3D Model Reconstruction using Neural Gas Accelerated on GPU

Sergio Orts-Escolano, Jose Garcia-Rodriguez, Antonio Jimeno-Morenilla, Jose Antonio Serra-Perez, Alberto Garcia

Department of Computing Technology at University of Alicante. Alicante (Spain)

Vicente Morell, Miguel Cazorla

Department of Computational Sciences and Artificial Intelligence at University of Alicante. Alicante (Spain)

Abstract

In this work, we propose the use of the Neural Gas (NG), a neural network that uses an unsupervised competitive hebbian learning (CHL), to develop a reverse engineering process. This is a simple and accurate method to reconstruct objects from the point cloud obtained from overlapped multiple views using low cost sensors. In contrast to other methods that may need several stages that include downsampling, noise filtering and many other tasks, the NG automatically obtains the 3D model of the scanned objects.

To demonstrate the validity of our proposal we tested our method with several models and performed a study of the neural network parameterization calculating the quality of representation and also comparing results with other neural methods like Growing Neural Gas and Kohonen maps or clasical

Email addresses: `sorts@dtic.ua.es` (Sergio Orts-Escolano), `jgarcia@dtic.ua.es` (Jose Garcia-Rodriguez), `jimeno@dtic.ua.es` (Antonio Jimeno-Morenilla), `jserra@dtic.ua.es` (Jose Antonio Serra-Perez), `agg180@alu.ua.es` (Alberto Garcia), `vmorell@dccia.ua.es` (Vicente Morell), `miguel@dccia.ua.es` (Miguel Cazorla)

methods like Voxel Grid. We also reconstructed models acquired by low cost sensors that can be included in virtual and augmented reality environments to redesign or manipulation purpose. Since the NG algorithm has a strong computational cost we propose its acceleration. We have redesigned and implemented the NG learning algorithm to fit it onto a Graphic Processor Unit using CUDA. A speed-up of 180x faster is obtained compared to the sequential CPU version.

Keywords:
Neural Gas, Topology Preservation, 3D Model Reconstruction, Graphics Processor Units.

1. Introduction

A 3D model of a free-form object [1] is constructed by acquiring multiple viewpoints so that its surface is completely covered. These views are then registered in a common coordinate basis and the implicit surface defined by the 3D data is computed obtaining a 3D mesh.

Although many algorithms have already been proposed for mesh generation from an unorganized point cloud [2, 3, 4, 5, 6, 7], existing techniques have various limitations mainly in terms of their applicability to free-form objects, accuracy, efficiency, and the discriminating capability of the generated representation. An excellent survey of free-form object representation and recognition techniques can be found in [8]. In addition, a brief survey is presented in Section 2 for completeness.

The three-dimensional representation of an object provides a higher degree of realism. This factor has been implemented at different stages of production processes in order to improve productivity and quality. Moreover, the combination of the acquired and reconstructed 3D models with virtual and augmented reality environments allows the users to interact with them and also developing a virtual manufacturing system as an application of augmented reality to improve several process stages [9], including quality control, human-machine interface and information flows, practicing e-commerce and possibility of implementing different production philosophies. [10] presented the Direct Modelling approach, which allows the user to intuitively select geometric entities in real time regardless of the modifications made. This was initially developed for conceptual design and architectural planning [11]. [12] showed the advantages derived from using a semi-immersive environment, combining stereoscopic vision with 3D inputs, addressed to conceptual and aesthetic design.

In the last years, neural networks approaches have become popular in different fields, including computer graphics problems. Neural networks have been extensively used for data clustering. In particular, self-organising models [13] place their neurons so that their positions within the network and connectivity between different neurons are optimised to match the spatial distribution of activations. As a result of this optimization, existing relationships within the input space will be reproduced as spatial relationships among neurons in the network. The final result of the self-organising or competitive learning process is closely related to the concept of Delaunay triangulation of the input space corresponding to the reference vectors of neurons in the network. Traditionally, it has been suggested that this triangulation, result of competitive learning, preserves the topology of the input space. However, [14] introduced a new condition which restricts this quality. Thus, self-organising maps or Kohonen maps are not Topology Preserving Networks (TPN) as has traditionally been considered, since this condition only would happen in the event that the topology or dimension of the map and the input space coincide. The Growing Cell Structures [15] are not TPN since the topology of the network is established a priori.

In the case of the Neural Gases like Neural Gas (NG) [16] or Growing Neural Gas (GNG) [17], the mechanism for adjusting the network through a competitive learning generates an induced Delaunay triangulation. Martinetz and Schulten [14] demonstrated that these models are TPN.

Neural Gases have been widely used in recent years for different applications, mainly for clustering or topology learning. In [18] the representation obtained with the neural network structure is used to represent road networks and build an intelligent navigation system. [19] and [20] use the NG to build character recognition systems. [21] applied the NG to reconstruct 3D surfaces but no quantitative quality of representation data is provided with only qualitative examples and comparisons with the GNG model.

There are also a number of works using GNG models for 3D reconstruction tasks [22, 5, 23, 24]. However, based on our previous experience [25, 26] we demonstrated that Neural Gas quality of representation capability is higher than the GNG one which makes NG model more accurate for the proposed problem.

Moreover, the NG learning algorithm has a higher level of parallelism and is more suitable to be implemented onto graphic processor units (GPUs) compared to other growing self-organising networks which have been already implemented onto the GPU [27, 28].

In general, in previous works [29, 30], we demonstrated using the GNG algorithm, that self-organizing maps, due to their stochastic nature can handle corrupted data and noisy information, being able to interpolate missing data from the input space. This feature makes NG model advantageous for working with noisy 3D data provided by low-cost RGB-D sensors.

We propose the use of the Neural Gases to reconstruct objects from the point cloud obtained from overlapped multiple views. As a difference with other methods that may need several stages that includes downsampling, noise filtering, surface triangulation, and many other tasks, the proposed NG-based algorithm automatically obtains the 3D model from the scanned objects. Moreover, overlapping scanned data may produce unsatisfactory results, as a simple connection of consecutive scan points would lead to wrong surfaces. The proposed algorithm therefore does not make any assumption about the topology of the data. It takes unorganized three-dimensional data points as input and generates a 3D mesh that represents the underlying surface.

The main contributions of this paper are the following. An extension of the NG algorithm considering colour information and producing meshes with faces during the learning stage. An accelerated implementation on the GPU, considerably accelerating the original algorithm compared to the CPU version. A fully integrated framework to perform 3D object reconstruction using low-cost RGB-D sensors and finally an extensive study of the proposed method, comparing the obtained results with related methods and under different noise conditions.

The remainder of the paper is organized as follows. Section 2 reviews the most used methods for 3D surface reconstruction. Section 3 presents the proposed NG-based algorithm and demonstrates its capacities to reconstruct 3D models. Moreover, Section 3 presents an accelerated implementation of the NG algorithm on graphics processor units using CUDA. Next, Section 4 presents several experiments, showing qualitative and quantitave results of the proposed method compared to related works. Finally, in Section 5, the conclusions of the work and further directions of the research are presented.

2. 3D Reconstruction methods

This section will review the most used methods and techniques for the reconstruction of three dimensional surfaces.

2.1. Delaunay and α shapes

Reconstruction with Delaunay methods in three dimensions consists on the extraction of tetrahedron surfaces from the initial point cloud. The concept of alpha-shape formalizes the intuitive notion of "shape" for a set of points in the space. One of the earliest approaches is based on α-shapes [31]. Given a finite point set S, and the real parameter α, the alpha-shape of S is a polytope (the generalization to any dimension of a two dimensional polygon and a three-dimensional polyhedron) which is neither convex nor necessarily connected. For a large α value, the α-shape is identical to the convex-hull of S. If the alpha value decreases progressively, non-convex shapes with cavities are obtained.

The algorithm proposed by [31] eliminates all tetrahedrons which are delimited by a surrounding sphere smaller than α. The surface is then obtained with the external triangles from the resulting tetrahedron. Another approach is based on labelling the initial tetrahedrons as interior and exterior. The resulting surface is generated from the triangles found in and out. This idea first appeared in [32] and was later performed by Powercrust in [33] and the algorithm called Tight Cocone [34]. Both methods have been recently extended for reconstructing point clouds with noise [34, 35].

The main advantage of most Delaunay based methods is that fit very accurately the surface defined by the original point cloud. However, the method is very sensitive to noise since it is an interpolation based method that produces undesirable results with noisy data. Therefore, the quality of the points obtained in the digitization process determines the feasibility of these methods. Due to the use of the whole point cloud set to obtain the most accurate triangulation, considering the Delaunay rule, the digitized points on the surface with an error considered above the limit, will be explicitly represented on the reconstructed surface geometry.

2.2. Zero-set methods

Implicit reconstruction methods (or zero-set methods) reconstruct the surface using a distance function which assigns to each point in the space a signed distance to the surface S. The polygonal representation of the object is obtained by extracting a zero-set using a contour algorithm. Thus, the problem of reconstructing a surface from a disorganized point cloud is reduced to the definition of the appropriate function f with a zero value for the sampled points and different to zero value for the rest. In [36] was established the use of such methods with the algorithm called Marching-Cubes. This algorithm has evolved in different variants, [2] used a discrete function f, in [6] a polyharmonic radial basis function is applied to adjust the initial point set. Other approaches include the adjustment function Moving Least Squares [37, 38] and basic functions with local support [39], based on the Poisson equation [7].

Those methods have the problem of loss of the geometry precision in areas with extreme curvature, i.e., corners, edges. Furthermore, pretreatment of information by applying some kind of filtering technique also affects the definition of the corners. There are several studies related to post-processing techniques used in the reconstruction for the detection and refinement of corners [38, 40] but these methods increase the complexity of the solution.

2.3. Voxel Grid

The Voxel Grid filtering technique is based on the input space sampling using a grid of 3D voxels to reduce the number of points. This technique has been traditionally used in the area of computer graphics to subdivide the input space and reduce the number of points [41, 42].

For each voxel, a centroid is chosen as the representative of all points. There are two approaches, the selection of the voxel centroid or the centroid calculation by using the points lying within the voxel. The calculation of the mean point has a higher computational cost, but offers better results. Thus, a subset of the input space is obtained that roughly represents the underlying surface. The Voxel Grid method presents the same problems as other filtering techniques: impossibility of defining the final number of points that represent the surface, geometric information loss due to the reduction of points inside a voxel and sensitivity to noisy input spaces. This method will be compared with our NG based 3D reconstruction method, as it offers similar features for efficient mesh downsampling.

2.4. Surface reconstruction based on Self-Organizing Maps

Considering the 3D representation problem from a computational intelligence approach and based on self-organization maps, a different perspective to obtain 3D reconstructions is proposed. These methods could be considered as flexible and growing models considering if the topology of the selected map has a priori topology or otherwise they grows until a condition is fulfilled. Moreover, we can find some similarities or correspondences between the neural network map and the 3D representation obtained. Nodes of the neural network map correspond to vertices of a mesh and connections between nodes correspond to the edges. Therefore, in this works the terms node and vertex, and connection and edge are used interchangeably. From this perspective, some methods were proposed based on self-organizing maps.

