# Vector Maps: A Lightweight and Accurate Map Format for Multi-robot Systems

Khelifa Baizid, Guillaume Lozenguez, Luc Fabresse, Noury Bouraqadi

# Vector Maps: a Lightweight and Accurate Map Format for Multi-Robot Systems

Khelifa Baizid, Guillaume Lozenguez, Luc Fabresse, and Noury Bouraqadi *

Mines Douai, Uni. Lille, France
firstName.lastName@mines-douai.fr
http://car.mines-douai.fr

**Abstract.** SLAM algorithms produce accurate maps that allow localization with typically centimetric precision. However, such a map is materialized as a large *Occupancy Grid*. Beside the high memory footprint, *Occupancy Grid Maps* lead to high CPU consumption for path planning. The situation is even worse in the context of multi-robot exploration. Indeed, to achieve coordination, robots have to share their local maps and merge ones provided by their teammates. These drawbacks of *Occupancy Grid Maps* can be mitigated by the use of topological maps. However, topological maps do not allow accurate obstacle delimitations needed for autonomous robots exploration. So, robots still have to handle with *Occupancy Grid Maps*. We argue that *Vector-based Maps* which materialize obstacles using collections of vectors is a more interesting alternative. *Vector Maps* both provide accurate metric information likewise *Occupancy Grid Maps*, and represent data as a graph that can be processed for path planning and maps merging as efficiently as with topological maps. Conclusions are backed by several metrics computed with several terrains that differ in size, form factor, and obstacle density.

## 1 Introduction

Metric SLAM attempts to model a given environment typically in terms of *Occupancy Grids* [1] They discretise the environment as a collection of cells of fixed size. Their size grows with the size of the mapped environments and the precision of range sensors. Beside memory overhead, metric maps are also resource intensive both in terms of CPU, and network bandwidth. Path planning based on metric maps is computationally intensive, even in environments with few obstacles. The larger is the map the more CPU is required for computing a path.

Additionally, more drawbacks of metric maps, based on *Occupancy Grids*, appears in the context of Multi-Robot Systems (MRS) that collaboratively explore unknown environments.

Collaboration requires robots to exchange local maps and merge them to build a global map faster Figures 1). However, the larger is the map, the larger

* This work was supported by Région Nord Pas-de-Calais as part of the SUCRé project (Human-robots cooperation in hostile environment).

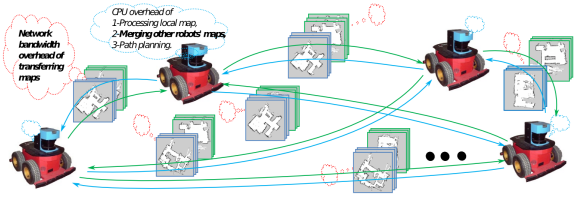



**Fig. 1.** Multi-robot exploration requires exchanging and merging maps.

is the network traffic to transfer it. Given that $required\_bandwidth(kbps) = transferred\_data\_size(kb)/time(s)$, in case of many large maps transferred simultaneously, the required bandwidth may exceed the network capacity.

Besides, map fusion is also computationally expensive. The larger is the robotic fleet, the more maps will be merged by each robot. As a result the amount of required CPU for map fusion can quickly become significantly high.

An alternative mapping approach relies on topological maps [2,3].A topological map can be modeled as a graph, where nodes denote locations in the environment, while edges connect nodes that are adjacent locations. Such a graph requires much less memory than an *Occupancy Grid*. It also requires less CPU for path planning and map fusion when a fleet of robots performs a complex coordination [4]. Indeed, the size of a topological map is independent from the size of the terrain. However, pure topological maps are unsuitable for localization or navigation because of their limited metric information. Thus, topological maps cannot always replace metric maps in heterogeneous multi-robot fleets context and they include metrical information.

In the context of multi-robot exploration, an ideal map should:

- provide metric information that accurately model the environment;
- minimize the CPU overhead for path planning;
- minimize the CPU overhead for map fusion;
- minimize the memory footprint;
- minimize the network bandwidth overhead for map sharing.

We focus on 2D maps produced out of laser range sensor scans. Our long-term goal is to build a SLAM algorithm that directly produces an ideal map from laser scans. But, before developing such an algorithm, we need first to identify a map format that meets requirements listed above.

