Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 972304, 11 pages
http://dx.doi.org/10.1155/2014/972304

Research Article

3D Maps Representation Using GNG

Vicente Morell, Miguel Cazorla, Sergio Orts-Escolano, and Jose Garcia-Rodriguez

University Institute for Computing Research, University of Alicante, P.O. Box 99, 03080 Alicante, Spain

Correspondence should be addressed to Miguel Cazorla; miguel.cazorla@ua.es

Received 6 March 2014; Accepted 24 July 2014; Published 27 August 2014

Academic Editor: Yi Chen

Current RGB-D sensors provide a big amount of valuable information for mobile robotics tasks like 3D map reconstruction, but the storage and processing of the incremental data provided by the different sensors through time quickly become unmanageable. In this work, we focus on 3D maps representation and propose the use of the Growing Neural Gas (GNG) network as a model to represent 3D input data. GNG method is able to represent the input data with a desired amount of neurons or resolution while preserving the topology of the input space. Experiments show how GNG method yields a better input space adaptation than other state-of-the-art 3D map representation methods.

1. Introduction

A 3D point is comprised of (X, Y, and Z) values representing the spatial coordinates. Moreover, if color information (R, G, and B) is available for each point, it is referred to as RGB-D data. RGB-D cameras provide this kind of data and have become very popular due to their low cost, like the Kinect sensor. This sensor could provide more than 300,000 3D points per capture. That feature is very useful to carry out a fundamental task in mobile robotics: mapping [1]. Mapping is a task that builds a map from the observations and movements of the robot. Each time the robot moves, an observation is linked to that movement. Then, using different methods, for example, registration, the map can be built, transforming each observation with respect to a common coordinates frame. This map is useful to develop subsequent tasks, like localization, navigation, and recognition. The use of RGB-D data as observations is referred to as RGB-D mapping and RGB-D maps.

The amount of data in RGB-D maps is huge since the number of poses is high. In a typical map with 10,000 poses, the data could consist of more than 3 billion of 3D points, which are unaffordable for representation and other tasks. Furthermore, as acquisitions frame rate is high, for a common area, a huge amount of redundant points is used to represent the input space. Due to the huge quantity of data, several methods have been proposed to reduce the number of points in the map while preserving the main features of the data, as it would be used in further tasks.

Elevation maps were a commonly used structure in the past [2]. These elevation maps are represented using a regular 2D cell grid where each cell value represents the elevation or height of the surface of that space. This compact model allows a simple representation of large areas but with low level of detail. Triebel et al. [3] presented an extension of the height maps to represent different surfaces at different heights. This multilevel surface map (MLS map) allows the representation of vertical structures and different surfaces in a 2D cell-based structure like the ones used in the traditional height maps. This approach focuses on the representation of planar surfaces to help mobile robotics tasks like robotics navigation.

Following this idea of 3D space representation, some other structures have been proposed like occupancy grids or Octrees. Occupancy grids represent the entire space as 3D cell grids. The cell information could consist of a single value of occupancy or contain more complex information as the probability of occupancy. Several works in mobile robotics have used this structure as a base for their applications [4–6]. Another common structure is the Octree [7]. The Octree is a tree structure in which each internal node has eight children. Each node of the tree is recursively subdivided into eight new nodes until a certain condition is fulfilled, like the size of the area represented by a node. This structure represents both occupied and empty space in the area represented by the Octree. It also allows some optimized operations like closest point searching or occupancy checking. In [8], an Octree based framework called OctoMap is presented. It uses probabilistic occupancy estimation where areas of the space are represented as occupied, empty, or uncertain. Another commonly used structure is the voxel grid (VG). The VG downsampling technique is based on the input space sampling using a grid of 3D voxels. This technique has been traditionally employed in the area of computer graphics to subdivide the input space and reduce the number of points [7, 9].

Wang et al. [10] presented a feature based 3D point cloud simplification method. They detect the points with more information (big curvature) and subsample the rest of the points using a uniform spherical sampling method. Therefore, they preserve the keypoints and subsample those points with less curvature information. This method is able to subsample 3D point clouds obtained from object surfaces. It does not work on scene maps since the spherical sampling and the feature selection process is usually harder and problem dependent.

Other approaches use self-organizing maps in order to reduce the input space. Viejo et al. [11] used a Growing Neural Gas (GNG) algorithm to filter and reduce single frontal point clouds. In this paper, we propose the extension of that work to manage complete maps. The GNG adapts its structure to the complete map, reducing its size, preserving the input space topology, and providing better adjustment than existing methods. To validate our method, we present several experiments comparing our method with map size reduction state-of-the-art methods.

The rest of this work is organized as follows. First, in Section 2 we introduce and describe the proposed GNG application and the Octree and voxel grid methods that we will use in the experimentation. Next, in Section 3, the validation of our method is carried out comparing it with the two previous mentioned methods. Finally, conclusions and future works are drawn.

2. 3D Representation Methods

One way of selecting points of interest in 3D point clouds is the use of a topographic mapping where a low-dimensional map is fitted to the high dimensional manifold of the model, whilst preserving the topographic structure of the data.

In this section, we review some typical methods to represent and compress 3D data. First, we propose the use of a Growing Neural Gas algorithm to reduce and represent 3D point cloud maps. Then, we briefly describe two well-known data structures in order to compare them with our method.

2.1. GNG Method. A common way to achieve data dimension reduction is by using self-organising neural networks where input patterns are projected onto a network of neural units such that similar patterns are projected onto units adjacent in the network and vice versa. As a result of this projection, a representation of the input patterns is achieved, which in postprocessing stages allows exploiting the similarity relations of the input patterns.

However, most common approaches do not provide good neighborhood and topology preservation if the logical structure of the input pattern is not known a priori. In fact, the most common approaches specify in advance the number of neurons in the network and a graph that represents topological relationships between them, for example, a two-dimensional grid, and seek the best match to the given input pattern manifold. When this is not the case, the networks fail to provide good topology preservation as in the case of Kohonen's algorithm [12].

