2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

3D Maps Representation using GNG

Vicente Morell Miguel Cazorla Sergio Orts-Escolano José García-Rodríguez

Abstract— **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 becomes unmanageable. In this work, we focus on 3D maps representation and we propose the use of a Growing Neural Gas (GNG) network as a 3D representation model of the input data. GNG method is able to represent the input data with a desired amount of neurons while preserving the topology of the input space. Experiments show how GNG method yields better input space adaptation than other state-of-the-art 3D map representation methods.**

I. Introduction

A 3D point is comprised of (X, Y, Z) representing the spatial coordinates. When color information (R, G, B) is available for each point, it is referred as RGB-D data. RGB-D cameras provide those kind of data and nowadays they are very popular due to their low value, like the Kinect sensor. At each time, a RGB-D sensor could provide more than 300,000 3D points. In mobile robotics, there is a fundamental task that must be carried out: 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, first transforming each observation with respect to common coordinates frame. A map is useful to make subsequent tasks, like localization, navigation, etcetera. When using RGB-D data as observations, it is referred as RGB-D mapping and RGB-D maps.

RGB-D maps amount of data is huge as the number of poses is high. In a typical map with 10.000 poses, the data could be more than 3 billions of 3D points which is unaffordable for representation and for other tasks. 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 this map, as it would be used in other tasks.

Elevation maps were a commonly used structure in the past [2], [3]. These elevation maps are represented using a regular 2D cell grid where each cell value represent the elevation or height of the surface of that space. This compact model allows a simply representation of large areas but with lower level of detail. Burgar et al. [4] present an extension of the height maps in order to represent different surfaces at different heights. This Multi-Level 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 applications like robotics navigation.

Vicente Morell and Miguel Cazorla are with the Computer Science and Artificial Intelligence Department of the University of Alicante (email: {vmorell, miguel}@dccia.ua.es).

Sergio Orts-Escolano and Jose Garcia-Rodriguez are with the Department of Computer Technology of the University of Alicante (email: {sorts, jgarcia}@dtic.ua.es).

Following this idea of 3D space representation, some other structures are used like occupancy grids or Octrees. Occupancy grids represent the entire space as a 3D cell grids. The cell information could be a simply value of occupancy or contain more complex information as the probability of occupancy. Several works in mobile robotics use this structure as a base of their applications [5], [6], [7]. Another common structure is the Octree [8]. The Octree is a tree structure in which each internal node has eight children. Each node of the tree is subdivided into eight new nodes until a certain condition is reached. This structure allows representation of both occupied and empty space in the area represented by the Octree and allows some optimized operations like closest point searching or occupancy checking. In [9], an Octree based framework called OctoMap is presented. They use a probabilistic occupancy estimation where areas of the space are represented as occupied, empty or uncertain. Another common used structure are the Voxel Grid (VG). The VG down-sampling technique is based on the input space sampling using a grid of 3D voxels. This technique has been used traditionally in the area of computer graphics to subdivide the input space and reduce the number of points [8], [10].

Wang et al. [11] present a feature based 3D point cloud simplification. They detect the points with more information (big curvatures) and they subsample the rest of the points using a uniform spherical sampling method. Therefore, they preserve the key points and subsample the points with less curvature information. This method is good to subsample 3D point clouds of object surfaces, but it will not work on scene maps due to the spherical sampling, and that the feature selection process is usually harder and problem dependent.

Another approaches use self-organizing maps in order to reduce the input space. Viejo et al. [12] use a Growing Neural Gas (GNG) algorithm to filter and the reduce single frontal point clouds. In this paper, we propose to extend that work to manage complete maps. The GNG will be able to adapt to the complete map, reducing its size, keeping the input topology and providing better adjustment than existing methods. To validate our method, we show several experiments comparing our method with state-of-the-art methods for reducing map size.

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

II. 3D Representation methods

One way of selecting points of interest in 3D point clouds is to use 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 commonly used data structures in order to compare our proposal application.

A. GNG Method

A common way to achieve a multi-dimensional 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 mapping, a representation of the input patterns is achieved that in post-processing stages allows one to exploit the similarity relations of the input patterns. However, most common approaches are not able to 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 preserving as for example in the case of Kohonen's algorithm [13]. The approach presented in this paper is based on self-organising networks trained using the Growing Neural Gas learning method [14], an incremental training algorithm. The links between the neurons in the network are established through competitive hebbian learning [15]. As a result the algorithm can be used in cases where the topological structure of the input pattern is not known a priori and yields topology preserving maps of feature manifold [16].

