# Generation of Probabilistic Graphs for Path Planning from Stochastic Maps

P. Urcola, M. T. Lázaro, J. A. Castellanos and L. Montano *
*{urcola, mtlazaro, jacaste, montano}@unizar.es
Instituto de Investigación en Ingeniería de Aragón
Universidad de Zaragoza

***Abstract*—This paper presents a method to automatically generate probabilistic graphs for path planning out of stochastic maps. A sample based technique is used to generate a set of paths from which a graph structure of the map is built. The nodes of the graph represent places in the map and the edges the paths between those places, which are labelled with the cost of traversing the edge and the probability of being navigable. We provide results of the proposed method using different types of stochastic maps such as feature-based maps, occupancy grid maps and pose graph maps.**

## I. Introduction

A graph is a very versatile and compact representation to keep the structure of a scenario. There are plenty of robotics applications such as path planning, localisation and task allocation where robust methods have been developed based on the prior knowledge of a graph representing the environment.

For path planning purposes, the problem of finding the best path between two vertexes has been extensively studied, developing very efficient algorithms such as [1] and [2]. The complexity of these algorithms increases with the size and density of the graphs used, so it is important to have a representative graph of the environment as simple as possible.

One commonly used representation of the environment are the occupancy grid maps. There have been approaches to solve the problem of obtaining graph representations from this kind of maps. For instance [3] propose a method to obtain a topological map from an occupancy grid by partitioning the free space using critical points and obtaining the Voronoi diagram to compute the connections between the partitions. Although the initial occupancy grid is labelled with probabilistic information, this information is not incorporated into the topological representation.

In [4] and [5], they apply learning techniques to classify different regions from the range sensor readings which, in the first case, are simulated from a given grid map. The use of this technique is limited to structured indoor scenarios, where typical elements such as corridors, rooms and doorways can be identified.

In [6], the authors consider a fuzzy occupancy grid map as an image and obtain a topological representation of the open spaces and their connections by applying morphological operators. In this approach, the probabilistic information in the original map is transferred to the graph by labelling the vertexes and the edges of the map with the values of the cells they represent. However, the method assumes independent distribution probability of each cell in the grid map which is not true in general.

**9781-1-4673-9163-4/15/$31.00 © 2015 IEEE**

Pose graph techniques for SLAM, very popular in the last years, are able to directly provide a graph representation of the environment. The main weakness of these graphs is that the connections between nodes are limited to the trajectories executed by the sensor during the map building phase. Thus, if they are used for path planning, the robot is strongly limited to the trajectories previously traversed. In addition, the resulting representation of the scenario is not uniform as there are high density of vertexes in the places that have been revisited several times meanwhile there are no vertexes in zones not visited. In [7], the authors propose a technique to address these limitations by extending the graphs to unvisited zones. However, the additional vertexes and edges added to the graph are assumed to be fully navigable, but they neither consider the obstacles during the graph extension, nor provide an estimation of the traversability. This assumption is critical for path planning because it relies on the traversability of the edges or, in case of probabilistic information, uses the traversability values for computing the optimal path.

The scenario considered in this paper is a navigation mission where the robot is given a previous stochastic map of the environment and the starting and the goal localisation of the mission in the map. The robot has to plan an optimal path between these places which will be used for navigation.

We propose a sample based method to obtain probabilistic graphs out of stochastic maps. The intrinsic uncertainty of the map is transmitted to the graph generated by labelling the edges with the probabilities of being traversable. Moreover, the algorithm has been designed in such a generic way that it can be applied to a very diverse types of stochastic maps such as feature-based, occupancy grids and pose graphs with few requirements.

## II. Probabilistic Graphs

Graphs are a common well known abstraction to represent the environment, where the nodes correspond to places in the space and edges represent the possibility of travelling from one place to another. In deterministic graphs, the nodes are labelled with the coordinates in the space of the place they represent and the edges with the cost of moving from one node to another.

When considering stochastic maps, the information about the space is not deterministic. Depending on the map, uncertainty is associated with different elements. For instance, feature maps represent the position of the features with a probability distribution; occupancy grid maps represent the probability of the presence of obstacles in each cell; pose graphs assume that there is uncertainty in the poses. In any case, the stochastic map provides a probability distribution of the obstacles (non-navigable elements) in the space.

The uncertainty of these maps must be represented into the graph. This is done by extending the labelling of the edges in the graph so that the labels are no longer just cost values but also probability distributions.