The approach presented in this paper is based on self-organising networks trained using the Growing Neural Gas learning method [13], an incremental training algorithm. The links between the neurons in the network are established through competitive Hebbian learning [14]. As a result, the algorithm is suitable in cases where the topological structure of the input pattern is not known a priori and yields topology preserving maps of feature manifold [15].

In GNG, the nodes of the network compete to determine the ones with the highest similarity to the input distribution. In our case, the input distribution is a finite set of 3D points extracted from different types of sensors. The highest similarity reflects which node together with its topological neighbors is the closest one to the input sample point which is the pattern generated by the network. The n-dimensional input signals are randomly generated from a finite input distribution.

The nodes move towards the input distribution by adapting their position to the input data geometry. During the learning process local error measures are gathered to determine where to insert new nodes. New nodes are inserted near the node with the highest accumulated error. At each adaptation step a connection between the winner and its topological neighbors is created as dictated by the competitive Hebbian learning method. This is continued until an ending condition is fulfilled, for example, the evaluation of the optimal network topology, a predefined network size or a deadline.

The network is specified as follows.

(i) It is a set N of nodes (neurons). Each neuron $c \in N$ has its associated reference vector $w_c \in R^d$. The reference vectors can be regarded as positions in the input space of their corresponding neurons.

(ii) It is a set of edges (connections) between pairs of neurons. These connections are not weighted and its purpose is to define the topological structure. The edges are determined using the competitive Hebbian learning algorithm. An edge-aging scheme is used to remove connections that are invalid due to the activation of the neuron during the adaptation process.

The GNG learning algorithm is as follows.

(1) Start with two neurons a and b at random positions w_a and w_b in R^d.

(2) Generate a random input signal ξ according to a density function $P(\xi)$.

(3) Find the nearest neuron (winner neuron) s_1 and the second nearest s_2.

(4) Increase the age of all the edges emanating from s_1.

(5) Add the squared distance between the input signal and the winner neuron to a counter error of s_1:

$$\Delta\text{error}\left(s_1\right) = \left\|w_{s_1} - \xi\right\|^2. \tag{1}$$

(6) Move the winner neuron s_1 and its topological neighbours (neurons connected to s_1) towards ξ by learning steps ε_w and ε_n, respectively, of the total distance:

$$\begin{aligned} \Delta w_{s_1} &= \varepsilon_w\left(\xi - w_{s_1}\right), \\ \Delta w_{s_n} &= \varepsilon_n\left(\xi - w_{s_n}\right). \end{aligned} \tag{2}$$

(7) If s_1 and s_2 are connected by an edge, set the age of this edge to 0. If it does not exist, create it.

(8) Remove the edges larger than $a_{\max}$. If this results in isolated neurons (without emanating edges), remove them as well.

(9) With every certain number λ of input signals generated, insert a new neuron as follows.

  (i) Determine the neuron q with the maximum accumulated error.

  (ii) Insert a new neuron r between q and its further neighbor f:

$$w_r = 0.5\left(w_q + w_f\right). \tag{3}$$

  (iii) Insert new edges connecting the neuron r with neurons q and f, removing the old edge between q and f.

  (iv) Decrease the error variables of neurons q and f multiplying them with a constant α. Initialize the error variable of r with the new value of the error variable of q and f.

(10) Decrease all error variables by multiplying them with a constant β.

(11) If the stopping criterion is not yet achieved, go to step 2.

In summary, the adaptation of the network to the input space takes place in step 6. The insertion of connections (step 7) between the two closest neurons to the input patterns establishes an induced Delaunay triangulation in the input space. The elimination of connections (step 8) eliminates the edges that no longer comprise the triangulation. This is made by eliminating the connections between neurons that are no longer activated or isolated. Finally, the accumulated error (step 5) allows the identification of those areas in the input space where it is necessary to increase the number of neurons to improve the mapping.

Using a Growing Neural Gas model to represent 3D data has some advantages over the traditionally used methods like voxel grid or Octrees. For example, we specify the number of neurons (representative points of the map), while other methods like the voxel grid or Octree get different number of occupied cells depending on the distribution and resolution of the cells (voxels on voxel grid and leaves on Octree based methods).

2.2. Octree Based Method. Most 3D point cloud mapping algorithms typically use the spatial organization of the points to encode them in a structure like an Octree to reduce the amount of information. An Octree is a tree data structure in which their internal nodes have exactly eight children. Octrees make a partition of the three-dimensional space by recursively subdividing it into eight octants. It starts from a user specified volume space or it computes the bounding box of the input set. Then, each node or cell is subdivided into 8 children nodes until a certain condition is reached. These conditions vary depending on the problem or the Octree implementation. A commonly used condition is to stop producing new children nodes when the volume or size of the corresponding cell node reaches the desired precision.

One of the main features of the Octree representation is that nodes not containing input space points are not subdivided and therefore those leaf nodes represent an empty volume of the space. This feature is useful for some mobile applications as robot navigation. There exist different approaches to select the representative point of the occupied nodes. A simple one is to get the center of the node cell but using the mean or centroid of the cell inner points improves the preservation of the topology. This approach offers better results but it has a higher computational and memory cost.

