Processing Point Cloud Sequences with Growing Neural Gas

Sergio Orts-Escolano Jose Garcia-Rodriguez Vicente Morell Miguel Cazorla Marcelo Saval Jorge Azorin

***Abstract*— We consider the problem of processing point cloud sequences. In particular, we represent and track objects in dynamic scenes acquired using low-cost sensors such as the Kinect. A neural network based approach is proposed to represent and estimate 3D objects motion. This system addresses multiple computer vision tasks such as object segmentation, representation, motion analysis and tracking. The use of a neural network allows the unsupervised estimation of motion and the representation of objects in the scene. This proposal avoids the problem of finding corresponding features while tracking moving objects. A set of experiments are presented that demonstrate the validity of our method to track 3D objects. Favorable results are presented demonstrating the capabilities of the GNG algorithm for this task.**

I. Introduction

Visual tracking is a very challenging problem due to some constraints of current acquisition systems like: the loss of information caused by the projection of the 3D world on a 2D image, noise in images, cluttered-background, complex object motion, partial or full occlusions, illumination changes as well as real-time processing requirements, etc. [1]. The availability of both depth and visual information provided by the Kinect sensor[1] opens up new opportunities to solve some of the inherent problems in 2D based vision systems. Objects tracking and motion estimation are some of the topics that can take advantage using these devices. A large number of researchers are adapting previous 2D tracking algorithms exploiting the combination of RGB and depth information to overcome some of the previously mentioned inherent problems in 2D vision systems.

A common problem to the different tracking methods is the background subtraction, when the camera is fixed. With the availability of the low-cost Kinect depth cameras, researchers immediately noticed that the nature of the depth information can help to establish a more stable background model, which is resistant to changes in illumination or lack of contrast [2]. Once the object has been segmented and located in the image using depth information, visual features are extracted from the RGB image and are then used for tracking the object in successive frames. As example, a large number of works have dealt with the problem of detecting and tracking people in intelligent environments using RGB-D sensors [3], [4], [5], [6].

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

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).

[1]Kinect for XBox 360: http://www.xbox.com/kinect Microsoft

Traditional methods to track and represent objects are adopting RGB-D cameras. In the next paragraphs, we summarize some of the last works in this field. In [7], it is proposed a baseline method and a dataset for testing new methods using RGB-D data. [8] segment and track objects using a rich feature set, including local image appearance, depth discontinuities, optical flow, and surface normal to inform the segmentation decision in a conditional random field model. In [9], it is proposed a fully automatic tracking system that is capable of detecting and tracking a large number of flying targets in a 3D volume using particle filters. A different approach is presented in [10] that uses an unsupervised cross-view action recognition using multi-view feature synthesis to represent human motion. [11] presents a method to register images from specular and poorly textured 3D environments using local descriptors or a descriptor field. [12] proposes a new scene flow estimation that exploits the local and piece-wise rigidity of real world scenes. [13] extends RGB-D based flow estimators that use robust norms, non-linearized data constraints and variational solutions. [14] applies an extended object tracking method based on Random Hypersurface Models (RHMs) for tracking people approximating their shapes as cylinders. In [15], the tracking problem is handled using a bag-of-pixels representation and a back-projection scheme with their STAR3D system. [16] proposes a texture-less object detection and 3D tracking method which automatically extracts on the fly the information it needs from color images and the corresponding depth maps using Dominant Orientation Templates, a fast 2D template detection method. In [17], it is proposed the use of an Ensemble of Collaborative Trackers (ECT) to solve multi-object 3D tracking problem with a high level of occlusions. However, all these methods mostly rely on feature descriptors, and therefore in most cases there are extremely computationally intensive and feature-dependent: geometry, texture, light conditions, etc.

A group of methods of particular interest to track objects and estimate its motion are the ones based on neural representations. Neural networks have been extensively used to represent objects in scenes and estimate their motion. In particular, there are several works that use the self-organizing models for the representation and tracking of objects. Fritzke [18] proposed a variation of the original Growing Neural Gas (GNG) model [19] to map non-stationary distributions that in [20] was applied to people representation and tracking. In [21], amendments to self-organizing models for the characterization of the movement are proposed. In a recent work, Frezza-Buet [22] modifies the original GNG algorithm

978-1-4799-1959-8/15/$31.00 ©2015 IEEE

to compute non-stationary distributions velocity fields but it is only applied to 2D data. [23] modifies the original SOM algorithm to adapt and track 3D hand and body skeleton structures. However, none of them exploit previous obtained maps to solve the correspondence problem or to improve segmentation. From the works cited, only [20] represent both local and global motion. However, there is no consideration of time constraints, and the algorithm does not exploit the knowledge acquired in previous frames for the purpose of segmentation or prediction. In addition, the structure of the neural network is not used to solve the feature correspondence problem through the following frames.

Considering our previous works in the area [24], [25] that demonstrated object representation and tracking capabilities of self-growing neural models for 2D data. We propose a neural method capable of segmenting and representing objects of interest in the scene, as well as analyzing the evolution of these objects over time to estimate their motion. The proposed method creates a flexible model able to characterize morphological and positional changes over the sequence. The representation identifies objects over time and establishes feature correspondence through the different observations. This allows the estimation of motion based on the interpretation of the dynamics of the representation model. It has been demonstrated that architectures based on the GNG [19] can be applied on problems with time restrictions such as tracking objects. These systems have the ability to process sequences of images and thus offer a good quality of representation that can be refined very quickly depending on the time available. In this line of work, some works have been already presented that propose both, algorithmic optimizations [26] and hardware parallelization [27].