In GNG, nodes in the network compete for determining the set of nodes 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 to the input sample point which is the signal 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 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, as for example evaluation of the optimal network topology, a predefined networks size or a deadline.

Next, we describe the growing neural gas algorithm and the ending condition as used in this work. The network is specified as:

- 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.
- A set of edges (connections) between pairs of neurons. These connections are not weighted and their purpose is to define the topological structure. An edge aging scheme is used to remove connections that are invalid due to the motion of the neuron during the adaptation process.

The GNG learning algorithm to map the network to the input manifold is as follows:

1) Start with two neurons a and b at random positions w_a and w_b in R^d.
2) Generate at random an input pattern ξ according to the data distribution $P(\xi)$ of each input pattern.
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 such as:

$$\triangle error(s_1) = \|w_{s_1} - \xi\|^2 \tag{1}$$

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

$$\triangle w_{s_1} = \epsilon_w(\xi - w_{s_1}) \tag{2}$$

$$\triangle w_{s_n} = \epsilon_w(\xi - w_{s_n}) \tag{3}$$

for all direct neighbors n of s_1.

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) Every certain number λ of input patterns generated, insert a new neuron as follows:
   - Determine the neuron q with the maximum accumulated error.
   - Insert a new neuron r between q and its further neighbor f:

$$w_r = 0.5(w_q + w_f) \tag{4}$$

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

10) Decrease the error variables of neurons q and f multiplying them with a consistent α. Initialize the error variable of r with the new value of the error variable of q and f.
11) Decrease all error variables by multiplying them by a constant γ.
12) If the stopping criterion is not yet achieved (in our case the stopping criterion is the number of neurons), go to step 2.

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 leafs on Octree based methods)

B. Octree based method

Most 3D point clouds mapping algorithms usually use the spatial organization of the points to encode them in a structure like an Octree in order 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 reach the desired precision. One of the main features of the octree representation is that nodes without input space points are not subdivided and therefore those leaf nodes represent a empty volume of the space and this feature can be 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 usually the mean or centroid of the cell inner points offers better results despite it has a higher computational and memory cost.

C. Voxel Grid method

The VG down-sampling technique is based on the input space sampling using a grid of 3D voxels. 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 by using one of the approaches described in the 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 than other sub-sampling 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 inside a voxel; and sensitivity to noisy input spaces.

D. Comparative

In this subsection we briefly describe the main differences of the below described methods. The GNG representation provides a set of neurons and their neighbors. This 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 this 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 fast reduction method, but it does not have any of the advantages of the Octree structure or GNG model like neighbor searching.

Figure 1 shows a 2D description of the representative selection points of the described methods. We observe as the GNG method assign more neurons on high density input values like in the bottom left area than the Voxel Grid and Octree methods. We also observe how the GNG is able to reduce some noisy values like the point near the center in contrast with the representative used in the VG and Octree methods.

III. Experimentation

In this section we are going to 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, quantitatively and qualitatively.

A. 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 data and ground-truth data with the goal of evaluating visual odometry and visual SLAM systems. The dataset contains the color and depth images of 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 are recorded in a typical office environment and the *fr2* datasets are recorded in a large industrial hall. Figure 2 shows a ground-truth reconstruction map of the “fr1 360” scene.

Table I 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 of the “fr1 xyz” to 6 million of the“fr1 desk”.