2.3. Voxel Grid Method. The VG downsampling technique is based on the input space sampling using a grid of 3D voxels [16]. VG algorithm defines a voxel grid in the 3D space and for each voxel a point is chosen as the representative of all points that lie on that voxel. It is necessary to define the size of the voxels as this size establishes the resolution of the filtered point cloud and therefore the number of points that form the new point cloud. The representative of each cell could be chosen by using one of the approaches described in the

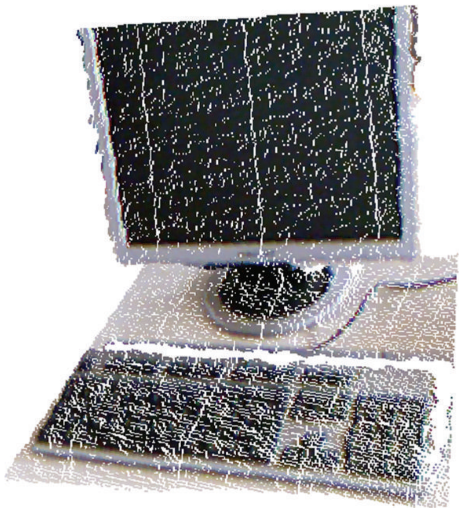


Figure 1: Partial point cloud example.

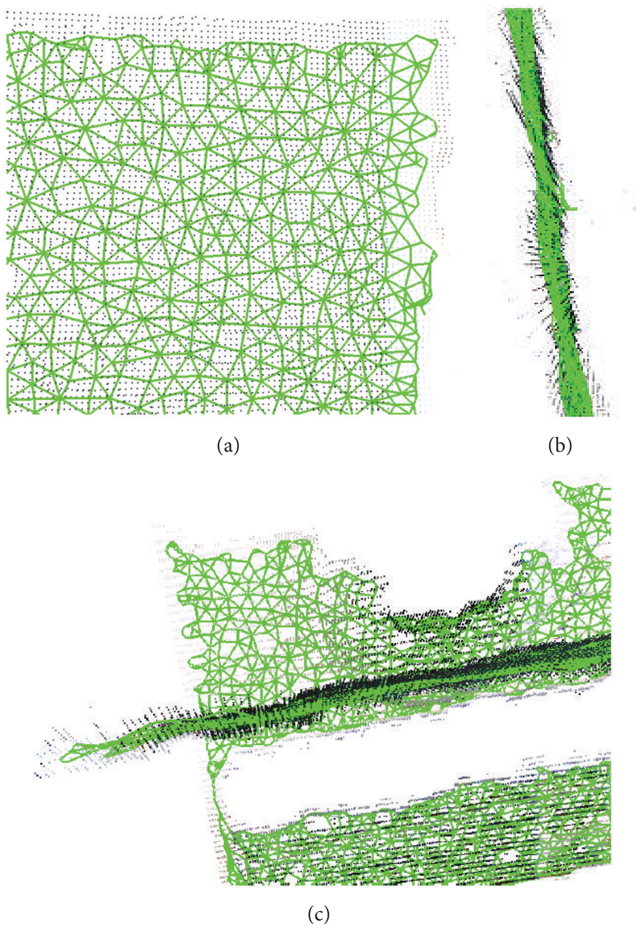


Figure 2: GNG representation example of the partial point cloud shown in Figure 1. Front, side, and top views are shown from (a) to (c), respectively.

previous section. Thus, a subset of the input space is obtained that roughly represents the underlying surface.

The VG method, as the Octree based methods, presents the same problems compared to other subsampling techniques: it is not possible to define the final number of points which represents the surface, geometric information loss due to the reduction of the points within a voxel and sensitivity to noisy input spaces.

2.4. Discussion. In this subsection we briefly describe the main differences of the above-described methods. Figures 2, 3 and 4 show several examples of using the three described methods. The GNG representation provides a set of neurons and their neighbors. These representatives and their connections can be used in some algorithms like 3D mesh reconstruction or feature extraction.

Both, voxel grid and Octree methods should provide similar results due to their final representation of the points. In a point cloud reduction application, the Octree gets their representatives of the leaf nodes and if we use the same resolution as the voxel grid method we get a similar division of the space in cubes or cells of the same dimension. The voxel grid method is the most simple and fastest reduction method but it does not have any of the advantages of the Octree structure or GNG model like neighbor searching facilities.

Figure 5 shows a 2D description of the representative points of the described methods. We observe that the GNG method assigns more neurons on high density input areas (bottom left area) than the voxel grid and Octree methods. We also observe how the GNG is able to eliminate some noisy values like the point near the center in contrast with the representatives used in the VG and Octree methods.

3. Experimentation

In this section we test the quality of adaptation of the three described methods. We first describe the data used in the experiments and then we analyze the results of the tested methods, both quantitatively and qualitatively.

3.1. Experimentation Setup. To test the implemented scene mapping systems on room map scenarios, we used the TUM RGB-D dataset [17]. This dataset provides RGB-D and ground-truth data with the goal of evaluating visual odometry and visual SLAM systems. The dataset contains the color and depth images obtained using a Microsoft Kinect sensor along the ground-truth trajectory of the sensor. It provides images at full frame rate (30 Hz) and sensor resolution (640 × 480). The ground-truth trajectory was obtained from a high-accuracy motion-capture system with eight high-speed tracking cameras (100 Hz).

This dataset contains 39 sequences recorded in two different scenarios. The fr1 datasets were recorded in a typical office environment (first scenario) and the fr2 datasets were