The rest of the paper is organized as follows: first, Section II describes the GNG algorithm and the proposed modifications. It is proposed a modification of the original algorithm to deal with point cloud sequences. Moreover, we combine our proposal with an existing optimization technique for reducing GNG computational complexity. Next, Section III presents some experiments and discuss results obtained using our novel approach. Finally, in Section IV, we give our conclusions and directions for future work.

II. GNG Algorithm

The Growing Neural Gas (GNG) [19] is an incremental neural model able to learn the topological relations of a given set of input patterns by means of Competitive Hebbian Learning (CHL). Unlike other methods, the incremental character of this model, avoids the necessity to previously specify the network size. On the contrary, from a minimal network size, a growth process takes place, where new neurons are inserted successively using a particular type of vector quantization [28].

To determine where to insert new neurons, local error measures are gathered during the adaptation process and a new unit is inserted near the neuron which has the highest accumulated error. At each adaptation step a connection between the winner (closest neuron) and the second-nearest neuron is created as dictated by the CHL rule. This is continued until an ending condition is fulfilled. In addition, in the GNG network the learning parameters are constant in time, in contrast to other methods whose learning is based on decaying parameters [28]. In the remaining of this section we describe the growing neural gas algorithm. The network is specified as:

- A set A of nodes (neurons). Each neuron $c \in A$ has its associated reference vector $w_c \in \mathbb{R}^d$. The reference vectors are 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 its 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 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:

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

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

   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 with 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.

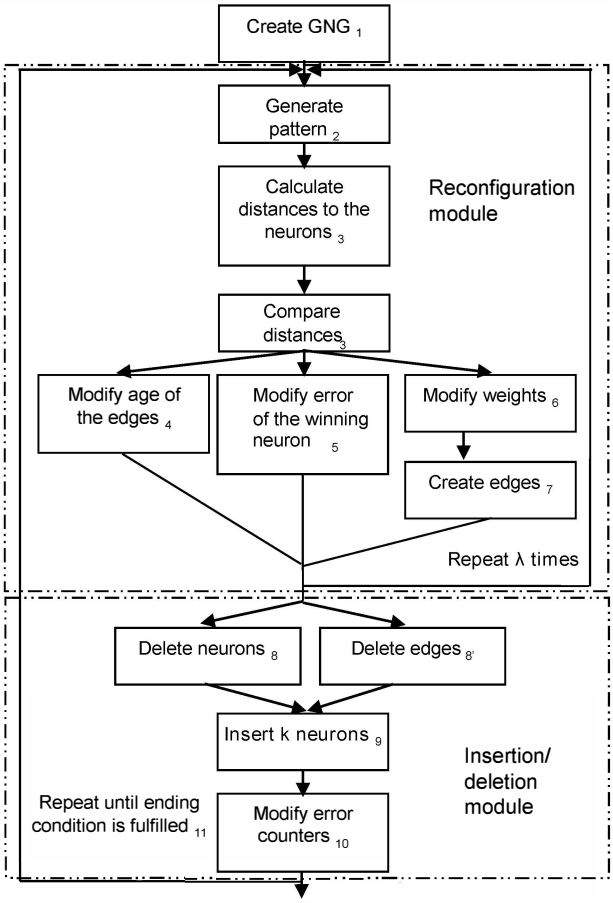

Fig. 1. GNG learning algorithm.

Figure 1 shows an scheme of the above presented GNG algorithm. This method offers further benefits due to the incremental adaptation of the GNG. Input space denoising and filtering is performed in such a way that only concise properties of the point cloud are reflected in the output representation [29], [30].

A. *Point cloud sequences processing*

GNG capabilities for representing static 3D observations have been already shown in previous works [31], [29], [32] but we propose the extension of this method to allow representing non-stationary observations. In this way, GNG representation can be used in complex computer vision problems as scene understanding by long-term observation, Simultaneous Localization and Mapping (SLAM) or 3D object tracking.

To obtain the representation of the first point cloud, the entire algorithm presented in Section II is computed, performing the complete learning process of the network. However, for the following point clouds, only internal loop (reconfiguration module in Figure 1) of the algorithm is processed. In this way, neurons that are already placed in the map are adjusted and moved towards changes in the observation (partial learning). Only weight modification and spatial relationship steps are processed keeping stable the number of neurons of the GNG. Figure 2 shows the proposed work-flow for processing point cloud sequences using the GNG algorithm.

