detected, which is the closest point of AGVD to the robot pose (R). This action is achieved by performing a Wavefront algorithm initiating from R, propagating through the X_{free}, and terminating when a pixel $x_i \in X_{\text{GVD}}$ is reached. This pixel is symbolized as x_{close}. The Wavefront algorithm is similar to BrushFire except that the propagation initiates from solely one point.



(a) Terminal Nodes

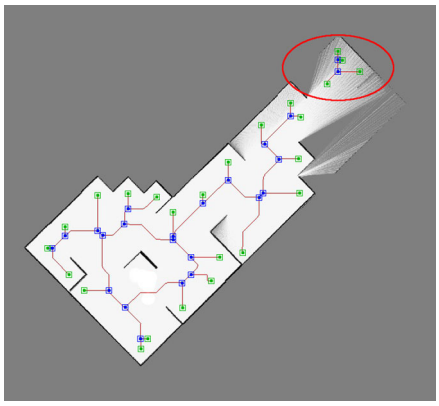


(b) Junction nodes

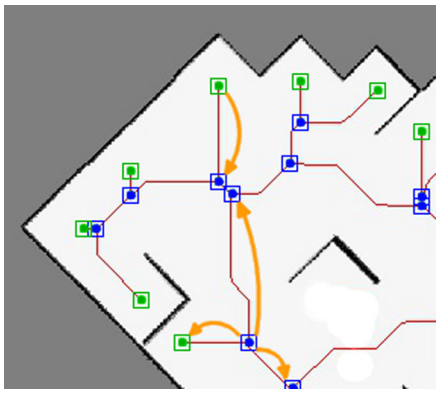
Fig. 4 Identification of AGVD's nodes

The next step is to execute a Wavefront algorithm that initiates from x_{close}, propagates in the X_{GVD} space and terminates when no more propagation is possible. During this propagation, every $x_i \in X_{\text{Nodes}}$ that is reached, is inserted in the set $X_{\text{Connected}}$. Thus, $X_{\text{Connected}}$ keeps the vertices that exist in the closest connected part of AGVD. Finally, $x_i \in X_{\text{Nodes}}$ is replaced by $X_{\text{Connected}}$, so the vertices that exist in the unconnected parts of the AGVD are eliminated. Figure 5a illustrates the unconnected part that is eliminated.

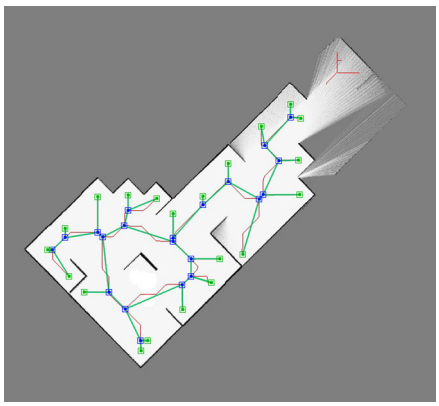
The final step to construct a graph, is to determine the valid connections between the vertices found by the previous methods, which vertices are symbolized as $x_i \in X_{\text{Nodes}}$. A connection between x_i and x_j, $[x_i, x_j] \in X_{\text{Nodes}}$ exists, if there is a path consisting of pixels $x_k \in X_{\text{GVD}}$, such that for each $x_k, x_k \notin X_{\text{Nodes}}$. In other words, a connection between two vertices is valid if there is a path



(a) Elimination of unconnected GVD part



(b) Determining neighbouring relations in the topological graph



(c) Final topological graph produced by AGVD. Each line is a neighbour connection between two vertices

Fig. 5 Steps of AGVD graph's creation. Robot's pose is indicated by R

through AGVD that connects them and no other vertex exists in this path. The proposed method is based on the Wavefront algorithm. Specifically, for each element of X_{Nodes} a Wavefront algorithm that initiates from each element x_i and propagates solely through the AGVD is employed. If a vertex x_j is approached, the edge e_{ij} is inserted in E and the propagation on the specific way terminates. The algorithm reaches an end when no further propagation is possible.

During each propagation initiating from a point x_i, let X_{Neigh_i} be the set of the pixels that are reached from the algorithm. Then for each $x_j \in X_{\text{neigh}_i}$ the edge e_{ij} is created and inserted in E. Two examples of the algorithm are depicted in Fig. 5b. The final topological graph constructed by the AGVD information is depicted in Fig. 5c.

4 Graph Manipulation

In Chapter 3, a description of the procedure regarding the topological graph construction based on AGVD was introduced. In this chapter a further issue is discussed, which involves the idea that the exploration is based on both the connected undirected graph and the information concerning the areas that have already been covered by the robot. Therefore, it would be effective to manipulate the graph such that to consist only of vertices with low coverage value. Following the naive approach, which is the direct elimination of all the vertices with high coverage value, implies a high probability for the graph to become unconnected, something which is undesirable. Thus, the graph will be minimized by taking into consideration not only the explored space of the environment, but also a manipulation procedure that guarantees the minimized graph connectivity.

The minimization procedure is presented via algorithms implemented in the OGM of Fig. 2a and specifically on the connected undirected graph presented in Fig. 5c. In addition, Fig. 6 illustrates an indicative coverage map.