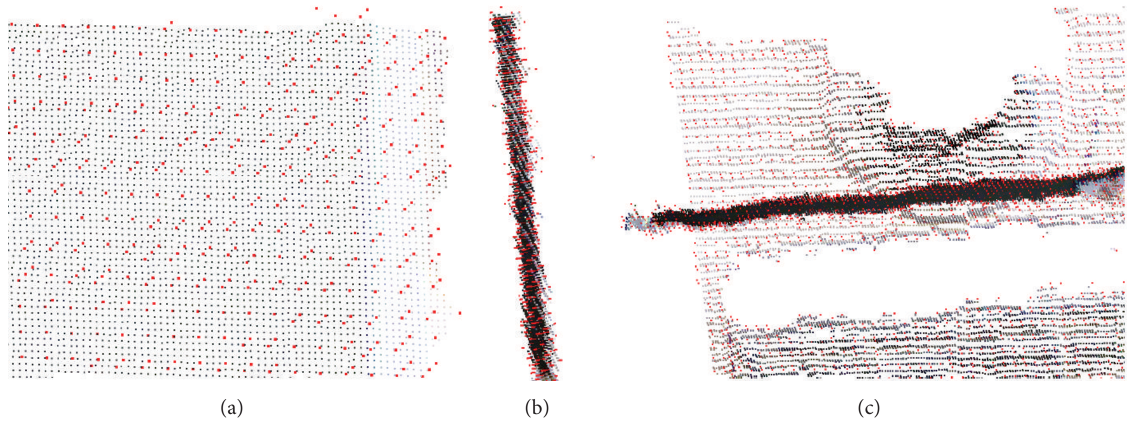


Figure 3: Octree representation example of the partial point cloud shown in Figure 1. Front, side, and top views are shown from (a) to (c), respectively.

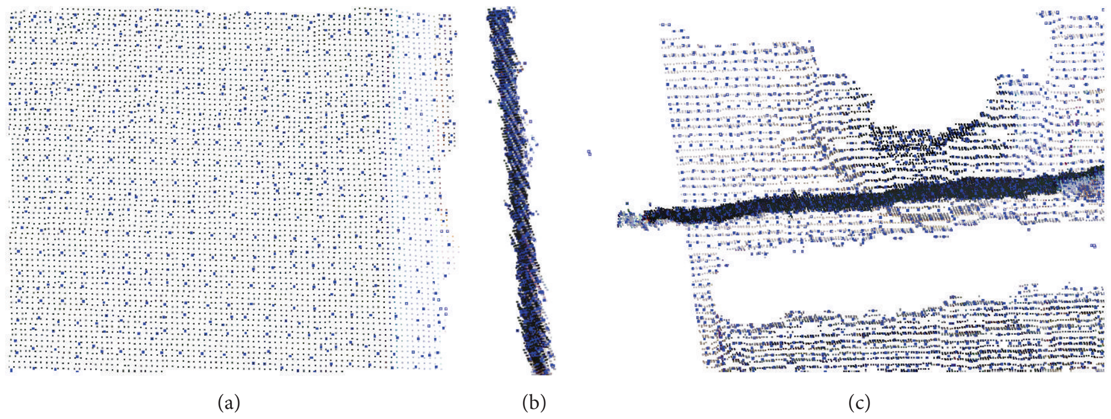


Figure 4: Voxel grid representation example of the partial point cloud shown in Figure 1. Front, side, and top views are shown from (a) to (c), respectively.

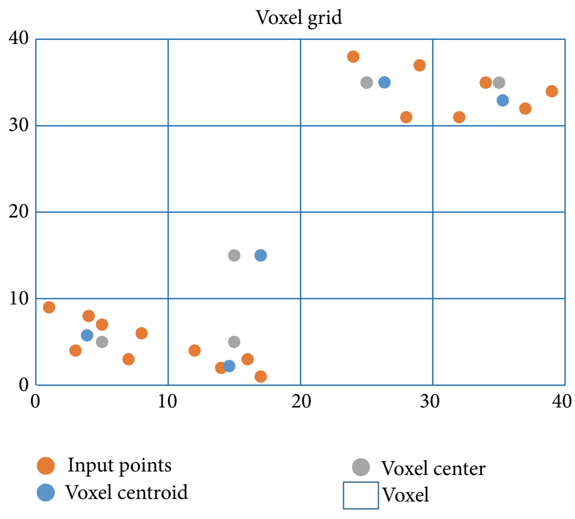
Table 1: Number of points of each ground-truth map dataset.

Dataset	Number of input points
fr1 *xyz*	1049739
fr1 desk	1952544
fr1 360	2357039
fr1 desk2	2751402
fr2 *xyz*	3492032
fr2 desk	5841800
fr3 long	1636623

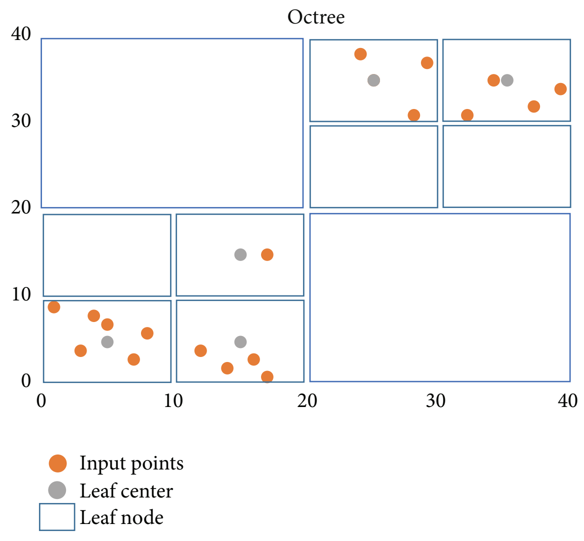
recorded in a large industrial hall (second scenario). Figures 6 and 7 show the ground-truth reconstruction maps of the “fr1 360” and “fr2 desk,” respectively.

Table 1 shows the number of points of the input maps used in the experimentation. We can observe that the number of input points ranges from one million (“fr1 *xyz*”) to 6 million (“fr2 desk”).

