A Study of a Soft Computing Based Method for 3D Scenario Reconstruction

Diego Viejo, Jose Garcia, Miguel Cazorla

Instituto de Investigación en Informática
University of Alicante
PO. Box 99, E-03080 Alicante, Spain
Tel.: +34-965-903400 ext: 2072
Fax: +34-965-903902

Abstract

Several recent works deal with 3D data in mobile robotic problems, e.g. mapping. Data comes from any kind of sensor (time of flight, kinect or 3D lasers) that provide a huge amount of unorganized 3D data. In this paper we detail an efficient approach to build complete 3D models using a soft computing method, the Growing Neural Gas (GNG). As neural models deal easily with noise, imprecision, uncertainty or partial data, GNG provides better results than other approaches. The GNG obtained is then applied to a sequence. We present a comprehensive study on GNG parameters to ensure the best result at the lowest time cost. From this GNG structure, we propose to calculate planar patches and thus obtaining a fast method to compute the movement performed by a mobile robot by means of a 3D models registration algorithm. Final results of 3D mapping are also shown.

Keywords: Egomotion, GNG, registration, 3D feature extraction

Email addresses: dviejo@dccia.ua.es (Diego Viejo), jgarcia@dtic.ua.es (Jose Garcia), miguel.cazorla@ua.es (Miguel Cazorla)

1. Introduction

In robotics, one of the main basic goal is the determination of the egomotion, i.e. the relative movement done by the robot between two consecutive poses. Using its internal sensors, the robot is able to estimate its odometry, but it is inaccurate and sometimes unavailable. Visual odometry [1] (sometimes also called pose registration) has been used as a good estimation for egomotion in the last years. It is a good starting point for automatic map building and for solving the Simultaneous Location And Mapping problem (SLAM) [2]. Our main goal in this work is to perform six degrees of freedom (6DoF) pose registration in semi-structured environments, i.e., man-made indoor and outdoor environments. Some examples of works that deal with this problem can be found in: [3], [4], [5], [6], [7], [8] and [9].

We use dense raw 3D data as input sets. Our method is developed for managing 3D point sets collected by any kind of sensor. For our experiments, we use three main data sources: a 2D SICK laser mounted on a sweeping unit for obtaining 3D data, an infrared time-of-flight (ToF) camera SR4000, and a Kinect sensor, mounted on a mobile robot. The sweeping laser provides 3D data with a low error and a higher range compared to ToF and Kinect sensor, but the data collection time of this sensor is much bigger than in the other two systems. One of the main objectives of this study is to create a generic method that can be used on any platform. Therefore, we do not use the information of the robot odometry since not all the robots have it.

We are also interested in dealing with outliers, i.e., environments with the presence of non-modeled moving objects, such as people, other vehicles, etc. Outliers also come from pose to pose where some data seen in the previous pose is not

observed in the next one, and the last pose presents data which has not been seen in the previous one. This challenge is hard to overcome as classic algorithms, like Iterative Closest Point (ICP) [10] and its variants, are very sensitive to outliers. Furthermore, handling raw 3D data is not suitable for most of the mobile robot methodologies, because usually this data contains a huge number of points. In this paper we use a soft computing based method to extract and model planar patches from 3D raw data [11]. Using this method we get two main advantages: first, a complexity reduction (when comparing with raw data) is done and time and memory consumptions are improved (we obtain over 500 planar patches from 100000 3D points); second, outliers are better overcome using these features, as points not supported by a planar patch are deleted. Planar patches are useful features as man-made environments are easily described with them. This provides a way to manage outliers, while other classical registration methods (like ICP) provide poor results in the presence of outliers.

Nevertheless, in some situations the planar patches extraction method can not obtain a complete environment model. As we will explain later in this article, this kind of problems arise when the 3D sensor used combines both, a short measurement range and a high measurement error. In these situations, we propose the use of a Growing Neural Gas [12, 13, 14]. By means of a competitive learning algorithm, it makes an adaptation of both the reference vectors of the neurons and the interconnection network among them [15], thus obtaining a mapping that tries to preserve the topology of an input space. Besides, they are capable of a continuous re-adaptation process even if new patterns are entered, with no need to reset the learning. These features are an easy way to represent fast and high quality 3D spaces, obtaining an induced Delaunay Triangulation of the input space very

useful to easily obtain features like corners, edges, etc. We modify the original GNG method to be applied to sequences: the GNG is adapted sequentially, i.e. the result in a given frame is taken as the input in the next frame. Modeling 3D scenes using GNG produces a more detailed result and thus further computations such as planar patches based egomotion are also improved. Traditionally, soft computing methods have been used in robotics for route planning or control [16], [17], [18], [19]. Some 3D reconstruction applications of neural networks can be found in [20], [21], [22], [23] and [24]. However, none of them processes 3D sequences of data.