## III. Graph Generation

To create a graph useful for path planning purposes, it is interesting to capture the structure of the scenario with all the possible interconnections between places so that a full set of possible paths is available for planning.

The idea behind the technique we propose is to analyse a set of valid paths across the scenario to extract the places and the connections among these places. Once the connectivity of the graph is computed, the labelling process takes place. Finally, due to the fact that the execution time of the path planning algorithms increases with the size and complexity of the graph, a simplification process is proposed to reduce the number of places and connections.

All the processes are detailed in the following subsections. Along the explanation, we guide the reader using as example a segment-based stochastic map of an office-like environment shown in Fig. 1. Note how blurred segments correspond to features with higher uncertainty level.

### A. Path Generation

First of all, we need a set of possible paths in the scenario. In a navigation mission, where only an *a priori* stochastic map, a starting point and a goal point are provided, we generate the set of paths from the starting point to the goal by sampling the map and computing the best path in each sample.

Due to the uncertainty of the map, obstacles of each sample are not placed in the same position with respect to the start and the goal points. This fact can block some zones as the width is not enough for the robot considered for the mission, leading to different optimal paths from start to goal.

Function GeneratePath shows the pseudo-code to generate one path.

```
function GeneratePath(map, start, goal)
    mapSample ← GenerateRandomSample(map)
    gridMap ← ProjectMap(mapSample)
    CSpace ← ComputeCspace(gridMap, robotSize)
    return ComputeShortestPath(CSpace, start, goal)
end function
```

We use a grid based A* algorithm to compute the optimal path between start and goal so the sample of the stochastic map must be first projected into a grid. This projection consists in setting to OCCUPIED the cells that represent the space where there are obstacles in the sample of the map. The configuration space (CSpace) is computed from the binary occupancy grid and the size of the robot. Assuming a circular shape, it corresponds to a morphological operation of dilation over the grid. Then, the A* algorithm computes the shortest path using only not OCCUPIED cells in the c-space. The heuristic function used is the Manhattan distance assuming no obstacles. The path is defined as an ordered list of the cells traversed.

The set of paths provides a rich information about the connectivity of the open space but there are many of them that represent the same route to the goal except for local variations. To reduce the number of paths with no loss of important information, we group the paths into equivalence classes.

### B. Path Clustering

The classification of the paths is done by defining homotopic groups. In other words, two paths belong to the same homotopic group if there is a transformation in the free space that deforms one into the other.

Instead of verifying the homotopy property for every pair of paths, we characterise it by defining $\mathcal{D}_{path}$ (1), a distance function between two paths, where $p_k\,(q_k)$ is the k-th equally distributed cell along the path $p\,(q)$.

$$\mathcal{D}_{path}(p,q) = \frac{1}{m}\sum_{k=1}^{m} \|p_k - q_k\|_2\,,\; m \geq 2 \qquad (1)$$

In simple words, the distance between two paths is defined as the average distance between $m$ pairs of cells equally distributed along the paths.

With this definition of distance between pairs of paths, we use an agglomerative hierarchical clustering (see [8] for details) to merge similar paths into the same cluster. The distance between two clusters of paths is defined as $\mathcal{D}_{cluster}$ (2), the maximum distance between any pair of paths, one from each cluster $c_i$ and $c_j$, which corresponds to a complete-link agglomerative hierarchical clustering algorithm.

$$\mathcal{D}_{cluster}(c_i,c_j) = \max_{\forall p\in c_i,\,\forall q\in c_j} \mathcal{D}_{path}(p,q) \qquad (2)$$

The clustering algorithm (ClusterPaths) keeps merging clusters that are close enough to be considered one until there are no more clusters to merge. This stop condition may happen because all the paths have been merged into one cluster or because the clusters are too far away one to each other to merge them.

Other clustering methods for trajectories like [9] require all the trajectories to have the same length, which does not apply in our case.

Figure 1 shows the set of random paths obtained between start and goal classified into different clusters using the method described above. From these clustered paths, we obtain a set of candidate vertexes for the graph, using the process explained in the next subsection.