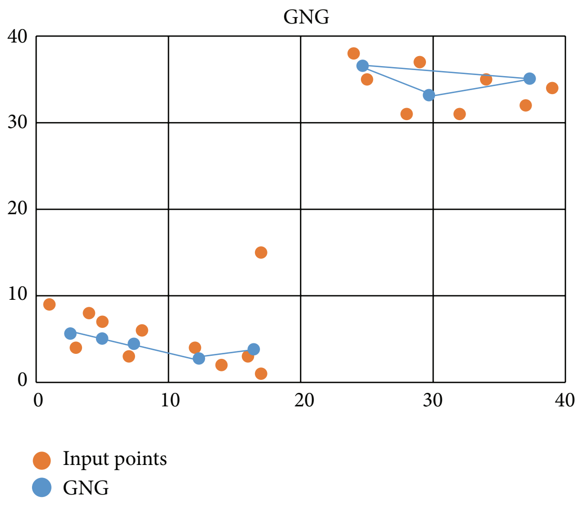
3.2. Quality Adaptation Experiment. As we previously mentioned, we are going to compare the proposed GNG adaptation with two commonly used data structures in the state-of-the-art, Octree, and voxel grid. The implementation of both methods is included in the Point Cloud Library (PCL) (the Point Cloud Library (or PCL) is a large scale, open project [18] for 2D/3D image and point cloud processing). The Octree implementation uses the center of the leaf nodes as representative points. However, the voxel grid implementation uses the centroid of the points of each nonempty voxel. Both implementations use a resolution parameter that represents the size of the voxel in the VG method and the side of the



(a)



(b)



(c)

FIGURE 5: Two-dimensional examples of the three tested methods.

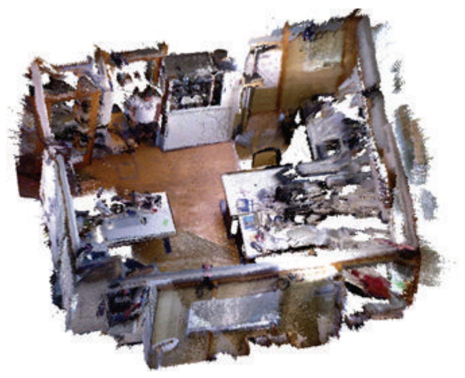


FIGURE 6: Example of the “fr1 360” ground-truth point cloud map.

Figure 7: Example of the "fr3 long" ground-truth point cloud map.

leaf cell of the Octree implementation. The GNG results are obtained using 10,000 λ input patterns.

We extensively tested the implemented methods using different number of representatives. Since the three tested methods reduce the amount of noise in the generated map, it is needed to know the real distance from the selected representatives to the original input space. The following measure specifies how close the representations are from the original model.

A quantitative measure of the input space adaptation of the generated map is obtained by computing the mean error (ME) of the reduced map against sampled points (input space):

$$\mathrm{ME} = \frac{1}{|V|} \sum_{p \in V} \min_{q \in A} \|p - q\|, \quad (4)$$

where V is the input space, p is a point that belongs to the input space, and q is the representative point with the minimum distance to p. Euclidean distances to closest points are averaged over the entire input space.

Figure 8 shows the RMS errors of the three methods on the six different tested maps. We observe that the three methods have similar behavior on the different datasets. The Octree method gets the worse results probably due to the selection of its cell-node center as representative. The voxel grid gets lower errors than Octree due to the use of the centroid of the inner points instead of using the center of the voxel or cell.

It is important to point out again that the representative selection method used in this comparison is given by the used implementations. But Octree and voxel grid methods can use a representative selection strategy. GNG adaption shows the best results on all datasets. It is noticeable that the GNG gets lower errors with different number of representatives but as the number of representatives increases the three different methods converge to the same error.

3.3. Qualitative Results. In this subsection we qualitatively analyze the results obtained with the three different methods. Figure 9 shows the original map and the three tested representations of the "fr1 360" scene. Part (a) shows the point cloud that we are trying to represent and reduce. Parts (b) and (c) are, respectively, the Octree and voxel grid representation, and part (d) is the GNG representation of the scene. The Octree representation, using the centers of the leaf nodes, gets a strongly structured point representation. This representation gets a more uniform distribution of the representatives but the adaption error is worse as we saw in Figure 5 and the mean error graphs. The voxel grid representation gets similar results compared to the Octree, where the points are uniformly distributed as it can be observed in the points that represent the floor. However, it gets better results on the boundaries compared with the Octree method. Both VG and Octree place representatives in isolated and noisy points. However, the GNG neurons are uniformly distributed over the input space and the impact of the noisy points and undefined borders on the obtained representation is reduced. We also observe the inherent triangulation of the space that the GNG algorithm gets using the neurons neighborhood.

Figure 10 shows two more experiments with GNG representation. In these experiments we have selected two maps with holes. GNG is able to adapt to these holes and it does not insert neurons in them.

Table 2: Runtimes and speed-up of GPU versus CPU implementation for different GNG versions.

Neurons	Patterns	CPU runtime (s)	GPU speed-up	GPU runtime (s)
5000	250	63	3×	21
12000	350	526	5×	105.2
18000	500	1448	6×	241.33

3.4. Execution Time. With respect to computational cost, our method is feasible to be included in a modern system using general purpose computing platforms. However, we designed in a previous work [17] a GPU-based implementation of the GNG algorithm that speeds up the sequential version several times. The speedup is increased as the number of neurons used for the representation grows.

In Table 2 some results can be appreciated with different number of neurons and input patterns with CPU and GPU runtimes and speedup obtained with GPU version with respect to the CPU ones. The GPU used was a GTX 480 NVIDIA graphic card with 480 cores, a global memory of 1.5 MB, and a bandwidth memory of 177.4 GB/sec.

