CrossMark

ORIGINAL ARTICLE

Fast 2D/3D object representation with growing neural gas

Anastassia Angelopoulou[1] · Jose Garcia Rodriguez[2] · Sergio Orts-Escolano[2] ·
Gaurav Gupta[3] · Alexandra Psarrou[1]

Abstract This work presents the design of a real-time system to model visual objects with the use of self-organising networks. The architecture of the system addresses multiple computer vision tasks such as image segmentation, optimal parameter estimation and object representation. We first develop a framework for building non-rigid shapes using the growth mechanism of the self-organising maps, and then we define an optimal number of nodes without overfitting or underfitting the network based on the knowledge obtained from information-theoretic considerations. We present experimental results for hands and faces, and we quantitatively evaluate the matching capabilities of the proposed method with the topographic product. The proposed method is easily extensible to 3D objects, as it offers similar features for efficient mesh reconstruction.

1 Introduction

The images captured of hand gestures, which are effectively a 2D projection of a 3D object, can become very complex for any recognition system. Systems that follow a model-based method [1, 32] require an accurate 3D model that captures efficiently the hand's high Degrees of Freedom (DOF) articulation and elasticity. The main drawback of this method is that it requires massive calculations which makes it unrealistic for real-time implementation. Since this method is too complicated to implement, the most widespread alternative is the feature-based method [16] where features such as the geometric properties of the hand can be analysed using either Neural Networks (NNs) [34, 36] or stochastic models such as Hidden Markov Models (HMMs) [6, 35].

However, for the accurate analysis of the hand's properties, a suitable segmentation that separates the object of interest from the background is needed. Segmentation is a pre-processing step in many computer vision applications. These applications include visual surveillance [5, 10, 18, 20], and object tracking [15, 17, 26]. While a lot of research has been focused on efficient detectors and classifiers, little attention has been paid to efficiently labelling and acquiring suitable training data. Existing approaches to minimise the labelling effort [19, 21, 24, 30] use a classifier which is trained in a small number of examples. Then the classifier is applied on a training sequence, and the detected patches are added to the previous set of examples. Levin et al. [21] start with a small set of hand labelled data and generate additional labelled examples by applying co-

✉ Anastassia Angelopoulou
agelopa@westminster.ac.uk

Jose Garcia Rodriguez
jgarcia@dtic.ua.es

Sergio Orts-Escolano
sorts@dtic.ua.es

Gaurav Gupta
guptaga@westminster.ac.uk

Alexandra Psarrou
psarroa@westminster.ac.uk

1   Faculty of Science and Technology, University of Westminster, 115 New Cavendish Street, Middlesex W1W 6UW, UK

2   Department of Computing Technology, University of Alicante, PO Box 99, 03080 Alicante, Spain

3   Faculty of Media, Arts and Design, University of Westminster, Northwick Park, Middlesex HA1 3TP, UK

training of two classifiers. Nair and Clark [24] use motion detection to obtain the initial training set. Lee et al. [21] use a variant of eigentracking to obtain the training sequence for face recognition and tracking. Sivic et al. [30] use boosting orientation-based features to obtain training samples for their face detector. A disadvantage of these approaches is that either a manual initialization [19] or a pre-trained classifier is needed to initialise the learning process. Having a sequence of images, this can be avoided by using an incremental model.

We decided to use NNs to represent the geometric properties of objects, and more specifically the self-organising maps (SOMs), due to their incremental nature. One of these SOM-based methods is the growing cell structures (GCS) algorithm [8], which is a model formed incrementally. However, it constrains the connections between the nodes, so any model produced during the training stage is always topologically equivalent to the initial topology. The Topology Representing Networks (TRN) approach, proposed by Martinez and Schulten [22], does not have a fixed structure and also does not impose any constraint on 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 distribute themselves like a gas within the data space. However, as the NG has a fixed number of nodes, it is necessary to have some a priori information about the input space to pre-establish the size of the network. This model was extended by Fritzke [9] proposing the Growing Neural Gas (GNG) network, which combined the flexible structure of the NG with a growing strategy. Moreover, the learning adaptation step was slightly modified. This extension enabled the neural network to use the already detected topological information while training in order to conform to the geometry. This approach has the capability to add neurons while preserving the topology of the input space.

Although the use of the SOM-based techniques of NG, GCS or GNG for various data inputs has already been studied and successful results have been reported [4, 13, 14, 27, 31, 32], there are some limitations that still persist. Most of these works assumed noise-free environments and low complexity distributions. Therefore, applying these methods on challenging real world data obtained using noisy 2D[1] and 3D[2] sensors is our main study. These particular non-invasive sensors have been used in the associated experiments and are typical, contemporary technology.