**function** CLUSTERPATHS(paths, $\alpha$)
  clusters $\leftarrow$ paths
  (a, b) $\leftarrow \underset{c_i, c_j \in \text{ clusters}}{\arg\min} \mathcal{D}_{cluster}(c_i, c_j)$
  **while** $\mathcal{D}_{cluster}(a, b) < \alpha$ **do**
    c $\leftarrow$ a $\cup$ b
    REMOVE(clusters, (a, b))
    INSERT(clusters, c)
    (a, b) $\leftarrow \underset{c_i, c_j \in \text{ clusters}}{\arg\min} \mathcal{D}_{cluster}(c_i, c_j)$
  **end while**
  **return** clusters
**end function**

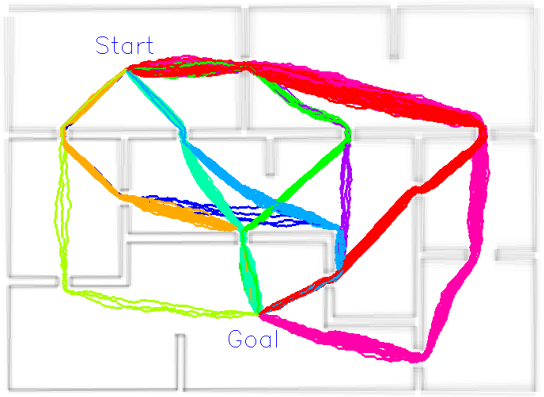


Figure 1. The different clusters of paths obtained from all the sampled paths. Different colours represent different clusters. Blurred walls represent the uncertainty of the stochastic map in the location of the obstacles.

### C. Vertex Generation

To obtain vertexes of the graph, we select $m$ points equally distributed along the paths (GENERATEVERTEXES). From the path clustering process, we know that the paths in the same class are close to each other and thus, the vertexes selected along them will also be close to the others. We use this fact to cluster together these vertexes that are expected to be close and to select the centroid of each group as the representative of the class and candidate to be a vertex in the final graph.

Figure 2 shows the initial set of candidate vertexes of our example and how they are aggregated into clusters inside the same path class.

**function** GENERATEVERTEXES(clusters, $m$)
  vertexes $\leftarrow \emptyset$
  **for** c $\in$ clusters **do**
    p $\leftarrow$ REPRESENTATIVE(c)
    **for** $k \in 0 \ldots m$ **do**
      INSERT(vertexes, $p_k$)
    **end for**
  **end for**
  **return** vertexes
**end function**

However, different classes of paths share some subpaths, resulting in vertexes that are too close each other. As these duplicated vertexes provide no extra information to the graph, we simplify the vertexes (SIMPLIFYVERTEXES) using the same hierarchical clustering process used for paths with the

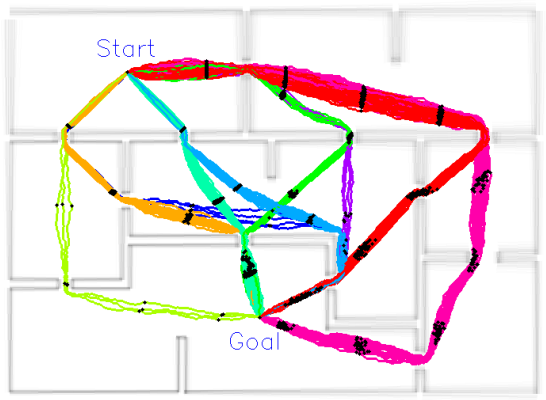


Figure 2. Initial set of vertexes defined along the paths. Vertexes of paths in the same cluster are also clustered.

corresponding Euclidean norm for vertexes. The REP function of a cluster of vertexes returns the closest vertex to the centroid as the representative of the cluster.

**function** SIMPLIFYVERTEXES(vertexes, $\beta$)
  clusters $\leftarrow$ vertexes
  (a, b) $\leftarrow \underset{u, v \in \text{ clusters}}{\arg\min} \|\text{REP}(u) - \text{REP}(v)\|_2$
  **while** $\|\text{REP}(a) - \text{REP}(b)\|_2 < \beta$ **do**
    c $\leftarrow$ a $\cup$ b
    REMOVE(clusters, (a, b))
    INSERT(clusters, c)
    (a, b) $\leftarrow \underset{u, v \in \text{ clusters}}{\arg\min} \|\text{REP}(u) - \text{REP}(v)\|_2$
  **end while**
  **return** clusters
**end function**

Figure 3 shows the final set of vertexes for the graph, after simplifying the vertexes. For instance, vertex 29 is the result of clustering four initial vertexes that belonged to the red, pink, green and purple path clusters.

In the next subsection, we will define the connectivity of the graph by creating the edges connecting the vertexes.