3.5. GNG Maps Applications. The experiments showed how the GNG is able to adapt their topology to represent the input map space. In [19] we showed many applications of the GNG structure in order to improve 3D data representation and computer vision methods. We proved that the GNG representation partially discards most of the noisy values provided by these RGB-D sensors. In addition, we performed some experiments to show the improvement obtained using

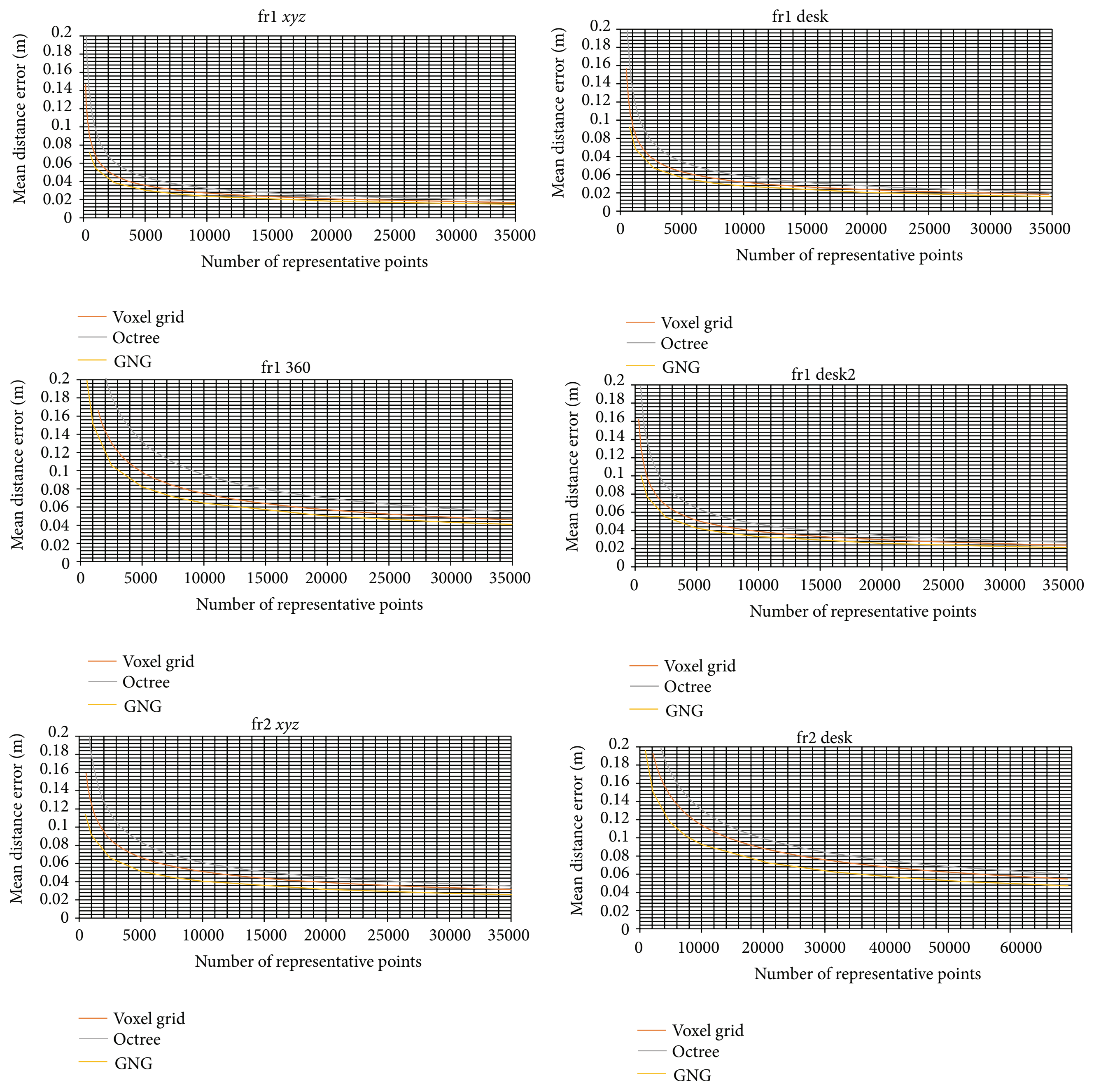


Figure 8: Closest neighbor distance mean errors of the different datasets.

the GNG representation on normal estimation using point neighborhood and keypoint detection and description.

Normal estimation methods are based on the analysis of the eigenvectors and eigenvalues of a covariance matrix created from the nearest neighbours and are very sensitive to noisy data. Therefore, we computed normals on raw and filtered point clouds in order to demonstrate how a simple 3D processing step like normal or curvature estimation is affected by the presence of noise.

Figure 11 shows how more stable normals are estimated using filtered point clouds produced by the GNG method. Normals are considered more stable as their distribution is smooth and also they have less abrupt changes in their directions. Moreover, a flat wall with some small changes in its surface was selected to appreciate changes in normal directions. We employed 20,000 neurons and 1,000 λ patterns as configuration parameters for the GNG method in the normal estimation experiment Figure 11(b).

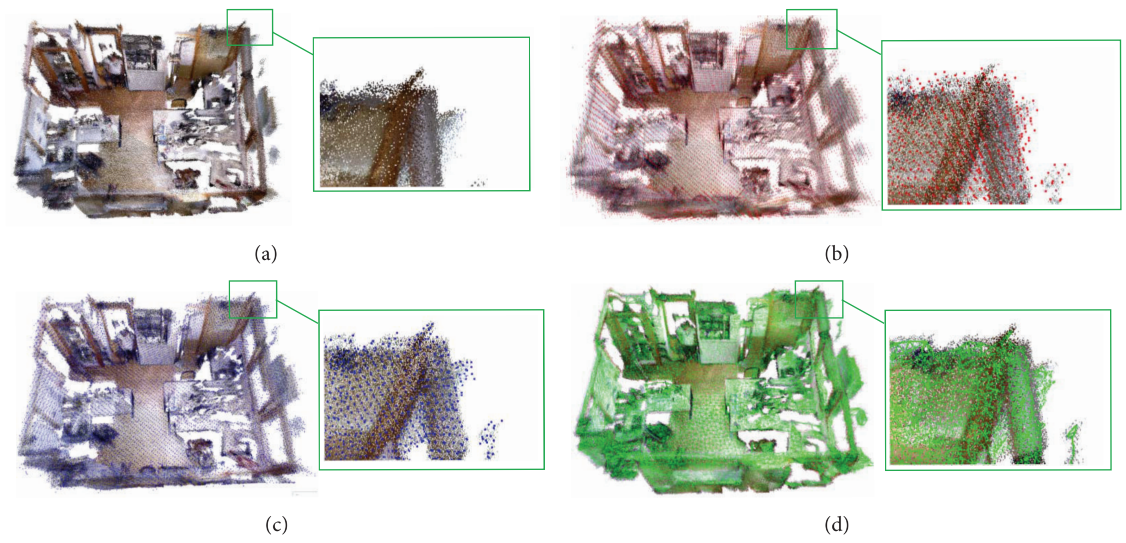


Figure 9: (a) Original point cloud map. (b) Octree reduction. (c) Voxel grid reduction. (d) GNG representation.