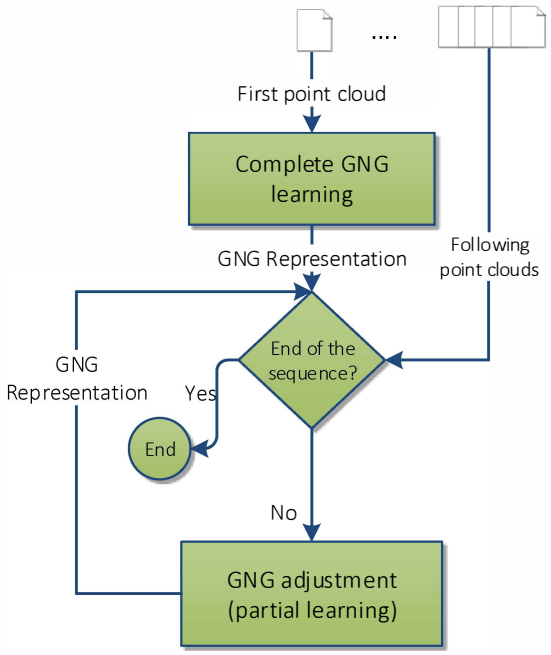

Fig. 2. An improved work-flow to manage point cloud sequences using the GNG algorithm.

This extension of the GNG algorithm has some advantageous features for processing sequences of point clouds. The most interesting is that is not required restarting the learning of the network for each new point cloud that is captured from the scene. Once the initial map that represent objects of interest in the scene is generated, this map can be used as the initial topology to represent and track present objects in the next point cloud. In order to take advantage of this feature a good sample rate is required, so that the changes between different captured point clouds should be small. Using this approach and considering recorded information from previous processed point clouds it is possible to obtain an updated representation of the new observation. This extension provides a speed-up in the runtime of the learning step as neurons are kept between frames. Moreover, this adaptive method is able to face real-time constraints, because the number of λ times that the reconfiguration module is performed can be chosen according to the time available between two successive frames that depends on the

acquisition rate. More results about runtimes for different number of λ patterns will be later presented.

B. Exponential time-decaying adaptation

In the original GNG algorithm it is considered a stationary input space distribution where neurons are placed directly over the input data. However, considering sequences of non-stationary data, neurons need to move towards (step 6) a changing input space caused by the motion of the scene (camera motion) or objects motion. So that, the use of the original constant values for ϵ_w and ϵ_n parameters takes long time (λ iterations) to move the whole map to the new input space (next frame). In this work, we propose the modification of these constant ϵ_w and ϵ_n parameters so neurons move fast towards the selected input patterns using an exponential decaying function based on a time function (λ iteration). A similar approach was previously used in the Neural Gas (NG) algorithm proposed in [28]. Thanks to this modification, the neural map considerably accelerates the learning step fitting new input data. The learning step is defined as a single iteration over the steps detailed above. The ϵ_w and ϵ_n parameters are computed every iteration as follows:

$$\epsilon_w(t) = \epsilon_{w_i}(\epsilon_{w_f}/\epsilon_{w_i})^{t/t_{max}} \tag{5}$$

$$\epsilon_n(t) = \epsilon_{n_i}(\epsilon_{n_f}/\epsilon_{n_i})^{t/t_{max}} \tag{6}$$

where ϵ_{w_i}, ϵ_{w_f}, ϵ_{n_i} and ϵ_{n_f} are the initial and final epsilon values for adapting winning neurons during the adaptation step. Using this exponentially decaying function we are able to initially move very fast the neurons towards the new input space (large ϵ parameters) and to finally perform a fine adaptation using a lower ϵ value.

C. Uniform grid optimization

As we previously introduced, the GNG algorithm has been successfully used in various applications, ranging from multi-dimensional data clustering to 3D object reconstruction. However, this type of learning is time-consuming, especially for high-dimensional data. Therefore, we propose the combination of our GNG extension for point cloud sequences (Section II-A), which already reduces the computational complexity, with other optimization techniques already presented in [26]. These optimization techniques proposed by Fišer et al. [26] provide a considerably speed-up of the GNG learning step.

In particular, we focused on accelerating the step for finding the nearest neurons (step 3), which is the algorithm most time-consuming part. We introduced a uniform grid, which allows to speed-up the searching process by dividing and indexing the search space. Although in the worst case the whole space is explored, in most cases the computational complexity is significantly lower. In contrast to Fišer et al., we propose the use of an uniform grid instead of a growing one, since we apply this optimization to adjust the topology of the initial map to a sequence of point clouds and therefore only the internal loop of the GNG algorithm is processed.

The uniform grid has some key issues for achieving a good performance. These key issues are related with the choice of an suitable voxel size and grid dimensions. In our case, grid dimensions are selected computing the bounding box of the input data. Using the minimum and maximum 3D points of the input data a bounding box is built aligned with the global reference system. Voxel size is selected based on the ratio between the mean point cloud resolution and the number of neurons. Once the grid is constructed, we can use input pattern coordinates as indexes for computing the voxel index in which that point is placed, then a local search for the nearest neurons is performed. This process is repeated increasing the number of visited voxels until the nearest neurons are found. More information about this optimization step can be found in [26]. Figure 4 shows an example of a 3D grid used to accelerate the GNG learning step.