In this paper, we investigate the use of *Vector-based Maps*[1] as an ideal format for maps. We have chosen the vector maps because they exhibit the advantages of both topological and metric maps. A *Vector Map* materializes obstacles as a collection of line segments. Such a map occupies significantly less memory (typically Kilobytes) than a metric map (typically Megabytes). As a result, transmitting a *Vector Map* over the network consumes less bandwidth. A map with smaller size

[1] We use *Vector-based Map* and *Vector Map* interchangeably throughout the paper.

and organized as a graph structure also drastically reduces CPU consumption for both path planning and map fusion.

In the following, we first give a brief overview of state of the art (Section 2). Then, we introduce the vector-based map format (Section 3). We show how to convert a *Vector Map* from and to a metric map. Next, we introduce a selection of metrics to compare vector and metric maps (Section 4). We then use these metrics to evaluate *Vector Maps* built for several terrains that differ in size, form factor, and obstacle density (Section 5). Our *Vector Maps* are built out of *Occupancy Grids* maps generated with the Karto SLAM algorithm [5]. Last, we draw conclusions and sketch some future works (Section 6).

## 2 State of the Art

The most used metric map consist in *Occupancy Grids* [6], where the environment is represented in terms of occupied cells for obstacles and unoccupied cells for free spaces. However, they are extremely expensive in terms of memory size. Moreover, they are unsuitable for data associations which implies a huge CPU load [7]. CPU overhead is even worse in multi-robot exploration missions, since robots have to merge maps. This makes it very expensive to compute complex coordination. Deliberative approaches are limited to sequential frontier attribution that requires efficient communications.

It is possible to reduce the size of the occupancy memory of metric maps by proposing an approach of compression using RANSAC map matching and sparse coding as building blocks [8]. Authors claim that the proposed approach performs well in terms of compression ratio, speed and retrieval performance of compressed/decompressed maps. While this compression reduces network overhead, the path planning and map fusion still require dealing with the full size *Occupancy Grid*. A relevant example of reducing the map size, applied in 3D environment, the one given by Armin et all [9]. The approach was based on Octrees tree data structure, where the space is partitioned by recursively subdividing it into eight octants. In fact, the structure is a tree of 3D spaces (e.g., parallelepiped). This approach proves to be efficient and it is used most in 3D graphics and 3D game engines, however, it explicitly represents free volumes in the tree. Moreover, it is expected to have computational complexity when the map precision is high and/or the environment is unstructured. In this paper, we compare vector maps to compressed occupancy grid maps (sparse matrices). This gives insights on similar other approaches such as Octree maps (in 2D space) and the RANSAC approaches.

Topological approaches [2,3] in map building have been considered as a great alternative to the metric approaches to reduce memory footprint. Moreover, they help to speed up path planning [10] and allow a fleet of robots to handle complex coordination as task allocation in punctual distributed path planning phases [4]. However, metric environment representation is usually mandatory for modeling the navigable free space and for localization. This leads to a new trend in SLAM algorithms to have both representations (i.e. topological and metric) [3, 11].

Hybrid metric-topological approaches take advantage from topological relations allowing to use algorithms based on graph theory and metric representation, which come with euclidean mathematics. A generic interface toward hybrid metric-topological SLAM was proposed by Blanco et al. [10]. The algorithm builds a graph of *Occupancy Grid Maps*. *Occupancy Grids* are nodes modeling local areas and edges materialize as topological transformation between *Occupancy Grids*. Such a representation is well suited for large-scale environments mapping and exploration. Similarly, Jongwoo et al. [12] proposed a vision-based mapping of large-scale environments. The approach builds a hybrid representation of the environment of topological and metric maps. The approach relies on optimizing the global map (topological), while it maintains the metric property of the local map. Authors claim that loop closures issues tackled in [10] are handled very efficiently, by benefiting from topological representation. Other alternatives hybrid metric-topological are proposed using collection of range scans [7] or vector-based features [13] but there are not deeply investigated. These approaches permit to model the surrounding environment using a graph structuring and positioned objects with relations between them. In [13] objects are frames of vector-based obstacle delimitation with distance and uncertainty relations. It implements the Extended Kalman Filter (EKF) to estimate the maps and robot poses.

While we appreciate benefits from these approaches, we believe that they rely on a heavy structure generally based on one large or several *Occupancy Grids* with the addition of a graph structure.