In [43] it is proposed the use of Kohonen's self-organizing map for surface reconstruction using as an input data unorganized point clouds. Moreover, since Kohonen's map does not produce regular faces, an edge collapse operation was introduced eliminating dangling faces. This approach presents some drawbacks as if the real object surface has concave structures, applying Kohonen's learning algorithm has some difficulties to correctly approximate those parts. In addition, as the Kohonen's algorithm has a high computational cost, the single thread CPU implementation presented in this work took more than one hour to represent the Stanford bunny model. Presented method was only tested with synthetic data and the bunny model, which is comprised of $34,834$ points. Junior et al. [44], extended [43] introducing new mesh operators that allowed it to perform improvements on the surface geometry: edge swap, edge collapse, vertex split and triangle subdivision. Moreover, the method introduced a new step to remove unstable vertices using the mean distance and the standard deviation of the 3D representation regarding the sampled input space. Although this new approach improved the surface geometry, the method does not deal with concave or non-convex regions and the initial structure of the representation has to be pre-established considering the topology of the input space. The fixed structure of the SOM does not learn the spatial relationships between the vertices and therefore does not generate a model that accurately represents the shape of the input space.

In [45] it is used the Growing Cell Structures (GCS) [15] algorithm to reconstruct objects surface. Meshes operators are used to change the connectivity of the mesh and therefore final topology is always equivalent to the initial mesh.

The Topology Representing Networks (TRN), proposed by [14], does not have a fixed structure and also does not impose any constraint about the connection between the nodes. In contrast, this network has a pre-established number of nodes, and therefore, it is not able to generate models with different resolutions. The algorithm was also coined with the term Neural Gas (NG) due to the dynamics of the feature vectors during the adaptation process, which distributes themselves like a gas within the data space. Barhak [46] proposed a NG-based surface reconstruction algorithm since this network has the ability to accurately represent the topology of a point cloud.

In [22], the GNG network is employed to model a point cloud and those regions that need further sampling in order to obtain a more accurate model.

Rescanning at higher resolution is performed for each identified region of interest and a multi-resolution model is built. In this work, only nodes of the generated map are used as the work is focused on sampling capabilities of the GNG. MGNG [23] applied some postprocessing steps in order to perform surface reconstruction once the map is generated using the original GNG algorithm. Most of these approaches were tested against CAD models or synthetic data and only few experiments were performed on objects acquired with 3D sensors. In [47], the GNG algorithm was modified in order to produce topological faces. The extended method was called Growing Self-Reconstruction Maps (GSRM) and some learning steps as CHL and the operation of vertex insertion and removal were also modified. Most experiments of this work were performed on the Stanford dataset, which had been previously filtered and therefore the surface reconstruction step does not have to deal with noisy input spaces produced by common 3D sensors. In [47, 46] the Competitive Hebbian Learning was extended considering the creation of 2-manifold meshes and face reconstruction. However, it was also required to apply some post-processing steps to create a complete model.

Although the use of the SOM-based techniques as NG, GCS or GNG for 3D input space representation and surface reconstruction has already been studied and successful results have been reported, there are some limitations that still persist. Most of these works assumed noise-free point clouds. Therefore, applying these methods on challenging real-world data obtained using noisy 3D sensors have not been object of study yet. Moreover, with the advent of low cost RGB-D cameras as the Microsoft Kinect [1] partial point clouds have to be considered. Besides providing 3D information, these devices also provide colour information, feature that was not considered in the revised works.

As the original NG algorithm does not produce faces and the generated map is a wire-frame representation model, we proposed to extend the original algorithm to produce full coloured meshes (faces). In addition, the proposed method will be tested using noisy data obtained from low-cost RGB-D sensors.

3. Neural Gas Based Surface Reconstruction

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 shape, whilst preserving the topographic structure of the data. A common way to achieve this 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 to exploit the relationship of similarity of the input patterns [48].

3.1. Neural Gas Algorithm

Neural Gas is an unsupervised soft competitive clustering algorithm that given some input distribution in $\mathbb{R}^d$, creates a graph, or network of nodes,

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

where each node in the graph has a position in $\mathbb{R}^d$. The model can be used for vector quantization by finding the code-vectors in clusters. These code-vectors are represented by the reference vectors (the position) of the nodes. It can also be used for finding topological structures that closely reflects the structure of the input distribution. Neural Gas learning is a dynamic algorithm in the sense that if the input distribution slightly changes over time, it is able to adapt, moving the nodes to the new input space.

Neural Gas uses Competitive Hebbian Learning (CHL). CHL assumes a number of centers (neurons) in R^n and successively inserts topological connections among them by randomly evaluating input signals from a data distribution, in this case the three-dimensional space. This scheme is combined with the Neural Gas algorithm, allowing the neurons movement based on a decaying scheme. For each input signal ξ adapt the k-nearest centers (neurons) whereby k is decreasing from a large initial to a small final value. This step allows the adaption of the neurons to the input manifold.
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 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 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 neurons during the adaptation process.

Neural Gas uses parameters that decay exponentially according to time and the distance to the input pattern.

The Neural Gas with CHL algorithm is the following:

1. Initialize the set A to contain N units c_i

$$A = \{c_1, c_2, ..., c_N\} \tag{1}$$

   With reference vectors $W_{c_i} \in \mathbb{R}^d$ chosen randomly according to $p(\xi)$.

   Initialize the connection set $C, C \subset A \times A$, to the empty set:

$$C = \emptyset \tag{2}$$

   Initialize the time parameter t

$$t = 0 \tag{3}$$

2. Generate at random an input signal ξ according to $p(\xi)$.

3. Order all elements of A according to their distance to ξ, i.e., find the sequence of indices $(i_0, i_1, ..., i_{N-1})$ such that w_{i_0} is the reference vector closest to ξ, w_{i_1}, is the reference vector second-closest to ξ and w_{i_k}, $k = 0, ..., N-1$ is the reference vector such that k vectors w_j exists with $||\xi - w_j|| \leq ||\xi - w_k||$. We denote with $k_i(\xi, A)$ the number k associated with w_i.

4. Adapt the reference vectors according to

$$\triangle w_i = \epsilon(t) \cdot h_\lambda(k_i(\xi, A)) \cdot (\xi - w_i) \tag{4}$$

With the following time-dependencies

$$\lambda(t) = \lambda_i(\lambda_f/\lambda_i)^{t/t_{max}} \tag{5}$$

$$\epsilon(t) = \epsilon_i(\epsilon_f/\epsilon_i)^{t/t_{max}} \tag{6}$$

$$h_\lambda(k) = exp(-k/\lambda(t)) \tag{7}$$

5. If it does not exist already, create a connection between i_0 and i_1:

$$C = C \cup \{(i_0, i_1)\} \tag{8}$$

   Set the age of the connection between i_0 and i_1 to zero (”refresh” the edge):

$$age(i_0, i_1) = 0 \tag{9}$$

6. Increment the age of all edges emanating from i_0

$$age_{(i_0,i)} = age_{(i_0,i)} + 1 \qquad \forall i \in N_{i_0} \tag{10}$$

   Thereby, N_c is the set of direct topological neighbors of c.

7. Remove edges with an age larger than the maximal age $T(t)$ whereby

$$T(t) = T_i(T_f/T_i)^{t/t_{max}} \tag{11}$$

8. Increase the time parameter t:

$$t = t + 1 \tag{12}$$

9. If $t \leq t_{max}$ continue with step 2

In summary, the learning step of the NG is based on the Euclidean distance between the input space (3D points) and the neurons structure. Every iteration neurons are sorted by their Euclidean distance to the input pattern that has been randomly selected (activated). Neurons are moved towards the selected input pattern during this step using a decaying function based on the previously calculated Euclidean distance and a time function (iteration). Thanks to this learning step, the neural map is adapted to the topology of the input data. The learning step is defined as a single iteration over the steps detailed above.

Furthermore, the proposed NG method was modified compared to the original version, considering also original point cloud colour information. Once NG network has been adapted to the input space and it has finished learning step, each neuron of the network takes colour information from nearest neighbours in the original input space. Colour information of each neuron is calculated as the average of weighted values of the K-nearest neighbours, obtaining a interpolated value of the surrounding point. Color values are weighted using Euclidean distance from input pattern to its closest neuron reference vector. In addition, this search is considerably accelerated using a Kd-tree structure [49]. Colour down-sampling is performed in order to transfer color information from points to mesh triangles.

3.2. Topology preservation

The final result of the self-organising or competitive learning process is closely related to the concept of Delaunay triangulation. The Voronoi region of a neuron consists of all points of the input space for what this is the winning neuron. Therefore, as a result of CHL a graph (neural network structure) is obtained whose vertices are the neurons of the network and whose edges are connections between them, which represents the Delaunay triangulation of the input space corresponding to the reference vectors of neurons in the network.

Traditionally, it has been suggested that this triangulation, result of competitive learning, preserves the topology of the input space. However, [14] introduces a new condition which restricts this quality.

It is proposed that Φ_w of V in A preserves the neighbourhood when vectors that are close in the input space V are mapped to nearby neurons from network A. It is also noted that the inverse mapping preserves the neighbourhood if nearby neurons of A have associated feature vectors close in the input space.

$$\Phi_w^{-1} : A \to V, c \in A \to w_c \in V \tag{13}$$

Combining the two definitions, it can be defined the Topology Preserving Network (TPN) as the network A whose mappings Φ_w and Φ_w^{-1} preserve the neighborhood.

Thus, self-organising maps or Kohonen maps are not TPN as has traditionally been considered, since this condition only would happen in the event that the topology or dimension of the map and the input space coincide. Since the network topology is established a priori, possibly ignoring the topology of the input space, it is not possible to ensure that the mappings Φ_w and Φ_w^{-1} preserve the neighborhood.

The Growing Cell Structures [15] are not TPN since the topology of the network is established a priori (triangles, tetrahedra, ...). However, it improves the performance compared to Kohonen maps [13], due to its capacity of insertion and removal of neurons.

In the case of the Neural Gases like Growing Neural Gas and Neural Gas, the mechanism for adjusting the network through a competitive learning generates an induced Delaunay triangulation, a graph obtained from the Delaunay triangulation, which has only edges of the Delaunay triangulation of points which belong to the input space V. [14] demonstrated that these models are TPN.

This capability can be used, for instance, in the application of these models to the representation of objects in different dimensions.

In a previous comparative study [25] with Kohonen Maps, Growing Cell Structures and Neural Gas, it was demonstrated that Kohonen Maps and Growing Cell Structures are not topology preserving neural networks. Only NG and GNG are topology preserving networks.

3.2.1. Topology preservation measures

The adaptation of the self-organising neural networks is often measured in terms of two parameters: 1) resolution and 2) degree of preservation of the topology of the input space.

The most widely used measure of resolution is the quantization error [13], which is expressed as:

$$E = \sum_{\forall \xi \in \mathbb{R}^d} \|w_{s_\xi} - \xi\| \cdot p(\xi) \tag{14}$$

where s_ξ is the closest neuron to the input pattern ξ.

However, as NG is used to reconstruct a model from an unorganized noisy point cloud in a reverse-engineering application, it is needed to know the real distance from the generated structure to the input data of the reconstructed model. This measure specifies how close our reconstructed surface is from the original model, removing noisy data generated by 3D sensors and the alignment process. Therefore, calculating the mean square error (MSE) between filtered point cloud and the input data shows the NG input space adaptation and noise removal capabilities. 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).

$$MSE_1 = \frac{1}{|V|} \sum_{\forall p \in V} \min_{i \in A} d(p - w_i)^2 \tag{15}$$

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 network topology learning step is completed.