The coverage map is similar to the OGM, although in the current case each cell holds the possibility of the pixel to be explored, or covered by the robots victim identification sensors. In contrast to the OGM, the initial values of the coverage map are set to 0, as it is assumed that the robot initially enters in an entirely unknown and unexplored space. In our implementation, the

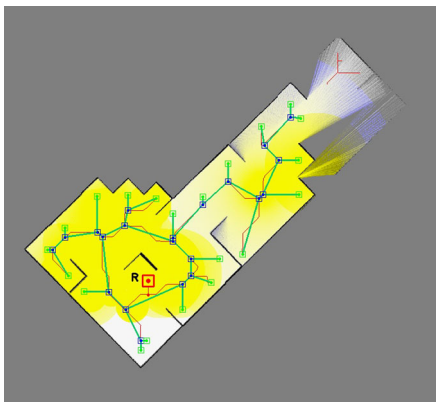


Fig. 6 OGM with topological graph and coverage information. Yellow cells are covered by the robots victim sensors (covered space)

explored space has a different meaning than the covered space. Specifically, the explored space consists of the free pixels of the OGM, meaning the space the LRF has perceived as empty. Covered space is a part of the explored space, as coverage is considered the area that was perceived by the victim sensors of the robot. Covered space is a subset of the explored space, as the sensors (such as cameras, microphones, thermal or CO_2 sensors–sensors that the robot P.A.N.D.O.R.A. is equipped with) that are able to recognize victims, have a smaller range of reliable operation than the LRF.

The steps for the minimization of the topological graph by means of coverage are firstly the recursive elimination of the covered terminal vertices, secondly the recursive elimination of the covered vertices that have exactly two neighbours and finally, the elimination of the vertices and edges that do not participate in any of the minimum paths connecting the remaining nodes of the AGVD. The results would be the same if we performed the final step in the graph, but the execution time would be much larger, as the possible pairs of nodes increase geometrically.

4.1 Elimination of Covered Terminal Vertices

This step eliminates iteratively all the terminal vertices with high coverage value from the set of graphs vertices. Nonetheless, some vertices that used to have more than one neighbour, become terminal after this elimination. This is possible if a vertex participates in a tree structure, which is a specialized form of a graph, characterized by the absence of any loops.

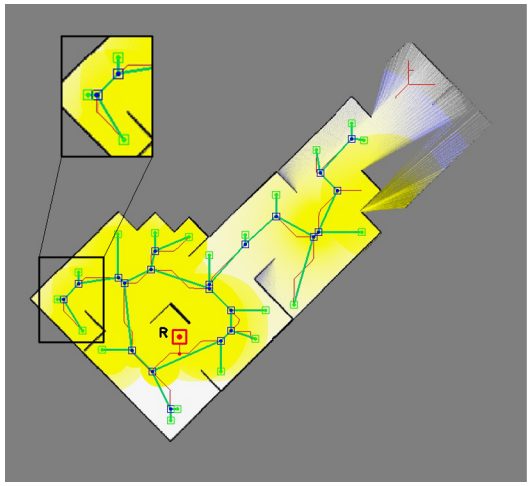
As it is mentioned, V is the set of the graphs nodes and E the set of the graphs edges. If $e_{ij} \in E$, there is a connection between vertices v_i, v_j. Then, these vertices can be described as neighbours. In addition $Coverage[v_i]$ holds the coverage value that vertex v_i has, which is bounded between [0, 1], as it represents the probability of that cell to be covered. Finally, the constant $Cov_{\min}$, states the minimum coverage value/probability that a vertex can have, in order to be considered as covered. In the specific implementation $Cov_{\min} = 0.5$. The weight of an edge $e_{ij} \in E$ is denoted as w_{ij} and is the total length of the path through AGVD that connects vertex v_i with vertex v_j.

The elimination method iterates through the members of the set V and identifies all terminal vertices, i.e. the vertices that have exactly one neighbour, by checking if $SizeOf(Neigh(v_i, V)) = 1$, $v_i \in V$. If the above is true and simultaneously $Coverage[v_i] > Cov_{\min}$, then v_i is a terminal vertex and it is covered by the robots sensors. The next step is to eliminate the edges e_{ij} that connect vertex v_i with every $v_j \in Neigh(v_i, V)$ by erasing v_i from the set $Neigh(v_j, V)$. Finally the vertex v_i itself is eliminated from V. The above routine must be executed iteratively until no eliminations in V are made, as it is possible for new terminal covered vertices to appear after each elimination. The result of the appliance of the above method to the graph produced by the AGVD appears in Fig. 7a, b and c.

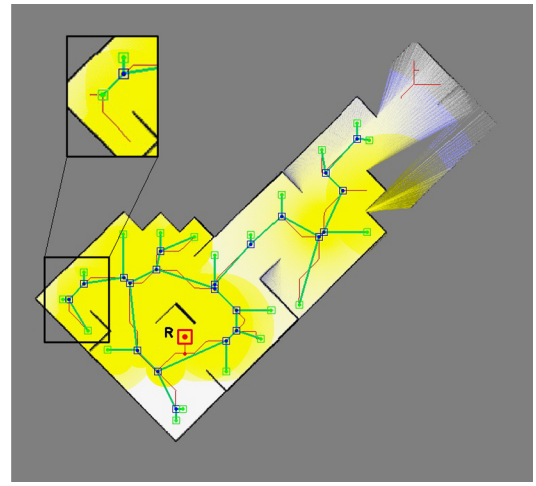
After the first step of elimination, the two terminal vertices in Fig. 7a are eliminated from the set of graphs vertices and their neighbour becomes a terminal vertex (Fig. 7b). After the iterative eliminations of the terminal vertices, the topological graph of the environment has the form of Fig. 7c.