In order to determine the limitations of our method, we have done several experiments with the GNG method. We also have compared our method with a robust ICP version, in order to show the advantages of using it.

The rest of the paper is organized as follows: first, Section 2 describes the physical systems used for the experiments and some methods and concepts used later in the paper are explained; then, Section 3 explains the GNG algorithm. Our method for the extraction of 3D features from GNG and their use in a problem of egomotion are discussed in Sections 4 and 5; Analysis of GNG parameters together with the results of applying our approach to the problem of mapping are shown in Section 6; We present our conclusions and future work in section 7.

2. Preliminaries

One of the goals of our work is to achieve the independence of our algorithm: it can be applied to any robot platform, any 3D measurement device and can be used in indoor and outdoor handmade environments. We present here the physical systems used in our experiments and some basic concepts used in the work.

2.1. Data acquisition

We have worked with several robot platforms, depending on the perception system used. In Figure 1, two of these platforms are shown. The left one is a Magellan Pro from iRobot used for indoor experiments. For outdoors we have used a PowerBot from ActiveMedia. Furthermore, PowerBot can carry heavy loads like the 3D sweeping laser unit. Both come with an onboard computer.

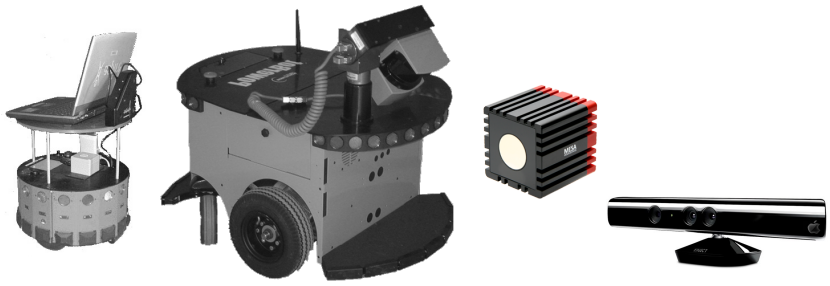

Figure 1: Mobile robots used for experiments. From left to right: Magellan Pro unit used for indoors; PowerBot used for outdoors. SR4000 camera used with both robots; Kinect sensor used in indoors.

We manage 3D data that can come from different sensor devices. For outdoor environments we use a 3D sweeping laser unit, a LMS-200 Sick laser mounted on a sweeping unit. Its range is 80 meters with an error of 1mm per meter. The main disadvantage of this unit is the data capturing time: it takes about one minute to get a complete frame. For indoor environments we use another two sensors. The first one is a SR4000 camera from Mesa Imaging, which is a time-of-flight camera, based on infrared light. Its range is limited to 5 or 10 meters, providing gray level color. Finally, a Kinect sensor has been included. This sensor provides 3D data together with RGB data, with a maximum range of 10 meters.

2.2. Basic concepts

We describe here some concepts used in this paper and review the state of the art. We are interested in calculating the egomotion between consecutive poses of a robot, using its 3D sensors. These sensors provide a 3D point set at each pose. Finding the egomotion of the robot is equivalent to the process of registration of these two point sets. Several works allow to find this registration. Two of the most used methods are the Iterative Closest Point (ICP) and Ransac method (both described below). The final goal of our work is to obtain an initial approximation for the Simultaneous Location And Mapping problem (SLAM). So, we first describe this term. SLAM is defined as the problem of building a map using the robots sensors, or improving an existing map, while at the same time localizing the robot within this map. Both questions cannot be solved independently, which reminds the chicken-egg problem. In this problem, robot actions increase the uncertainty in the robot position (location), while perception reduces that uncertainty. Uncertainty is reduced when a previously visited place is seen again. This means that the proposed methods are highly dependent on perception. In this work, we present a method which improves the perception, reducing the error in calculating correspondences.