### D. Edge Generation

The interconnection of the vertexes using edges defines the structure and complexity of the graph. We know that two consecutive vertexes that come from the same path class are connected. But, for path planning purposes, it is interesting to have as much interconnections as possible so that there is a bigger set of paths to select the optimal in each case, even if they are not in the initial set of paths computed.

So, to increase the inherited connectivity from the paths sampled, we select as candidate edges all the possible interconnections between vertexes that are closer than a given radius (GENERATEEDGES).

This first set of edge candidates does not represent the connectivity of the environment because the distance between vertexes is measured in the whole space, not only in the free space. However, this set is an upper bound because in the best case, where all the space is free, all the edges would have the estimated distance.

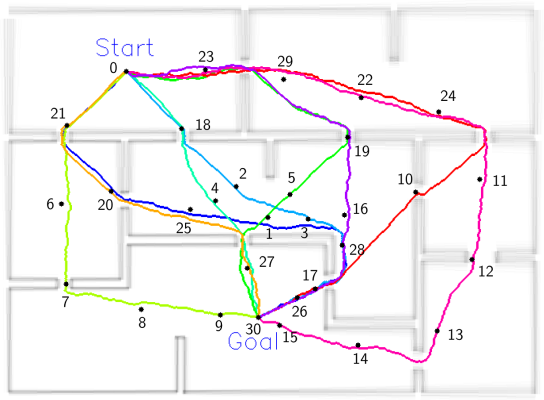


Figure 3. Final set of vertexes after the simplification of the initial clustering. To simplify the figure we have only plotted a representative path of each class.

**function** GENERATEEDGES(vertexes, $r$)
  edges $\leftarrow \emptyset$
  **for** u, v $\in$ vertexes **do**
    **if** $\|u - v\|_2 < r$ **then**
      INSERT(edges, (u,v))
    **end if**
  **end for**
  **return** edges
**end function**

Computing the real distance between two vertexes along an edge is considered during the labelling process.

### *E. Labelling the Graph*

As mentioned above, a probabilistic graph that tries to extract the structure of a stochastic map needs information about the coordinates in the space of the places represented by the vertexes as well as the distance and traversability probability of the edges in the free space.

Thus, vertexes are directly labelled with their location in the map space. Instead, labelling the edges is not trivial because the shortest path between two vertexes can be different depending on the realisation of the stochastic map.

So, resembling the process of path generation proposed above, we sample the stochastic map and, for each sampled instance, we compute the shortest paths between the two vertexes of every edge in the graph. A new sampling process is required for labelling the edges because not all of them are included in the paths previously sampled. Some of them are edges connecting vertexes in different paths and clusters and thus we have no information about the possible paths connecting them directly. With all the $n$ paths sampled for an edge, we classify them using the very same technique described before, in subsection III-B. Then, we select the class with shortest paths as the one that actually represents the edge. The length of the edge is the average length of the paths in the selected class and the probability for that edge to be navigable is the ratio between the number of paths in the selected class over the total number of sampled paths. This process is detailed in function LABELEDGES, where $len(p)$ is the number of cells in the path $p$. As example, Fig. 4 shows the resulting classified paths for an edge.

**function** LABELEDGES(map, edges, $n$)
  **for** (u, v) $\in$ edges **do**
    paths = $\emptyset$
    **for** $i \in 1, \cdots, n$ **do**
      p $\leftarrow$ GENERATEPATH(map, u, v)
      INSERT(paths, p)
    **end for**
    clusters $\leftarrow$ CLUSTERPATHS(paths, $\alpha$)
    c = $\underset{c_j \in clusters}{\arg\min} \frac{1}{|c_j|} \sum_{p \in c_j} len(p)$
    length(u, v) $\leftarrow \frac{1}{|c|} \sum_{p \in c} len(p)$
    probability(u, v) $\leftarrow \frac{|c|}{n}$
  **end for**
  **return** edges
**end function**

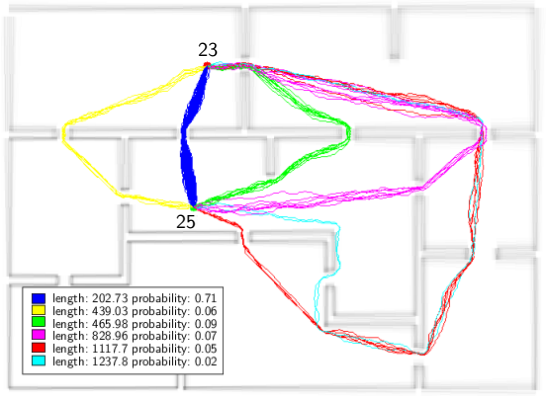


Figure 4. Labelling process for the edge (23, 25). The number of paths in the shortest class is used to estimate the probability of the edge.