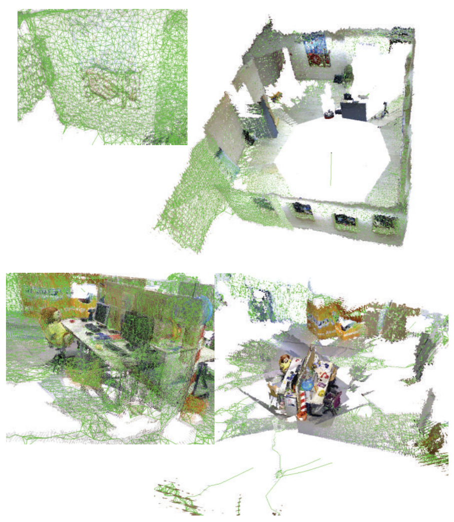


Figure 10: Other examples of GNG representation with two additional maps. Left part is a zoomed detail.

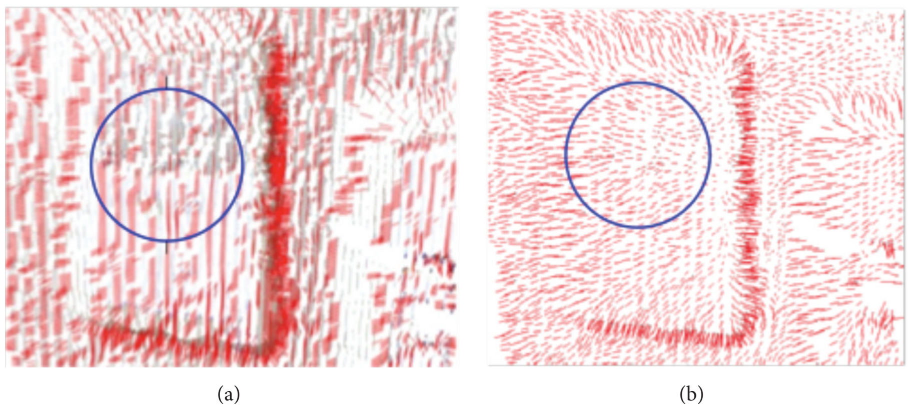


Figure 11: Normal estimation comparison. (a) Normal estimation on raw point cloud. (b) Normal estimation on filtered point cloud produced by the GNG method.

Table 3: RMS deviation error (meters) is obtained using different detector-descriptor combinations. Combinations are computed on the original point cloud (raw) and different filtered point clouds using voxel grid and the proposed GNG method. Keypoint detector search radius is equal to 0.05 meters. Feature extractor search radius is equal to 0.02 meters.

		GNG			VG			Raw
		5000p	10000p	20000p	5000p	10000p	20000p	All points
SIFT3D	FPFH	0.168	0.092	0.231	0.239	0.073	0.139	0.103
	CSHOT	0.052	0.037	0.019	0.063	0.07	0.037	0.039
	PFH	0.185	0.367	0.255	0.54	0.171	0.375	0.082
	PFHRGB	0.106	0.029	0.057	0.08	0.05	0.027	0.041
Harris3D	FPFH	0.151	0.114	0.079	0.404	0.088	0.18	0.128
	CSHOT	0.085	0.046	0.052	0.038	0.033	0.069	0.066
	PFH	0.109	0.177	0.153	0.305	0.097	0.469	0.144
	PFHRGB	0.054	0.047	0.042	0.033	0.058	0.043	0.093
Tomasi3D	FPFH	0.049	0.054	0.27	0.383	0.127	0.148	0.659
	CSHOT	0.043	0.023	0.063	0.022	0.047	0.046	0.049
	PFH	0.189	0.308	0.319	0.067	0.258	0.123	0.765
	PFHRGB	0.112	0.121	0.066	0.066	0.049	0.078	0.062
Noble3D	FPFH	0.143	0.186	0.199	0.387	0.188	0.289	0.114
	CSHOT	0.098	0.052	0.063	0.085	0.059	0.048	0.06
	PFH	0.273	0.127	0.239	0.073	0.244	0.188	0.099
	PFHRGB	0.188	0.117	0.064	0.096	0.082	0.079	0.077
Lowe3D	FPFH	0.143	0.117	0.076	0.387	0.188	0.093	0.203
	CSHOT	0.065	0.052	0.062	0.04	0.059	0.107	0.067
	PFH	0.273	0.12	0.239	0.073	0.244	0.167	0.24
	PFHRGB	0.188	0.135	0.016	0.077	0.082	0.076	0.027
Curvature3D	FPFH	0.099	0.228	0.113	0.228	0.103	0.093	—
	CSHOT	0.048	0.032	0.022	0.033	0.042	0.05	—
	PFH	0.262	0.151	0.123	0.244	0.101	0.057	—
	PFHRGB	0.139	0.071	0.053	0.068	0.083	0.023	—

In order to test the keypoint detector/descriptor improvement, we performed a transformation estimation algorithm and we compared the GNG results against the voxel grid representation and against the entire source point cloud. We used the available descriptors [18, 20, 21] and detectors [22] provided by the Point Cloud Library [23]. The minimum, median, mean, and maximum of RMS transformation error (with respect to different keypoint detectors) are presented in Table 3. These results show how filtered point clouds using the GNG method generally improved the precision of the estimated transformation. Moreover, the worst estimated transformations (maximum errors) were also slightly improved using the GNG compared to the other techniques.