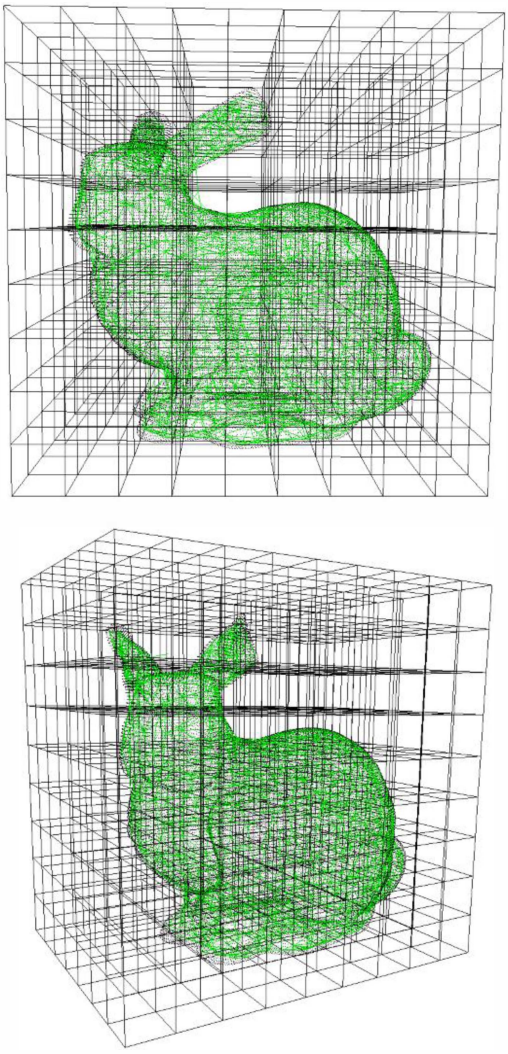

Fig. 4. Example of a 3D uniform grid structure used to accelerate the GNG algorithm. It considerably accelerates finding the closest neurons reducing the search space.

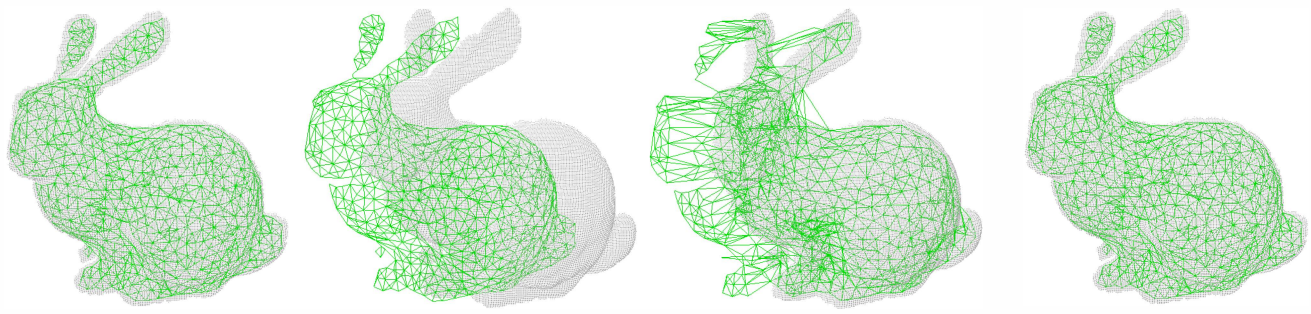

Fig. 3. From left to right: GNG representation of the initial input space. Input space translated over the x-axis. Intermediate state of the GNG structure during the adaptation process to fit the new input data (GNG extension for managing point clouds). Final GNG representation after activating 200000λ patterns.

This extended version of the algorithm will be further validated in the experiments section. We will show how this optimized version of the GNG method is useful for dealing with point cloud sequences and reducing the runtime for processing following frames.

III. Experiments

In this section, different experiments are shown validating the capabilities of our extended GNG-based method for managing point cloud sequences. First, we studied the input space representation error of the proposed method for tracking a 3D object along a sequence. Next, we show how the proposed method is able to estimate the trajectory of a 3D non-stationary distribution along the time. Moreover, we tested the proposed method using different point clouds with different shape and texture information. 3D non-stationary distributions were simulated using the Blensor software [33]. Blensor allowed us to generate synthetic scenes and to obtain partial views of the generated scene as if a real 3D sensors, such as the Kinect or a Time-of-Flight camera, were used. The main advantage of this software is that it provides ground truth (absence of noise) information. Moreover, it allows to capture point cloud sequences simulating the movement of an object in the scene or the camera movement. It also provides us with the translation and rotation values that are applied to the distribution on every frame of the sequence, so we can use this information to evaluate the proposed method.

A. Representation error

In this section, we show the input adaptation error of the generated map along some point cloud sequences. The adaptation of the self-organizing neural networks is often measured in terms of preservation of the topology of the input space, so it is needed to know the real distance from the generated structure to the input data. This measure specifies how close our generated model is from the original input data. In order to obtain a quantitative measure of the input space adaptation of the generated map, we computed the Mean Square Error (MSE) of the map against sampled points (input space).

$$MSE = \frac{1}{|V|} \sum_{\forall p \in V} \min_{i \in A} d(p - w_i)^2 \tag{7}$$

where V is the input space, p is a sample that belongs to the input space, i is the neuron with the minimum distance to the input space sample and w_i is the neuron's weight. $d(p-w_i)^2$ is the squared euclidean distance between the closest neuron and the input point p. This measure is computed once the network topology learning step is completed.