Moreover, we computed the mean squared error considering distances from the neurons that conform the map to the input space, considering both directions and therefore having a more accurate metric of the input space adaptation.

$$MSE_2 = \frac{1}{|A|} \sum_{\forall c \in A} \min_{i \in V} d(c - w_i)^2 \tag{16}$$

where A is the generated self-organizing map, c is a neuron that belongs to the map, i is sample of the input space with minimum distance to the selected neuron and w_i is the sample weight vector. $d(c-w_i)^2$ is the squared euclidean distance between the closest input point and the neuron c. This measure is also computed once network topology learning step is completed.

Combining mean squared distances in both directions we obtain a useful measure of the map adaptation to the input space. In experiments section we will consider the MSE as the product of MSE_1 and MSE_2.

3.3. NG 3D Surface Reconstruction

3D reconstruction can be considered as a cluster-seeking problem in which the goal is finding a finite number of points that describe the surface precisely. A representation structure (graph) is automatically created by minimizing the error of the self organising map adaptation to the input space to be represented (point cloud).

The ability of neural gases to preserve the topology of the input space will be evaluated in this work with the representation of 3D objects. Identifying the points of the input data that belong to objects allows the network to adapt its structure to this input subspace, obtaining an induced Delaunay triangulation of the object.

Figure 1 shows different 3D models reconstructed using original NG method. Different number of neurons have been used since original models are represented with different number of points.

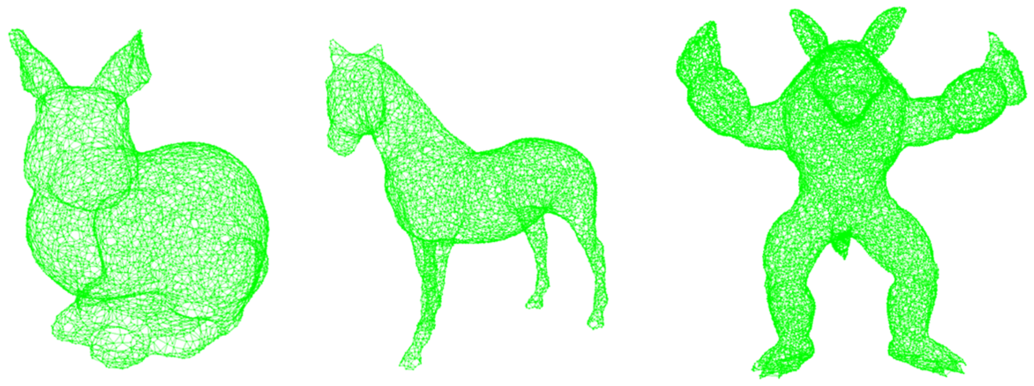


Figure 1: Reconstructed 3D models using NG method (wire-frame representation). From left to right: Bunny, Horse and Armadillo. The number of neurons of the reconstructed models is 5k, 8k and 15k.

As we can see in Figure 1 and as we commented above, original NG algorithm only produces a wire-frame representation of the input data. In order to create a complete mesh, we extended the original algorithm considering the face creation step during the learning process. Steps 5, 6 and 7 of the original algorithm introduced above were modified considering face creation.

Below we can find the modified steps for considering face creation during the learning process of the NG algorithm

5. If it does not exist already, create a connection between i_0 and i_1:

$$C = C \cup \{(i_0, i_1)\} \tag{17}$$

   5.1. if i_0 and i_1 have two common neighbours n_0 and n_1

      5.1.1. if n_0 and n_1 are already connected then

Remove edge between n_0 and n_1

$$C = C - \{(n_0, n_1)\} \tag{18}$$

Coincident faces to the vertices n_0,n_1 are removed

$$F = F - \{incidentFaces(n_0, n_1)\} \tag{19}$$

Create two faces using n_0, n_1, i_0 and i_1

$$F = F \cup \{(n_0, i_0, i_1), (n_1, i_0, i_1)\} \tag{20}$$

5.1.2. if n_0 and n_1 are not connected

Create two faces using n_0, n_1, i_0 and i_1

$$F = F \cup \{(n_0, i_0, i_1), (n_1, i_0, i_1)\} \tag{21}$$

5.2. if i_0 and i_1 have one common neighbours n_0

Create one face using n_0, i_0 and i_1

$$F = F \cup \{(n_0, i_0, i_1)\} \tag{22}$$

6. If a connection between i_0 and i_1 already exist:

Set the age of the connection between i_0 and i_1 to zero (”refresh” the edge):

$$age(i_0, i_1) = 0 \tag{23}$$

6.1. if i_0 and i_1 have common neighbours $N_n = n_0, n_1, n_2, ..., n_i$

For each common neighbour n_i create a face f using n_i, i_0, i_1.

$$F = F \cup \{(n_i, i_0, i_1)\} \tag{24}$$

6.2. if i_0 and i_1 have zero common neighbours

6.2.1. if there are two neighbours n_0 and n_1 of i_1 and i_2 respectively that are connected but are not common to i_1 and i_2 then Triangulate hole: create two faces.
Create two faces using n_0, n_1, i_0 and i_1

$$F = F \cup \{(i_0, i_1, n_0), (i_1, n_0, n_1)\} \tag{25}$$

6.2.2. if there are two neighbours n_0 and n_1 of i_1 and i_2 respectively that are not connected between them and are not common to i_1 and i_2 and also have a common neighbour n_2 then Triangulate hole: create three faces.
Create three faces using n_0, n_1, n_2 i_0 and i_1

$$F = F \cup \{(i_0, i_1, n_2), (i_1, n_1, n_2), (i_1, n_0, n_2)\} \tag{26}$$

Finally, we also extended step 7 removing those faces incident to the edges with an age larger than the maximal age $T(t)$ that is defined in the original algorithm (step 7).

Thanks to these new changes, the proposed version of the NG algorithm is able to create a mesh instead of a wire-frame representation. Figure 2 shows different 3D models reconstructed using the proposed NG extension for 3D surface reconstruction.

4. Neural Gas implementation onto Graphics Processor Units

As the NG learning algorithm has a high computational cost, we propose a method to accelerate it using GPUs and taking advantage of the many-core architecture provided by these devices, as well as their parallelism at

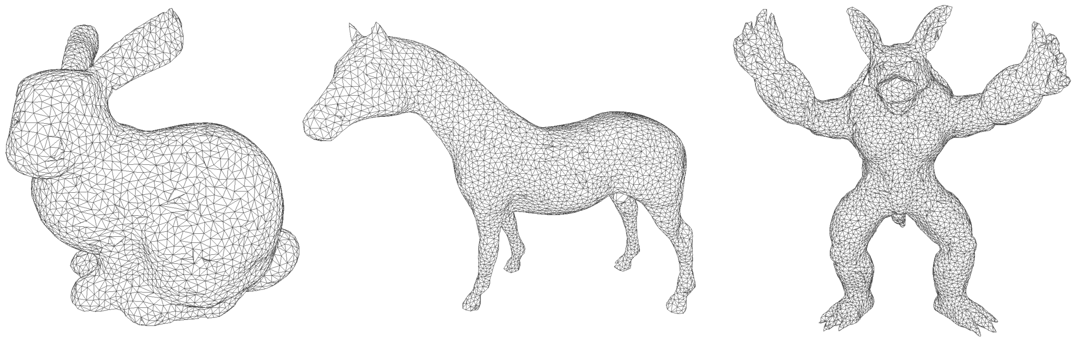


Figure 2: Reconstructed 3D models using the proposed NG-based method (mesh). From left to right: Bunny, Horse and Armadillo. The number of neurons of the reconstructed models is 5k, 8k and 15k.

the instruction level. GPUs are specialised hardware for computationally intensive high level parallelism that uses a higher number of transistors to process data and less for flow control or management of the cache, unlike in CPUs. We have used the architecture and the programming tools (language, compiler, development environment, debugger, libraries, etc.) provided by NVIDIA to exploit their hardware parallelism.

4.1. Cuda architecture

A CUDA compatible GPU is organised in a set of multiprocessors [50]. These multiprocessors called Streaming Multiprocessors (SMs) are highly parallel at thread level. Each SM consists of a series of Streaming Processors (SPs) that share the control logic and cache memory. Each of these SPs can be launched in parallel with a huge amount of threads. CUDA architecture reflects a SIMT model: Single Instruction, Multiple Threads. These threads are executed simultaneously working on large data in parallel. Each of them runs a copy of the kernel on the GPU and uses local indexes to be identified. Threads are grouped into blocks to be executed. The number of blocks that are executed depends on the resources available on the multiprocessor, scheduled by a system of priority queues. CUDA architecture also has a memory hierarchy. Different types of memory can be found: constant, texture, global, shared and local registries. In the last few years, a large number of applications have used GPUs to speed up the processing of neural network algorithms [51, 52, 53, 54, 55, 56], face representation and tracking [57] or pose estimation [58].

4.2. Neural Gas onto GPUs

In order to identify algorithm steps with a larger computational complexity a profiling analysis was performed. Steps 3, 4 and 7, corresponding to Euclidean distances calculation, distances sorting, weights update and edges removing, were identified as steps with the larger computational cost. In addition, many of these operations performed in the NG algorithm can be parallelised because they act on all the neurons of the network simultaneously. That is possible because there is no direct dependence between neurons at the operational level. However, there exists dependence in the adjustment of the network, which makes necessary the synchronization after each iteration t.

The first stage of the algorithm that was accelerated is the calculation of Euclidean distances. This stage calculates the Euclidean distance between a random pattern and each of the neurons. This task takes place in parallel by running the calculation of each neuron distance on as many threads as neurons the network contains. It is possible to calculate more than one distance per thread. This approach increases the level of parallelism as the number of neurons is increased, obtaining a larger speed-up for a large number of neurons.

The second parallelized task was the neurons sorting step, corresponding to step 3 of the NG method. For this task, we used a parallel sorting technique implemented in the Thrust library [59]. Thrust dispatches a highly-tuned Radix Sort algorithm [60] which is considerably faster than alternative comparison-based sorting algorithms such as Merge Sort [61].

Once neurons have been sorted, weights update step is computed in parallel, launching as many threads as neurons comprise the network. Before continuing with the next possible parallelizable step, NG learning algorithm establishes that an edge between first and second winner neurons has to be created (step 5). Moreover, all edges emanating from winner neuron increment their age (step 6). Steps 5 and 6 are not possible to implement in parallel since they affects to only two neurons and their computational cost is really low. Therefore, in order to avoid data communication between CPU and GPU, steps 5 and 6 are performed sequentially on the GPU, launching one GPU thread that performs edge creation and age increment. Finally, step 7 which corresponds to edge age checking is performed in parallel. Parallelizing the algorithm in this way, communication between CPU memory and GPU memory is avoided during the algorithm execution and therefore the main computational bottleneck is prevented.