4.2 Elimination of Covered Vertices Having Two Neighbours

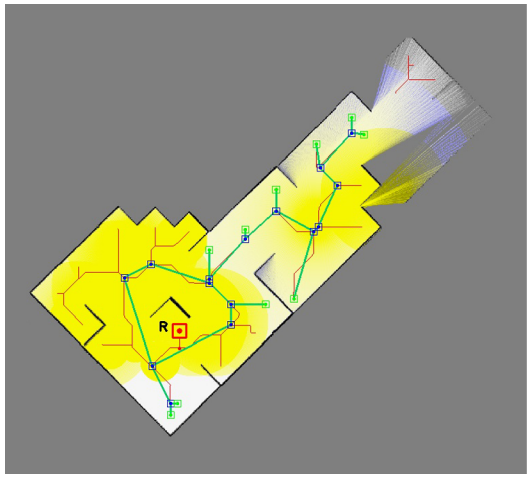
The second step detects all the covered vertices that have exactly two neighbours and replaces the path between the two limit vertices with a



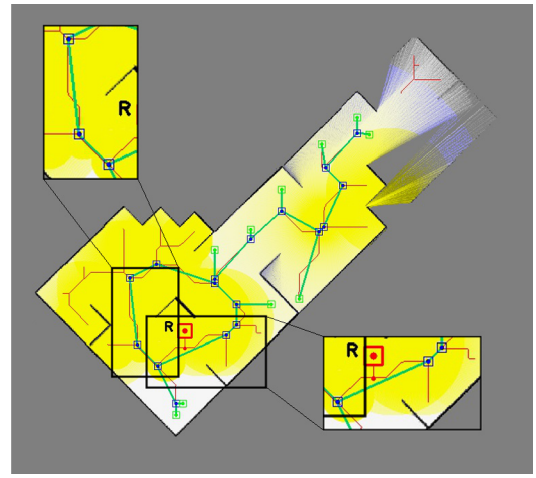
(a) Initial step of terminal vertices elimination



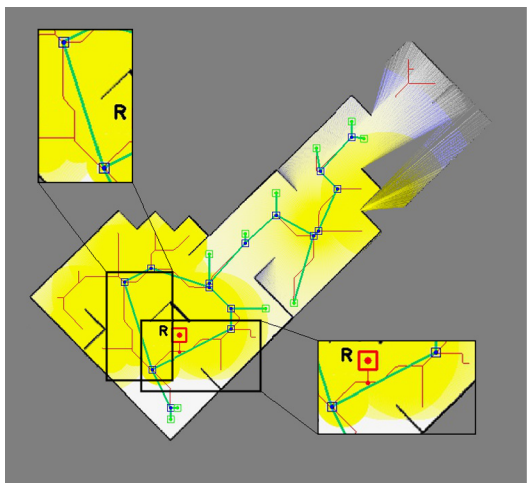
(b) Result of recursive terminal vertices elimination



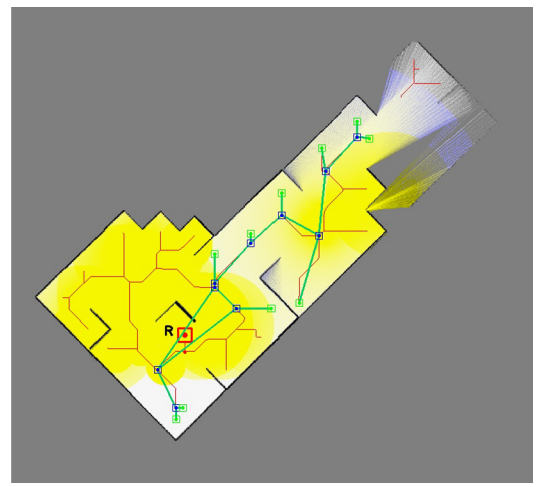
(c) Redundant terminal vertices eliminated



(d) Example of AGVD joints in covered area



(e) Elimination of covered joint nodes



(f) Final topological graph after the recursive joint vertex elimination

Fig. 7 Steps of AGVD graphs minimization

direct one and the covered intermediate vertex is eliminated. The vertices that have been detected are eliminated from the set of graphs vertices (Fig. 7d and e).

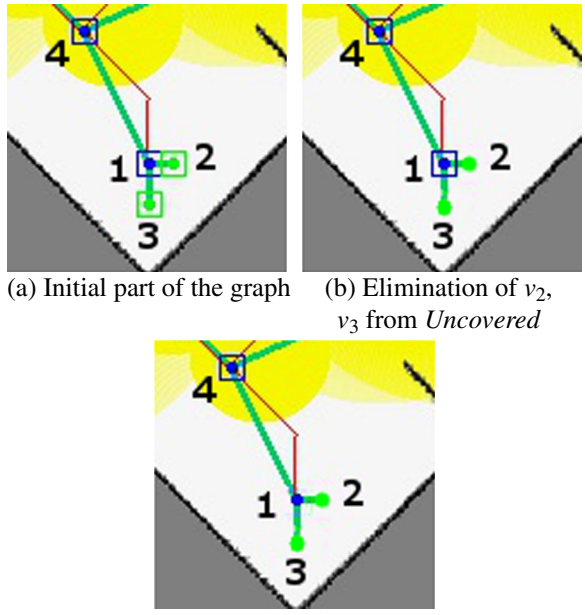
A similar approach is used to the requested covered vertices to the one applied in the elimination of covered terminal vertices. Considering a vertex v_i with a high coverage value that has exactly two neighbours v_{j_1} and v_{j_2}, then the connections of this vertex to its adjacent vertices are eliminated (e_{ij_1} and e_{ij_2}). Furthermore, the two neighbours are connected by inserting the edge e_{j_1,j_2} in E, and updating the new edges weight by summing the weights of the two eliminated edges. It should be noted that none of the neighbours v_{j_1} and v_{j_2} are checked regarding their coverage value since their common covered neighbour (v_i) should be eliminated no matter if they are covered or not. Finally, the vertex v_i is eliminated from the vertices set V. Identically to the previous step, the above routine is iteratively executed until no more elimination occurs.