The Iterative Closest Point (ICP) is a classical algorithm used to register two 3D point sets. It iteratively revises the transformation (translation, rotation) needed to minimize the distance between the points of two 3D point sets. Given the two sets as input, an initial transformation (can come from odometry) and a stopping criteria (minimum error, number of iterations), ICP applies iteratively these steps:

- Apply the current transformation to one point set.
- Match points from one set to the other, using a nearest neighbor criteria.

- Estimate transformation parameters (translation and rotation) using a mean square cost function.

This method is not always able to find the optimal transformation between sets. One reason for this is the initial transformation: giving different initial transformation the method could finish in a different solution. Furthermore, the presence of outliers provides a bad registration due to, mainly, the use of mean square. There are several works that try to minimize the outliers influence. A survey on ICP based methods can be found in [25].

Another method used for this task is the RANdom SAmple Consensus (RANSAC) algorithm [26]. It is an iterative method that estimates the parameters of a mathematical model from a set of observed data which contains outliers. In our case, we look for a 3D transformation (our model) that best explain the data (matches between 3D features). Thus, we do need a set of matches between points (or features) in both sets. This method is used when we have visual features which can be used to find the most probable (or closest) feature in the other set. At each iteration of the algorithm, a subset of data elements (matches) is randomly selected. These elements are considered as inliers; a model (3D transformation) is fitted to these elements; all other data is then tested against the fitted model and included as inliers if its error is below a threshold; if the estimated model is reasonably good (its error is low enough and it has enough matches), it is considered as a good solution. This process is repeated a number of times and then, the best solution is returned. It is not suitable for our problem, as we use raw 3D points and the only information provided are the coordinates.

3. GNG Algorithm

In this section, we review GNG and highlight the main parts of the algorithm as used in this work. 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 inputs 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.

3.1. Point cloud sequences representation

GNG has been adapted to represent Point Cloud Sequences using models learnt from previous data acquisitions, we have introduced several improvements to the network in order to accelerate the representation and allow the architecture to work faster.

The main difference with the GNG algorithm is the omission of insertion/deletion actions (steps 8 to 11) after the first frame. Since no neurons are added or deleted the system keeps correspondence during the whole sequence, solving intrinsically the problem of correspondence. For the initial moment t_0 the representation is obtained making a complete adaptation of a GNG. However, for the

following frames the previous network structure is employed. So, the new representation of the object is obtained by performing the iteration of the internal loop of the learning algorithm of the GNG, relocating the neurons and creating or removing edges. This adaptive method is also able to face real-time constraints, because the number λ of times that the internal loop is performed can be chosen according to the time available between two successive frames that depends on the acquisition rate. The mean time to obtain a GNG on a frame is about 10ms., using the adaptive method.

For the experiments, the GNG parameters used are: $N = 2000$, $\lambda = 2000$, $\epsilon_w = 0.1$, $\epsilon_n = 0.001$, $\alpha = 0.5$,$\alpha_{max} = 250$. In Figure 2 a result of applying GNG to a 3D points from a SR4000 is shown, in Figure 3 the same for kinect data and, finally, Figure 4 for 3D laser data.

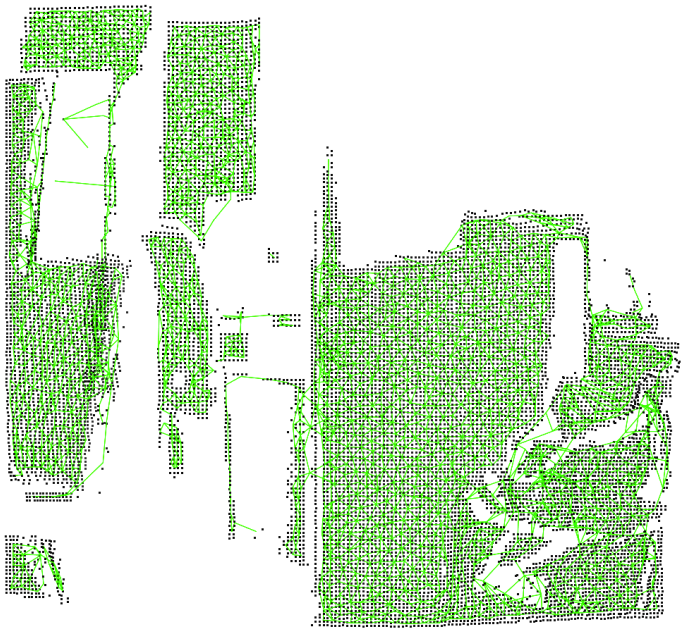

Figure 2: Applying GNG to SR4000 data set.