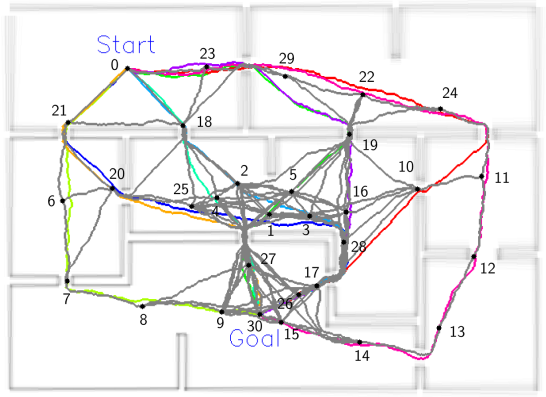


Figure 5. All the edges obtained after labelling process. In light grey, a sample of the selected class of paths for each edge is depicted to represent the connection between two vertexes.

Note that some vertexes that are close to obstacles such as number 7, 12 and 19 in Fig. 3, due to uncertainty, might not be in the free space for some samples during the labelling process. This fact reduces the probability of every edge that connects these vertexes.

Figure 5 shows all the edges after the labelling process. As mentioned above, the set of edges can be simplified because the candidate edges where chosen using all the space, not only

the free space. This leads to overlapping edges in the resulting graph which can be simplified by removing the redundant edges (SIMPLIFYEDGES).

An edge $(u, v)$ is said to be redundant if there exists a vertex $w$ close enough to the shortest path between $u$ and $v$ and there exists a chain of edges in the graph not using the edge $(u, v)$ that connects the vertexes $u$, $w$ and $v$, $w$ respectively. A path $p$ is close to a vertex $w$ if $\exists i \in 1, \cdots, len(p)$ such that $\|p_i - w\|_2 < \gamma$, given $\gamma$ a threshold distance. This is exemplified in Fig. 6.

**function** SIMPLIFYEDGES(edges, $\gamma$)
  **for** e $\in$ edges **do**
    **if** ISREDUNDANT(e, $\gamma$) **then**
      REMOVE(edges, e)
    **end if**
  **end for**
  **return** edges
**end function**

For instance, in Fig. 5, there are several edges depicted between vertexes 10 and 11 which actually correspond to edges from 11 to 16, 19 and 28 which are forced to first pass through the vertex 10 as it is placed in the shortest path on the free space between those pairs of vertexes. Thus, edges $(11, 16)$, $(11, 19)$ and $(11, 28)$ are redundant and can be removed as they can be built up by joining the edge $(10, 11)$ with $(10, 16)$, $(10, 19)$ and $(10, 28)$, respectively.

Figure 7 shows the graph after removing all the redundant edges. Notice how after this process the graph has been enriched with new edges like $(10, 19)$ and $(6, 20)$ connecting the original set of sampled paths which increase the connectivity of the graph and provides short-cuts between close vertexes.

Further simplification of the graph can take place. For instance, replacing cliques, sets of vertexes fully connected with edges in an open zone, i.e. with probability 1, by only one representative vertex placed in the centroid. For example, vertexes 1, 3 and 5 in Fig. 7 are close enough and fully connected in a open area so they could be simplified. Although this and other simplifications reduce the size of the graph and thus its complexity, the benefits of this simplification of the graphs are application dependant and thus out of the scope of this paper.

## IV. THE FULL ALGORITHM

Although we have presented the process to obtain a probabilistic graph representation of a scenario out of a feature-based stochastic map, the design is open and generic so different components can be used by only keeping an interface between them.

Here we present the complete algorithm of the process (GENERATEGRAPH). The inputs are the stochastic map, the starting point of the mission and the goal point. The only requirement for the stochastic map is to be able to be sampled and projected into a binary occupancy grid. These sampling and projection of the map are used only during the path generation and the graph labelling processes. The algorithm also depends on a set of intuitive parameters which are summarized below.

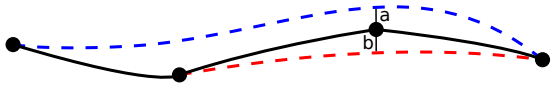


Figure 6. Redundant edge simplification. Dashed blue and red edges are already represented by the individual black edges. They can be removed if the distances $a$ and $b$ are lower than $\gamma$.