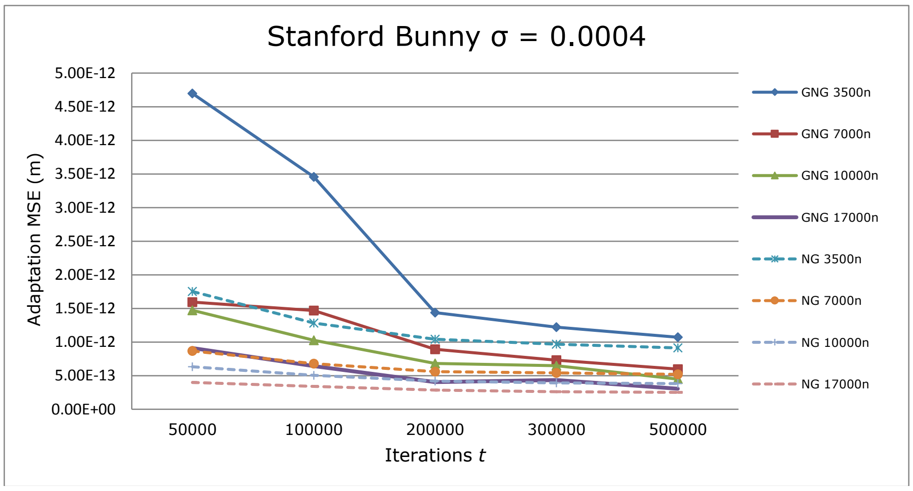
5. Experiments

In this section, some experiments related with the input space adaptation and 3D reconstruction capabilities of the NG method are presented. Firstly, it is performed a comparison with other competitive self-organizing models such as the traditional Self-Organizing Map and the Growing Neural Gas. Quality of representation is studied depending on the parameters established for the selected method (iterations and number of neurons), obtaining different adaptation mean square errors (MSEs). Furthermore, a comparison between state-of-the-art Voxel Grid filtering method and the NG is presented demonstrating the accuracy of the representation generated by the NG network structure. Experiments were performed using noise-free models with added Gaussian noise to simulate noisy data commonly present in low-cost sensors. This scenario provides us with a validation set for testing the NG capability to deal with noisy inputs. Moreover, 3D surface reconstruction capabilities of the proposed method are evaluated and compared against the state-of-the-art Poisson surface reconstruction method. Finally, a performance analysis of the GPGPU accelerated implementation is carried out in order to show the achieved speed-up compared to a CPU implementation.

5.1. Quality of representation

In the first experiment, using the 3D Stanford bunny model, a comparison with other self-organizing models was carried out. Stanford bunny model has 34,834 points. Different levels of Gaussian noise, number of neurons and number of iterations were exhaustively tested obtaining the above commented adaptation MSE error compared to the original model (noise-free). In this way, a downsampled model of the noisy input space is created using different self-organizing models and then input space error is computed against noise-free input space.

Figures 3 and 4 show obtained results for the mean square adaptation



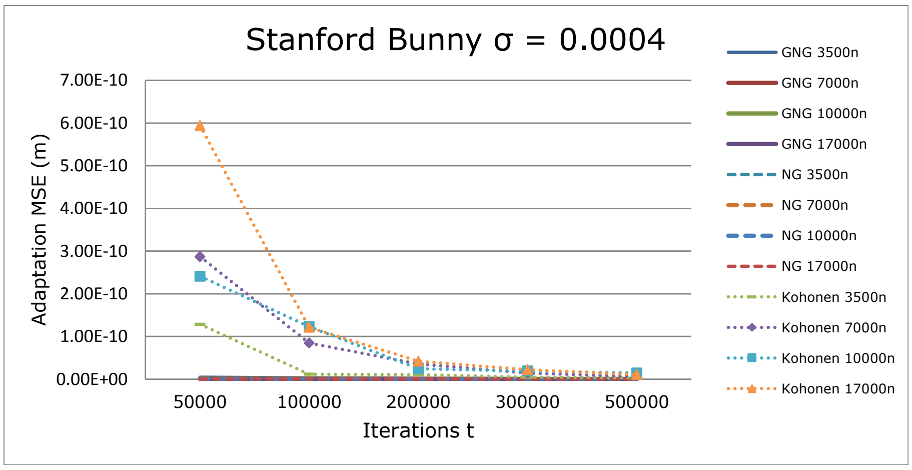
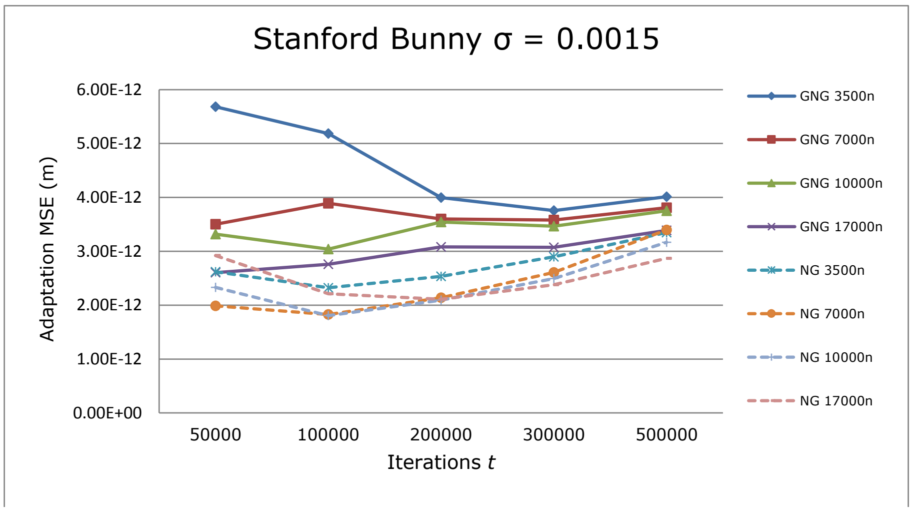


Figure 3: Input space adaptation comparison between different self-organizing models. A level of noise $\sigma = 0.0004$ meters is applied on the Stanford Bunny. **Top**: comparison between NG and GNG. **Bottom**: same comparison including Kohonen and zooming out the chart.



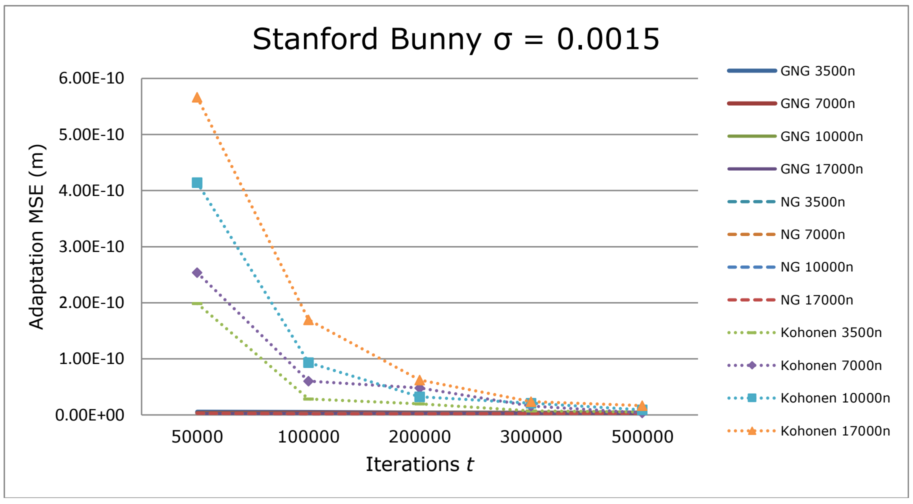


Figure 4: Input space adaptation comparison between different self-organizing models. A level of noise $\sigma = 0.0015$ meters is applied on the Stanford Bunny. **Top**: comparison between NG and GNG. **Bottom**: same comparison including Kohonen and zooming out the chart.

error of adaptation of different self-organizing models. On the top of both figures we can see a comparison between NG and GNG algorithms. On the bottom it is shown the same comparison but including results from the Kohonen representation. Since the Kohonen representation obtained a higher error compared to GNG and NG, it was necessary to zoom out the presented data (Figures 3 and 4 (bottom)) to make it clear. From presented results it can be extracted that the number of iterations parameter converges to a similar spatial approximation error after a defined number of iterations for different models but the NG achived a lower error compared to GNG or Kohonen (traditional SOM). Besides, this was demonstrated using different number of neurons and levels of Gaussian noise. Number of chosen neurons represents 10%, 20%, 30% and 50% of the original number of points in the input space. Finally, it is also demonstrated how as the number of neurons is increased, the MSE error decreased converging to a similar error for latest configurations, 10,000 and 17,000 number of neurons. However, using a large number of neurons will cause that present error in the original model to be also present in the reconstructed model. Therefore, using a reduced number of points to represent noisy models is convenient.

In the case of the Kohonen map it can be also visually apreciated how the adaptation of the generated map is worse compared to the one obtained by GNG or NG algorithms. Figure 5 shows the reconstructed map created using Kohonen algorithm (top) and NG (bottom). As it can be seen, Kononen map does not adapt in an accurate way to the input space, creating some gaps and holes in the representation.

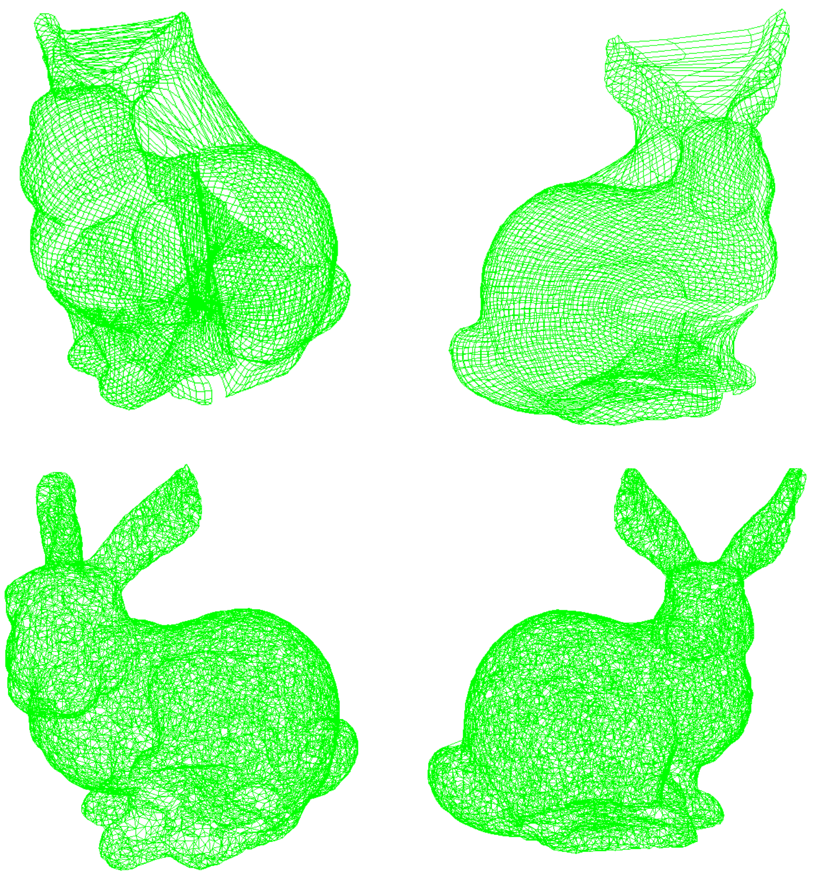


Figure 5: Front and back views of the generated representation using Kohonen (top) and Neural Gas (bottom).

5.2. Neural Gas vs Voxel Grid

The second experiment demonstrates the accuracy of the representation generated by the NG network structure compared with other filtering methods (not based on self-organizing models) such as the Voxel Grid.