To speed up global path planning and help to achieve faster navigation and localization several researchers have attempted to extract topological map models from grid maps [11, 14–17]. Fabrizi and Saffiotti [15] proposed an approach based on fuzzy grid map to deal with sensors uncertainties and benefits from image processing to define free and occupied space regions. While, Myunget et al. [14] extract virtual doors by overlapping the Generalized Voronoi Diagram (GVD) and a configuration space eroded by the half size of the door. Kwangro et al. [17] have attempted to improve results given in [14] by building their approach on an algorithms given in [18] to detect edges and obstacle curvature. The resulting map is optimized by genetic algorithms to merge nodes and reduce edges. While we are aware that this approach is dedicated to path planning of a robotic cleaner, we argue that the quality of the generated map (e.g., obstacle counters) is rather poor.

Many small areas (e.g., non free space) were not represented. This decrease of map precision is not suitable for a multi-robot systems that explores and builds a shared map in a coordinated way. Indeed errors on a local map accumulate for the whole team.

There is in the literature a work that builds a map based on geometric primitives [19]. Vandorpe et al. used lines and circles to represent a given environment. Other work [20] presents a technic to create a map of the surroundings of a mobile robot by converting the raw data of a scanning sensor to a map composed of polygonal curves. Nevertheless, there exist no evaluation or comparison of these approaches with *Occupancy Grids*.

## 3 Vector-based Map

A *Vector Map* is a graph that delimits the navigable space (i.e. free space) and the un-navigable one (i.e. obstacles). The graph is defined by vertices and edges where each vertex $v \in V$ is a particular delimitation position (e.g., corner) and each edge $e \in E$ is a continuous delimitation. Assuming a two dimensional *Vector Map*, a vertex $v$ defines a position at the coordinates $(x_v,\ y_v) \in \mathbb{R}^2$ and an undirected edge $e(v_e,\ v'_e)$ defines a line-segment obstacle delimitation. Each point $p_e$ of an edge $e$ is at the frontier between navigable space and obstacle ($p_e = v_e + k \cdot \overrightarrow{v_e v'_e}$ with $k \in [0,\ 1]$).

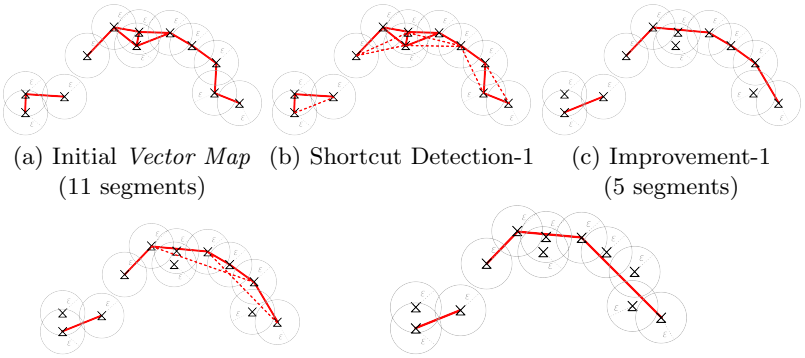
Using a range sensor or *Occupancy Grid Map*, obstacle delimitation is first obtained in a more or less structured points cloud format. Obstacle delimitations are defined as a finite set of points $P \subset \mathbb{R}^2$. A valid *Vector Map* $\langle V,\ E \rangle$ has to respect a constraint of $\epsilon$-error threshold and a constraint of continuity (example Figure 2e). The constraint of $\epsilon$-error threshold defines the resolution of the generated map. Whatever a point $p \in P$ in the initial obstacle delimitation, there is at least one segment $e \in E$ in the *Vector Map* where the distance between the point $p$ and the segment $e$ is less than $\epsilon$:

$\forall p \in P, \quad \exists e \in E, \quad \mathit{distance}(p,\ e) < \epsilon.$

The constraint of $\epsilon$-error threshold is mainly responsible for the growing on the number of edges in the *Vector Map*. The smaller is $\epsilon$, the less is the number of edges that are necessary to approximate the initial points cloud. The continuous constraint ensures that *Vector Map* edges do not introduce obstacles that are considered as free in the initial points cloud. For each position $v''_e$ of any edge segment $e$, there is at least a position $p$ in the initial points cloud $P$ the distance between $v''_e$ and $p$ is shortest than $\epsilon$: $\forall e \in E, \quad \forall v'' \in e, \quad \exists p \in P, \quad \mathit{distance}(p,\ v'') < \epsilon$.