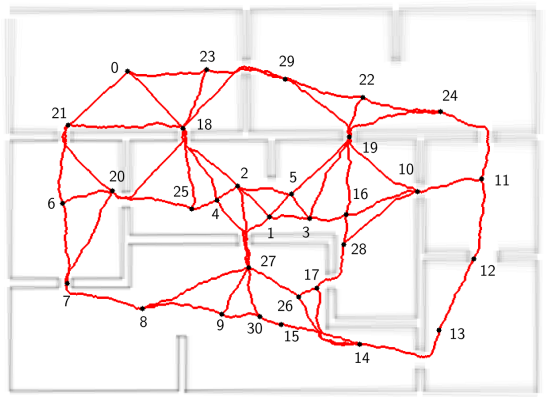


Figure 7. Simplified graph after removing the redundant edges.

**function** GENERATEGRAPH(map, start, goal)
  paths = $\emptyset$
  **for** $i \in 1, \cdots, N$ **do**
    new_path $\leftarrow$ GENERATEPATH(map, start, goal)
    INSERT(paths, new_path)
  **end for**
  clusters $\leftarrow$ CLUSTERPATHS(paths, $\alpha$)
  initial_vertexes = GENERATEVERTEXES(clusters, $m$)
  vertexes = SIMPLIFYVERTEXES(initial_vertexes, $\beta$)
  initial_edges = GENERATEEDGES(vertexes, $r$)
  labelled_edges = LABELEDGES(map, initial_edges, $n$)
  edges = SIMPLIFYEDGES(labelled_edges, $\gamma$)
  **return** GRAPH(vertexes, edges)
**end function**

### A. *Parametrisation of the algorithm*

These are the parameters used in the algorithms:

- $N$ Number of samples of the stochastic map used to compute the set of paths
- $n$ Number of samples used to label each edge
- $m$ Number of vertexes selected in each path
- $\alpha$ Minimum distance between two paths in different classes
- $\beta$ Minimum distance between two vertexes
- $\gamma$ Maximum distance to determine if a path passes through a vertex
- $r$ Maximum distance between two connected vertexes

The first two parameters $N$ and $n$ are the number of samples used to obtain the paths and to label the edges respectively. As for other sample-based algorithms, the higher these values are the better the result resembles the original probability distribution. The chosen values should trade off

the computational cost which increases with the number of samples. The KullbackLeibler divergence can be used to determine the compromise between improvement of the sampling and the computational cost. For instance, Table I shows the divergence and consistency measurements for different number of samples for the feature-based stochastic map.

Table I. RESULTS OF THE ANALYSIS OF THE SAMPLE SIZE

| Samples | 10 | 100 | 1000 |
|---|---|---|---|
| Median of the KL-divergence (nats) | 46.512 | 34.705 | 33.648 |
| Percentage of consistent sampling | 53.639 | 98.379 | 99.862 |

The values for 100 samples are clearly better than the ones obtained for 10 samples but the improvement in using 1000 samples is very small. Thus, using 100 samples seems to be a good compromise between acceptable approximation and computation costs.

The parameters $m$ and $\alpha$ control the classification of the paths. The number of clusters increases as $m$ increases and $\alpha$ decreases. $m$ sets the detail used in the path classification process. $\alpha$ sets the size of the smallest obstacle in the environment that causes a deviation in the paths that causes the paths to be clustered in different classes.

The last three parameters determine the type of graph that will be produced in terms of number of vertexes and connectivity between them. The parameters $\beta$ and $r$ control the size and connectivity of the graph. Depending on the application, there is a trade off between the complexity of the graph and the detail in the representation of the environment. Increasing $\beta$ reduces the number of vertexes in the graph while increasing $r$ the degree of the vertexes is increased.

The parameter $\gamma$ tunes the simplification of redundant edges. The bigger it is, the more redundant edges will be considered.

## V. RESULTS

In this section we show the results of applying the technique presented in this paper to different types of stochastic maps. In the previous sections, we have shown the results of the proposed method on a feature-based map, where features are 2D segments. Now, we provide the graphs obtained from probabilistic occupancy grid maps and from pose graphs.

In the first experiment, we apply our technique to an occupancy grid map obtained from [10] which represents the SFU campus. The original map has no uncertainty, so we applied a Gaussian filter to add some artificial uncertainty. Figure 8 shows the paths obtained from the samples after classification. To obtain a realization of the map we sample independently every cell with a uniform distribution and using the value of uncertainty as a threshold to decide if the cell is free or occupied.