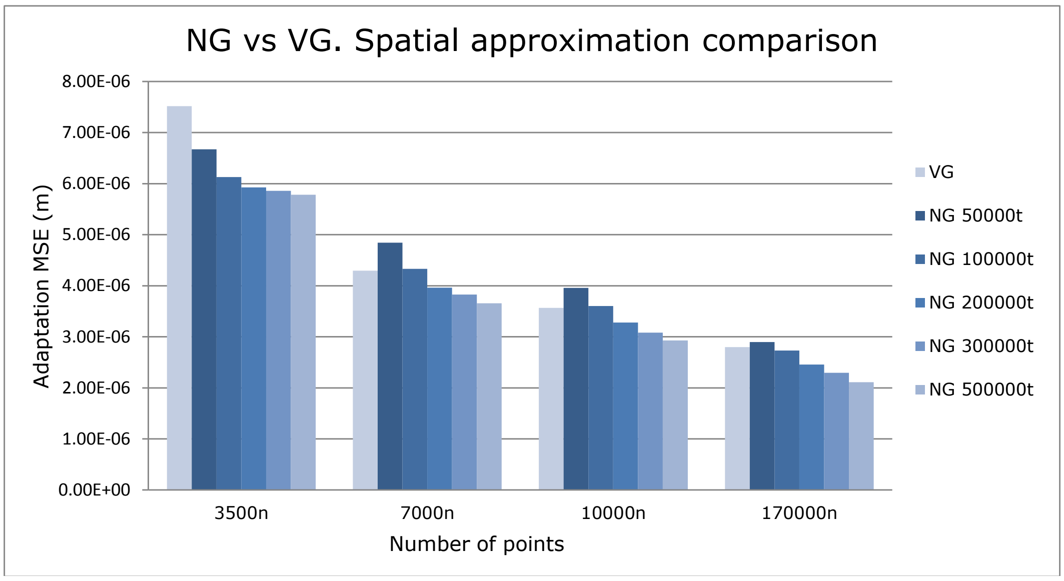


Figure 6: NG parameters study using Stanford bunny model. Three different levels of noise α are applied.

In Figure 6 it is showed how the NG method provides a lower MSE and therefore better adaptation to the original input space, maintaining a better quality of representation in areas with a high degree of curvature and eliminating the noise generated by the sensor. The Voxel Grid method eliminates noise with the sacrifice of information loss of the input space and therefore a worse adaptation. These experiments were performed using different number of points, and in the case of the NG it has been tested with different number of iterations t, obtaining better results with a larger number of iterations with the sacrifice of larger computation times. Voxel Grid method presents other drawbacks as it does not allow specifying a fixed number of points, as the number of points are given by the voxel size used for building the grid. We used the same number of points to perform a fair comparison. By contrast, the NG allows to specify the exact number of points that represent the input space. In Figure 7 it can be visually appreciated the results discussed in Figure 6. It can be observed how the adaption of the filtered points obtained with the Voxel Grid method produces less accurate results than the ones obtained using the NG method.

Another advantage of the NG method for surface reconstruction is showed in Figure 8. NG method allows establishing the number of vertex that represents the reconstructed model, obtaining models with different resolution depending on the number of neurons that comprises the network. Moreover, as the input data is not entirely processed, but it is only sampled one point at time, the time performance of the algorithm is practically independent of the size of the input data set, enabling the use of large datasets.

5.3. Evaluation of surface reconstruction accuracy

We performed an experiment to evaluate the accuracy of the surface reconstruction results obtained by the proposed method compared against one of the state-of-the-art methods, the Poisson surface reconstruction algorithm. We have used Poisson algorithm for creating meshes from different noisy input clouds that were shown above (synthetic model). As in Poisson method it is possible to define the level of depth (accuracy) we want to obtain in the final reconstruction, we performed experiments with different levels of accuracy and therefore number of points. Poisson uses, during the first stage

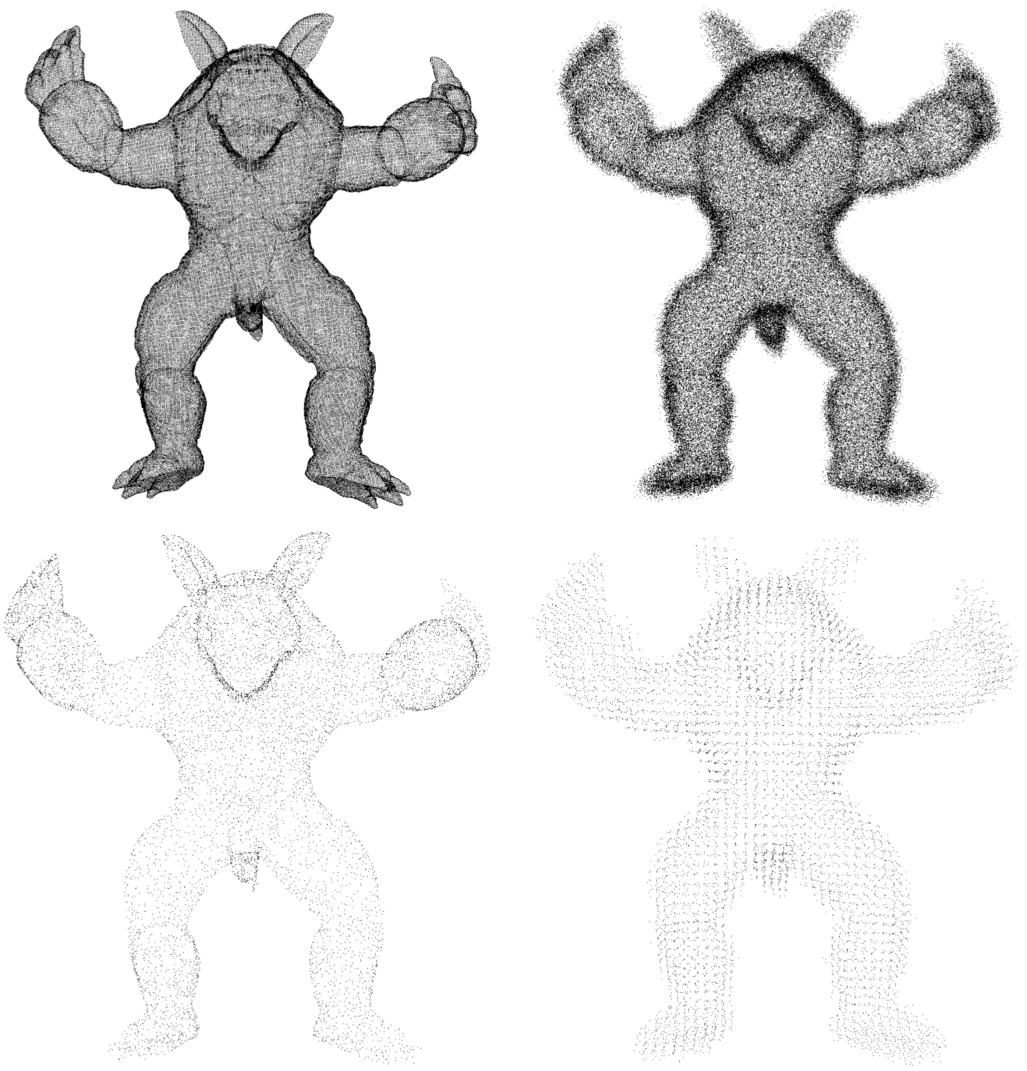


Figure 7: Filtering quality using 10,000 points. NG vs Voxel Grid comparison. Top left: original noise-free model. Top right: noisy model α=0.6 mm. Bottom left: filtered model using NG method $(200,000)$. Bottom right: filtered model using Voxel Grid.

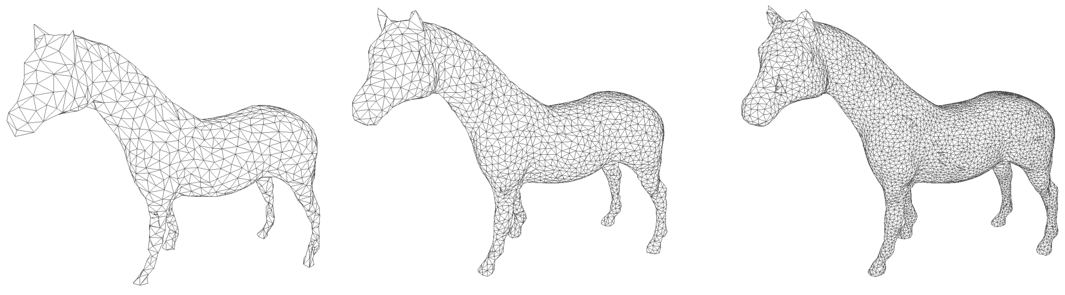


Figure 8: Horse model reconstructed incrementally from an unorganized point cloud of 148k points. From left to right, the number of vertex of the reconstructed mesh is 1k,2.5k and 8k.

of the process, a voxel grid approach to subdivide the space, so similar drawbacks as the ones commented in Section 5.2 were found.

As in previous experiments, we performed the evaluation using noise-free models with added Gaussian noise to simulate noisy data. In order to evaluate the accuracy of the 3D surface reconstruction (mesh) we used a new metric tool to measure the quality of the generated mesh. We computed Hausdorff distance (using the Metro tool [62]) between the synthetic mesh model and the reconstructed ones. In this way, we can see how well methods perform surface reconstruction in presence of noise, and also to measure the adaptation of the reconstructed mesh against the original one (validation set).

Figure 9 shows the results of applying both reconstruction methods to a noisy synthetic point cloud with Gaussian error equals to 1 millimetres (first row). It can be seen how the Poisson algorithm (second row, left) is not able to correctly reconstruct the horse legs due to the amount of error in that area, creating a deformed shape. The rest of the model is successfully reconstructed, but as we will see later, this reconstruction is not an accurate one since most generated surfaces are approximated, and therefore there exist error in terms of Euclidean distance to the original point cloud. We can appreciate this problem on the top part of the horse head, where the geometry of the horse head is not really accurate. The Growing Neural Gas (second row, right) is able to generate a more accurate reconstruction of the original point cloud compared to Poisson. Finally, on the bottom part of the Figure 9 we can see a color map of the distribution error computed using Haussdorff distance. It ranges from red to blue, using red for the areas with largest error and blue for the areas with lowest error. From this color map it can be appreciated how Poisson is not able to reconstruct some parts of the horse model. This happens just because Poisson creates an approximation of the input data and with the presence of noise these approximated surfaces do not accurately represent the original model.

Since both approaches allow us to create meshes with different resolution, we performed the same experiment showed above but increasing the precision of the reconstructed meshes. In Poisson method, we can adjust the depth level parameter, which allows the creation of smaller voxels during the construction of the grid and therefore a more accurate mesh with larger detail. In the case of the proposed method, we can modify the number of neurons (points) and iterations (sampling) during the learning step, so the final reconstructed mesh will increase its precision having a larger number of triangles.