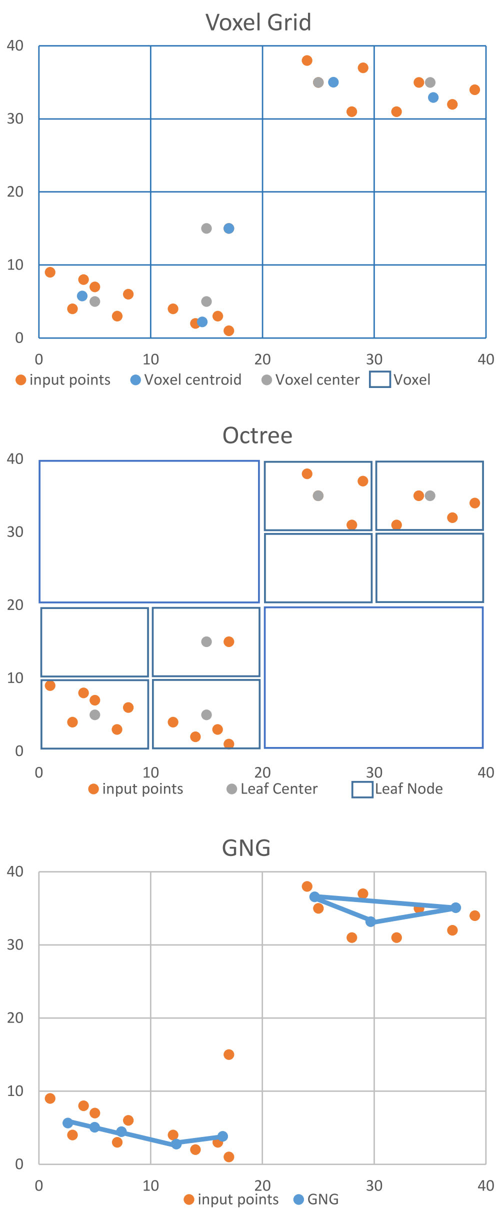

Fig. 1. Two dimensional samples of the three tested methods.

Dataset	Number of Input Points
fr1 xyz	1049739
fr1 desk	1952544
fr1 360	2357039
fr1 desk2	2751402
fr2 xyz	3492032
fr2 desk	5841800

TABLE I

NUMBER OF POINTS OF EACH GROUND-TRUTH MAP DATASET.

B. *Quality adaptation experiment*

As we previously mentioned we are going to compare the proposed GNG adaptation against two common used data structures in the state-of-the-art, Octree and Voxel Grid. The implementation of both methods are included in the

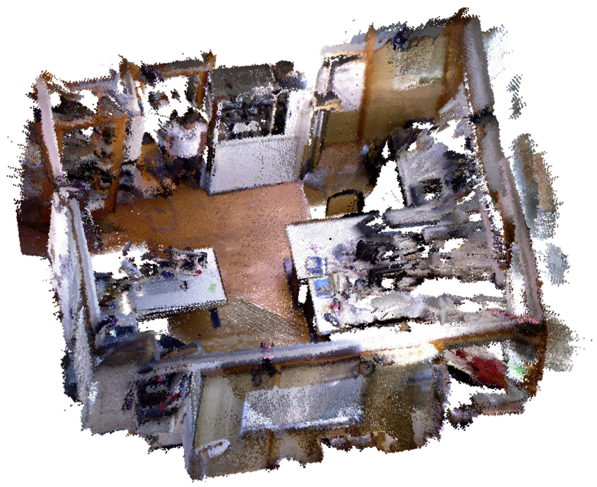

Fig. 2. Example of the "fr1 360" ground-truth point cloud map.

Point Cloud Library [1] (PCL). 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 non-empty voxel. Both implementations use a resolution parameter that represents the size of the voxel in the VG method and the side of the leaf cell of the Octree implementation. The GNG results are obtained using 10000λ input patterns.

We extensively tested the implemented methods using different numbers of representatives. As the three tested methods are reducing the amount of real noisy point cloud maps, it is needed to know the real distance from the selected representatives to the original input space. This measure specifies how close the representations are from the original model. In order to have a quantitative measure of the input space adaptation of the generated map, we computed the Mean Error (ME) of the reduced map against sampled points (input space).

$$ME = \frac{1}{|V|} \sum_{p \in V} \min_{q \in A} \|p - q\| \tag{5}$$

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

Figure 3 shows the RMS errors of the three methods on the six different tested maps. We observe that the three methods have the same behavior on the different datasets. The Octree method gets the worse results probably because 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 the use of the center of the voxel or cell. It is important to point out again that the representative selection used in this comparative is given by the implementations but both Octree and Voxel Grid

[1]The Point Cloud Library (or PCL) is a large scale, open project [18] for 2D/3D image and point cloud processing.