In this work, we extend the method presented in [2] for object representation using the GNG algorithm. This work extends the already proposed method by considering elimination of noisy connections during the learning process and by applying it to 3D datasets. The method is used for the representation of two-dimensional outline of hands and ventricles, which is extended to 3D. Furthermore, we are interested in the minimisation of the user intervention in the learning process; thus, we utilise an automatic criterion for maximum node growth based on topological parameters. We achieve that by taking into consideration that human skin has a relatively unique colour and the complexity or simplicity of the proposed model is decided by information-theoretic measures.

The remainder of the paper is organised as follows. Section 2 introduces the framework for object modelling using topological relations. Section 3 proposes an approach to minimise the user intervention in the termination of the network using knowledge obtained from information-theoretic considerations. In Sect. 4 a set of experimental results is presented that includes 2D and 3D representations before conclusions are drawn in Sect. 5.

2 Characterising 2D objects with modified GNG

GNG [9] is an unsupervised incremental self-organising network independent of the topology of the input distribution or space. It uses a growth mechanism inherited from the Growth Cell Structure [8] together with the Competitive Hebbian Learning (CHL) rule [22] to construct a network of the input date set. In the GNG algorithm [9], the growing process starts with two nodes, and new nodes are incrementally inserted until a predefined conditioned is satisfied, such as the maximum number of nodes or available time. During the learning process, local error measures are gathered to determine where to insert new nodes. New nodes are inserted near the node with the highest accumulated error and new connections between the winner node and its topological neighbours are created.

Identifying the points of the image that belong to objects allows the GNG network to obtain an induced Delaunay triangulation of the objects. In other words, to obtain an approximation of the geometric appearance of the object. Let an object $O = [O_G, O_A]$ be defined by its geometry and its appearance. The geometry provides a mathematical description of the object's shape, size, and parameters such as translation, rotation, and scale. The appearance defines a set of the object's characteristics such as colour, texture, and other attributes.

Given a domain $\mathbf{S} \subseteq \mathbb{R}^2$, an image intensity function $\mathbf{I}(x, y) \in \mathbb{R}$ such that $\mathbf{I} : \mathbf{S} \rightarrow [0, \mathbf{I}_{max}]$, and an object O, its standard potential field $\Psi_T(x, y) = f_T(I(x, y))$ is the

---

transformation $\Psi_T : \mathbf{S} \to [0, 1]$ which associates with each point $(x, y) \in \mathbf{S}$ the degree of compliance with the visual property T of the object O by its associated intensity $\mathbf{I}(x, y)$.

Considering:

- The input distribution as the set of points in the image:

$$\mathbf{A} = \mathbf{S} \tag{1}$$

$$\xi_w = (x, y) \in \mathbf{S} \tag{2}$$

- The probability density function according to the standard potential field obtained for each point of the image:

$$p(\xi_w) = p(x, y) = \Psi_T(x, y) \tag{3}$$

Learning takes place with our modified GNG algorithm where wrong edges in the network are eliminated and the final graph is normalised. Algorithms 1 and 2 describe our extended GNG. During this process, the neural network is obtained which preserves the topology of the object O from a certain feature T. Therefore, from the visual appearance O_A of the object is obtained an approximation to its geometric appearance O_G. Henceforth, the Topology Preserving Graph $TPG = \langle A, C \rangle$ is defined with a set of vertices (nodes) A and a set of connections (edges) C. To speed up the learning, we used the faster Manhattan distance [23] compared to the Euclidean distance in the original algorithm [9].

Figure 1 compares the original GNG algorithm with the modified GNG in 5 simple shapes with curvatures and corners.

We test the performance of the modified GNG by quantitative measures as shown in Table 1. The two measures of topological correctness that we used are the mean Quantisation Error (qe) and the Topology Preservation Error (te) [33], shown in Eqs. 4 and 5 respectively. There are N pixels, or reference vectors $\overrightarrow{x_c}$, representing the input space in the GNG network. Each node $c \in N$ has its associated reference vector $\{x_c\}_{c=1}^{N} \in \mathbb{R}^q$. The reference vectors

Table 1 Topology Preservation measures of the original vs. modified GNG with respect to frames per second (fps)

Shape	Nodes	Original GNG			Modified GNG		
		Fps	QE	TE	Fps	QE	TE
Star-4	71	1.16	2.7551	0	7.30	2.6375	0
Star-6	74	1.11	2.9564	0	6.06	2.9073	0.0014
Cloud	97	0.61	2.7275	0	5.26	2.6561	0
Heart	70	1.38	2.9337	0	5.24	2.9347	0
Lightning	71	1.04	2.9391	0	6.99	2.8138	0

indicate the nodes' position or *receptive field centre* in the input distribution. We first analyse the quantisation error for each node with the Euclidean distance to its Best Matching Unit $(\mathrm{BMU})_{m_{\overrightarrow{x_c}}}$. The Best Matching Unit $(\mathrm{BMU})_{m_{\overrightarrow{x_c}}}$ is the node whose reference vector is closest to the input signal (ξ_w). The mean quantization error (qe) is the average distance between each reference vector and its BMU. For the calculation of topographic error, there is a function $u(\overrightarrow{x_c})$ that is 1 if $\overrightarrow{x_c}$ data vectors first and second BMUs are adjacent and 0 otherwise. The modified version of GNG produces a significant speed increase, with better connections in corners and angles and better topology preservation (less error).

