PolyMap: A 2D Polygon-based Map format for Multi-Robot Autonomous Indoor Localization and Mapping

Johann Dichtl, Luc Fabresse, Guillaume Lozenguez, and Noury Bouraqadi

IMT Lille-Douai, 59500 Douai, France

Abstract. Autonomous exploration is an important tasks in many robotic fields such as disaster response scenarios. In time critical situations, the use of multiple robots can reduce the time to create a complete map of the environment. However among the most popular map formats in use today, none are ideal for the multi-robot autonomous indoor localization. In terms of memory usage, visualization, and usability in navigation and exploration tasks, all formats have some strengths and weaknesses.
In this paper we introduce PolyMap, a map format that is based on *simple* polygons. Since the polygons are based on line segments, this is a special case of vector-based map formats. This format provides advantages in terms of memory footprint over occupancy grids, while not falling behind in visualization. Its sparse nature is also an advantage for navigation tasks, in particular when the map needs to be shared over a wireless network connection. Additionally the explicit modeling of frontiers helps with autonomous exploration.

Keywords: Vector Maps, Indoor Mapping, Exploration, Multi-Robot Systems.

1 Introduction

In many indoor applications, robots have to autonomously map an environment without human assistance. For example, fire fighters can use robots to build a map before entering a building that caught fire. Rapid exploration is critical in such scenarios to minimize damage and maximize survival chances of victims. This is why it is interesting to rely on multiple robots to explore the environment and collaboratively build a map.

Currently popular 2D map formats have various shortcomings for this tasks. Occupancy grids, the most commonly used format, have a large memory footprint, which is particularly limiting when relying on wireless network connectivity to share maps between robots. Feature-based maps tend to not model the shape of obstacles, making navigation and visualization for human use difficult or impossible. Furthermore they don't define frontiers, making autonomous exploration more difficult. Vector-based map formats also don't have frontiers, and often have gaps between vectors, making it impossible to clearly distinguish explored space from unknown space.

In this paper, we investigate 2D map formats that are the most appropriate for such applications. Starting from our reference scenario we draw a list of requirements (Section 2), that exhibit the shortcomings of the state of existing map formats (Section 3).

We then introduce PolyMap (Section 4), a 2D map format that represent the environment as a collection of polygons. We also evaluate PolyMap by showing how it addresses our requirements. The paper ends with a conclusion (Section 5) that summarizes our contributions and sketches some future work.

2 Requirements

To speed up mapping by using a robotic fleet, robots have to spread and explore different parts of the environment. This means that they have to somehow decide which one goes where. This decision relies on *frontiers* [14] between explored space and the unknown one. Each reachable frontier is assigned to one or more robots. Robots then navigate towards their respective target frontiers, which lead to exploring new areas of the environement. Each robot then shares its local map with others to build a bigger map gathering all explored areas. The list of frontiers is updated and again assigned to robots. This process is repeated until we get a map with no reachable frontier left.

From the above description, we can infer that a map format for autonomous exploration and mapping should meet the following requirements:

Explicit Exploration Frontiers. Explicit frontiers enable collaboration. They materialize the exploration tasks assigned to each robot. This is why the map format should allow representing frontiers.

Support for Path Planning. To evaluate frontiers' reachability, as well as the cost to reach them, robots should be able to perform path planning. The map format should then unambiguously distinguish free navigable areas from obstacles.

Lightweight. To cover large areas, robotic fleets can include many robots, possibly dozens. All these robots need to share their local maps over a wireless network, to achieve collaborative exploration. The map format should be lightweight memory wise to save network bandwidth and support large fleets of robots.

Visualization for Human Use. In many scenarios, a human supervises the robots' mission or uses the built map to perform some task. For example, responders might use the map built by robots to go rescue trapped humans. The map format should then be usable by humans to identify different areas of the explored environment.

3 State of the Art

The purpose of a map varies from use case to use case. Therefore it is of no surprise, that different map formats emerged to satisfy different use cases. The two major classifications that are of interest to us are *metric* and *topological* map formats. Metric maps model geometric information of the environment such as obstacles and free space, in some form of coordinates that allow to measure distances. Topological maps model connectivity between different locations. As shown in Figure 1, metric maps can be further split into three sub-groups: occupancy grids, feature-based, and vector-based.