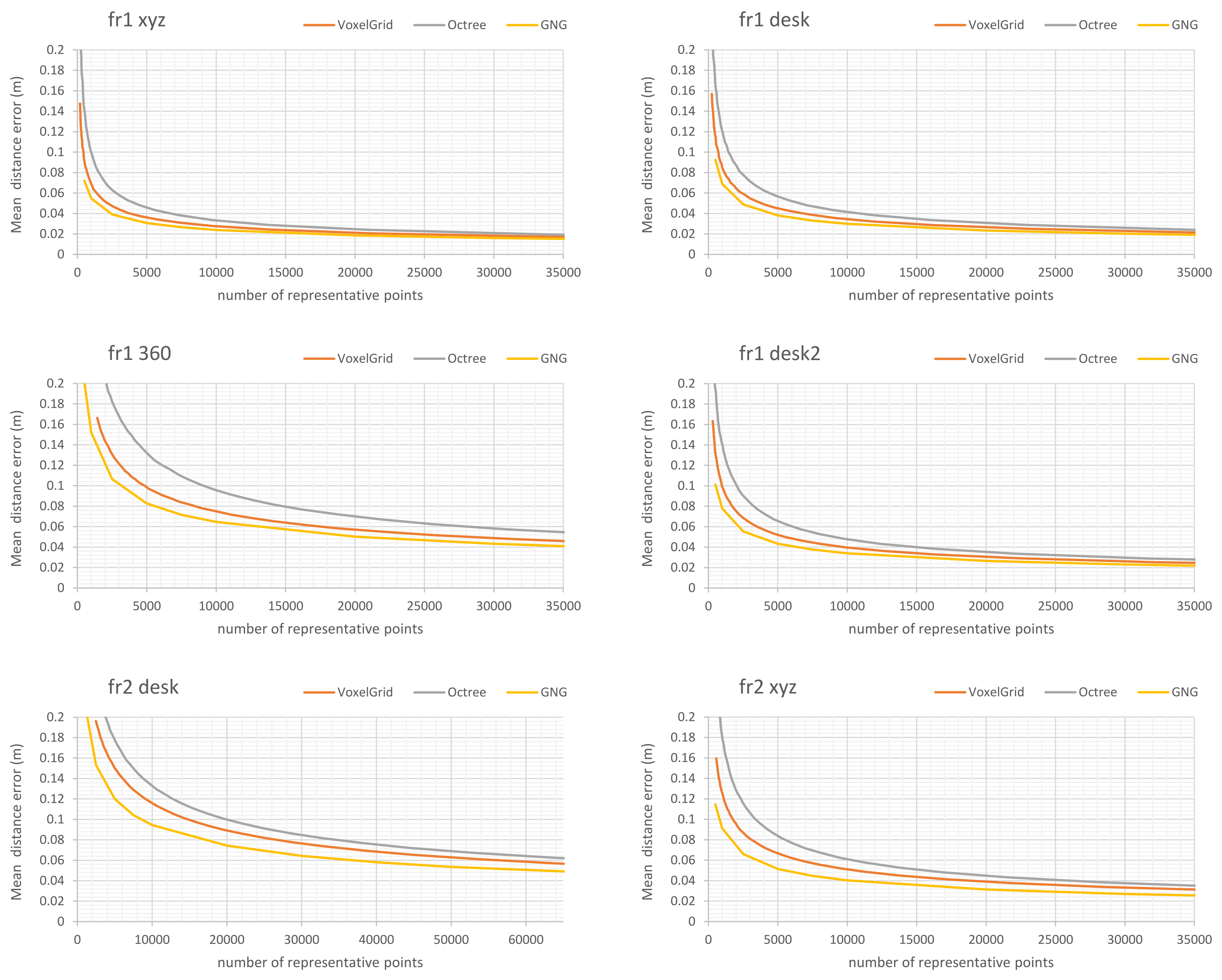

Fig. 3. Closest neighbor distance mean errors of the different datasets.

methods can use both representative selection strategy. GNG adaption shows the best results on all datasets. It is noticeable that the GNG gets lower errors on the different number of representatives but as the number of representatives increases the three different methods converge to the same error.

C. *Qualitative Results*

In this subsection we analyze qualitatively the results of the three different methods. Figure 4 shows the original map and the three representations of the tested method 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 obtains a more uniform distribution of the representatives, but the error adaption is worse, as we saw in Figure 1 and the Mean Error graphs. The Voxel grid representation gets similar results than the Octree where the points are uniformly distributed like the points of the floor but it gets better results on the border points than 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 it reduces the impact of the noisy points and undefined borders on the reduced representation. We also observe the inherent triangulation of the space that the GNG algorithms gets of the neighborhood of the neurons.