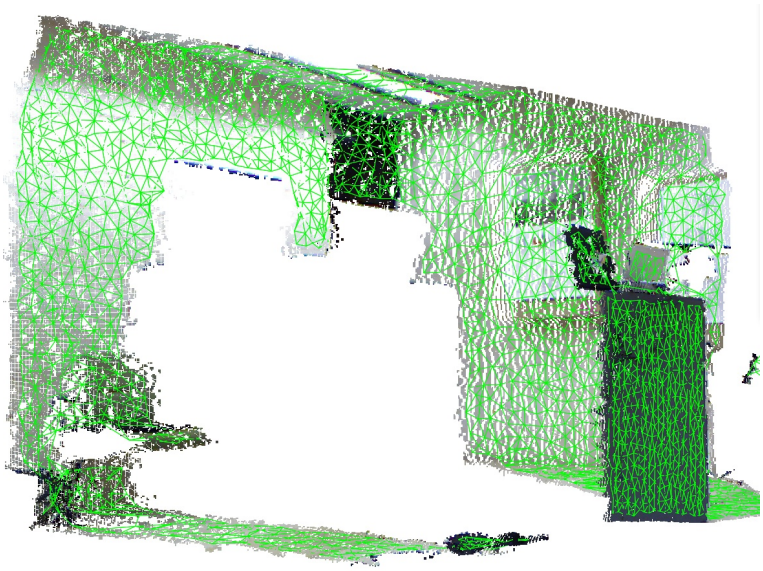

Figure 3: Applying GNG to kinect set.

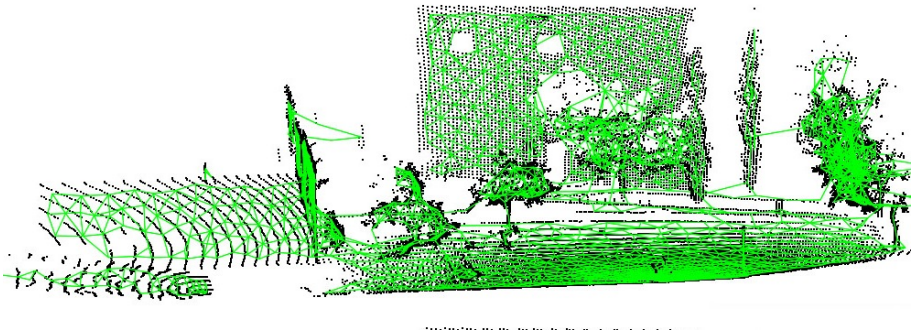

Figure 4: Applying GNG to 3D laser set.

GNG provides a reduction of the input data, while preserving its structure. This gives us two advantages. First, we have to process less points, so we speed up the next step of feature extraction. Second, outliers are reduced. Outliers are one of the main source of errors in this kind of application.

4. Features extraction method

We can reduce the size of the input sets at the same time we obtain new information of the scenes by modeling the data sets. To do this we calculate the normal vector to the surfaces for each of the points of the set. Normal vectors are estimated from a local area around each 3D point using a Singular Value Decomposition (SVD) [27]. Using this approach, when the underlying surface is a plane, the minimum singular value is quite smaller than the other two singular values, and the singular vector related to the minimum singular value is the normal vector of the surface at this point. From this information we can label each point in a 3D scene with two possible categories: belonging to a planar surface, when one of the singular values is much smaller than the others, or, in other case, as belonging to a not defined object.

This process retrieves the underlying surface normal vector of a given set of points. Furthermore, a threshold called thickness [27] can be defined from singular values in order to determine in which situations a point, as well as its neighborhood, belong to a planar surface or not. This thickness value can be used to measure the fitting of a 3D point set to a plane. The lower thickness value we find, the better is the fitting between points and planar surface. The size of the window used to obtain neighbor points has an important impact on the results. As it is considered in [28], sample density of 3D laser range finder data presents large variations due to the divergence of consecutively sampled beams. In general, this characteristic is present in any 3D data set, independently on the sensor used. A complete study on the impact of different window sizes was performed in [11]. Summarizing, a depth-based adaptive window provides better results. Depending on the sensor measurement error, this window has to be setup at different starting

sizes. The bigger the measurement error is, the bigger window size has to be set up.