$$qe = \frac{1}{N} \sum_{c=1}^{N} \| \overrightarrow{x_c} - m_{\overrightarrow{x_c}} \| \tag{4}$$

$$te = \frac{1}{N} \sum_{c=1}^{N} u(\overrightarrow{x_c}) \tag{5}$$

As reflected in Table 1, GNG modified version provided lower quantization and topology preservation errors due to the deletion of wrong edges for most cases. However, in a few cases, wrong edges provide a shorter distance between input space and the Delaunay triangulation obtained (see Star-6 TE).
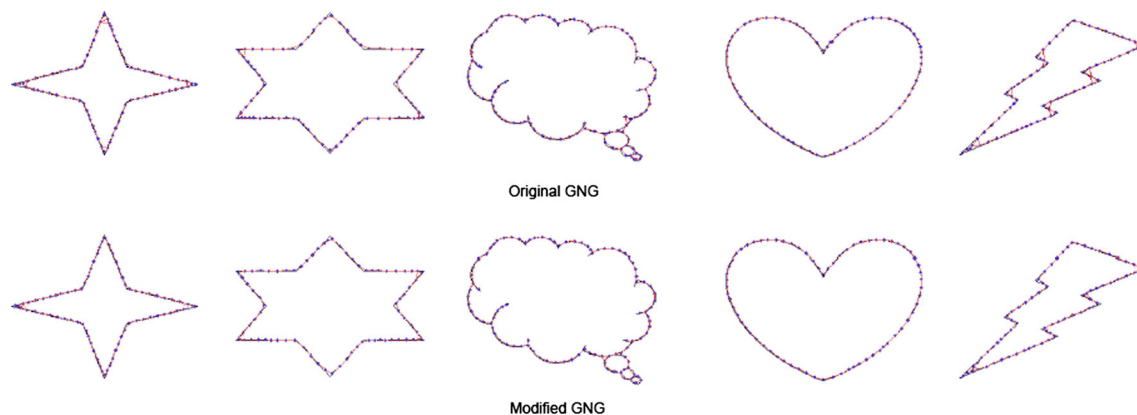

Original GNG

Modified GNG

Fig. 1 The *first row* shows the original GNG while the *second row* shows the modified GNG. With the modified GNG any wrong corrections to corners and curvatures have been eliminated
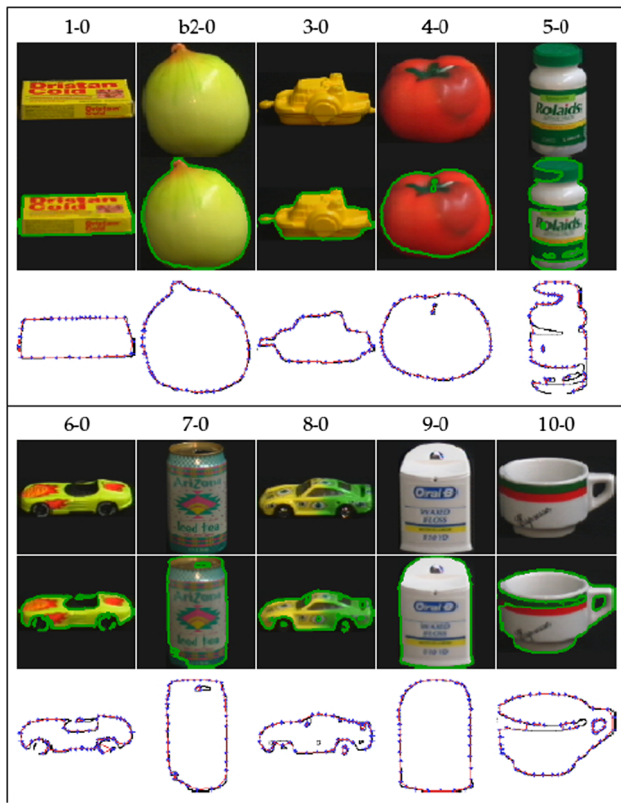
Fig. 2 First shape of each of the first 10 objects in coil-100, showing the original image, the thresholded region, and the modified GNG contour representation