The result of the elimination is presented in Fig. 7a, b and c. Figure 7d shows the first step of the algorithm where two vertices with high coverage value and exactly two neighbours are found. These vertices are eliminated from V and their neighbours are connected with an edge (Fig. 7e). After the method is completed, the graph of the environment has the form of Fig. 7f.

4.3 Elimination Through Minimum Cost Paths

The third and final step of the graph manipulation procedure is to eliminate the covered vertices and those edges which are not essential for the exploration. This is performed by the detection of the minimum cost paths that connect all the uncovered vertices and the elimination of all the covered vertices and the edges that do not participate in any minimum cost path. The term minimum cost paths, denotes the minimum cost routes that connect two vertices with low coverage value. Before extracting the minimum paths of all the nodes of V in pairs, the cardinality of V, by constructing the *Uncovered* set, is reduced. This holds only the essential nodes for the minimum paths algorithm, in order to avoid overlap and consecutively increase of execution time.

As *EssV* (Essential Vertices) is denoted the set of the vertices that participate in at least one minimum path and as *EssE* (Essential Edges) the set of the edges that was visited in the pre-mentioned paths. Initially, all the uncovered nodes are detected and enumerated, as well as inserted in the set *Uncovered*. The first step is to insert in *EssV* each $x_i \in Uncovered$, $SizeOf(Neigh(x_i, Uncovered)) \leq 1$, i.e. all the nodes of *Uncovered* set that have exactly one or no neighbours contained exclusively in *Uncovered* and replace them in the *Uncovered* set with their only neighbour if $SizeOf(Neigh(x_i, Uncovered)) = 1$, or with their only high coverage neighbour if $SizeOf(Neigh(x_i, Uncovered)) = 0$. This procedure if done iteratively may boost the time of the total algorithm, as it is possible to shrink the number of the vertices, the minimal paths of which must be calculated. For example in Fig. 8a, b and c, three vertices (v_1, v_2, v_3) are elements of the *Uncovered* set.



(a) Initial part of the graph (b) Elimination of v_2, v_3 from *Uncovered*

(c) Elimination of v_1 from *Uncovered* and insertion of v_4

Fig. 8 Steps of detecting *Uncovered* elements

The first iteration of the above step discovers that v_2 and v_3, are members of the *Uncovered* set and have exactly one neighbour, which is $v_1 \in$ *Uncovered*. Thus it erases v_2, v_3 from *Uncovered*, adds them to *EssV* and adds v_1 in *Uncovered*, action that has no impact since v_1 belongs already to *Uncovered* set (Fig. 8a). The second iteration finds the vertex which has no neighbours in *Uncovered*. So v_1 is erased from *Uncovered* set, inserted in *EssV* set, and the vertex v_4 is inserted in *Uncovered*. Of course this raises a paradox, as the set *Uncovered* eventually contains vertices that are actually covered, but this has no impact in the overall algorithm, as it will be explained. The result of the first step in the *Uncovered* set, is the reduction of the set by two elements, which is the desirable outcome.

Subsequently, all possible combination of vertices $v_i \in$ *Uncovered*, in pairs (v_i, v_j) where $i < j$ are identified. For each pair, the minimum cost path that connects the two vertices is calculated. The algorithm used to find the minimum path is the popular A* algorithm [21], which is an efficient generalization of the Dijkstra algorithm. For each pair, A* returns a set of vertices which consists of the initial vertex, the terminal one and the vertices that participate in the minimum cost path, that connects the initial and the terminal vertex. The minimum cost path between vertex v_i and v_j is denoted as $Path_{ij}$. Then the vertices contained in $Path_{ij}$ are inserted in the *EssV* set and each transition from x_i to x_{i+1}, where $x_i, x_{i+1} \in Path_{ij}$, is inserted to *EssE*. Finally, the vertices that are contained in V, but not in *EssV*, are removed from V, as well as all the participating edges. The pseudo code for the elimination through minimum cost paths follows (Algorithm 2). The final result of the graph manipulation is presented in Fig. 9.

In order to prove the usefulness of the steps before the execution of the minimum paths algorithm, we performed an experiment counting for each step the number of nodes of the GVD in the map shown in Fig. 3. As stated in the end of the previous chapter, the final topological graph consisted of 42 nodes. After the recursive elimination of the covered terminal nodes the remaining nodes were 27 and after the deletion of the covered nodes with two neighbours, 20.

The *Uncovered* set contained only 8 nodes, representing the number of nodes to be used in the minimum paths algorithm. As stated, this algorithm checks all the possible pairs of nodes and

Algorithm 2 Graph manipulation algorithm

Input: *Coverage*, $Cov_{\min}$, V, E
Output: V, E

```
for all v_i ∈ V do
    if Coverage[v_i] < Cov_min then
        Insert v_i in Uncovered
    end if
end for
for all v_i ∈ Uncovered do
    for all v_i ∈ Uncovered do
        if SizeOf(Neigh(v_i,Uncovered)):=0
         then
            Insert Neigh(v_i,V) in Uncov-
            ered
            Erase v_i from Uncovered
        else if SizeOf(Neigh(v_i,Uncovered)):=1
         then
            Erase v_i from Uncovered
        end if
    end for
end for
for all v_i ∈ Uncovered do
    for all v_j ∈ Uncovered, i < j do
        Path_ij = A*(v_i, v_j)
        ∀v_k ∈ Path_ij insert v_k in EssV
        ∀x_i, x_{i+1} ∈ Path_ij insert e_{i,i+1} in EssE
    end for
end for
for all v_i ∈ V do
    for all v_j ∈ Neigh(v_i, V) do
        if v_j ∉ EssV or e_{i,j} ∉ EssE then
            Erase e_{i,j} from E
        end if
    end for
    if v_i ∉ EssV then
        Erase v_i from V
    end if
end for
```