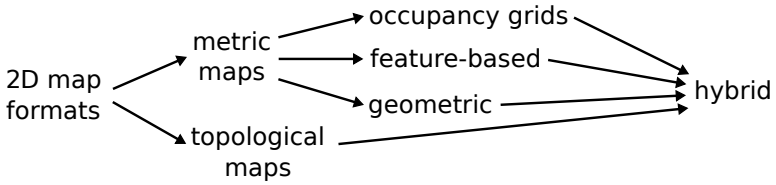

Fig. 1. Map formats families

Occupancy Grid. The dominating 2D map format is the *occupancy grid*, also called *grid map* [12]. An occupancy grid is a matrix, where each cell contains the estimated probability that the space it represents is traversable. In this context *traversable* only means that the space has been observed (i.e. it is not unknown space) and is not occupied by an obstacle. This propability is typically quantized into three possible states: traversable/free, occupied, and unknown/unexplored.

Figure 2 shows an example of a three-state grid map. This quantization reduces the required resources to create the map and distribute it in a network.

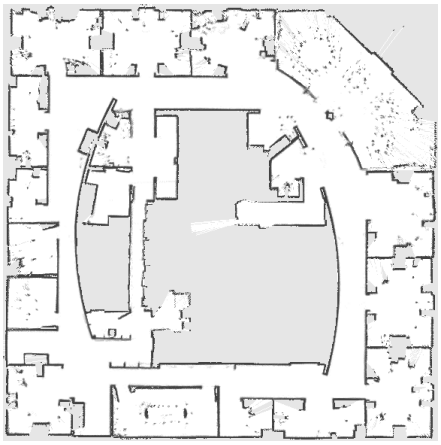

Fig. 2. A 3-state occupancy grid created from the *Intel Research Lab* data set. The cell states are: free (white), unexplored (gray), and occupied (black). Image and data set source: http://www2.informatik.uni-freiburg.de/~stachnis/datasets.html

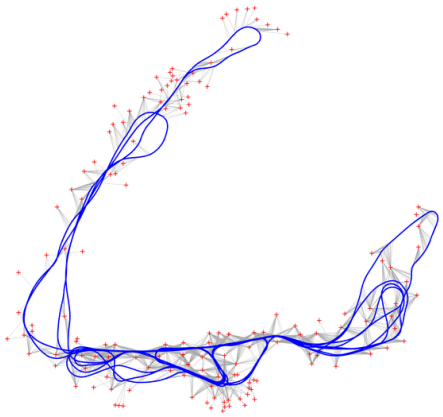

Fig. 3. An example of a landmark-based map from the *Victoria Park* data set. The red crosses show the detected landmarks, while the robot's estimated trajectory is displayed in blue. The black lines indicate at which position (i.e. which keyframe) the landmark was observed. Image source: https://sourceforge.net/p/slam-plus-plus/wiki/Compiling%20and%20running/?version=14

This format is easy to visualize for human use (as seen in Figure 2), and can also be used for navigation purposes. This makes the map format very versatile. It is how-

ever relatively expensive in terms of CPU and memory requirements, which limits the maximum size of the created maps. The map resolution (i.e. how big a single cell is) influences both resources as well. The overall size of the matrix storing the grid is limited by the hardware, and if it needs to be shared between multiple robots/computers, by the available network bandwidth.

Topological information is embedded only implicitly, by treating the grid as a big graph. Here we have edges between two neighboring cells if and only if both cells are marked as traversable space. This results in a relatively large graph that is not optimized for navigation tasks. Therefore, occupancy grids are only used for navigation tasks in relatively small environments, e.g. in an office building.

Feature-based Maps. The second map format familiy is the *feature-* or *landmark-based* map format. This map format stores distinct features or landmarks of the environment (e.g. corners or artificial beacons) with their relative position. For example Castellanos et al. [2] uses line segments as features. Unlike the *occupancy grid*, this is a sparse representation of the environment (see Figure 3).

The advantages of this format stem from the sparse nature of the landmarks, typically translating into significantly smaller memory footprints and computation power. This in return allows larger areas to be covered by this map format before being limited by the computational hardware.

The main disadvantage of this format is, that the transition from traversable space to unexplored space is not modeled. In particular, frontiers are not part of the map format, so frontier-based exploration is not possible. Furthermore, features don't necessarily model obstacles, even though some examples exist that use vectors as features, e.g. [2,8,10]. If visualization is meaningful with this map format depends on whether the chosen feature models obstacle shapes.