Finally, Table 1 shows all computed Hausdorff distances for all the tests performed using the synthetic model and different levels of noise and accu-

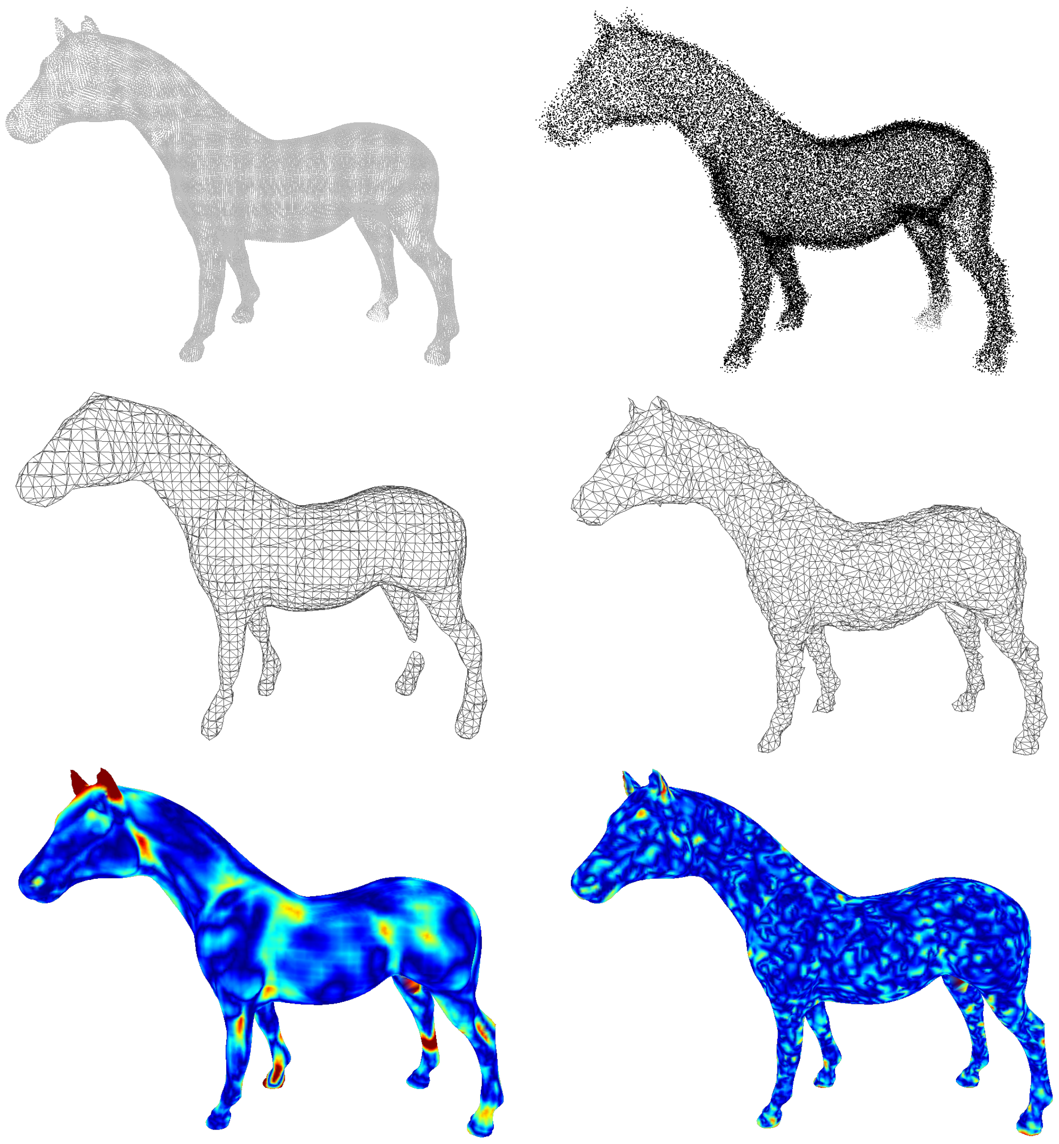


Figure 9: Evaluation of the surface reconstruction accuracy. First row, left column, noise-free point cloud. Right column, point cloud with added Gaussian noise (1 mm). Second row: left, surface reconstruction results using the Poisson method (depth level 6, 3800 points). Right: using the proposed NG-based method (3800 neurons, 200000 iterations). Third row: color maps of the error distribution, left: color map showing the error of the Poisson reconstruction, right: error of the NG-based reconstruction.

racy. It can be seen how the NG-based algorithm outperforms the Poisson surface reconstruction algorithm for models with a certain amount of noise. In most cases as we previously shown, the reconstructed models obtained using Poisson-based method are not able to correctly represent the noisy input data, and therefore in most of these cases the Hausdorff distance (error) is larger compared to the one obtained using the NG algorithm. In addition, we also observed that not only the mean distance is larger, but also the final reconstructed model contains more outliers. This fact is extracted from the maximum error computed using the Hausdorff distance, which is also shown in Table Table 1 . Errors presented in Table 1 are in millimeters. Moreover, Poisson method does not allow to define the number of points for the final reconstructed model while in contrast the proposed method allows to define the number of points that the generated model will have. For these experiments and to establish a fair comparison between both methods we defined for the GNG method the same number of points that Poisson algorithm created for different levels of depth, which is the parameter that enable us to define the accuracy of the reconstructed model.

5.4. Object reconstruction using low cost sensors

The combination of the acquired and reconstructed 3D models with virtual and augmented reality environments allows the users to interact with them and also developing a virtual manufacturing system as an application of augmented and virtual reality to improve several process stages. In this experiment, we applied the proposed method to an unorganized point cloud of an object captured using the Kinect sensor. Obtained point clouds from different views of the object were aligned using an existing marker-based ap-

Method	Error	Points	Min	Max	Mean	RMS
Poisson	1	3,800	0	9.333	0.626	0.958
GNG	1	3,800	0	5.075	**0.307**	**0.41**
Poisson	2	3,800	0	12.651	1.805	2.69
GNG	2	3,800	0	4.175	**0.569**	**0.701**
Poisson	1	10,000	0	11.975	0.726	1.202
GNG	1	10,000	0	5.354	**0.419**	**0.55**
Poisson	2	10,000	0	12.652	1.513	2.411
GNG	2	10,000	0	3.869	**0.528**	**0.654**

Table 1: Hausdorff distances (millimeters) from the reconstructed models to the original noise-free models.

proach [63]. Since the Kinect is essentially a stereo camera, the expected error on its depth measurements is proportional to the distance squared. It ranges from a few millimeters up to about 4-5 cm at the maximum range of the sensor (4 meters). More information about the Kinect sensor error can be found in [64]. Therefore, the proposed method is able to deal with noisy data provided by the Kinect sensor obtaining an accurate 3D surface reconstruction of the captured object without applying other filtering or downsampling additional techniques. Figure 11 shows a 3D reconstructed model computed using the NG method and acquired with a low cost sensor.

The builder helmet model reconstructed in Figure 10 and the textured model presented in Figure 11 could be used in virtual and augmented reality applications, allowing its rendering in a very realistic way. The same occurs in entertainment applications as video-games. The proposed method allows

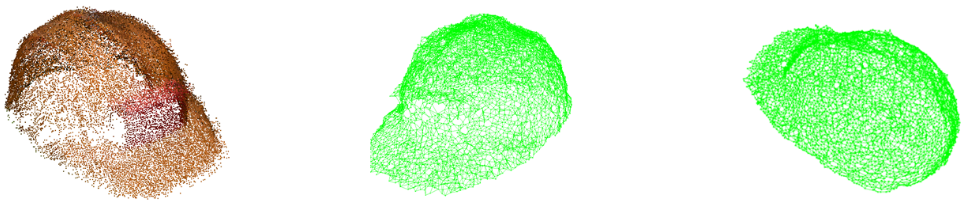


Figure 10: Builder helmet reconstructed using the proposed method. Left: Point cloud obtained using the Kinect sensor and the marker based-approach. Center and right: Different views of the 3D reconstructed surface using the NG method.

designers to capture real objects that are present in our environment using low cost sensors like the Kinect and introduce the reconstructed model in their CAD applications.

The three-dimensional representation of the models provides a higher degree of realism. This factor has been implemented at different stages of production processes in order to improve productivity and product quality.

In Figure 11 it is showed the final reconstructed models of some objects captured using the Kinect sensor. Models texture was transferred from color points of the neurons to mesh triangles interpolating point colors.

5.5. Accelerated GPU Implementation

Finally, in this section, some experiments were performed to validate the accelerated GPU implementation of the proposed NG algorithm. Moreover, some performance tests were carried out comparing the speed-up obtained by our implementation with a naive CPU implementations of the original NG algorithm.

The accelerated version of the NG algorithm has been developed and

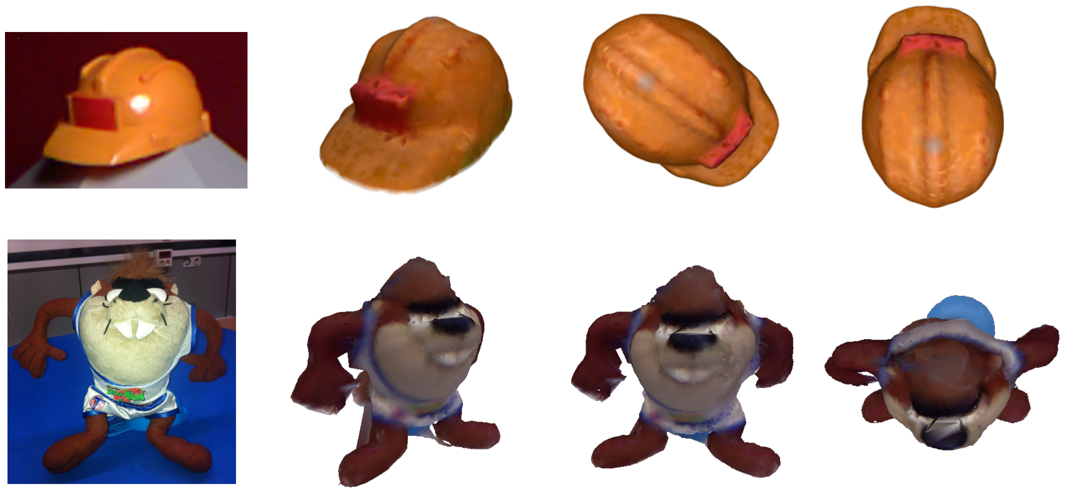


Figure 11: (Left) RGB image. (Right) Different views of the reconstructed objects using the proposed method.

tested on a machine with an Intel Core i3 540 3.07 GHz and different CUDA capable devices. Table 5.5 shows different models that we have used and their features.