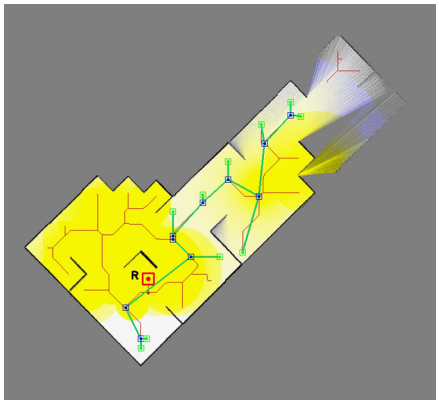


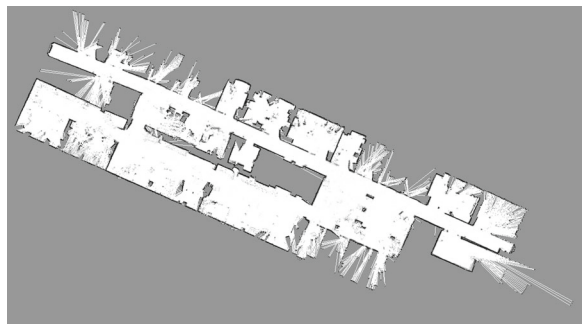
Fig. 9 OGM with topological graph and coverage information. Yellow cells are covered by the robots victim sensors (covered space)

eliminates the elements that do not participate in at least one path. If the calculation was performed without the construction of the *Uncovered* set, the possible permutations would be 190 (20 nodes in pairs with no repetition and order significance). Instead, with the *Uncovered* set the paths needed to be computed are only 28.

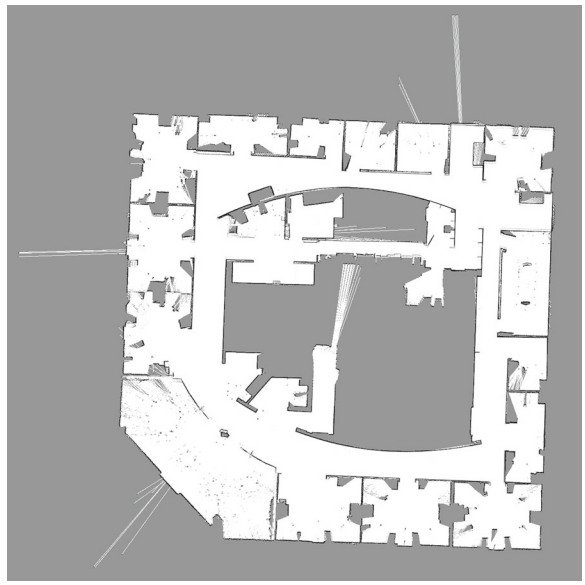
5 Experiments

In the current section the results from three different types of experiments are presented. The first type will be used to compare the execution time of the AGVD construction in two environments, with different characteristics, compared with the most common skeletonization technique, Thinning, as well as a tool for constructing GVDs hosted in OpenSlam webpage, evg-thin [22, 23]. Afterwards, the execution time of the topological graph extraction is presented, as well as the minimization of the graph and the reduction in the number of the total nodes. Finally, a new metric is introduced, effCov (effective coverage), in order to investigate if the proposed method of the topological graph construction leads to topological information loss (i.e. to omit nodes in the summation of the free space). That is the percentage of the free space covered by the victim identification sensors, given that the robot visits all the nodes of the graph. We assume that the robots victim sensors have an effective range of 2 m and each pixels side has length equal to 0.02 m. Environment 1 has dimensions of 1856×1024 pixels with 742.1 m^2 of free space and environments 2 size is 1392×1336 pixels with 732 m^2.

The experiments were performed in a PC with Intel Core i7 CPU at 2.80GHz, 4GB RAM, running Ubuntu 9.10 Karmic distribution. The described system is equipped with 4 cores and hyper-threading, but as the proposed algorithms have no parallel procedures, only one core was used.



(a) Environment 1



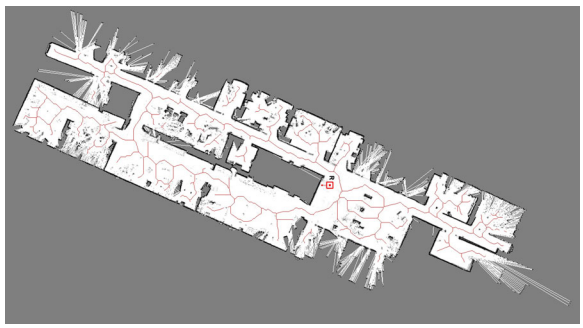
(b) Environment 2

Fig. 10 Experimental environments

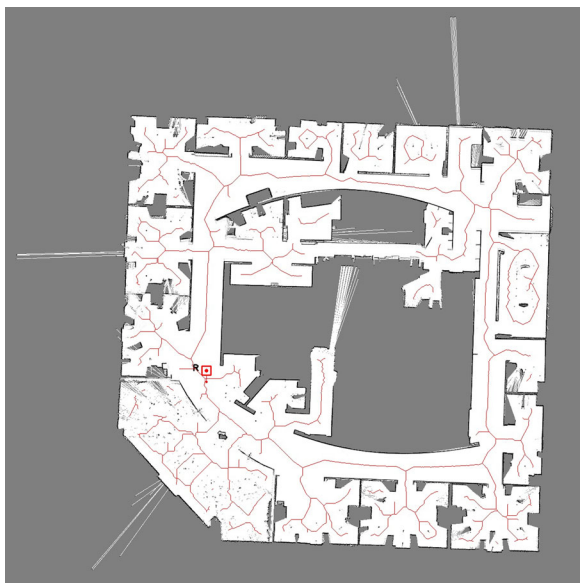
The two environments in which the experiments will be performed are illustrated in Fig. 10a and b.