Vector Maps. The third and currently least used map format family of the three is a parametric representation of the environment. Parametric in this context refers to parametric curves (splines and bezier curves), and line segments.

In the context of SLAM line segments are often also called vectors. They are the only ones used for SLAM [5], since they are significantly easier to handle for computations such as collision control.

Line segment based maps model the borders between traversable space and obstacles via line segments. This representation has been used since early robotic Localization and Mapping research [3]. The resulting maps is very usable by humans, since it allows them to identify obstacles in the environment.

Compared with occupancy grids, line segment based maps are sparse in nature, since only the boundaries are explicitly modeled. This sparse nature makes vector map lightweight and hence very appropriate for sharing over wireless.

Existing vector-based map formats [4,11,6,8,10,9,1] suffer from two major limitations. First, they lack a representation of frontiers between explored and unexplored spaces. Besides, the resulting map is not always navigable because of possible gaps between vectors representing boundaries of the same obstacle. A notable exception is the map format of TvSLAM [4], where segments delimiting a given obstacle are always

connected. However, likewise all other vector-based format, TvSLAM maps don't distinguish between unexplored areas from explored traversable ones.

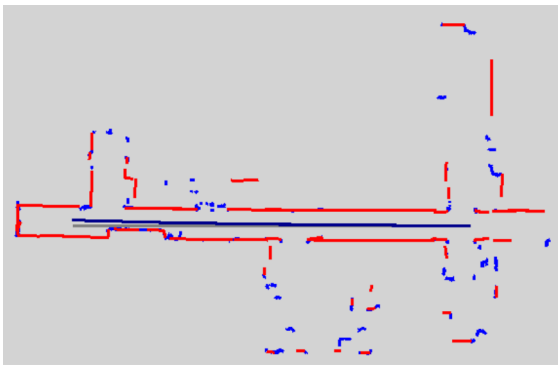

Fig. 4. A small vector map created with edge-extraction from point clouds. Long vectors are colored red, shorter vectors are blue. Source: [6].

Hybrid Maps. Hybrid map formats, as indicated in Figure 1, are a mix of two or more maps formats. They are not explicitly included in Table 1, because their properties can be deduced from the type of maps that they are composed of. Two examples for hybrid map formats are: Ravankar et al. [10] and Lv et al. [8], both combining feature-based and vector-based maps.

Summary. Table 1 summarizes our evaluation of map format families with regard to requirements of Section 2.

Table 1. Evaluation of Existing Map Formats

	occupancy grids	feature-based	vector-based
Exploration Frontiers	Yes	No	No
Path Planning	Yes	Not Always[1]	Not Always[2]
Lightweight	No	Yes	Yes
Visualization	Yes	Not Always[3]	Yes

[1] The map needs to embed topological information to support path planning since features alone typically are not sufficient to model traversable space.
[2] Path planning can be challenging if there are gaps between vectors.
[3] In general, feature-based maps don't model obstacle shapes, and are not suitable for visualization. However exceptions exist, e.g. [8,10]

Explicit Exploration Frontiers. Occupancy grids make it easy to make frontiers explicit. Unexplored space is well differentiated from free space and obstacles. Thus, frontiers are *free* cells that have neighboring *unexplored* cells. The classic feature-based map format does not contain any frontiers. Vector-based map formats could model frontiers via vectors that are marked as such, but we are currently not aware of any implementations that make use of this.

Support for Path Planning. Typically, the first step for path planning is to create a topological graph of the environment. Building this graph from an occupancy grid is technically not needed since the format already implicitly contains such a graph: each free cell represents a node in the graph, and two neighboring cells are connected if both are free. This method however is relatively expensive in terms of computational and memory requirements for path finding, since the graph is relatively large. Hence, often a dedicated topological map is created from the metric map instead. Feature based maps do not allow to create topological maps. Vector maps do not allow to build topological maps. One reason are the gaps between the vectors (as shown for example in Figure 4). Path planning is also challenging with vector maps, since there is no clear distinction between free space and unexplored space.

Lightweight. Occupancy grids have the highest memory requirements among the discussed formats. The memory depends only on the size of the area covered by the map and the chosen resolution. It does not depend on the number of obstacles, or the shape of the obstacles. Feature-based maps have a low memory consumption which correlates with the number of features in the environment/map. As a result, the memory consumption can be altered by choosing a different feature detector, or tweaking the parameters of the selected feature detector. Vector-based map formats are sparse representation of the environment, and as such have low memory requirements. Furthermore, the number of line segments in both formats depends on the amount of obstacles (and their shape), but not on the size of the environment.