Using SVD based normal vector estimation method we can obtain a model that represents the planar surfaces in the scene. We propose an optimal method that can obtain a planar patch model from a 3D point set in O(log n). This method is based on automatic seed selection methods [29] [30]. The idea consists in performing a selection of the most representative points in the whole 3D scene. These selected points must belong to planar surfaces. To ensure this we use the thickness value. In order to find out the most representative points, we randomly select points in the scene until all points are visited. For each non-visited point we compute its normal vector and thickness value. If its thickness value is low enough, the point is inserted into the most representative points list and its neighbors inside the window used for computing its normal vector are marked as visited. When this process ends, the planar patches model is directly computed from the most representative points and its normal vectors. The size of the planar patches depends on the size of the window around a point used to compute its normal vector.

As we stated before, the window size is a key factor. It depends on both the depth of the point and the 3D sensor measurement error. Bigger windows will produce better results for noisy 3D sets but also will discard small objects, compared to window size, in the scene. This lack of small details may lead to problems in further computations, specially when 3D sensors with a short range (up to 10 meters) are used. To overcome this problem we introduce the use of GNG in order to improve the feature extraction method. As we have explained in the previous section, GNG produces a Delaunay Triangulation which can be used as a representation for the points neighborhood. In this way we can state the

neighbor searching window size according to the GNG model and produce more detailed and accurate planar patches descriptions. Figure 5 shows planar patches extraction from a 3D image obtained with a SR4000 camera. Bottom image shows the results of combining GNG with the features extraction procedure. It can be compared with the upper image in which no GNG has been used. Patches are represented by blue circles. Radius of each patch depends on the size of the window used to compute the patch. The image at the bottom uses GNG to improve planar patches extraction. As a result we obtain a more detailed planar patches descriptions.

5. Using 3D models: 6DoF egomotion

In the previous section we described a method for building 3D models from scenes captured by a 3D sensor. Therefore, we want to use these models to apply them to a mobile robot applications in real 3D environments. The basic idea is to take advantage of the extra knowledge that can be found in 3D models such as surfaces and their orientations. This information is introduced in a modified version of an ICP-like algorithm in order to reduce the incidence of outliers in the results. ICP is widely used for geometric alignment of a pair of three-dimensional points sets. From an initial approximate transformation, ICP iterates the next three steps until convergence is achieved: first, closest points between sets are stated; then, best fitting transformation is computed from paired points; finally, transformation is applied. In the mobile robotics area, the initial transformation usually comes from odometry data.

Nevertheless, our approach does not need an initial approximate transformation like ICP based methods do. We can use the global model structure to recover

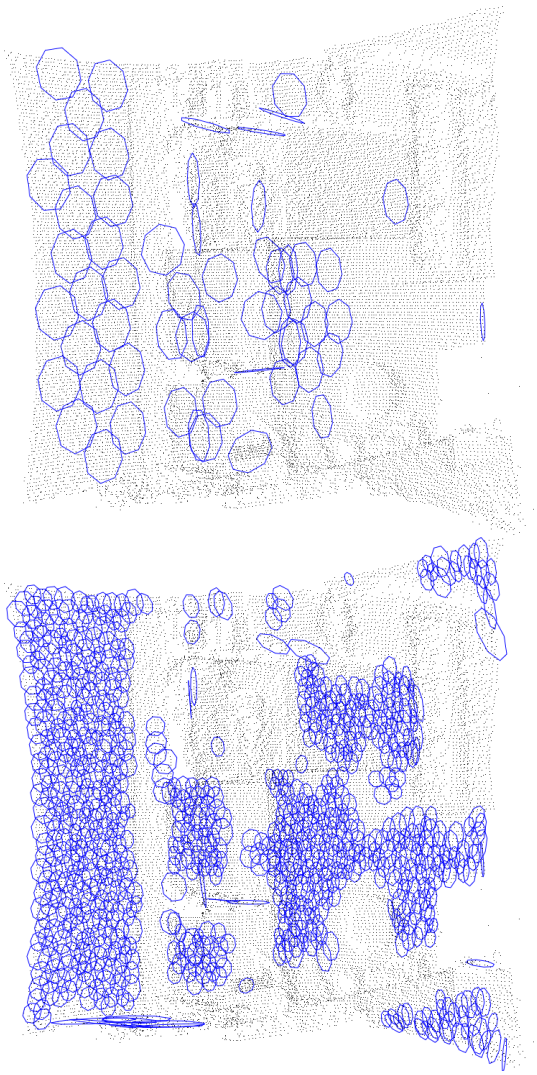

Figure 5: Planar patches extracted from SR4000 camera using and not using GNG.