The first one is the result of a SLAM procedure in an environment built in the open-source simulator USARSim, and is characterized by a simple corridor-like structure, with lack of distinct rooms. In contrast, the second environment is an occupancy grid map of the Intel lab in Seattle, characterized by increased complexity as it contains many corridors, junctions and rooms/offices.

The first set of experiments refers to the construction of the AGVD. Comparison is made between the proposed, the default method of Thinning and the open-source tool evg-thin. It must be noted that before executing the Thinning method it is essential to perform a series of erosion and dilation iterations in order to smoothen the foreground space, because of the irregularities inserted between the Laser Range Finder rays.

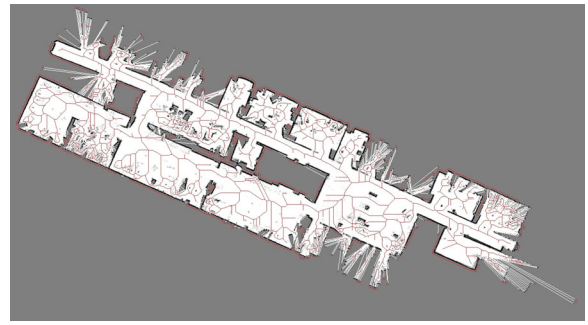


(a) AGVD of Environment 1 (Hybrid BrushFire)

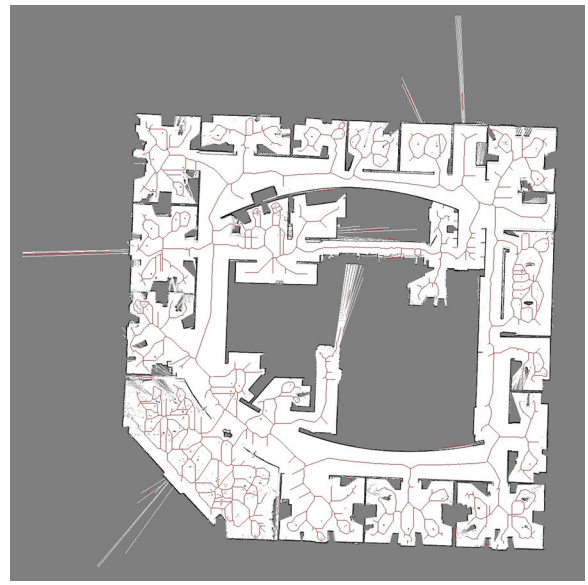


(b) AGVD of Environment 2 (Hybrid BrushFire)

Fig. 11 Produced AGVDs with the use of the hybrid BrushFire method



(a) AGVD of Environment 1 (Thinning)

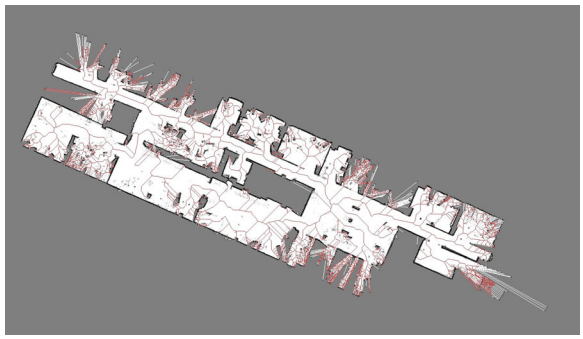


(b) AGVD of Environment 2 (Thinning)

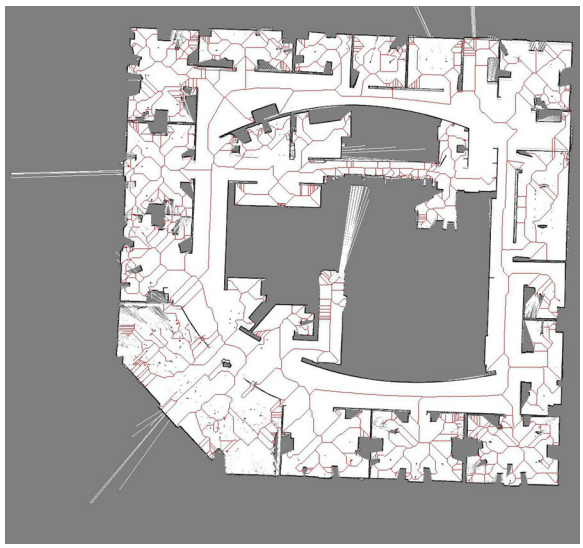
Fig. 12 Produced AGVDs with the use of the Thinning method

The resulting AGVDs of the two environments are illustrated in Fig. 11a and b for the proposed method, in Fig. 12a and b for Thinning, and Fig. 13a and b for the evg-thin output respectively.

The execution time of each algorithm and the extreme values for both environments are illustrated in Table 1, as the mathematical mean of 20 experiments.