Visualization for Human Use. All presented map formats except feature-based maps are considered suitable for visualization. For feature-based maps, it depends on the type of feature: those that do model the position and shape of obstacles are suitable for visualization, the rest is not. Occupancy grids can easily be exported as bitmaps (typically in the *portable network graphic* (PNG) format), vector maps are either rasterized as bitmaps or exported in a vector format such as *scaleable vector graphics* (SVG). Either provide an easy-to-understand top-down-view of the covered area.

4 PolyMap

Looking at the previous section, we can see that vector-based map format is promising. Vector-based maps are both lightweight and appropriate for human use. In this section, we introduce PolyMap, a vector-based map format that addresses the two limitations of vector maps. We show that PolyMap is appropriate for path planning and for exploration.

4.1 Polygon-based Map format

PolyMap represents the environment using simple polygons. *Simple* polygons are polygons that consist of non-intersecting line segments that are joined pair-wise to form a closed path [13].

Formally, a polygon-based map M is defined as:

$$type: \quad T = \{border, frontier\} \tag{1}$$

$$vector: \quad V = \{(x_i, t) | 1 \leq i \leq 2; x_i \in \mathbb{R}^2; x_1 \neq x_2\} \tag{2}$$

$$polygon: \quad P = \{v_i | i \geq 3; i \in V\} \tag{3}$$

$$map: \quad M = \{p_i | 1 \leq i \leq n; p \in P\} \tag{4}$$

with P forming a *simple* polygon, and n being the number of polygons in the map.

We define two types of line segments :

Obstacles: they represent the outline of obstacles.
Frontiers: they model the transition from free/traversable space to unknown/unexplored space.

Every line segment has exactly one type, polygons can contain line segments of different types. The direction of line segments is chosen, so that clockwise oriented polygons contain traversable space inside. Figure 5 shows an example of map containing two different types of line segments: red line segments represent obstacles and green line segments represent frontiers. A map is composed of a single (clockwise oriented) large polygon that encompasses the entire explored area. Inside this polygon are typically multiple other polygons with a counter-clockwise orientation, modeling obstacles.

Using directed line segment like this, has the advantage that it is easy to determine whether a given point is inside free or unexplored space; a property that feature-based and vector-based map formats are missing. For this, we only need to find the closest line segment to the point and look at which side of the polygon the point is located. Figure 6 illustrates this with two points: one is inside the explored area and the other is outside. This not only helps for visualization, but also allows localization with particle filters to discard particles that are inside obstacles or otherwise in unexplored space.

4.2 Evaluation of PolyMap

PolyMap does address all the requirements presented in Section 3. Being based on vectors grouped into polygons, a PolyMap has by definition the advantages of vector-based map format. It is lightweight and can be easily visualized by humans. The remaining of this section first discusses how PolyMap meets the two other requirements and then how to build polygon-based maps.

Explicit Exploration Frontiers. The types of line segments of a polygon are either obstacles or frontiers. Outside a polygon is unexplored space, while inside represents the explored space. Passages between the two spaces, if any, are represented thanks to frontier segments.

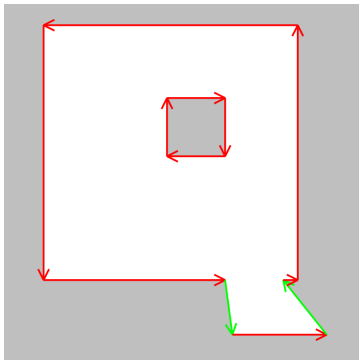

Fig. 5. An example of a map consisting of only simple polygons. Obstacles are red and frontiers are green.

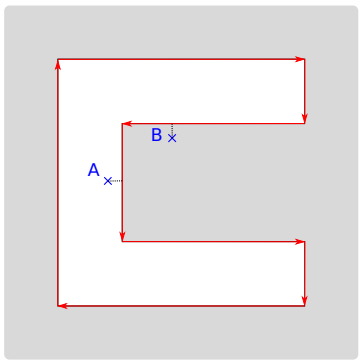

Fig. 6. Two points, one inside free space, the other in unexplored space.

Support for Path Planning. PolyMaps are suitable for navigation because polygons separate traversable areas and unexplored ones. The sparse nature of this map format makes it easy to create a topological graph using visibility graphs, or random sampling points in traversable space [7]. Such topological graphs are then suitable for navigation and path planning.