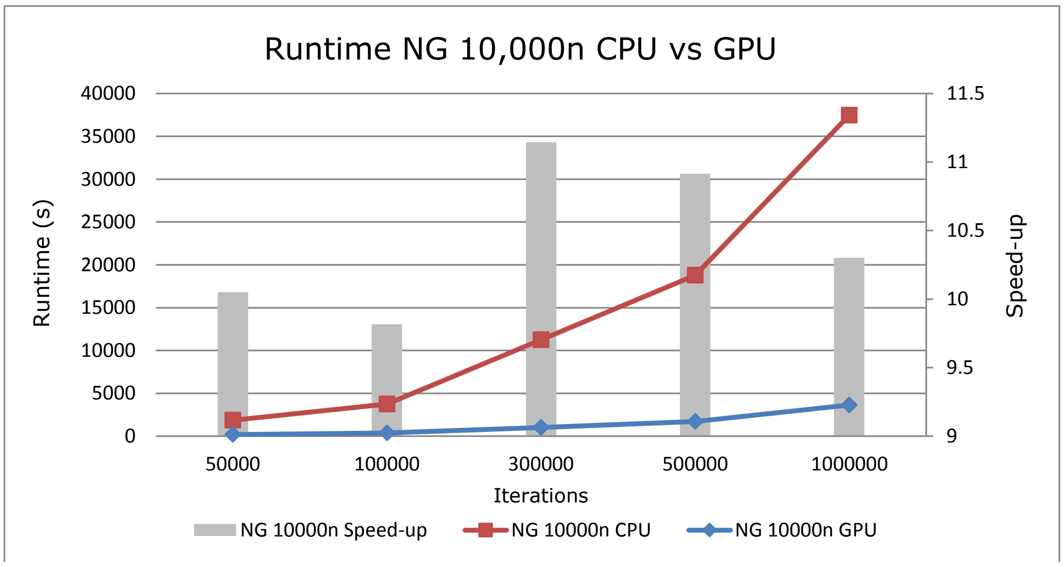
Device Model	Capability	SMs	cores per SM	Global Mem	Bandwidth Mem
Quadro 2000	2.1	4	192	1 GB	41.6 GB/s
GeForce GTX 480	2.0	15	480	1.5 GB	177.4 GB/s

Table 2: CUDA capable devices used in experiments

5.5.1. Speed-up

First performed experiment demonstrates how the speed-up obtained using GPU-based NG implementation is affected by the algorithm learning parameters. As we discussed in Section 4, parallelism of the GPU implementation is mainly limited by the number of neurons that we use and the number of iterations that the algorithm performs does not affect the speed-up obtained. Experiments performed demonstrated that larger number of neurons obtains larger acceleration. This statement is demonstrated in the Figure 12, where two different experiments were performed. First one (top), given a fixed number of neurons runtime execution is measured using different number of iterations. Second one (bottom), runtime execution is measured establishing a fixed number of iterations and increasing the number of neurons. On the top part of the Figure 12, it can be seen how the obtained speed-up is in the range of 9-11x not changing considerably as the number of iterations increases, while on the bottom part it is shown how as the number of neurons increases the speed-up obtained is considerably improved, obtaining a speed-up close to 180x for $100,000$ neurons.

In addition, GPU-based implementation was tested exhaustively on different 3D models with different features and using different learning parameters in order to compare speed-up obtained related to sequential CPU version. Table 3 shows runtime executions for different models with different number of points and using different learning parameters. We can conclude that as we increase the number of neurons and number of iterations the speed-up achieved by the GPU-based NG implementation is considerably increased. Moreover, it is demonstrated again in Table 3, that the number of neurons



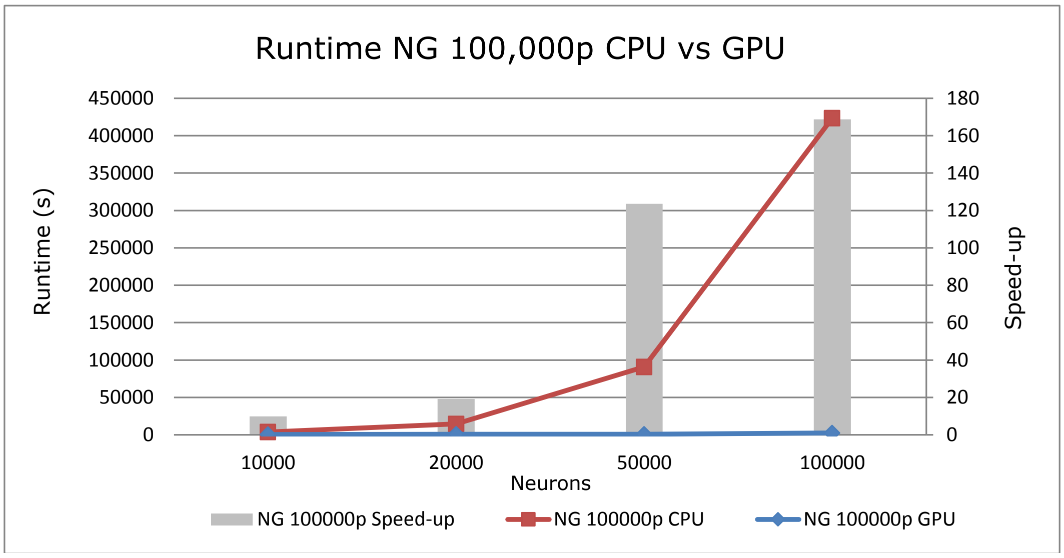


Figure 12: (Top) Runtime execution (GPU GTX480) (lines) and speed-up (bars) for a fixed number of neurons (10, 000) and different number of iterations. (Bottom) Runtime execution (GPU GTX480) (lines) and speed-up (bars) for a fixed number of iterations (100, 000) and different number of neurons.

parameter has more influence in the speed-up achieved than the number of iterations parameter. These results also demonstrate how the proposed methods is suitable for massively parallel architectures such as the GPU, where each thread processes one of the neurons.

Model	Npoints	Nneurons	Iterations	CPU time (s)	GTX480 time (s)	Speed-up	Q2k time (s)	Speed-up
Bunny	34,834	3,000	60,000	233.406	114.468	**2.039x**	109.802	**2.125x**
Bunny	34,834	5,000	80,000	611.39	192.213	**3.180x**	160.771	**3.802x**
Horse	48,485	5,000	80,000	605.561	153.669	**3.940x**	169.053	**3.582x**
Horse	48,485	8,000	100,000	1918.979	309.3116	**6.204x**	256.864	**7.470x**
Armadillo	172,974	10,000	150,000	4306.184	572.912	**7.516x**	526.822	**8.173x**
Armadillo	172,974	15,000	200,000	12881.386	1040.818	**12.376x**	1005.559	**12.810x**
Happy Buda	543,652	20,000	300,000	34189.155	2257.832	**15.142x**	1096.175	**31.189x**
Dragon	3,609,600	50,000	600,000	431100.45	4453.178	**96.807x**	8011.181	**53.812x**

Table 3: Speed-up obtained using accelerated version of the NG algorithm on different 3D models.

In Table 3 it is also shown the speed-up achieved for different GPUs. For a small number of neurons devices behave similarly, but as the number of neurons is increased and therefore the parallelism is also higher, the device with more number of cores (GTX480) achieves better performance (speed-up).

5.5.2. Adjustments per second

We have also performed another experiment that shows how the accelerated version of the NG is not only capable of learning faster than CPU, but also obtains more adjustments per second than the sequential CPU implementation. For instance, learning a network of $20,000$ neurons we can perform 130.71 adjustments per second using the GPU while the single-core CPU gets 8.77 adjustments per second. This means that GPU implementation can obtain a good topological representation with time constraints.

Figure 13 shows the different adjustments rates per second performed by GPU compared to CPU implementation. It is also shown that when increasing the number of neurons in the CPU, it cannot handle a large rate of adjustments per second.

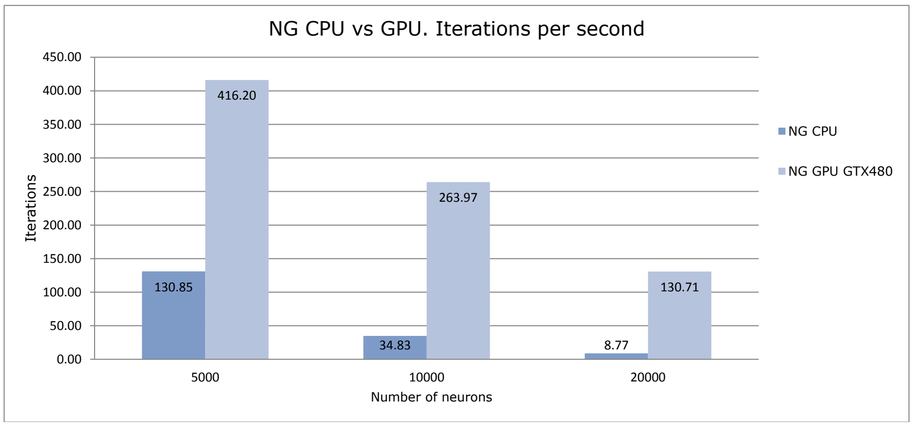


Figure 13: Computed iterations per second. NG CPU vs GPU.(GTX480).

6. Conclusions

In this paper, we have proposed the use of the Neural Gas model to reconstruct objects from the point cloud obtained from overlapped multiple views using low cost sensors. A difference to other methods that may need several stages that includes downsampling, noise filtering and many other tasks, the NG automatically obtains the 3D model of the scanned objects.

To demonstrate the validity of our proposal we tested our method with several models and performed a study of network parameterization calculating the quality of representation and also comparing results with other neural methods like GNG and Kohonen maps and also traditional methods like Voxel Grid. We also reconstruct models acquired with low cost sensors that can be included in virtual and augmented reality environments to redesign or manipulation purpose.

In order to accelerate the NG algorithm we have redesigned and implemented the NG learning algorithm to fit it onto a Graphic Processor Unit using CUDA obtaining in the best case a speed-up of 180x relative to CPU version.

As a further work we plan to use different sensors to acquire models and accelerate the whole process onto GPUs.

7. Acknowledgments

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

References

[1] 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.

[2] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, Surface reconstruction from unorganized points, SIGGRAPH Comput. Graph. 26 (2) (1992) 71–78. doi:10.1145/142920.134011.
URL `http://doi.acm.org/10.1145/142920.134011`

[3] K. Zhou, M. Gong, X. Huang, B. Guo, Data-parallel octrees for surface reconstruction, IEEE Transactions on Visualization and Computer Graphics 17 (5) (2011) 669–681. doi:10.1109/TVCG.2010.75.

[4] W. Saleem, O. Schall, G. Patan, A. G. Belyaev, H.-P. Seidel, On stochastic methods for surface reconstruction., The Visual Computer 23 (6) (2007) 381–395.

[5] R. L. M. do Rego, A. F. R. Araujo, F. B. de Lima Neto, Growing self-organizing maps for surface reconstruction from unstructured point clouds, in: Proc. Int. Joint Conf. Neural Networks IJCNN 2007, 2007, pp. 1900–1905. doi:10.1109/IJCNN.2007.4371248.

[6] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, T. R. Evans, Reconstruction and representation of 3d objects with radial basis functions, in: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, SIGGRAPH '01, ACM, New York, NY, USA, 2001, pp. 67–76. doi:10.1145/383259.383266.

[7] M. Kazhdan, M. Bolitho, H. Hoppe, Poisson surface reconstruction, in: Proceedings of the fourth Eurographics symposium on Geometry processing, SGP '06, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2006, pp. 61–70.

[8] R. J. Campbell, P. J. Flynn, A survey of free-form object representation and recognition techniques, Comput. Vis. Image Underst. 81 (2) (2001) 166–210. doi:10.1006/cviu.2000.0889.

[9] A. R. A. Khan, W.A., K. Cheng, Virtual Manufacturing, Vol. XVIII, 802 p, Springer-Verlag, London., 2011.

[10] J.-Y. Oh, W. Stuerzlinger, J. Danahy, Sesame: towards better 3d conceptual design systems, in: Proceedings of the 6th conference on Designing Interactive systems, DIS '06, ACM, New York, NY, USA, 2006, pp. 80–89. doi:10.1145/1142405.1142419.