(a) AGVD of Environment 1 (Evg-thin)



(b) AGVD of Environment 2 (Evg-thin)

Fig. 13 Produced AGVDs with the use of the Evg-thin software

Table 2 Graph minimization

Execution time		Environment 1	Environment 2
ms	Minimum	516.589	611.437
	Mean	525.27	623.63
	Maximum	549.711	653.896
Initial number of vertices		160	209
Final number of vertices		114 (−28,75 %)	124 (−40.66 %)

The results indicate that the proposed method manages to construct the AGVD, or the topological skeleton, of both environments, in a much lesser execution time than the Thinning method or the evg-thin software. As far as the quality of the produced AGVD is concerned, it can be seen that Thinning and evg-thin methods produce a much more detailed skeleton than the proposed method. The drawback of the exceedingly detailed skeletons is that they are prone to environment irregularities, examples of which are extremely small obstacles in the rooms like chairs, tables or even human feet. These comments are backed up by the experimental results concerning the vertices of the AGVD as well as its total length, where both of these metrics resulted in much higher values for the Thinning and Evg methods compared to the proposed method. Thus the topological graph can be very complicated al-

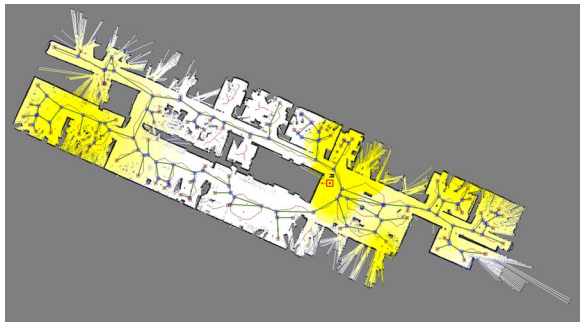
Table 1 Comparison of AGVD construction methods

Methods (ms)		Environment 1	Environment 2
Proposed	Minimum	456.76	414.756
	Mean	465.376	416.914
	Maximum	480.184	418.111
	Vertices AGVD	160	209
	Length AGVD (pixels)	11553	7415
Thinning	Minimum	6046.01	4049.96
	Mean	6076.14	4074.15
	Maximum	6136.6	4128.8
	Vertices AGVD	634	435
	Length AGVD (pixels)	18790	17991
Evg-thin	Minimum	3010.5	4368.5
	Mean	3019.5	4390.2
	Maximum	3021.8	4475.0
	Vertices AGVD	514	498
	Length AGVD (pixels)	22772	24965

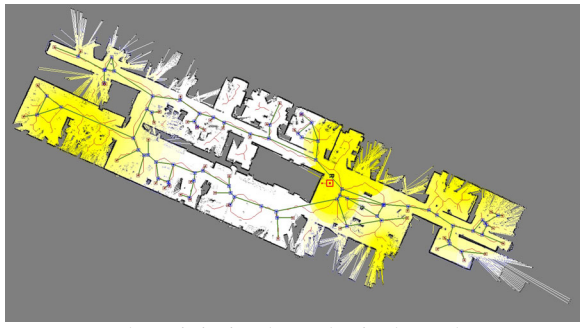
though its nodes are not valid topological features. In addition, the proposed method prohibits the construction of AGVD through very narrow passages, in contrast to the other methods, enabling the robot to perceive the valid routes it can follow.

The second type of experiments is the presentation of the execution time of the graph minimization procedure and the reduction percentage of the graph shown in Table 2. The coverage field and the initial topological graph of the two environments are illustrated in Figs. 14a and 15a. The final topological graphs are depicted in Figs. 14b and 15b.

Finally, the results of the effective coverage metric, are presented in Table 3, for all three AGVD methods applied to both environments. As the results of Table 3 indicate, our method manages to create a topological graph with much less nodes than Thinning and Evg-thin algorithms. Nevertheless, the topological information is not reduced, as the difference between the effective coverage percentages is trivial. Conclusively, the proposed method creates a sparser graph, which helps in the reduction of the planning algorithms complexity, but contains nodes in the ensemble of the free space, as the effective coverage values are close to 100 %.

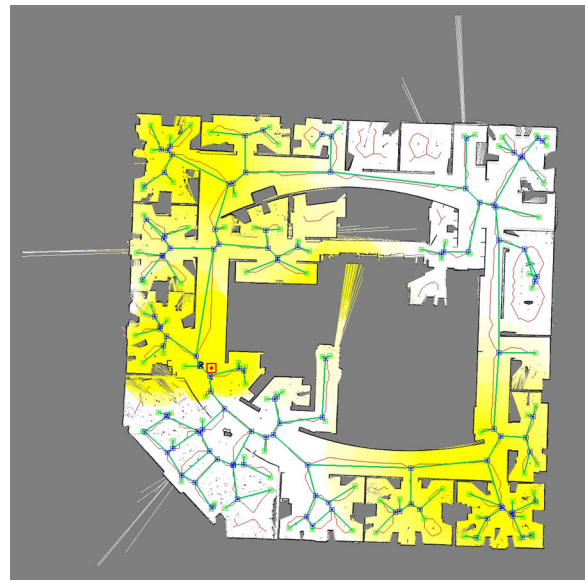


(a) Initial topological graph



(b) Minimized topological graph

Fig. 14 Minimization of the topological graph of Environment 1



(a) Initial topological graph

(b) Minimized topological graph

Fig. 15 Minimization of the topological graph of Environment 2

Table 3 Effective coverage

AGVD	Environment 1		Environment 2	
	Nodes (initial)	EffCov (%)	Nodes (initial)	EffCov (%)
Proposed	160	99.65	209	99.64
Thinning	634	100	435	99.96
Evg-thin	514	99.96	498	99.94

6 Conclusions—Further Work

The results indicate that the proposed method manages to construct the topological graph of the environments, and perform its minimization in a small amount of time, fact that leads us to state that the method can be efficiently used in robotic systems. It can also be noticed that the algorithm performs a satisfactory minimization of the graph, which is proportional to the amount of free space which is covered by the robots sensors. To conclude, this paper proposes a novel and fast method to construct a space skeletonization, as well as to extract topological information from it. In addition, the innovation of the coverage map, based on victim sensors, is introduced, which is afterwards used to minimize the topological graph, retaining the essential connected part, aiming at the total planning execution time reduction.