IV. Conclusions

RGB-D 3D maps are useful for robotics tasks, like robot navigation. But this kind of maps contains a huge amount of data, which must be reduced to process properly 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 that has been adapted to the 3D input space. The experiments carried out show the validity of our method, as it provides better adaptation than two of the most used methods

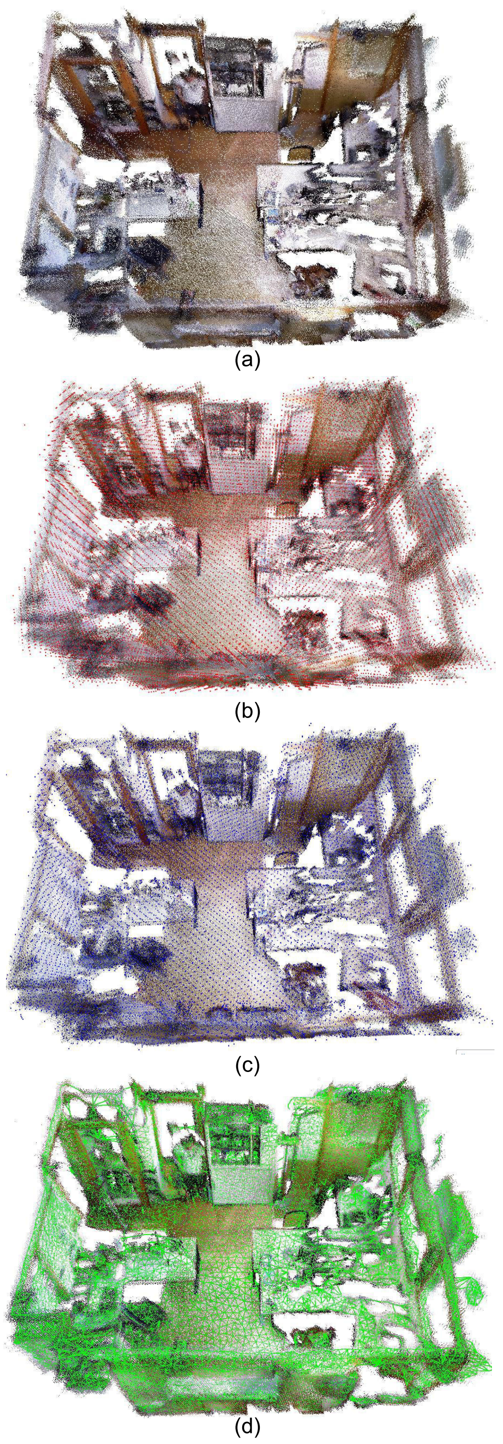

Fig. 4. a) Original point cloud map. b) Octree reduction. c) Voxel Grid reduction. d) GNG representation.

for this tasks: Voxel Grid and Octree.

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

Acknowledgements

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. F. Sebastian Thrun, Wolfram Burgard, *Probabilistic Robotics*. The Mit Press, 2005.

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

[3] I. S. Kweon and T. Kanade, "High resolution terrain map from multiple sensor data," in *Intelligent Robots and Systems '90. 'Towards a New Frontier of Applications', Proceedings. IROS '90. IEEE International Workshop on*, 1990, pp. 127–134 vol.1.

[4] R. Triebel, P. Pfaff, and W. Burgard, "Multi-level surface maps for outdoor terrain mapping and loop closing," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, 2006, pp. 2276–2282.

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

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

[7] B. Schiele and J. L. Crowley, "A comparison of position estimation techniques using occupancy grids," *Robotics and Autonomous Systems*, vol. 12, no. 34, pp. 163 – 171, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/092188909490023X

[8] C. I. Connolly, "Cumulative generation of octree models from range data," in *Proceedings of the First International Conference on Robotics and Automation*. IEEE, Mar 1984, p. 25.

[9] 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 *In Proc. of the ICRA 2010 workshop*, 2010.

[10] L. Kobbelt and M. Botsch, "A survey of point-based techniques in computer graphics," *Computers & Graphics*, vol. 28, pp. 801–814, 2004.

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

[12] D. Viejo, J. Garcia, M. Cazorla, D. Gil, and M. Johnsson, "Using GNG to improve 3d feature extraction-application to 6dof egomotion." *Neural Networks*, 2012.

[13] T. Kohonen, *Self-Organising Maps*. Springer-Verlag, 1995.

[14] B. Fritzke, *A Growing Neural Gas Network Learns Topologies*. MIT Press, 1995, vol. 7, pp. 625–632.

[15] T. Martinetz, "Competitive Hebbian Learning Rule Forms Perfectly Topology Preserving Maps," in *Proc. ICANN'93, Int. Conf. on Artificial Neural Networks*, S. Gielen and B. Kappen, Eds. London, UK: Springer, 1993, pp. 427–434.

[16] T. Martinetz and K. Schulten, "Topology representing networks," *Neural Networks*, vol. 7, no. 3, 1994.

[17] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[18] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.