Figure 5 shows the GNG input space adaptation error for a different number of iterations (λ patterns). The representation error is computed for the network adjustment between two different frames. In addition, Figure 5 shows the obtained representation error using three different strategies for selecting the ϵ_w and ϵ_n parameters of the GNG algorithm. We can observe how the time-decaying strategy introduced in our proposal is able to obtain a low representation error using the lowest number of iterations. Figure 5 also shows how the original GNG scheme, which uses low ϵ values is not suitable for managing point clouds sequences, since it takes a large number of iterations for getting a good representation error. Using a large ϵ value achieves a fast adaptation but the final error is worse compared to the time-decaying epsilon.

Figure 6 shows the representation error for a whole sequence of 30 frames. This sequence was synthetically generated using the Blensor tool for simulating a Kinect device. We animated the 3D Stanford bunny, translating the model over the x axis a known distance. This ground truth information about the object movement is later used to study the capabilities to track the trajectory of an object along the sequence. In addition, Figure 6 shows how the 'low epsilon' strategy is not able to track the input distribution and soon starts producing a non accurate representation of the input data. In particular, in the middle of the sequence, when the motion velocity increases, the representation error dramatically raises. As it was previously demonstrated, the time-decaying strategy obtains the best adaptation error for a fixed number of iterations (100000λ per frame).

Figure 7 shows some frames of the generated point cloud

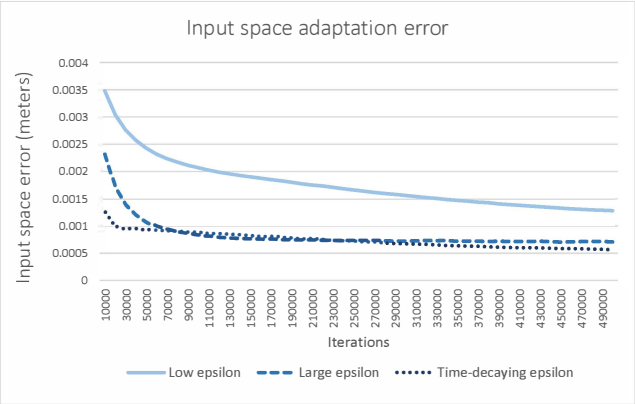

Fig. 5. GNG representation error (MSE) between two different frames of a sequence. We used a fixed network size of 2000 neurons. Large ϵ strategy uses $\epsilon_w = 0.15$, $\epsilon_n = 0.005$; low epsilon uses $\epsilon_w = 0.05$, $\epsilon_n = 0.0005$; time-decaying epsilon uses $\epsilon_{w_i} = 0.15$, $\epsilon_{w_f} = 0.05$, $\epsilon_{n_i} = 0.005$ and $\epsilon_{n_f} = 0.0005$

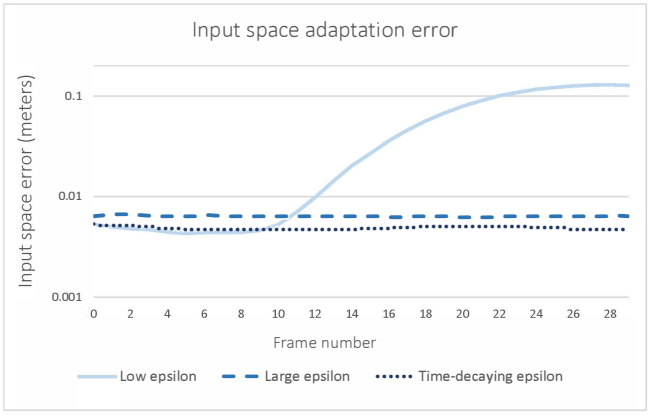

Fig. 6. GNG representation error for a whole sequence of 30 frames. MSE was computed for the three different strategies, which were previously commented, selecting ϵ values. $100,000\lambda$ patterns were activated at each frame.

sequence using the Blensor software for simulating a Kinect device. On the left side it is shown the initial acquired point cloud (red), whereas on the right side it is shown (blue) the last acquired point cloud.

B. Tracking error

In this section, we study the capability of the proposed method to compute the translational movement of a non-stationary distribution along a sequence. For that purpose, we compute the centroid of the generated map using the GNG algorithm at each new frame. To evaluate the neural network capacity to follow the input space motion, we compare the movement performed by the whole object using the centroid trajectory. Thanks to the Blensor software, we are able to obtain ground truth information about the object position in the scene. This information is used for obtaining the translational error between the real movement and the computed one using the GNG.

For evaluating the tracking capabilities of the presented

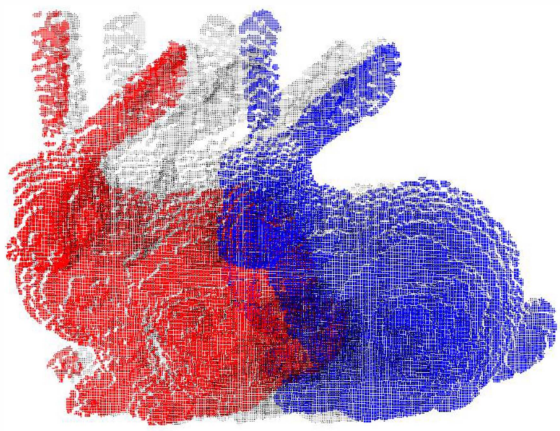

Fig. 7. Point cloud sequence of a moving object (3D Stanford bunny) generated using the Blensor software for simulating a Kinect device. The Bunny model is moved along the x axis 1.1 meters. The sequence was generated using realistic kinematics of an object movement (acceleration).