the correct transformation. This feature is useful for those situations where no odometry is available or it is not accurate enough, such as legged robots. In our case, we are going to exploit both the information given by the normal vector of the planar patches and their geometric position. Whereas original ICP computes

both orientation and position at each iteration of the algorithm, we can take an advantage of the knowledge about planar patches orientation for decoupling the computation of rotation and translation. So, we first register the orientation of planar patch sets and when the two planar patches sets are aligned we address the translation registration. Next section will show experiments using the complete method.

6. Experimentation

For experimentation, we have used a robot with an arm, with which we can have a ground truth. The arm rotates around its axis, so we can give an angular displacement and then calculate the angle returned by the registration method. We calculate the registration between two consecutive poses and compare that registration with the ground truth. Thus, we are able to estimate the error produced by different algorithms. We have not made the same for translation, because it is difficult to find a ground truth in translation.

The first experiment tries to determine which are the best GNG parameters. To establish that, we apply the original feature extraction method without GNG and with GNG, but with different learning parameters. Figure 6 shows the result of calculating the angular error using the ground truth. We can observe that the best parameters are 500 input patterns and 1000 neurons, which will be used in the following experiments. Learning time increases as we increase both parameters.

For the next experiment, we compare our method with the ICP. We have a complete 2π rads. turn with poses taken every $\pi/45$ rads. We try to register consecutive poses with different angles. The result is shown in Figure 7. Our algorithm obtains better results than ICP and it also is faster.

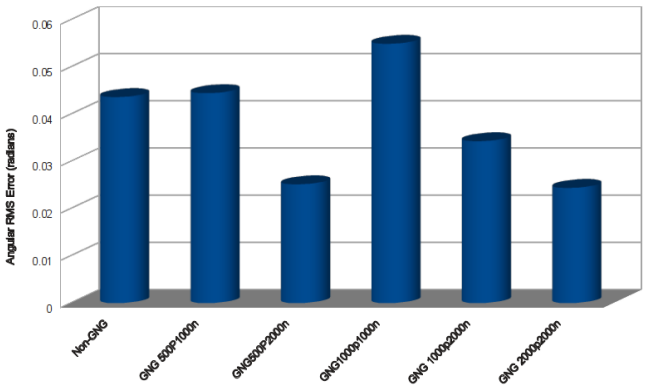

Figure 6: Angular error using different number of neurons and iterations.

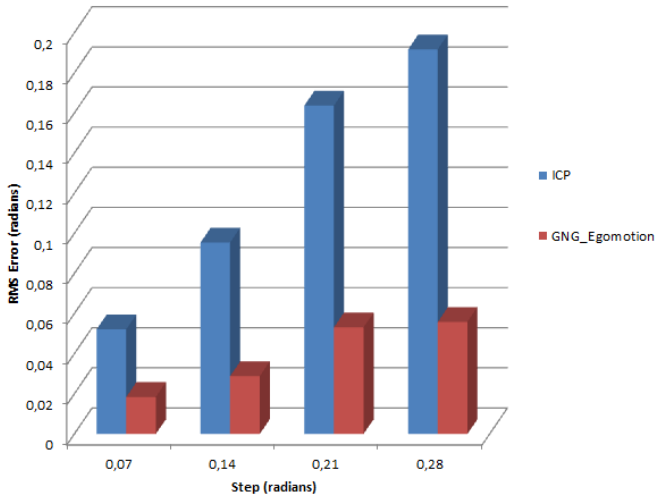

Figure 7: Comparison of our method and the ICP. The equivalent angles in degrees are 4, 8, 12 and 16, respectively.