In conclusion, the contributions of this paper are, in the field of AGVD construction, the introduction of "parenthood" in the propagation of BrushFire algorithm, the contamination step and the important speed-up of the AGVD construction in comparison with known methods. In addition, a topological graph construction based on the AGVD and its minimization based on coverage information is presented.

Future work may include the improvement in execution time of the proposed methods. Also the actual planning method may be implemented, based on the reduced topological graph constructed by this paper. Finally, some of the algorithms can be parallelized, such as the construction of the AGVD, or the graph minimization by computing all the minimum paths.

References

1. Fortune, S.: A sweepline algorithm for Voronoi diagrams. In: Proc. of the 2nd Annu. Symposium on Computational geometry, pp. 313–322. ACM Press, New York, 2–4 June 1986
2. Mohammadi, S., Hazar, N.: A Voronoi-based reactive approach for mobile robot navigation. In: Advances in Computer Science and Engineering, Communications in Computer and Information Science, vol. 6, part 2, pp. 901–904 (2009)
3. Ahn, S., Doh, N.L., Chung, W.K., Nam, S.Y.: The robust construction of a generalized Voronoi graph (GVG) using partial range data for guide robots. Ind. Robot. Int. J. **35**(3), 259–265 (2008)
4. Telea, A., van Wijk, J.J.: An augmented fast marching method for computing skeletons and centerlines. In: Proc. Symposium on Data Visualisation, pp. 251–259 (2002)
5. Rumpf, M., Telea, A.: A continuous skeletonization method based on level sets. In: Proc. Symposium on Data Visualisation, pp. 151–158 (2002)
6. Lihong, M., Yinglin, Y., Yu, Z.: A skeletonization algorithm based on Euclidean distance maps and morphological operators. J. Electron. **18**(3), 272–276 (2001)
7. Kalra, N., Ferguson, D., Stentz, A.: Incremental reconstruction of generalized Voronoi diagrams on grids. Robot. Auton. Syst. **57**(2), 123–128 (2009)
8. Boris, L., Sprunk, C., Burgard, W.: Improved updating of Euclidean distance maps and Voronoi diagrams. In: 'IROS', pp. 281–286. IEEE (2010)
9. Garrido, S., Moreno, L., Blanco, D.: Exploration of 2D and 3D Environments using Voronoi transform and fast marching method. J. Intell. Robot. Syst. **55**(1), 55–80 (2009)
10. Cheong, H., Park, S., Park, S.K.: Topological map building and exploration based on concave nodes. In: International Conference on Control, Automation and Systems (ICCAS), pp. 1115–1120. IEEE (2008)
11. Poncela, A., Perez, E.J., Bandera, A., Urdiales, C., Sandoval, F.: Efficient integration of metric and topological maps for directed exploration of unknown environments. Robot. Auton. Syst. **41**(1), 21–39 (2002)
12. Choset, H., Nagatani, N., Rizzi, A.: Sensor based planning: using a honing strategy and local map method to implement the generalized Voronoi graph. In: SPIE Conference on Systems and Manufacturing Mobile Robots XII, pp. 72–83 (1997)
13. Kim, J., Zhang, F., Egerstedt, M.: A provably complete exploration strategy by constructing Voronoi diagrams. Auton. Robot. **29**(3–4), 367–380 (2010)
14. Tao, T., Tully, S., Kantor, G., Choset, H.: Incremental construction of the saturated-GVG for multi-hypothesis topological SLAM. In: Proc. IEEE

International Conference on Robotics & Automation, pp. 3072–3077 (2011)
15. Ko, B.Y., Song, J.B., Lee, S.: Real-time building of a thinning-based topological map with metric features. In: Intelligent Robots and Systems (IROS), pp. 1524–1529 (2004)
16. Shi, L., Kodagoda, S.: Towards generalization of semi-supervised place classification over generalized Voronoi graph. Robot. Auton. Syst. **61**(8), 785–796 (2013)
17. Narayanan, V., Vernaza, P., Likhachev, M., LaValle, S.M.: Planning under topological constraints using beam-graphs. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (2013)
18. Gonzalez, R.C., Woods, R.E.: Digital Image Processing, 2nd edn. Addison-Wesley Longman Publishing Co., Inc., Boston, MA (1992)
19. Mingas, G., Tsardoulias, E., Petrou, L.: An FPGA implementation of the SMG-SLAM algorithm. Microprocess. Microsyst. **36**(3), 190–204 (2012)
20. Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., Thrun, S.: Principles of robotic motion: theory, algorithms, and implementation, MIT Press, Cambridge, MA (2004)
21. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. **4**(2), 100–107 (1968)
22. Zhang, T.Y., Suen, C.Y.: A fast parallel algorithm for thinning digital patterns. Commun. ACM **27**(3), 236–239 (1984)
23. Beeson, P., Jong, N.K., Kuipers, B.: Towards autonomous topological place detection using the Extended Voronoi Graph. In: Proc. of the International Conference on Robotics and Automotation (ICRA), pp. 4373–4379. IEEE (2005)