Figure 2 shows another example of the modified GNG applied to shapes extracted from the Columbia Object Image Library (coil-100) dataset. The 100 object coil-100 dataset consists of colour images of 72 different poses for each object. The poses correspond to 5° rotation intervals. Figure 3 shows the modified GNG from our own dataset of hands and shapes. Any wrong connections to corners have been accurate eliminated.

```
input  : N-dimensional input data
output : N-dimensional map
1 We start with a set of nodes A_N defined by coordinates (x_i, y_i) and edges C(A_s, A_t) s ≠ t, and aim to
  represent these in terms of a sequence of lengths l_j and turning angles θ_j. The distance between
  two nodes x_v and x_u is their Manhattan distance d(x_v, x_u). If
  ∃A_i ∈ S : S = {C(A_i, A_y) ∪ C(A_y, A_i)}, |S| = 1, then the contour network has hanging ends
  consisting of some nodes with only one emanating edge. We select one of these as the starting
  node. If not, then every node x_c has two neighbours, so the contour is closed and we arbitrarily
  select some starting node x_c.;
2 Initialise new representation with A_s = x_c, j = 0 and previously considered node f = -1;
3 Set θ_j = 0 and l_j = d(x_v, x_u);
4 Set f = c and c = y;
5 Find new neighbour x_z of node x_c along edge C(A_z, A_c) such that z ≠ f;
6 for j = 1, ......, N do
7     Set θ_j = Θ(A_j, A_{j+1});
8     if [Θ(A_{j-1}, A_j) and Θ(A_j, A_{j+1})] < [Θ(A_{j-1}, A_j) and Θ(A_j, A_{j+2})] then
9         update GNG, remove C(A_j, A_{j+2})
10    end
11 end
12 Repeat from Step 4;
```
Algorithm 1: Eliminating wrong edges.

To normalise the graph that represents the contour we must define a starting point, for example the node on the left-bottom corner. Taking that node as the first we must follow the neighbours until all the nodes have been added

to the new list. If necessary we must apply a scale and a rotation to the list with respect to the centre of gravity of the list of nodes. We achieved the required alignment by applying a transformation T composed by a translation (t_x, t_y), rotation θ, and a scaling s. The normalisation is given by Algorithm 2.

$$T\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} s(\cos\vartheta)x_i & -s(\sin\theta)y_i \\ s(\sin\vartheta)x_i & s(\cos\theta)y_i \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \quad (6)$$

```
input  : N-dimensional map
output : N-dimensional nodes re-ordered
1 Start with N-dimensional map = ⟨A, C⟩;
2 Assign starting node one of the two terminating nodes defined by coordinates A_N(x_i, y_i) and edges
  C(A_s, 1);
3 Find nextNode, the one connecting from the last found node (in this case first node) and update
  previousNode;
4 for i = 1, ....., S do
5     if C(A_i, 1) == nextNode then
6         previousNode == nextNode;
7         nextNode == C(A_i, 2);
8     end
9     if C(A_i, 2) == nextNode then
10        previousNode == nextNode;
11        nextNode == C(A_i, 1);
12    end
13 end
14 Remove the connection that has already been followed;
15 Reduce the number of connections and update list;
16 while the stopping criteria such as the network size is not satisfied do
17     Find if there are more than one follow-on;
18     if none exist then
19         C(A_i, 1) == 0;
20     else if decide which one is next based on their difference then
21         prevPrevNode = prevNode;
22         prevNode = nextNode;
23         nextNode = C(A_i, 2);
24         C(A_i, :) = [];
25         for condition do
26             xVec = Nodes(prevPrevNode, :);
27             yVec = Nodes(prevNode, :);
28             zVec = Nodes(nextNode, :);
29             grad1 = diff(xVec - yVec);
30             grad2 = diff(yVec - zVec);
31             angleVals(j) = grad1 - grad2;
32             min{angleVals(j)};
33         end
34     end
35     Update list, update connections.;
36 end
```
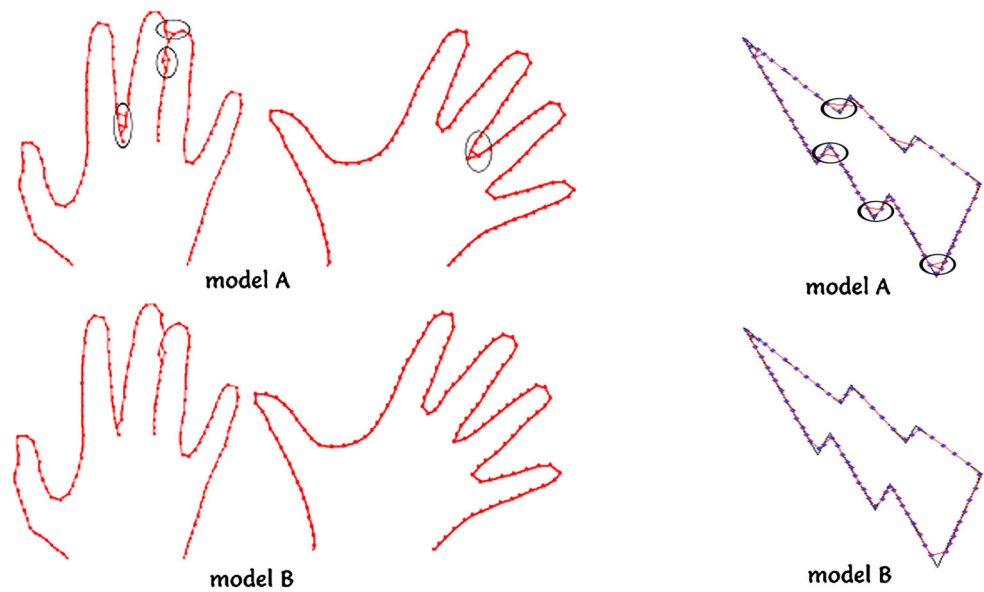**Algorithm 2:** Graph normalisation.