How to build a PolyMap? To use PolyMaps in practice, we need to build such maps. One idea is to build it from an existing occupancy grid map. Doing so can be achieved in several ways, for example by extending the approach of Baizid *et al.* [1] to form closed polygons. Another possibility would be to utilize a wall-following strategy to outline the borders of traversable space, creating closed polygons in the process. Yet another method is to create a vector for every transition from free space into non-free space, and build polygons from the collected vectors. We used this idea in our implementation (programmed in Pharo[1]) to build PolyMaps from grid maps. The pseudo-code for this is shown in Algorithm 1.

Figure 7 shows the result on applying our algorithm to convert a grid map of the Intel Lab data set[2] to a PolyMap. Obstacles are shown with red vectors and frontiers with green vectors. The resulting PolyMap in this example consists of 829 polygons, with a total of 18054 vectors. The right-most figure is a zoomed-in section of the PolyMap, displaying individual vectors. Interestingly, the size of the full PolyMap is approximately the size of the compressed grid map. In this particular example, the grid map (in PNG format) takes about 100kB, and the uncompressed PolyMap requires about 160kB. Compressed, the PolyMap shrinks to approximately 44kB (ZIP[3]) and 20kB (7z[4]). Nevertheless, this algorithm only creates either horizontal or vertical vectors.

[1] https://pharo.org/
[2] http://ais.informatik.uni-freiburg.de/slamevaluation/datasets.php
[3] ISO/IEC 21320-1:2015, https://www.iso.org/standard/60101.html
[4] https://www.7-zip.org/

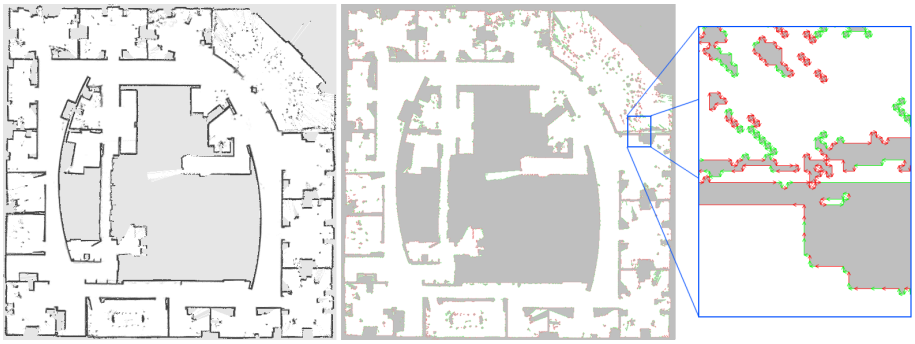

Fig. 7. Occupancy grid (left), full converted PolyMap (middle) and zoom in the PolyMap (right)

Further optimization that allows diagonal vectors could result in even smaller memory footprints of the maps. Such optimizations will be considered in future work.

Multi-robot exploration using PolyMaps. Being a lightweight map format with a sparse representation of the environment, PolyMaps have low network bandwidth requirements. In multi-robot exploration scenarios, this allows us to use more robots, cover larger areas, or exchange maps more frequently than when compared with occupancy grids. Also, due to the sparse nature of the map format combined with explicit frontiers, frontiers can be selected easily and assigned to individual robots as exploration goals. Priority can be based on size of the frontier (i.e. length of the vectors) and proximity to the robot. The process of merging multiple PolyMaps from different robot is an interesting topic for future work.

5 Conclusion

Polygon-based map formats provide advantages by employing a sparse representation of the environment while still modeling obstacles with an accuracy that is comparable to occupancy grids. The reduction in memory consumption allows to cover larger areas compared to occupancy grids, and helps to do deal with the bandwidth bottleneck when broadcasting the map over a network. This is particularly helpful if maps need to be distributed within a robot fleet.

Explicitly modeling frontiers and enforcing closed polygons makes the maps much more useful for navigation and exploration tasks. Using directed line segments also makes it easy to determine whether a given point in located in free or unexplored space.

For future work, we plan to implement a full SLAM application that utilizes the PolyMap format and to test it in autonomous exploration scenarios. We further want to use the PolyMap format in multi-robot systems to rapidly explore an environment with a fleet of robots. This includes merging PolyMaps that have been created bu different robots in the fleet.