After running our algorithm with the parameters $n = 20, m = 100, \alpha = 5, \beta = 20, \gamma = 10$ and $r = 60$, the resulting graph is depicted in Fig. 9.

Note that the graph is more complex than the one obtained in the feature-based map scenario. This is due to the different parametrisation used and shows that the output of our algorithm can be adapted to the needs of the scenario.

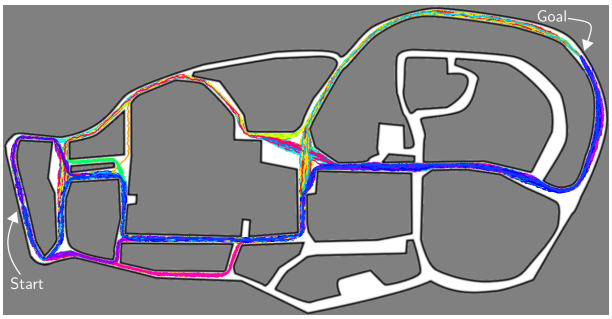


Figure 8. 1000 paths sampled from the occupancy grid map of SFU and clustered in 22 classes.

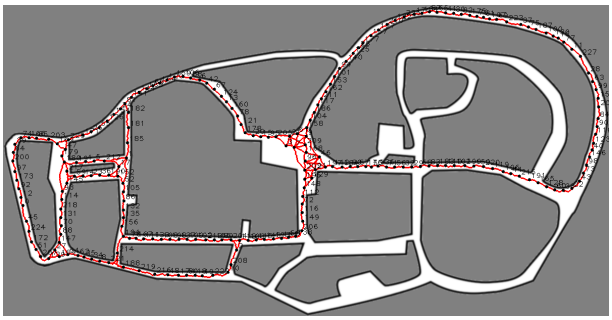

Figure 9. Resulting graph of the SFU scenario with 228 vertexes and 263 edges.

Figure 10. Stochastic map obtained by using a pose graph SLAM technique with the MIT CSAIL dataset.

It is also clear that our method, due to the fact that the set of paths are the shortest path between the starting and goal points on every sample, have left part of the scenario with no representation on the resulting graph.

In the second experiment, we apply our technique to the well-known map of the MIT CSAIL building, (raw data was obtained from the Robotics Data Set Repository (Radish) [11]). Previously to the navigation mission, we have obtained a pose graph map of the environment, shown in Fig. 10 where each graph node contains a robot position and an associated laser scan acquired at that position.

We have sampled the joint covariance matrix of all graph

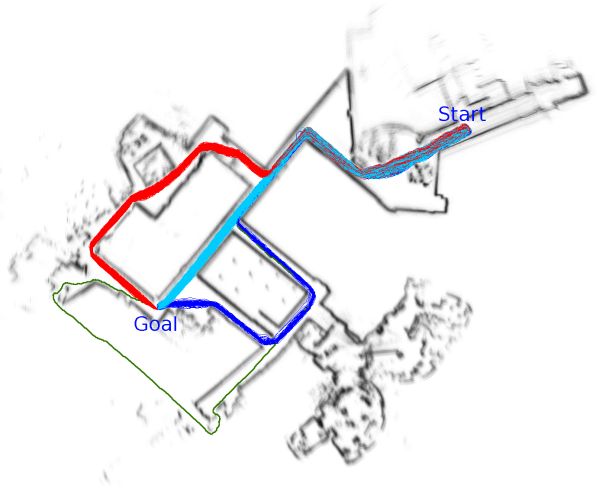


Figure 11. The set of classified paths obtained from all the sampled maps. A total of 4 clusters were obtained, represented in different colours.