An optimal *Vector Map* is a valid map that minimizes first the number of edges and second, distances between the initial points cloud and edge segments. The number of edges responsible for the required resources for map processes (map transfer, map fusion, path planning, etc.) and it is the main criterion to be optimized in our approach. While minimizing the distances allows to minimize errors due to the vectorization. Thus, given the minimal number of edges, an optimal *Vector Map* minimizes the following sum: $\sum_{p \in P} (\min_{e \in E} \mathit{distance}(p,\ e))$. A *Vector Map* is similar to $k$-means clustering method of vector quantization, where means are replaced by segments. This may imply that an optimal *Vector Map* is hard to compute. Therefore, we use a naïve heuristic method to generate the vector map from a given initial points cloud.

From an initial complete *Vector Map*, we use a naïve greedy algorithm to minimize the number of edges (cf. Figure 2). The initial *Vector Map* involves all point in points cloud $P$ and a edge each times the distance between two points is smaller than $\epsilon$. The algorithm improves the map by detecting and removing shortcuts. A shortcut is defined by three successive vertices $v_1,\ v_2$ and $v_3$ ($\exists e_{12} = (v_1,\ v_2)$ and $\exists e_{23} = (v_2,\ v_3)$) if the distance between $v_2$ and the segment $(v_1,\ v_3)$ is less than $\epsilon$. In such a case, the edges $e_{12}$ and $e_{23}$ are reduced



(a) Initial *Vector Map* (11 segments) (b) Shortcut Detection-1 (c) Improvement-1 (5 segments)

(d) Shortcut Detection-2 (e) Improvement-2 and Final (4 segments)

**Fig. 2.** Generation of a *Vector Map* from points cloud with a fixed $\epsilon$ distance.

to one edge linking $v_1$ and $v_3$ and vertex $v_2$ is removed. This process is repeated until no more shortcut is found.

## 4 Benchmarking Metrics

To compare maps, we used several metrics described in [21–23]. However, we've modified some of them and express them as percentage to ease understanding. We also consider two additional metrics which are the unoccupied picture distance [21] and the memory footprint as the size of the map.

*Map Score (MS)* [24]: It compares two maps cell-by-cell. It starts from the value of 0, and then it increases by a ratio of one divided by the total cells number, if the chosen cells are similar. It can be described by the following equation: $MS = \frac{100}{n} \cdot \sum_{i=1}^{n} R_i - G_i$, where $R_i$ and $G_i$ are the robot generated and the ground-truth maps, respectively, and $n$ is the number of pixels in the ground-truth map.

*Cross Correlation (CC)* [25]: It consists of calculating the coefficient of similarity between two maps. A fitness measure of the robot-generated map is calculated using Barons *cross correlation coefficient*. It is normalized between 0 and 1 and it is based on the averaging cell values: $CC = \frac{\overline{R_i \cdot G_i} - \overline{R_i} \cdot \overline{G_i}}{\sigma(R) \cdot \sigma(G)} \cdot 100$, where ,$\overline{R_i \cdot G_i}$ is the mean of $R_i \cdot G_i$ and $\sigma$ is the standard deviation.

*Pearson's Correlation (PC)* [26]: It gives the measure of how likely it is possible to infer a map from another and it is applied only for occupied space. According to [21] this metric suffers from two drawbacks, one because it requires similar occupied pixels between maps, and the other is that it is perturbed by the outliers (e.g, [27]): $PC = \frac{\sum_{i=1}^{n} (R_i - \overline{R_i})(G_i - \overline{G_i})}{\sqrt{\sum_{i=1}^{n} (R_i - \overline{R_i})^2} \cdot \sqrt{\sum_{i=1}^{n} (G_i - \overline{G_i})^2}} \cdot 100.$

*Occupied Picture-Distance-Function (OPDF)* [21, 22]: It calculates the ratio of the sum of the closest Manhattan-distance for each occupied cell of the robot-generated map to an occupied cell in the ground-truth divided by the number of occupied pixels of the robot-generated map. It comes with the advantages of removing the inherent problem of pixel-to-pixel comparison. It is worth to note that we make a limit in the search space (e.g., a rectangle of width *wscpace* and height *hsspace*): $OPDF = (1 - \frac{1}{no \cdot r} \sum_{i=1}^{no} d_i) \cdot 100$, where $no$ is the number of occupied cells, $d_i$ is the Manhattan-distance of each occupied cell of the robot-generated map to the closest occupied cell on the ground-truth map. In case no closest cell was found we set $d_i = r$, where $r$ is the maximum search space distance $r = \sqrt{(wsspace^2 + hsspace^2)}$.