4. Conclusions

3D maps obtained from RGB-D data are useful for robotics tasks, like robot navigation. But this kind of maps contains a huge amount of data, which must be reduced to properly process the map. In this paper, we have presented a method to represent and reduce 3D maps. Our method is based on a GNG neural network which is adapted to the 3D input space. The experiments carried out demonstrated the validity of our method, as it provided better adaptation than two of the most used methods for these tasks: voxel grid and Octree.

As future works, we propose to extend our method to provide a useful map for robot navigation. We also plan to provide the GNG with a way to revert the reduction or compression of the points, storing information in the neurons neighborhood (color, point distribution, etc.).

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was partially funded by the Spanish Government DPI2013-40534-R grant. Experiments were made possible with a generous donation of hardware from NVDIA.

References

[1] D. Fox, S. Thrun, and W. Burgard, *Probabilistic Robotics*, The MIT Press, Cambridge, Mass, USA, 2005.

[2] M. Herbert, C. Caillas, E. Krotkov, I. S. Kweon, and T. Kanade, "Terrain mapping for a roving planetary explorer," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, pp. 997–1002, 1989.

[3] R. Triebel, P. Pfaff, and W. Burgard, "Multi-level surface maps for outdoor terrain mapping and loop closing," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '06)*, pp. 2276–2282, October 2006.

[4] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[5] P. Stepan, M. Kulich, and L. Preucil, "Robust data fusion with occupancy grid," *IEEE Transactions on Systems, Man, and Cybernetics C*, vol. 35, no. 1, pp. 106–115, 2005.

[6] B. Schiele and J. L. Crowley, "A comparison of position estimation techniques using occupancy grids," *Robotics and Autonomous Systems*, vol. 12, no. 3-4, pp. 163–171, 1994.

[7] C. I. Connolly, "Cumulative generation of octree models from range data," in *Proceedings of the 1st International Conference on Robotics and Automation*, vol. 1, pp. 25–32, March 1984.

[8] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: a probabilistic, flexible, and compact 3D map representation for robotic systems," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '10)*, 2010.

[9] L. Kobbelt and M. Botsch, "A survey of point-based techniques in computer graphics," *Computers and Graphics*, vol. 28, no. 6, pp. 801–814, 2004.

[10] L. Wang, J. Chen, and B. Yuan, "Simplified representation for 3D point cloud data," in *Proceedings of the IEEE 10th International Conference on Signal Processing (ICSP '10)*, pp. 1271–1274, October 2010.

[11] D. Viejo, J. Garcia, M. Cazorla, D. Gil, and M. Johnsson, "Using GNG to improve 3D feature extraction: application to 6DoF egomotion," *Neural Networks*, vol. 32, pp. 138–146, 2012.

[12] T. Kohonen, *Self-Organizing Maps*, vol. 30 of *Springer Series in Information Sciences*, Springer, Berlin, Germany, 1995.

[13] B. Fritzke, "A growing neural gas network learns topologies," in *Advances in Neural Information Processing Systems*, vol. 7, pp. 625–632, MIT Press, 1995.

[14] T. Martinetz, "Competitive hebbian learning rule forms perfectly topology preserving maps," in *Proceedings of the International Conference on Artificial Neural Networks (ICANN '93)*, pp. 427–434, 1993.

[15] T. Martinetz and K. Schulten, "Topology representing networks," *Neural Networks*, vol. 7, no. 3, pp. 507–522, 1994.

[16] G. Xu, B. Mourrain, R. Duvigneau, and A. Galligo, "Analysis-suitable volume parameterization of multi-block computational domain in isogeometric applications," *Computer-Aided Design*, vol. 45, no. 2, pp. 395–404, 2013.

[17] S. Orts, J. Garcia-Rodriguez, D. Viejo, M. Cazorla, and V. Morell, "GPGPU implementation of growing neural gas: Application to 3D scene reconstruction," *Journal of Parallel and Distributed Computing*, vol. 72, no. 10, pp. 1361–1372, 2012.

[18] F. Tombari, S. Salti, and L. Di Stefano, "A combined texture-shape descriptor for enhanced 3D feature matching," in *Proceedings of the 18th IEEE International Conference on Image Processing (ICIP '11)*, pp. 809–812, September 2011.

[19] S. Orts-Escolano, *A Three-Dimensional Representation Method for Noisy Point Clouds Based on Growing Self-Organizing Maps Accelerated on GPUs*, University of Alicante, 2014.

[20] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 08)*, pp. 3384–3391, Nice, France, September 2008.

[21] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '09)*, pp. 3212–3217, Kobe, Japan, May 2009.

[22] I. Sipiran and B. Bustos, "Harris 3D: a robust extension of the Harris operator for interest point detection on 3D meshes," *Visual Computer*, vol. 27, no. 11, pp. 963–976, 2011.

[23] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *Proceeding of the IEEE International Conference on Robotics and Automation (ICRA '11)*, pp. 1–4, Shanghai, China, May 2011.

Advances in Operations Research
Advances in Decision Sciences
Journal of Applied Mathematics
Algebra
Journal of Probability and Statistics
The Scientific World Journal
International Journal of Differential Equations
International Journal of Combinatorics
Advances in Mathematical Physics
Journal of Complex Analysis
Journal of Mathematics
Mathematical Problems in Engineering
Abstract and Applied Analysis
Discrete Dynamics in Nature and Society
International Journal of Mathematics and Mathematical Sciences
Journal of Discrete Mathematics
Journal of Function Spaces
International Journal of Stochastic Analysis
Journal of Optimization