GNG extension, we used the sequence presented in Figure 7. Figure 8 shows the translational error obtained using the proposed GNG extension for point clouds sequence processing. Best results were obtained by using the exponential time-decaying strategy, which obtained for most frames of the sequence a translational error below 0.005. Moreover, the proposed GNG modification was able to follow the trajectory of the non-stationary distribution obtaining a global translation error for the whole trajectory below 0.04 meters in a complete movement of 1.1 meters along the x axis.

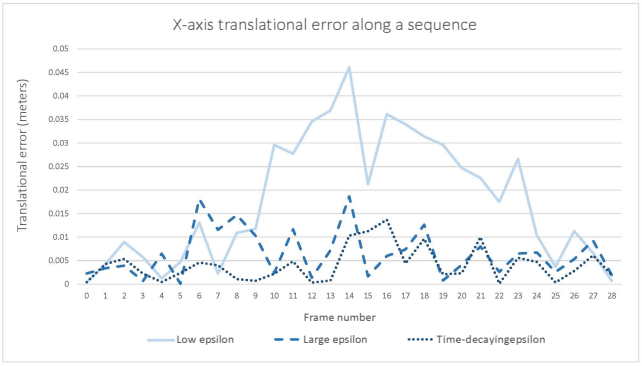

Fig. 8. x-axis translational error for the simulated sequence presented in Figure 7. Translational error was evaluated using the three different strategies for choosing the ϵ values. A constant number of 200000λ input patterns were activated between each frame of the sequence for adjusting the topology of the previous existing GNG structure.

Figure 9 shows the box-plot of the mean error for the whole sequence. The box-plot shows that not only the translational mean error is lower but also it shows how the exponential time-decaying strategy produces less outliers and then lower errors for all the analyzed frames.

Finally, some experiments were carried out using real information from a Microsoft Kinect v2 sensor. Figure 10 shows the result of applying the proposed method to a

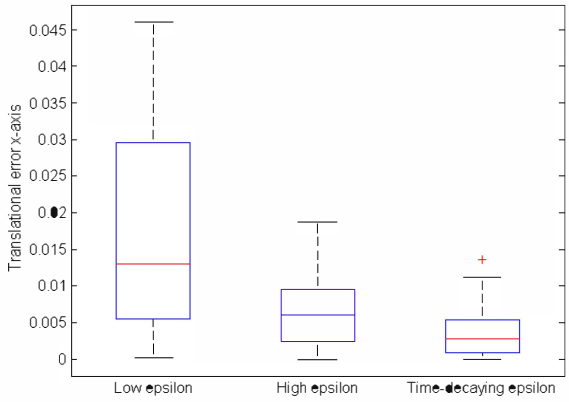

Fig. 9. Mean translational error in the x axis for the simulated sequence presented in Figure 7

sequence of real 3D data. The GNG extension is able to represent and track the two moving objects in the scene. The proposed method takes advantage of motion difference from previous frames to focus on moving objects and depth information for separating object representations within the same scene. We can see how the GNG is able to represent the objects in the scene and keep track of them along the sequence. No qualitative results are presented since real sensors do not provide us with ground truth information for error validation. That is the main reason why first experiments were performed using a simulated 3D sensor using the Blensor software.

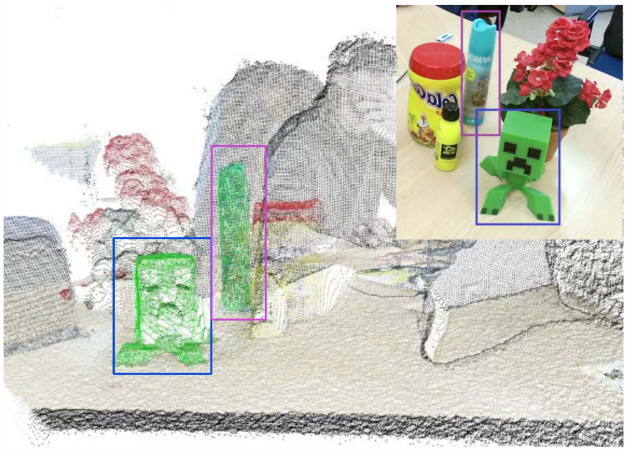

Fig. 10. Multiple object representation and tracking from real data obtained using the Kinect v2. The proposed method was used to represent and track objects in the scene using as a priori knowledge motion from previous frames. GNG representation (green wire-frame structure) is shown over the tracked objects: Creeper figure and the air freshener.

C. Optimized GNG algorithm for point cloud sequences

Finally, in this section, some experiments were performed to validate the optimized implementation of the proposed GNG algorithm. Moreover, some performance tests were carried out comparing the speed-up obtained by our implementation with a naive one (Original GNG).

The accelerated version of the GNG algorithm has been developed and tested on a machine with an Intel Core i3 540 3.07 GHz. Tested methods were developed using C++ and the Visual C++ compiler.