*Unoccupied Picture-Distance-Function (UPDF)* : It has the same concept of the OPDF however it is applied to unoccupied pixels: $UPDF = (1 - \frac{1}{nu \cdot r} \sum_{i=1}^{nu} d_i) \cdot 100$, where $nu$ is the number of unoccupied cells, $d_i$ is the Manhattan-distance of each unoccupied cell of the robot-generated map to the closest unoccupied cell on the ground-truth map. In case no closest cell was found we set $d_i = r$.

*Memory Occupancy Size* : This metric evaluates the size of the map.

## 5 Experiments

### 5.1 Setup

We conducted multiple experiments to evaluate and compare *Vector Maps* and *Occupancy Grid* maps using the previously defined metrics. Our objective is to cover as much as we can in terms of diversity, complexity and scalability. Therefore, we used two different axes of evaluation: terrain topology and terrain size as shown in Figure 3. We used three different terrain topologies: maze like, office like and unstructured like, to assess *Vector Maps* on different shapes. We also used three different sizes for each terrain topology: small (10mx10m), medium (40mx40m) and large (80mx80m).

Those parameters lead to 9 experiments that we have performed using the MORSE [28] robotics simulator. We created the 2D ground-truth maps and we generated the 3D MORSE map files out of them. All our experiments use the Pioneer3dx robot model. The control architecture of the robot has been developed using ROS[2] packages such as Karto SLAM for the construction of the *Occupancy Grid* map. We chose Karto SLAM because it provides better results compared to other SLAM algorithms implementations [22]. During each experiment, the simulated robot was automatically driven through the same predefined trajectory according to the current terrain.

The output of each experiment is one *Occupancy Grid* map. Based on each occupancy grid map, we generated the corresponding *Vector Map*. Indeed, we don't have a SLAM algorithm that directly builds a vector map from laser scans

[2] `http://www.ros.org`

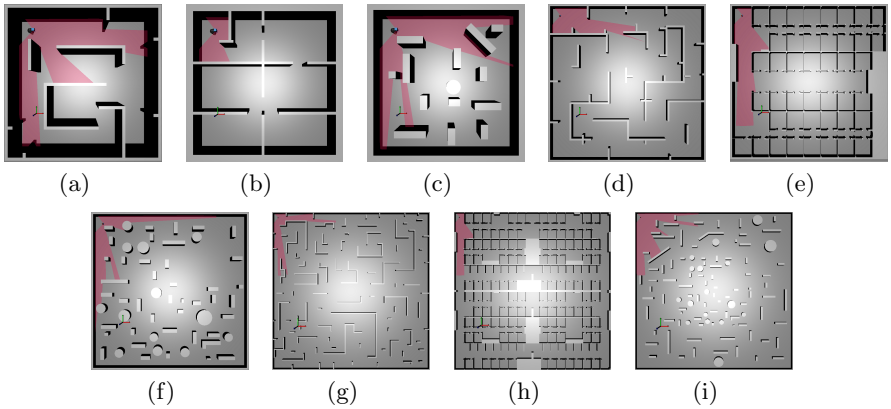



**Fig. 3.** Snapshots of the experiment terrain taken from MORSE simulator: (a) maze $10 \times 10$ m, (b) office $10 \times 10$ m, (c) unstructured $10 \times 10$ m, (d) maze $40 \times 40$ m, (e) office $40 \times 40$ m, (f) unstructured $40 \times 40$ m, (g) maze $80 \times 80$ m, (h) office $80 \times 80$ m and (i) unstructured $80 \times 80$ m.

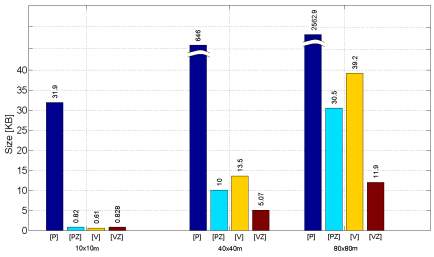
since we first need to assess whether the use of vectors are a good alternative to occupancy grids. The points cloud matches occupied cell centers and comes with regular discrete coordinates. The value $\epsilon$ is initialized according to the *Occupancy Grid* map resolution $\epsilon'$ (*i.e.* the size of a cell in the real word). $\epsilon$ is the minimal distance to potentially connect the eight neighbors of an occupied cell ($\epsilon = \sqrt{2} \times \epsilon'$).