3 Adaptive learning

The determination of accurate topology preservation requires the determination of best similarity threshold and best network map without overfitting. Let $\Omega(x)$ denote the set of pixels in the objects of interest based on the configuration of x (e.g. colour, texture, etc.) and Υ the set of all image pixels. The likelihood of the required number of nodes to describe the topology of an image y is:

$$p(y|x) = \left\{ \prod_{u \in \Omega(x)} p_{skin}(u) \prod_{v \in \Upsilon \setminus \Omega(x)} p_{bkgd}(v) \right.$$
$$\left. \propto \prod_{u \in \Omega(x)} \frac{p_{skin}(u)}{p_{bkgd}(u) + p_{skin}(u)} \right\} * e_T \quad (7)$$

Fig. 3 Modification of the GNG network to eliminate multiple connections and to attempt to reduce the network to a single series of sequentially linked nodes. Model A is the original network with the wrong connections (*circled corners*), while model B is our modified network

model A

model A

model B

model B

and $e_T \leq \prod_{u \in \Omega(x)} p_{skin}(u) + \prod_{v \in \Upsilon \setminus \Omega(x)} p_{bkgd}(v)$.

Figure 4 shows the network map for images with different skin to background ratio.

e_T is a similarity threshold and defines the accuracy of the map. If e_T is low, the topology preservation is lost and more nodes need to be added. On the contrary, if e_T is too big, then nodes have to be removed so that Voronoï cells become wider. For example, let us consider an extreme case where the total size of the image is $I = 100$ pixels and only one pixel represents the object of interest. Let us suppose that we use $e_T = 100$ then the object can be represented by one node. In the case where $e_T \geq I$ then overfit occurs since twice as many nodes are provided.

In our experiments, the numerical value of e_T ranges from $100 \leq e_T \leq 900$ and the accuracy depends on the size of the objects' distribution. The difference between choosing manually the maximum number of nodes and selecting e_T as the similarity threshold, is the preservation of the object independently of scaling operations. Algorithm 3 shows the steps of the automatic criterion added to

the modified GNG algorithm to minimise user intervention in the learning process.

```
input  : Segmented pixels Ω(x) from an unknown image I
output : likelihood p(y|x), TPG
1   Obtain skin colour pixels x by a Mixture of Gaussians in CIE L*, a*, b* space;
2   Obtain e_T value ≤ ∏_{u∈Ω(x)} p_skin(u) + ∏_{v∈Υ\Ω(x)} p_bkgd(v);
3   for every pixel x do
4       if 100 ≤ e_T ≤ 900 then
5           Use Equation 5 to find x_c;
6           if Number of prototypes x_c ≤ 50 then
7           |   Increment e_T until x_c ≥ 120;
8           end
9           if Number of prototypes x_c > 200 then
10          |   Decrement e_T until 120 ≤ x_c < 200;
11          end
12      else
13      |   x_c = maximum prototypes, update list;
14      |   update connections;
15      end
16  end
```
Algorithm 3: Stopping Criterion for Maximum Node Growth.

We can describe the optimum number of similarity thresholds, required for the accuracy of the map for different objects, as the unknown clusters K, and the network parameters as the mixture coefficients W_K, with d-dimensional means and covariances Θ_K. To do that, we use a heuristic criterion from statistics known as the Minimum Description Length (MDL) [28], which does not require an
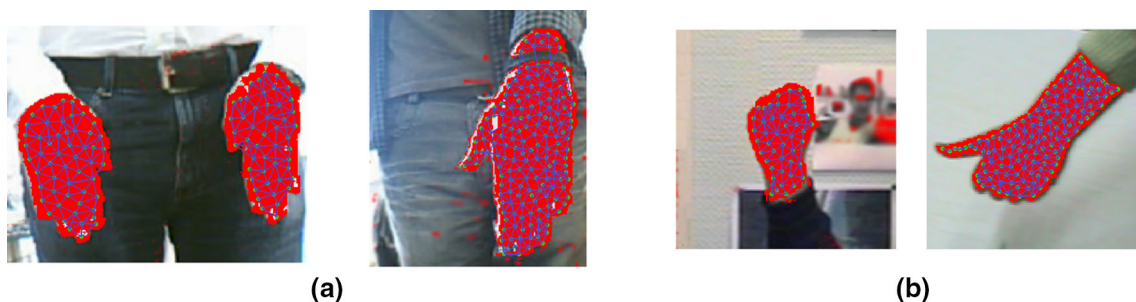