[11] M. Fiorentino, A. Uva, M. D. Fabiano, G. Monno, Improving bimanual 3d input in cad modelling by part rotation optimisation, Computer-Aided Design 42 (5) (2010) 462 – 470, advanced and Emerging Virtual and Augmented Reality Technologies in Product Design. doi:http://dx.doi.org/10.1016/j.cad.2008.12.002.

[12] M. Fiorentino, R. de Amicis, G. Monno, A. Stork, Spacedesign: A mixed reality workspace for aesthetic industrial design, in: Proceedings of the 1st International Symposium on Mixed and Augmented Reality, ISMAR '02, IEEE Computer Society, Washington, DC, USA, 2002, pp. 86–.

[13] T. Kohonen, Self-Organising Maps, Vol. 3rd ed. 2001, XX, 502 p, Springer-Verlag, 2001.

[14] T. Martinetz, K. Schulten, Topology representing networks, Neural Netw. 7 (3) (1994) 507–522. doi:10.1016/0893-6080(94)90109-0.

[15] B. Fritzke, Growing cell structures - a self-organizing network for unsupervised and supervised learning, Neural Networks 7 (9) (1994) 1441 – 1460. doi:http://dx.doi.org/10.1016/0893-6080(94)90091-4.

[16] T. M. Martinetz, S. G. Berkovich, K. J. Schulten, 'neural-gas' network for vector quantization and its application to time-series prediction 4 (4) (1993) 558–569.

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

[18] J. Vaščák, Using neural gas networks in traffic navigation, Acta Technica Jaurinensis 2 (2) (2009) 203–215, iSSN 1789-6932.

[19] F. Camastra, A. Vinciarelli, Combining neural gas and learning vector quantization for cursive character recognition., Neurocomputing 51 (2003) 147–159.

[20] B. ling Zhang, M. yue Fu, H. Yan, Handwritten signature verification based on neural 'gas' based vector quantization, in: ICPR '98 Proceedings of the 14th International Conference on Pattern Recognition-Volume 2 - 1862-1864, 1998, pp. 17–20.

[21] M. Melato, B. Hammer, K. Hormann, Neural gas for surface reconstruction, Tech. Rep. II-07-08, Department of Informatics, Clausthal University of Technology (Nov. 2007).

[22] A.-M. Cretu, E. M. Petriu, P. Payeur, Evaluation of growing neural gas networks for selective 3d scanning, in: Proc. Int. Workshop Robotic and Sensors Environments ROSE 2008, 2008, pp. 108–113.

[23] Y. Holdstein, A. Fischer, Three-dimensional surface reconstruction using meshing growing neural gas (mgng), Vis. Comput. 24 (2008) 295–302.

[24] V. Morell, M. Cazorla, S. Orts-Escolano, J. Garcia-Rodriguez, 3d maps representation using GNG, in: 2014 International Joint Conference on Neural Networks, IJCNN 2014, Beijing, China, July 6-11, 2014, 2014, pp. 1482–1487. doi:10.1109/IJCNN.2014.6889828.

[25] F. Florez, J. M. Garcia-Rodriguez, J. García, A. Hernandez, Hand gesture recognition following the dynamics of a topology-preserving network, in: Proc. Fifth IEEE Int Automatic Face and Gesture Recognition Conf, 2002, pp. 318–323.

[26] J. G. Rodríguez, A. Angelopoulou, J. M. G. Chamizo, A. Psarrou, S. Orts-Escolano, V. Morell-Giménez, Autonomous growing neural gas for applications with time constraint: Optimal parameter estimation, Neural Networks 32 (2012) 196–208. doi:10.1016/j.neunet.2012.02.032.

[27] G. Parigi, A. Stramieri, D. Pau, M. Piastra, Gpu-based parallel implementation of a growing self-organizing network., in: J.-L. Ferrier, A. Bernard, O. Y. Gusikhin, K. Madani (Eds.), ICINCO (1), SciTePress, 2012, pp. 633–643.

[28] S. Orts, J. Garcia-Rodriguez, D. Viejo, M. Cazorla, V. Morell, Gpgpu implementation of growing neural gas: Application to 3d scene reconstruction, J. Parallel Distrib. Comput. 72 (10) (2012) 1361–1372.

[29] S. Orts-Escolano, V. Morell, J. Garcia-Rodriguez, 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. doi:10.1109/IJCNN.2013.6706719.

[30] J. Garcia-Rodriguez, M. Cazorla, S. Orts-Escolano, 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. doi:10.1007/978-3-642-38682.

[31] H. Edelsbrunner, E. P. Mücke, Three-dimensional alpha shapes, ACM Trans. Graph. 13 (1) (1994) 43–72. doi:10.1145/174462.156635.

[32] J.-D. Boissonnat, Geometric structures for three-dimensional shape representation, ACM Trans. Graph. 3 (4) (1984) 266–286. doi:10.1145/357346.357349.

[33] N. Amenta, S. Choi, R. K. Kolluri, The power crust, in: Proceedings of the sixth ACM symposium on Solid modeling and applications, SMA '01, ACM, New York, NY, USA, 2001, pp. 249–266. doi:10.1145/376957.376986.

[34] T. K. Dey, S. Goswami, Tight cocone: a water-tight surface reconstructor, in: Proceedings of the eighth ACM symposium on Solid modeling and applications, SM '03, ACM, New York, NY, USA, 2003, pp. 127–134. doi:10.1145/781606.781627.

[35] B. Mederos, N. Amenta, L. Velho, L. H. de Figueiredo, Surface recon-

struction for noisy point clouds, in: Third Eurographics Symposium on Geometry Processing, Vienna, Austria, July 4-6, 2005, 2005, pp. 53–62.

[36] W. E. Lorensen, H. E. Cline, Marching cubes: A high resolution 3d surface construction algorithm, in: SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques, Vol. 21, ACM Press, New York, NY, USA, 1987, pp. 163–169. doi:10.1145/37401.37422.

[37] C. Shen, J. F. O'Brien, J. R. Shewchuk, Interpolating and approximating implicit surfaces from polygon soup, ACM Trans. Graph. 23 (3) (2004) 896–904. doi:10.1145/1015706.1015816.

[38] S. Fleishman, D. Cohen-Or, C. T. Silva, Robust moving least-squares fitting with sharp features, ACM Trans. Graph. 24 (3) (2005) 544–552. doi:10.1145/1073204.1073227.

[39] C. Walder, B. Schlkopf, O. Chapelle, Implicit surface modelling with a globally regularised basis of compact support., Comput. Graph. Forum 25 (3) (2006) 635–644.

[40] C. C. L. Wang, Incremental reconstruction of sharp edges on mesh surfaces, Comput. Aided Des. 38 (6) (2006) 689–702. doi:10.1016/j.cad.2006.02.009.

[41] C. Connolly, Cumulative generation of octree models from range data, in: Robotics and Automation. Proceedings. 1984 IEEE International Conference on, Vol. 1, 1984, pp. 25–32. doi:10.1109/ROBOT.1984.1087212.

[42] L. Kobbelt, M. Botsch, A survey of point-based techniques in computer graphics, Computers & Graphics 28 (2004) 801–814.

[43] Y. Yu, Surface reconstruction from unorganized points using self-organizing neural networks yizhou yu, in: In IEEE Visualization 99, Conference Proceedings, 1999, pp. 61–64.

[44] A. Junior, A. D. D. Neto, J. de Melo, Surface reconstruction using neural networks and adaptive geometry meshes, in: Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on, Vol. 1, 2004, pp. –807. doi:10.1109/IJCNN.2004.1380023.

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

[46] J. Barhak, Freeform objects with arbitrary topology from multirange images., Ph.D. thesis, Israel Institute of Technology, Haifa, Israel (2002).

[47] R. L. Mendona Ernesto Rego, A. F. R. Araujo, F. B. de Lima Neto, Growing self-reconstruction maps 21 (2) (2010) 211–223. doi:10.1109/TNN.2009.2035312.

[48] A. Angelopoulou, J. G. Rodriguez, A. Psarrou, Learning 2d hand shapes using the topology preservation model gng, in: Computer Vision ECCV 2006, Vol. 3951 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2006, pp. 313–324.

[49] J. H. Friedman, J. L. Bentley, R. A. Finkel, An algorithm for finding

best matches in logarithmic expected time, ACM Transactions on Mathematics Software 3 (3) (1977) 209–226.

[50] D. B. Kirk, W.-m. W. Hwu, Programming Massively Parallel Processors: A Hands-on Approach, 1st Edition, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2010.

[51] H. Jang, A. Park, K. Jung, Neural network implementation using cuda and openmp, in: Proc. DICTA '08.Digital Image Computing: Techniques and Applications, 2008, pp. 155–161.

[52] S. Oh, K. Jung, View-point insensitive human pose recognition using neural network and cuda 3 (12) (2009) 643 – 646.

[53] J. M. Nageswaran, N. Dutt, J. L. Krichmar, A. Nicolau, A. Veidenbaum, Efficient simulation of large-scale spiking neural networks using cuda graphics processors, in: Proc. Int. Joint Conf. Neural Networks IJCNN 2009, 2009, pp. 2145–2152.

[54] C.-F. Juang, T.-C. Chen, W.-Y. Cheng, Speedup of implementing fuzzy neural networks with high-dimensional inputs through parallel processing on graphic processing units, IEEE T. Fuzzy Systems 19 (4) (2011) 717–728.

[55] J. Garcia-Rodriguez, A. Angelopoulou, V. Morell, S. Orts, A. Psarrou, J. M. Garcia-Chamizo, Fast image representation with gpu-based growing neural gas, in: IWANN (2), 2011, pp. 58–65.

[56] J. Igarashi, O. Shouno, T. Fukai, H. Tsujino, 2011 special issue: Real-time simulation of a spiking neural network model of the basal ganglia

circuitry using general purpose computing on graphics processing units, Neural Netw. 24 (2011) 950–960.

[57] F. Nasse, C. Thurau, G. A. Fink, Face detection using gpu-based convolutional neural networks, in: Proceedings of the 13th International Conference on Computer Analysis of Images and Patterns, CAIP '09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 83–90.

[58] S. Oh, K. Jung, View-point insensitive human pose recognition using neural network and cuda 3 (12) (2009) 643 – 646.

[59] N. Bell, J. Hoberock, Thrust: A Productivity-Oriented Library for CUDA,, Elsevier, 2011, Ch. ch. 26, pp. pp. 359–371.

[60] D. G. Merrill, A. S. Grimshaw, Revisiting sorting for gpgpu stream architectures, in: Proceedings of the 19th international conference on Parallel architectures and compilation techniques, PACT '10, ACM, New York, NY, USA, 2010, pp. 545–546. doi:10.1145/1854273.1854344.

[61] N. Satish, M. Harris, M. Garland, Designing efficient sorting algorithms for manycore gpus, in: Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing, IPDPS '09, 2009, pp. 1–10.

[62] P. Cignoni, C. Rocchini, R. Scopigno, Metro: Measuring error on simplified surfaces, Tech. rep., Paris, France, France (1996).

[63] J. Kramer, N. Burrus, F. Echtler, M. Parker, D. H. C., Hacking the Kinect, Technology in Action, 2012, Ch. 9. Object Modeling and Detection, pp. 173–206.

[64] K. Khoshelham, S. O. Elberink, Accuracy and resolution of kinect depth data for indoor mapping applications, Sensors 12 (2) (2012) 1437–1454.