Figure 4(a) shows the size of the Karto map (grid cell), zipped Karto map, *Vector Map* and zipped *Vector Map*, where each value is an average of the three terrain types of the same size (P (.PGM), PZ (Zipped .PGM) V (.Vector Map), VZ (Zipped .Vector Map)). Figures 4(b) to 4(j) present our experimental results through comparisons between maps generated by Karto SLAM (left bar with blue color), *Vector Maps* based on Karto (middle bar with green color) and *Vector Maps* based on ground-truth (right bars with brown color) for the three different terrain sizes and the three different types of terrain.

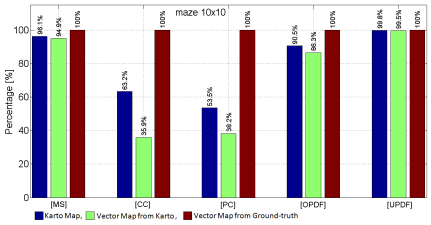
### 5.2 Discussion

As one might expect, a vector-based map has a much smaller memory footprint ($39.2KB$ in average for a 80x80m terrain) compared to the one of an *Occupancy Grid* map produced by Karto ($2532.4KB$ in average). Indeed, a single edge in the *Vector Map* can replace many points of the *Occupancy Grid*. Moreover, the number of edges in a vector increases only if there are more obstacle corners. In contrast, the memory footprint of an *Occupancy Grid* map increases with the size of the modeled terrain.

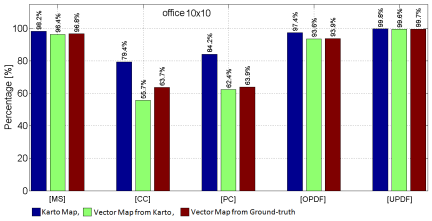
When it comes to sending a map over the network, which is a critical issue in Multi-Robotics exploration, a large file might be compressed to save the bandwidth. Interestingly, the non-compressed *Vector Map* requires an amount



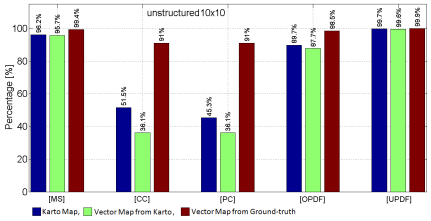

(a) Average on maps sizes.



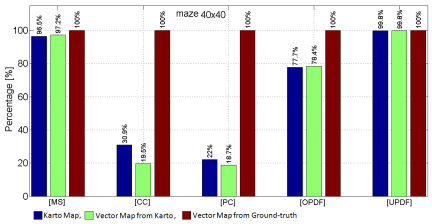

(b) Small maze terrain (10x10 m).



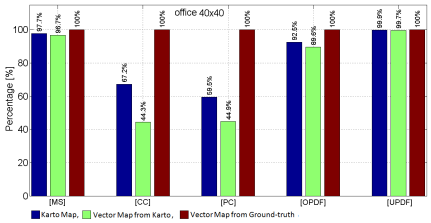

(c) Small office terrain (10x10m).



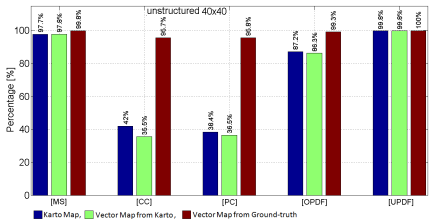

(d) Small unstructured terrain (10x10m).



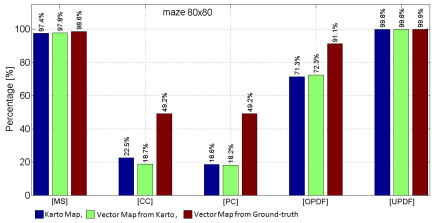

(e) Medium maze terrain (40x40m).



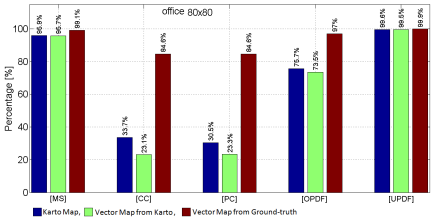

(f) Medium office terrain (40x40m).



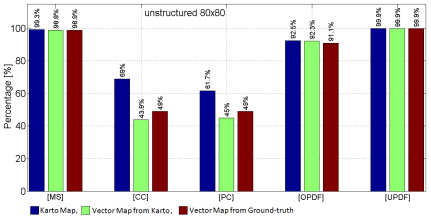

(g) Medium unstructured terrain (40x40m).