In Figure 8 and 9 we show an example of 3D map building using this 6DoF egomotion approach and the ICP algorithm. The used ICP is a state-of-the-art implementation, using a outlier reduction [31], a subsampling of the input sets [32], and we also use a KD-tree structure to speed up. For this experiment, 100

3D images from a 5 meter range SR4000 camera have been used. The first figure shows a 3D view of the reconstructed environment using 6DoF egomotion using ICP. The second one, Figure 9, shows the advantage of using GNG to improve feature extraction. While in the first experiment the registration of the sequence was almost impossible, in the last one the reconstruction was reasonably good enough. Computing time for obtaining planar patches descriptions after applying GNG is almost the same as without GNG and is about 100 ms per image, but quality is much better.

Figure 8: 6DoF egomotion results using ICP.

Figure 9: Planar based 6DoF egomotion results using planar patches after computing a GNG mesh.

7. Conclusions and future work

We have presented a new soft computing method to create 3D models from unorganized raw 3D data. We do not need to know anything about the kind of sensor used to obtain data so the method we propose can be used with the most of the 3D scanner devices. In order to reduce the huge quantity of data we present two reduction phases.

First, we apply a Growing Neural Gas to the 3D points for two main reasons:

to reduce the number of points, thus accelerating the method, and to reduce noise from the capture system. We have experimented with different number of GNG parameters to determine the optimal ones. We have modified the original GNG method to represent 3D data sequences, which accelerates the learning algorithm and allows the architecture to work faster. This is possible because the system does not restart the map for each frame in the sequence, but instead readjusts the network structure starting from previous structure without inserting or deleting neurons.

Second, we have described an algorithm for computing the planar patches that fits with the planar surfaces in the 3D scene. This is a low complexity method that can be used to obtain online 3D models. Using the planar patches from this step, we are able to calculate registration (or egomotion). We show that our algorithm improves a classical registration method, as we state in the experimentation section.

The qualitative results of our method are demonstrated by applying a 6DoF egomotion algorithm that uses these models as input for computations. It have been proved that the use of GNG improves 6Dof mapping results making it possible to use this approach with any kind of 3D sensor.

As future work we plan to improve the accuracy and performance of our method in order to use it in a real 6D SLAM, using a GPU accelerated implementation.

Acknowledgments

This work has been supported by grant DPI2009-07144 from Ministerio de Ciencia e Innovacion of the Spanish Government, by the University of Alicante's

projects GRE09-16 and GRE10-35 and Valencian Government project GV/2011/034.

References

[1] D. Nister, O. Naroditsky, J. Bergen, Visual odometry, in: Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on, Vol. 1, 2004, pp. I–652 – I–659 Vol.1. doi:10.1109/CVPR.2004.1315094.

[2] M. Dissanayake, M. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, M. Csorba, A solution to the simultaneous localization and map building (slam) problem, Robotics and Automation, IEEE Transactions on 17 (3) (2001) 229–241.

[3] J. W. Weingarten, G. Gruener, R. Siegwart, Probabilistic plane fitting in 3D and an application to robotic mapping, in: Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on, Vol. 1, 2004, pp. 927–932 Vol.1.

[4] D. Hähnel, W. Burgard, S. Thrun, Learning compact 3d models of indoor and outdoor environments with a mobile robot, Robotics and Autonomous Systems 44 (1) (2003) 15–27.

[5] R. B. Rusu, Z. C. Marton, N. Blodow, M. Beetz, Learning Informative Point Classes for the Acquisition of Object Model Maps, in: Proceedings of the 10th International Conference on Control, Automation, Robotics and Vision (ICARCV), Hanoi, Vietnam, December 17-20, 2008.

[6] B. Steder, G. Grisetti, M. Van Loock, W. Burgard, Robust on-line model-based object detection from range images, in: Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems, IROS'09, IEEE Press, Piscataway, NJ, USA, 2009, pp. 4739–4744.

[7] B. Steder, G. Grisetti, W. Burgard, Robust place recognition for 3D range data based on point features, in: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 2010.

[8] S. May, D. Droeschel, D. Holz, S. Fuchs, E. Malis, A. Nüchter, J. Hertzberg, Three-dimensional mapping with time-of-flight cameras, J. Field Robotics 26 (11-12) (2009) 934–965.

[9] M. Ruhnke, B. Steder, G. Grisetti, W. Burgard, Unsupervised learning of compact 3d models based on the detection of recurrent structures, in: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010, pp. 2137–2142.