Acknowledgment

This work is part of the CPER DATA project that is supported by Région Hauts de France, and the French state.

References

1. Baizid, K., Lozenguez, G., Fabresse, L., Bouraqadi, N.: Vector Maps: A Lightweight and Accurate Map Format for Multi-robot Systems. In: International Conference on Intelligent Robotics and Applications. pp. 418–429. Springer International Publishing (2016)
2. Castellanos, J.A., Montiel, J., Neira, J., Tardós, J.D.: The spmap: A probabilistic framework for simultaneous localization and map building. IEEE Transactions on Robotics and Automation **15**(5), 948–952 (1999)
3. Chatila, R., Laumond, J.P.: Position referencing and consistent world modeling for mobile robots. In: Robotics and Automation. Proceedings. 1985 IEEE International Conference on. vol. 2, pp. 138–145. IEEE (1985)
4. Chen, Y., Qu, C., Wang, Q., Jin, Z., Shen, M., Shen, J.: Tvslam: An efficient topological-vector based slam algorithm for home cleaning robots. In: International Conference on Intelligent Robotics and Applications. pp. 166–178. Springer (2017)
5. Group., I.R.M.D.R.W.: Ieee standard for robot map data representation for navigation, sponsor: Ieee robotics and automation society. (June 2016), http://standards.ieee.org/findstds/standard/1873-2015.html
6. Jelinek, A.: Vector maps in mobile robotics. Acta Polytechnica CTU Proceedings **2**(2), 22–28 (2015)
7. LaValle, S.M.: Planning Algorithms. Cambridge University Press, New York, NY, USA (2006)
8. Lv, J., Kobayashi, Y., Ravankar, A.A., Emaru, T.: Straight line segments extraction and ekf-slam in indoor environment. Journal of Automation and Control Engineering **2**(3) (2014)
9. Pfister, S.T., Roumeliotis, S.I., Burdick, J.W.: Weighted line fitting algorithms for mobile robot map building and efficient data representation. In: Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on. vol. 1, pp. 1304–1311. IEEE (2003)
10. Ravankar, A., Ravankar, A.A., Hoshino, Y., Emaru, T., Kobayashi, Y.: On a hopping-points svd and hough transform-based line detection algorithm for robot localization and mapping. International Journal of Advanced Robotic Systems **13**(3), 98 (2016)
11. Sohn, H.J., Kim, B.K.: VecSLAM: An Efficient Vector-Based SLAM Algorithm for Indoor Environments. Journal of Intelligent and Robotic Systems **56**(3), 301–318 (2009). https://doi.org/10.1007/s10846-009-9313-2, http://dx.doi.org/10.1007/s10846-009-9313-2
12. Stachniss, C., Grisetti, G., Hähnel, D., Burgard, W.: Improved rao-blackwellized mapping by adaptive sampling and active loop-closure. In: In Proc. of the Workshop on Self-Organization of AdaptiVE behavior (SOAVE) (2004)
13. Toussaint, G.: Efficient triangulation of simple polygons. The Visual Computer **7**(5-6), 280–295 (1991)
14. Yamauchi, B.: A frontier-based approach for autonomous exploration. In: Proceedings of CIRA'97 (1997)

Data: occupancy grid map G
Result: PolyMap M
create dictionary D
`/* create vectors */`
foreach *cell* $c \in G$ **where** *c is free space* **do**
  foreach *neighbor n of c* **where** *n is not free space* **do**
    `/* vector orientation is CCW with respect to c */`
    if *n is obstacle* **then**
      create vector v at border of c and n of type obstacle
    else if *n is unexplored* **then**
      create vector v at border of c and n of type frontier
    end
    add vector v to dictionary D with v start point as key
    in case of two vectors sharing the same start point, both are stored alongside
  end
end
`/* create polygons */`
while D *is not empty* **do**
  create empty polygon P
  take random vector v from D
  remove v from D
  w := v
  while w *end point* $\neq$ v *start point* **do**
    P add w
    w := D at key w end point
    remove w from D
  end
  M add P
end
`/* aggregate vectors */`
foreach *polygon* $P \in M$ **do**
  foreach *vector* $v \in P$ **do**
    w := vector in P **where** w start point = v end point
    if w *orientation* = v *orientation* **and** w *type* = v *type* **then**
      set v end point to w end point
      remove w from P
    end
  end
end

Algorithm 1: How to build a PolyMap from a grid map