(a)

(b)

Fig. 4 Likelihood node ratios for images with same image resolution but different skin to background ratio. **a** Network adaptation to images of 46,332 pixels with maps of 102 and 162 nodes. **b** Network adaptation to images of 21,903 pixels with maps of 46 and 132 nodes

estimation of the probability $p(Y)$ as is the case for the *conditional entropy* heuristic criterion [3]. The MDL criterion takes the general form of a prediction error, which consists of the difference between two terms:

$$E = model\_likelihood - complexity\_term \quad (8)$$

a likelihood term that measures the model fit and increases with the number of clusters, and a complexity term, used as a penalty, that grows with the number of free parameters in the model. Thus, if the number of cluster is small, we get a low value for the criterion because the model fit is low, while if the number of cluster is large, we get a low value because the complexity term is large.

The information-criterion MDL of Rissanen [28], is defined as:

$$MDL(K) = -\ln[L(X|W_K, \Theta_K)] + \frac{1}{2}M\ln(N) \quad (9)$$

where

$$L(X|W_K, \Theta_K) = \max \prod_{i=1}^{N} p(x_i|W_K, \Theta_K) \quad (10)$$

The first term $-\ln[L(X|W_K, \Theta_K)]$ measures the model probability with respect to the model parameter W_K, Θ_K defined for a Gaussian mixture by the mixture coefficients W_K and d-dimensional means and covariances Θ_K. The second term $\frac{1}{2}M\ln(N)$ measures the number of free parameters needed to encode the model and serves as a penalty for models that are too complex. M describes the number of free parameters and is given for a Gaussian mixture by $M = 2dK + (K-1)$ for $(K-1)$ adjustable mixture weights and 2D parameters for d-dimensional means and diagonal covariance matrices.

The optimal number of similarity thresholds can be determined by applying the following iterative procedure:

- For all K, $(K_{\min} < K < K_{\max})$
  (a) Maximize the likelihood $L(X|W_K, \Theta_K)$ using the EM algorithm to cluster the nodes based on the similarity thresholds applied to the dataset.
  (b) Calculate the value of MDL(K) according to Eqs. 9 and 10
- Select the model parameters (W_K, Θ_K) that correspond to minimisation of the MDL(K) value.

Figure 5 shows the value of MDL(K) for clusters within the range of $(1 < K < 18)$. We have doubled the range in the MDL(K) minimum and maximum values so we can represent the extreme cases of 1 cluster which represents the whole dataset, and 18 clusters which over classify the distribution and corresponds to the overfitting of the network with similarity threshold $e_T = 900$. A global minimum and therefore optimal number of clusters can be

determined for $K = 9$ which indicates that the best similarity threshold that defines the accuracy of the map without overfitting or underfitting the dataset is $e_T = 500$. To account for susceptibility for the *EM* cluster centres as part of the MDL(K) initialisation of the mixture coefficients, the measure is averaged over 10 runs and the minimal value for each configuration is selected. Algorithm 4 summarises the steps.

---
input : likelihood $p(y|x)$, TPG
output: MDL(K)
1 Initialise Algorithm 3;
2 **while** *current number of prototypes* $= x_c$ **do**
3   Calculate MDL(K) according to Equations 9 and 10;
4   Save position of all prototypes and average MDL(K) over 10 runs;
5 **end**

Algorithm 4: MDL(K) Value.

We can now use this optimal network to track objects locally wherever common regions are found. To do that, shape information and colour information from the 1*st* and any subsequent frames are added to the *TPG* map and can be used for the learning in a sequence of k frames. The segmented frame and the stored shape and colour information in each node is given by:

$$S(x; P(g(x, y); t) = p(k|x) \propto P(g(x, y), t-1), TPG_{t-1} \quad (11)$$

Figure 6 shows the convergence of the network with shape and posterior probability per node.

4 Experiments

In this section, different experiments are shown validating the capabilities of our extended GNG method to represent 2D and 3D hand models. The proposed method by considering elimination of noisy connections during the learning process is able to define an optimal number of nodes using the MDL criterion. The method has also been used in 3D datasets. First, a quantitative study is performed adding different levels of noise to the ground truth model (datasets). Using the ground truth models and the generated ones adding noise, we are able to measure the error produced by our method. In addition, our method is compared against the state-of-the-art algorithms Active Shape Models and Poisson surface reconstruction.