Table I shows the execution times of the original GNG algorithm and our optimized version for adjusting an initial map to a non-stationary input data. We can see how the time that takes to the network to adjust the map to a new frame is considerably reduced by using the uniform grid structure. Considerable speed-ups were obtained adjusting the map, up to 23 times faster for a map comprised of 5000 neurons. We can also observe how as the number of neurons that comprises the map is increased the obtained speed-up is also increased. The uniform grid structure allows to increase the number of neurons getting a similar runtime, since neurons are equally split into different voxels.

TABLE I

EXECUTION TIMES (SECONDS) FOR THE ORIGINAL GNG AND THE OPTIMIZED VERSION (ONLY RECONFIGURATION MODULE RUNTIME). DIFFERENT NUMBER OF NEURONS AND λ PATTERNS WERE EVALUATED.

	Orig GNG	**Optimized GNG**	**Speed-up**
$1000n$ - 50000λ	0.34	0.06	5.66
$1000n$ - 100000λ	0.58	0.12	4.83
$1000n$ - 200000λ	0.98	0.22	4.46
$2000n$ - 50000λ	0.70	0.07	10.15
$2000n$ - 100000λ	1.07	0.12	9.40
$2000n$ - 200000λ	1.88	0.18	10.44
$5000n$ - 50000λ	1.78	0.08	23.42
$5000n$ - 100000λ	2.81	0.13	21.61
$5000n$ - 200000λ	5.07	0.22	23.04

IV. CONCLUSIONS AND FUTURE WORK

Depth information provided by RGB-D cameras significantly increases the robustness of the proposed method. This is mainly due to that the depth information is inherently resistant to lighting condition changes and environment clutters, which are difficult problems for conventional algorithms.

In this paper we have presented a novel computing method to manage complete sequences of organized raw noisy 3D data. Previous knowledge about the sensor is not necessary and it has been demonstrated how Growing Self-Organizing Maps (GSOM) are capable to deal with point cloud sequences by adjusting its topology and therefore reducing the computational time taken by creating a new structure for each new frame (new input data). Moreover, the proposed method is able to estimate the trajectory of the moving objects along the sequence with a low translational error. Furthermore, The method was validated with several models ranging from synthetic objects (Simulated sensor) to real ones acquired using the new Kinect v2 sensor. Obtained models were able to accurately represent objects trajectory with an error below 5 millimeters.

Finally, the proposed method was accelerated by means of a grid structure considerable reducing the computational

complexity and obtaining acceleration factors up to 23 times faster than the original version of the GNG algorithm.

Future work includes the evaluation of the proposed method using existing data sets and performance comparison with other state-of-the-art 3D object tracking techniques. In addition, we plan to explore the possibility of including visual and geometric descriptors to the neurons code vector, combining the traditional learning approach from the original GNG with new feature matching techniques.

ACKNOWLEDGMENTS

This work was partially funded by the Spanish Government DPI2013-40534-R grant.

REFERENCES

[1] H. Yang, L. Shao, F. Zheng, L. Wang, and Z. Song, "Recent advances and trends in visual tracking: A review," *Neurocomput.*, vol. 74, no. 18, pp. 3823–3831, Nov. 2011.

[2] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced Computer Vision With Microsoft Kinect Sensor: A Review," *Cybernetics, IEEE Transactions on*, vol. 43, no. 5, pp. 1318–1334, Oct. 2013.

[3] Z. Zhang, W. Liu, V. Metsis, and V. Athitsos, "A viewpoint-independent statistical method for fall detection," in *Pattern Recognition (ICPR), 2012 21st International Conference on*, Nov 2012, pp. 3626–3630.

[4] P. Wang, S. Ma, and Y. Shen, "Performance study of feature descriptors for human detection on depth map," *International Journal of Modeling, Simulation, and Scientific Computing*, vol. 05, no. 03, p. 1450003, 2014.

[5] L. Spinello and K. O. Arras, "People detection in rgb-d data," in *In IEEE/RSJ Int. Conf. on*, 2011.

[6] L. Spinello, C. Stachniss, and W. Burgard, "Scene in the loop: Towards adaptation-by-tracking in RGB-D data," 2012.

[7] S. Song and J. Xiao, "Tracking revisited using rgbd camera: Unified benchmark and baselines," in *Proceedings of the 2013 IEEE International Conference on Computer Vision*, ser. ICCV '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 233–240.

[8] A. Teichman, J. Lussier, and S. Thrun, "Learning to segment and track in rgbd," *Automation Science and Engineering, IEEE Transactions on*, vol. 10, no. 4, pp. 841–852, Oct 2013.

[9] Y. Liu, H. Li, and Y. Q. Chen, "Automatic tracking of a large number of moving targets in 3d," in *Proceedings of the 12th European Conference on Computer Vision - Volume Part IV*, ser. ECCV'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 730–742.

[10] A. Gupta, A. Shafaei, J. Little, and R. Woodham, "Unlabelled 3d motion examples improve cross-view action recognition," in *Proceedings of the British Machine Vision Conference*. BMVA Press, 2014.

[11] A. Crivellaro and V. Lepetit, "Robust 3d tracking with descriptor fields," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, June 2014, pp. 3414–3421.