[10] P. Besl, N. McKay, A method for registration of 3-d shapes, IEEE Trans. on Pattern Analysis and Machine Intelligence 14 (2) (1992) 239–256. doi:10.1109/34.121791.

[11] D. Viejo, M. Cazorla, 3d model based map building, in: International Symposium on Robotics, ISR 2008, 2008.

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

[13] J. Garcia-Rodriguez, A. Angelopoulou, J. M. Garcia, A. Psarrou, Fast autonomous growing neural gas, in: IJCNN 2011. International Joint Conference on Neural Networks, 2011.

[14] J. Garcia-Rodriguez, J. M. Garcia-Chamizo, Surveillance and humancomputer interaction applications of self-growing models, Applied Soft Computing 11 (7) (2011) 44134431.

[15] T. Kohonen, Self-Organising Maps, Springer-Verlag, 2001.

[16] Q. Zhu, J. Hu, W. Cai, L. Henschen, A new robot navigation algorithm for dynamic unknown environments based on dynamic path re-computation and an improved scout ant algorithm, Applied Soft Computing 11 (8) (2011) 46674676.

[17] J. Faigl, L. Preucil, Inspection planning in the polygonal domain by self-organizing map, Applied Soft Computing 11 (2011) 50285041.

[18] M. P. Garcia, O. Montiel, O. Castillo, R. Sepulveda, P. Melin, Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation, Applied Soft Computing 9 (3) (2009) 11021110.

[19] T. Kondo, Evolutionary design and behavior analysis of neuromodulatory neural networks for mobile robots control, Applied Soft Computing 7 (1) (2007) 189202.

[20] A.-M. Cretu, E. Petriu, P. Payeur, Evaluation of growing neural gas networks for selective 3d scanning, in: Robotic and Sensors Environ-

ments, 2008. ROSE 2008. International Workshop on, 2008, pp. 108 –113. doi:10.1109/ROSE.2008.4669190.

[21] R. do Rego, A. Araujo, F. de Lima Neto, Growing self-organizing maps for surface reconstruction from unstructured point clouds, in: Neural Networks, 2007. IJCNN 2007. International Joint Conference on, 2007, pp. 1900 – 1905. doi:10.1109/IJCNN.2007.4371248.

[22] I. Ivrissimtzis, W.-K. Jeong, H.-P. Seidel, Using growing cell structures for surface reconstruction, in: Shape Modeling International, 2003, 2003, pp. 78 – 86. doi:10.1109/SMI.2003.1199604.

[23] M. I. Fanany, I. Kumazawa, A neural network for recovering 3d shape from erroneous and few depth maps of shaded images, Pattern Recogn. Lett. 25 (2004) 377–389. doi:10.1016/j.patrec.2003.11.001.

[24] Y. Holdstein, A. Fischer, Three-dimensional surface reconstruction using meshing growing neural gas (mgng), Vis. Comput. 24 (2008) 295–302. doi:10.1007/s00371-007-0202-z.

[25] J. Salvi, C. Matabosch, D. Fofi, J. Forest, A review of recent range image registration methods with accuracy evaluation, Image Vision Comput. 25 (5) (2007) 578–596.

[26] M. A. Fischler, R. C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Commun. ACM 24 (6) (1981) 381–395. doi:http://doi.acm.org/10.1145/358669.358692.

[27] M. Martín, J. Gómez, E. Zalama, Obtaining 3d models of indoor environments with a mobile robot by estimating local surface directions, Robotics and Autonomous Systems 48 (2-3) (2004) 131–143.

[28] A. R. H. Cole, David M., P. M. Newman, Using naturally salient regions for slam with 3d laser data, Proc. of the IEEE International Conference on Robotics and Automation.

[29] F. Y. Shih, S. Cheng, Automatic seeded region growing for color image segmentation, Image and Vision Computing 23 (2005) 877–886.

[30] J. Fan, M. Zeng, M. Body, M. S. Hacid, Seeded region growing: an extensive and comparative study, Pattern Recognition Letters 26 (2005) 1139–1156.

[31] K. Pulli, Multiview registration for large data sets, in: Proc. Second International Conference on 3-D Digital Imaging and Modeling, 1999, pp. 160–168. doi:10.1109/IM.1999.805346.

[32] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, D. Fulk, The digital michelangelo project: 3d scanning of large statues, in: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co., 2000, pp. 131–144.