All methods have been developed and tested on a desktop machine of 2.26 GHz Pentium IV processor. These methods have been implemented in MATLAB and C++. The Poisson surface reconstruction method has been implemented using the PCL library[3] [29].
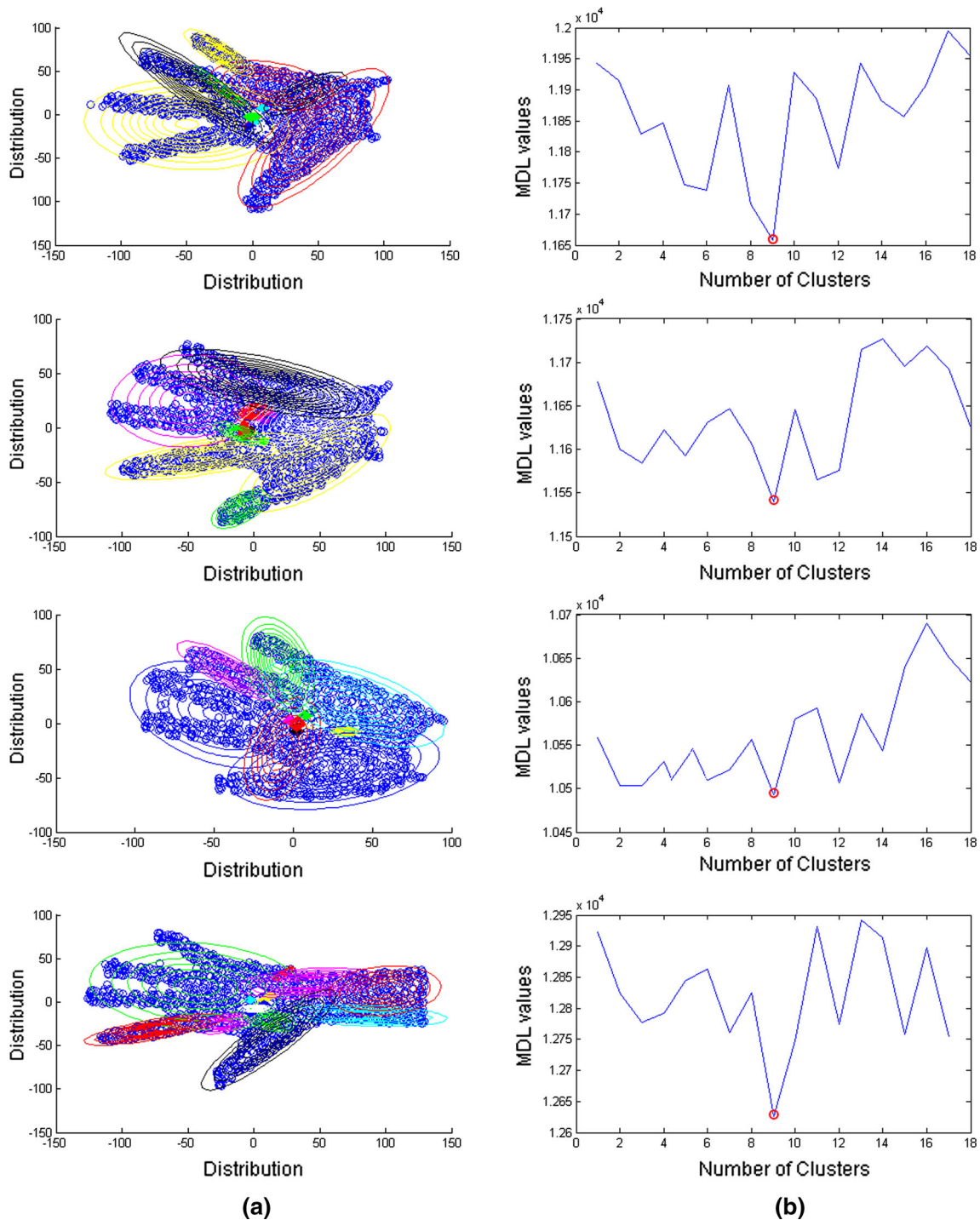

Fig. 5 **a** Plot of hand distributions. **b** Plot of the MDL values versus the number of cluster centres. The Minimum Description Length MDL(K) is calculated for all cluster configurations with $(1 < K < 18)$ clusters, and a global minimum is determined at 9 (*circled point*)

4.1 Benchmark data

We tested our modified GNG network on a dataset of hand images recorded from 5 participants each performing different gestures (Fig. 7) that frequently appear in sign language. To create this dataset, we have recorded images