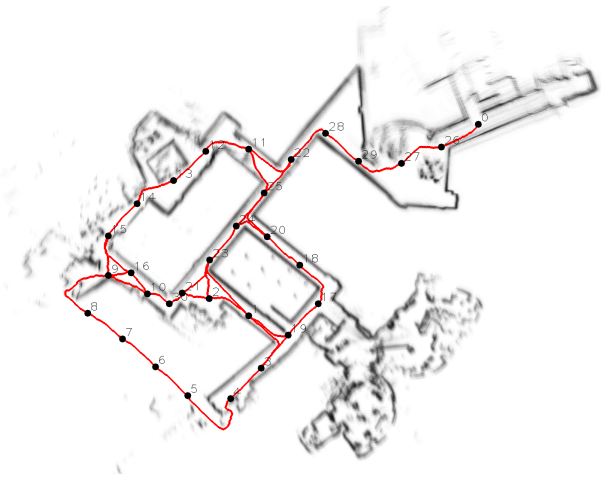

Figure 12. The probabilistic graph for path planning obtained out of the pose graph of the MIT CSAIL dataset with 31 nodes and 40 edges.

nodes and projected their corresponding scans into a grid, used as input data to the A* algorithm to obtain the set of possible paths between the start and goal positions. Figure 11 shows the set of paths sampled and clustered.

Finally, Fig. 12 shows the resulting probabilistic graph for path planning by applying the proposed algorithm with parameters $n = 20, m = 10, \alpha = 20, \beta = 100, \gamma = 20$ and $r = 200$.

We can observe how a possible path connecting the starting and goal position traversing a non-explored area is considered (green path of Fig. 11). Although this path has very low probability (it only appeared in 1 out of 1000 samples), it is represented in the final graph since it would be a plausible option when the rest of the paths were non-traversable.

## VI. Conclusions

We have proposed a technique to obtain a graph representation of the environment from the information in a stochastic map. The method is sample-based and it is able to transmit the probabilistic information in the original map to the resulting graph by labelling the edges with the expected distance between neighbour vertexes and the probability of each edge of being navigable.

Due to its generic nature, the method is able to use any type of stochastic map that is able to be sampled and projected into a binary occupancy grid. Moreover, the flexibility in the parametrisation allows to obtain a great diversity of graphs with different sizes and connectivity degrees. The technique has been tested with a feature-based, an occupancy grid map and a pose graph map.

In its current form, the graph obtained is limited by the variety of shortest paths obtained from the start to the goal point in each sample. This limitation could be avoided by selecting an initial set of points to sample paths between them, but it is not trivial to select a good set of points to ensure that the structure of the whole map is captured in the graph. This will be the main focus of the future works on this topic as well as the reduction or simplification of the parametrisation to facilitate the usage.

## Acknowledgements

This work has been partially funded by Grupo DGA T04-FSE and MINECO-FEDER project DPI2012-36070. The authors would like to sincerely thank to our colleague Marta Salas for her suggestions in path clustering.

## References

[1] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959. [Online]. Available: http://dx.doi.org/10.1007/BF01386390

[2] D. Ferguson, M. Likhachev, and A. Stentz, "A guide to heuristic-based path planning," in *Proceedings of the International Workshop on Planning under Uncertainty for Autonomous Systems, International Conference on Automated Planning and Scheduling*, 2005, pp. 9–18.

[3] S. Thrun and A. Bücken, "Integrating grid-based and topological maps for mobile robot navigation," in *Proceedings of the National Conference on Artificial Intelligence*. AAAI, 1996, pp. 944–951.

[4] O. Mozos and W. Burgard, "Supervised learning of topological maps using semantic information extracted from range data," in *Proceedings of the International Conference on Intelligent Robots and Systems*, Beijing, China, Oct 2006, pp. 2772–2777.

[5] A. Romeo and L. Montano, "Environment understanding: Robust feature extraction from range sensor data," in *Proceedings of the International Conference on Intelligent Robots and System*, Beijing, China, Oct 2006, pp. 3337–3343.

[6] E. Fabrizi and A. Saffiotti, "Extracting topology-based maps from gridmaps," in *Proceedings of the International conference on Robotics and Automation*. San Francisco, USA: IEEE, April 2000, pp. 2972–2978.

[7] R. Valencia, M. Morta, J. Andrade-Cetto, and J. M. Porta, "Planning reliable paths with pose slam," *Transactions On Robotics*, vol. 29, no. 4, August 2013.

[8] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A survey," *ACM Computing Surveys*, vol. 31, no. 3, September 1999.

[9] G. Tanzmeister, D. Wollherr, and M. Buss, "Environment-based trajectory clustering to extract principal directions for autonomous vehicles," in *Proceedings of the International Conference on Itelligent Robots and Systems*, Chicago, USA, September 2014, pp. 667–673.

[10] R. Vaughan, "Stage: Mobile robot simulator," 2009. [Online]. Available: https://github.com/rtv/Stage/

[11] A. Howard and N. Roy, "The robotics data set repository (radish)," 2003. [Online]. Available: http://radish.sourceforge.net/