[12] J. Quiroga, T. Brox, F. Devernay, and J. Crowley, "Dense semi-rigid scene flow estimation from rgbd images," in *Computer Vision ECCV 2014*, ser. Lecture Notes in Computer Science, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Springer International Publishing, 2014, vol. 8695, pp. 567–582.

[13] E. Herbst, X. Ren, and D. Fox, "Rgb-d flow: Dense 3-d motion estimation using color and depth," in *2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6-10, 2013*, 2013, pp. 2276–2282.

[14] F. Faion, M. Baum, and U. Hanebeck, "Tracking 3d shapes in noisy point clouds with random hypersurface models," in *Information Fusion (FUSION), 2012 15th International Conference on*, July 2012, pp. 2230–2235.

[15] C. Yuheng Ren, V. Prisacariu, D. Murray, and I. Reid, "Star3d: Simultaneous tracking and reconstruction of 3d objects using rgb-d data," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[16] Y. Park, V. Lepetit, and W. Woo, "Texture-less object tracking with online training using an rgb-d camera," in *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, Oct 2011, pp. 121–126.

[17] N. Kyriazis and A. Argyros, "Scalable 3d tracking of multiple interacting objects," in *CVPR*, 2014, to appear.

[18] B. Fritzke, "A self-organizing network that can follow non-stationary distributions," in *Artificial Neural Networks ICANN'97*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 1997, vol. 1327, pp. 613–618.

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

[20] H. Frezza-Buet, "Following non-stationary distributions by controlling the vector quantization accuracy of a growing neural gas network," *Neurocomput.*, vol. 71, pp. 1191–1202, March 2008.

[21] X. Cao and P. N. Suganthan, "Hierarchical overlapped growing neural gas networks with applications to video shot detection and motion characterization," in *Proc. Int. Joint Conf. Neural Networks IJCNN '02*, vol. 2, 2002, pp. 1069–1074.

[22] H. Frezza-Buet, "Online computing of non-stationary distributions velocity fields by an accuracy controlled growing neural gas," *Neural Networks*, vol. 60, no. 0, pp. 203 – 221, 2014.

[23] F. Coleca, A. State, S. Klement, E. Barth, and T. Martinetz, "Self-organizing maps for hand and full body tracking," *Neurocomputing*, vol. 147, no. 0, pp. 174 – 184, 2015, advances in Self-Organizing Maps Subtitle of the special issue: Selected Papers from the Workshop on Self-Organizing Maps 2012 (WSOM 2012).

[24] J. Garcia-Rodriguez and J. M. Garcia-Chamizo, "Surveillance and human-computer interaction applications of self-growing models," *Appl. Soft Comput.*, vol. 11, no. 7, pp. 4413–4431, Oct. 2011.

[25] J. Garcia-Rodriguez, S. Orts-Escolano, A. Angelopoulou, A. Psarrou, J. Azorin-Lopez, and J. Garcia-Chamizo, "Real time motion estimation using a neural architecture implemented on gpus," *Journal of Real-Time Image Processing*, pp. 1–19, 2014.

[26] D. Fišer, J. Faigl, and M. Kulich, "Growing neural gas efficiently," *Neurocomput.*, vol. 104, pp. 72–82, Mar. 2013.

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

[28] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten, "'neural-gas' network for vector quantization and its application to time-series prediction," vol. 4, no. 4, pp. 558–569, 1993.

[29] S. Orts-Escolano, V. Morell, J. Garcia-Rodriguez, and M. Cazorla, "Point cloud data filtering and downsampling using growing neural gas," in *The 2013 International Joint Conference on Neural Networks, IJCNN 2013, Dallas, TX, USA, August 4-9, 2013*, 2013, pp. 1–8.

[30] J. Garcia-Rodriguez, M. Cazorla, S. Orts-Escolano, and V. Morell, "Improving 3d keypoint detection from noisy data using growing neural gas," in *Advances in Computational Intelligence - 12th International Work-Conference on Artificial Neural Networks, IWANN 2013, Puerto de la Cruz, Tenerife, Spain, June 12-14, 2013, Proceedings, Part II*, 2013, pp. 480–487.

[31] R. L. M. E. Do Rego, A. F. R. Araujo, and F. B. De Lima Neto, "Growing self-reconstruction maps," *Trans. Neur. Netw.*, vol. 21, no. 2, pp. 211–223, Feb. 2010.

[32] S. Orts-Escolano, J. Garcia-Rodriguez, V. Moreli, M. Cazorla, and J. Garcia-Chamizo, "3d colour object reconstruction based on growing neural gas," in *Neural Networks (IJCNN), 2014 International Joint Conference on*, July 2014, pp. 1474–1481.

[33] M. Gschwandtner, R. Kwitt, A. Uhl, and W. Pree, "BlenSor: Blender Sensor Simulation Toolbox Advances in Visual Computing," ser. Lecture Notes in Computer Science, G. Bebis, R. Boyle, B. Parvin, D. Koracin, S. Wang, K. Kyungnam, B. Benes, K. Moreland, C. Borst, S. DiVerdi, C. Yi-Jen, and J. Ming, Eds. Berlin, Heidelberg: Springer Berlin / Heidelberg, 2011, vol. 6939, ch. 20, pp. 199–208.