(h) Large maze terrain (80x80m).




(i) Large office terrain (80x80m).




(j) Large unstructured terrain (80x80m).

**Fig. 4.** Benchmarking Results

of memory that is close to what is required for a compressed *Occupancy Grid*. Thus, *Vector Map* makes it possible to send the uncompressed map to avoid the computational overhead of the compression/decompression.

The evaluation of *Vector Maps* includes the analysis of the quality. Our goal was to highlight potential degradations of the map quality due to the approximation by line segments. Therefore, we have chosen five different metrics: Map Score (MS), Cross Correlation (CC), Pearson's Correlation (PC), Occupied Picture-Distance-Function (OPDF) and Unoccupied Picture-Distance-Function (UPDF). We performed a normalization to provide results as percentages with best value being 100%. So we can compare them for different terrain sizes.

For the three metrics MS, OPDF, and UPDF, the quality is always over 91%. The quality of the ground-truth vectorized map is often 100% as one might expect, while the vectorized Karto maps obtain values that always equal or very close to scores of the actual Karto map (max 1% difference). For the two remaining metrics (CC and PC), the quality of the regenerated map may decrease in some terrains. Surprisingly, this is even true for ground-truth vectorized map. CC and PC reach the value of 100% in three terrains, over 63% in other three terrains and around 49% in the two remaining terrains. However, these two metrics are criticized as suffering from some drawbacks given in [21], which seem to be reinforced by behaviors drown in this paper. We suspect that line segments approximation is one possible source for these results.

However, we believe that reaching 100% quality for all setups is also a matter of trade-off with the memory footprint. A map of higher quality is likely to require more edges, and hence occupy more memory. Still, since *Vector Maps* have a small memory footprint, scarifying some more memory in favor of an increase in quality is definitely acceptable.

## 6 Conclusion and Future Work

In this paper, we show that *Vector Maps* are an interesting alternative to occupancy grids for representing the environment. Indeed, *Vector Maps* which are based on line segments defined in metric coordinates can be processed based on graph theory likewise topological maps. They are also lightweight and accurate when generated from either a map produced by a SLAM algorithm (i.e. Karto) or the ground truth. While this is already interesting in the context of a single robot, it is even more critical for multi-robot exploration. Indeed, to achieve coordination robots have to share their local maps and merge ones provided by their teammates. Vector maps outperform occupancy grids by requiring less network bandwidth for the transfer and are promising less CPU consumption for maps merging.

We used several metrics to compare maps produced by Karto, the best laser SLAM algorithm available for ROS [22] and our *Vector Maps*. As expected, *Vector Maps* have drastically reduced memory footprint. We also compared quality using 5 metrics. For 3 of them, we can conclude that the quality of the *Vector Map* is similar to the *Occupancy Grid* map. However, the 2 remaining quality

metrics show a loss that can be important under some conditions. Considering the overall quality, the memory reduction and the possibility to manipulate maps as graphs, we can conclude that the use of *Vector Maps* is an interesting alternative to *Occupancy Grid* maps. However, the 2 remaining quality metrics show a loss that can be important under some conditions. Considering the overall quality, the memory reduction and the possibility to manipulate maps as graphs, we can conclude that the use of *Vector Maps* is an interesting alternative to *Occupancy Grid* maps, that can be used in many robotics settings.

As for future work, we aim at evaluating *Vector Maps* in merging and path planning process during multi-robot exploration missions. In the long term, we plan to investigate a SLAM algorithm that directly produces *Vector Maps* out of laser scans. By suppressing the intermediate step of building *Occupancy Grid* maps, we expect an improvement of the final *Vector Map* quality.

## References


1. C. Stachniss, *Robotic Mapping and Exploration.* Springer, 2009.
2. H. Choset and K. Nagatani, "Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 2, pp. 125–137, Apr 2001.
3. F. Savelli and B. Kuipers, "Loop-closing and planarity in topological map-building," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 2, Sept 2004, pp. 1511–1517 vol.2.
4. G. Lozenguez, L. Adouane, A. Beynier, A.-I. Mouaddib, and P. Martinet, "Punctual versus continuous auction coordination for multi-robot and multi-task topological navigation," *Autonomous Robots*, pp. 1–15, 2015.
5. B. Gerkey, "Karto slam," http://www.ros.org/wiki/karto, July 2015.
6. H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," *Robotics and Automation*, vol. 2, pp. 116–121, 1985.
7. F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous Robots*, vol. 4, pp. 333–349, 1997.
8. N. Tomomi and T. Kanji, "Dictionary-based map compression for sparse feature maps," in *IEEE International Conferenc on Robotics and Automation (ICRA)*, May 2011, pp. 2329–2336.
9. A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap:an efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, pp. 1–17, 2013.
10. J.-L. Blanco, J.-A. Fernandez-Madrigal, and J. Gonzalez, "A new approach for large-scale localization and mapping: Hybrid metric-topological slam," in *Robotics and Automation, 2007 IEEE International Conference on*, April 2007, pp. 2061–2067.
11. S. Thrun, "Leaning metric-topological maps for indoor mobile robot navigation," *Artificial Intelligence*, vol. 99, pp. 21–71, 1985.
12. M. P. Jongwoo Lim, Jan-Michael Frahm, "Online environment mapping using metric-topological maps," *The International Journal of Robotics Research*, pp. 1–15, Nov 2007.
13. P. Moutarlier and R. Chatila, "Stochastic multisensory data fusion for mobile robot location and environment modeling," in *International Symposium on Robotics Research (ISRR)*, 1989, pp. 207–216.

14. H. Myung, H. Jeon, and W. Jeong, "Virtual door algorithm for coverage path planning of mobile robot," in *IEEE International Symposium on Industrial Electronics (ISIE)*, July 2009, pp. 658–663.
15. E. Fabrizi and A. Saffiotti, "Extracting topology-based maps from gridmaps," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3, 2000, pp. 2972–2978 vol.3.
16. J. Choi, M. Choi, and W. K. Chung, "Incremental topological modeling using sonar gridmap in home environment," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2009, pp. 3582–3587.
17. K. Joo, T. kyeong Lee, S. Baek, and S. young Oh, "Generating topological map from occupancy grid-map using virtual door detection," in *IEEE Congress on Evolutionary Computation (CEC)*, July 2010, pp. 1–6.
18. P. Nunez, R. Vazquez-Martin, J. del Toro, A. Bandera, and F. Sandoval, "Feature extraction from laser scan data based on curvature estimation for mobile robotics," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2006, pp. 1167–1172.
19. H. X. J. Vandorpe, H. Van Brussel, "Exact dynamic map building for a mobile robot using geometrical primitives produced by a 2d range finder," in *IEEE International Conferenc on Robotics and Automation (ICRA)*, April 2011, pp. 901–908.
20. R. Lakaemper, L. Latecki, X. Sun, and D. Wolter, "Geometric robot mapping," in *Discrete Geometry for Computer Imagery*, ser. Lecture Notes in Computer Science, E. Andres, G. Damiand, and P. Lienhardt, Eds. Springer Berlin Heidelberg, 2005, vol. 3429, pp. 11–22. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-31965-8_2
21. B. Balaguer, S. Balakirsky, S. Carpin, and A. Visser, "Evaluating maps produced by urban search and rescue robots:lessons learned from robocup," *Springer Autonomous Robots*, pp. 449–464, 2009.
22. J. M. Santos, D. Portugal, and R. Rocha, "An evaluation of 2d slam techniques available in robot operating system," in *IEEE International Symposium on Safety, Security (SSRR)*, Oct 2013, pp. 1–6.
23. Z. Yan, L. Fabresse, J. Laval, and N. Bouraqadi, "Metrics for performance benchmarking of multi-robot exploration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2015.
24. M. C. Martin and H. P. Moravec, "Robot evidence grids," in *Techni-cal Report CMU-RI-TR-96-06, Robotics Institute–Carnegie Mellon University*, 1996.
25. S. O'Sullivan, *An empirical evaluation of map building methodologies in mobile robotics using the feature prediction sonar noise filter and metric grip map benchmarking suite.* Master Thesis, University of Limerick, 2003.
26. I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature Extraction: Foundations and Applications (Studies in Fuzziness and Soft Computing).* Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.
27. A. Miranda Neto, A. Correa Victorino, I. Fantoni, D. Zampieri, J. Ferreira, and D. Lima, "Image processing using pearson's correlation coefficient: Applications on autonomous robotics," in *International Conference on Autonomous Robot Systems (Robotica)*, April 2013, pp. 1–6.
28. G. Echeverria, N. Lassabe, A. Degroote, and S. Lemaignan, "Modular open robots simulation engine: MORSE," in *IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011, pp. 46–51.