over several days and a simple webcam was used with image resolution 800×600. In total, we have recorded over 12000 frames, and for computational efficiency, we have resized the images from each set to 300×225, 200×160, 198×234, and 124×123 pixels. We obtained the dataset from the University of Alicante, Spain and the
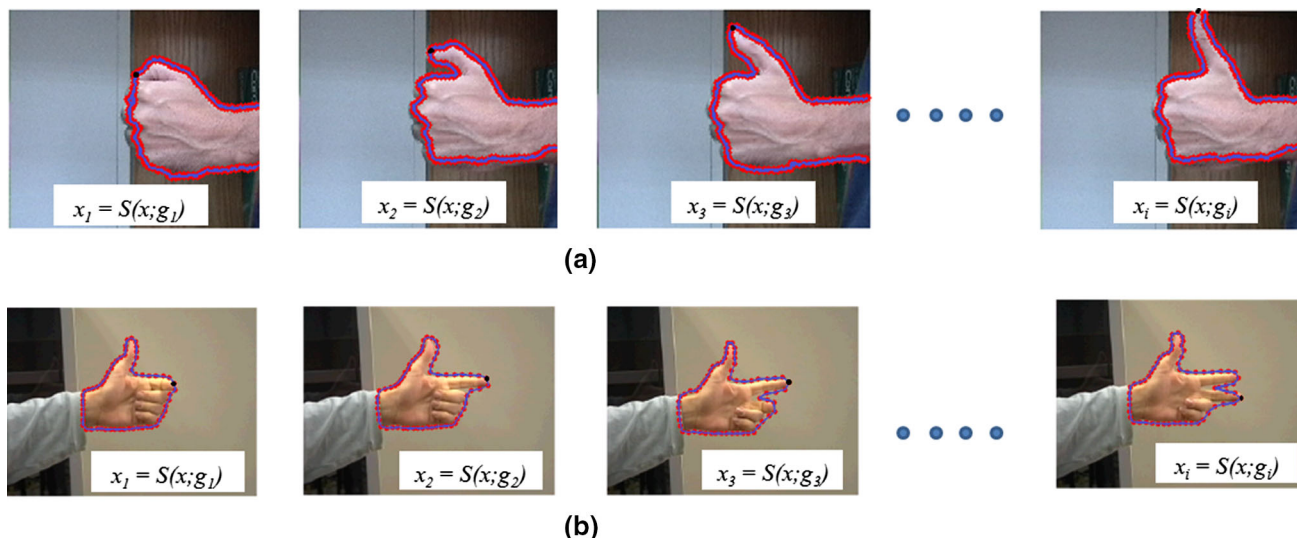
Fig. 6 Network convergence for two sets of images after a sequence of k frames. The network is defined by the shape $S(x; P(g(x, y)))$ and the movement of the nodes depend on the posterior probability $P(g(x, y))$. The higher the probability of a node to belong to the skin prior probability, the faster the node will re-adjust its position to the new input distribution (*black dot*)
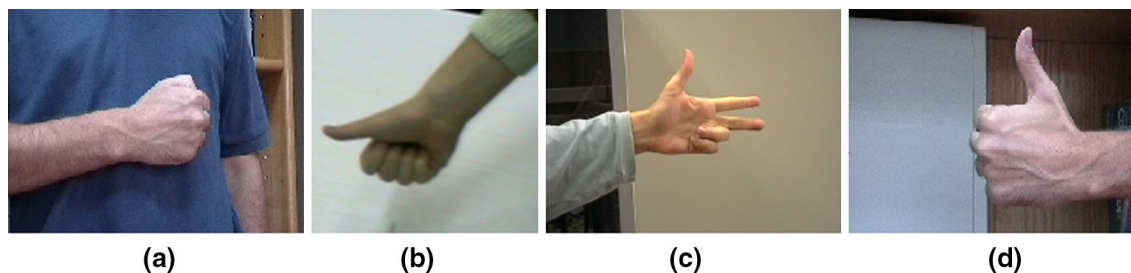

Fig. 7 Some common gestures used in sign language

University of Westminster, UK. Also, we tested our method with 49 images from Mikkel B. Stegmann[4] online dataset. In total we have run the experiments on a dataset of 174 images. Since the background is unambiguous, the network adapts without occlusion reasoning. For our experiments, only complete gesture sequences are included. There are no gestures with partial or complete occluded regions, which means that we do not model multiple objects that interact with the background.

Furthermore, we have performed the experiments having in mind specific applications, thus limiting its applicability. The quality and stability of the results at close range makes it worthwhile for webcam or green screen sign language applications which share a close range viewing distance and a relatively uncluttered background.

We have also tested the system in a more generic background where shadows, changes in lighting and extremely cluttered backgrounds are common. Figure 8 shows that when colour information is incorporated into the network, the system is able to represent the gesture and

only a few nodes adjust to nearby similar pixels. Gesture representation is possible as long as no homogeneity is applied around the gesture.

To classify a region as a hand or face, we take into account domain knowledge information that always respects some proportions found in hands and human faces [11]. To do that we find the centroid, height and width of the connected nodes in the networks as well as the percentage of skin in the rectangular area (Fig. 9). Since the height to width ratio for hands and human faces fall into a small range, we are able to reject or accept if the topology of a network does or does not represent a hand. Studies [7, 11] have shown that the height to width ratio of human face and hands fall within a range based on the well known Golden Ratio (Eq. 12). Thus, we consider a network as a hand or not if the height to width ratio of the region falls within a range of the Golden Ratio ± Tolerance. In the case where the hand is in a folded posture the rule still applies but with different percentage for the *Tolerance*. The values for the *Tolerance* were found by experimentation, and range from 0.5 to 0.7 based on the hand posture.
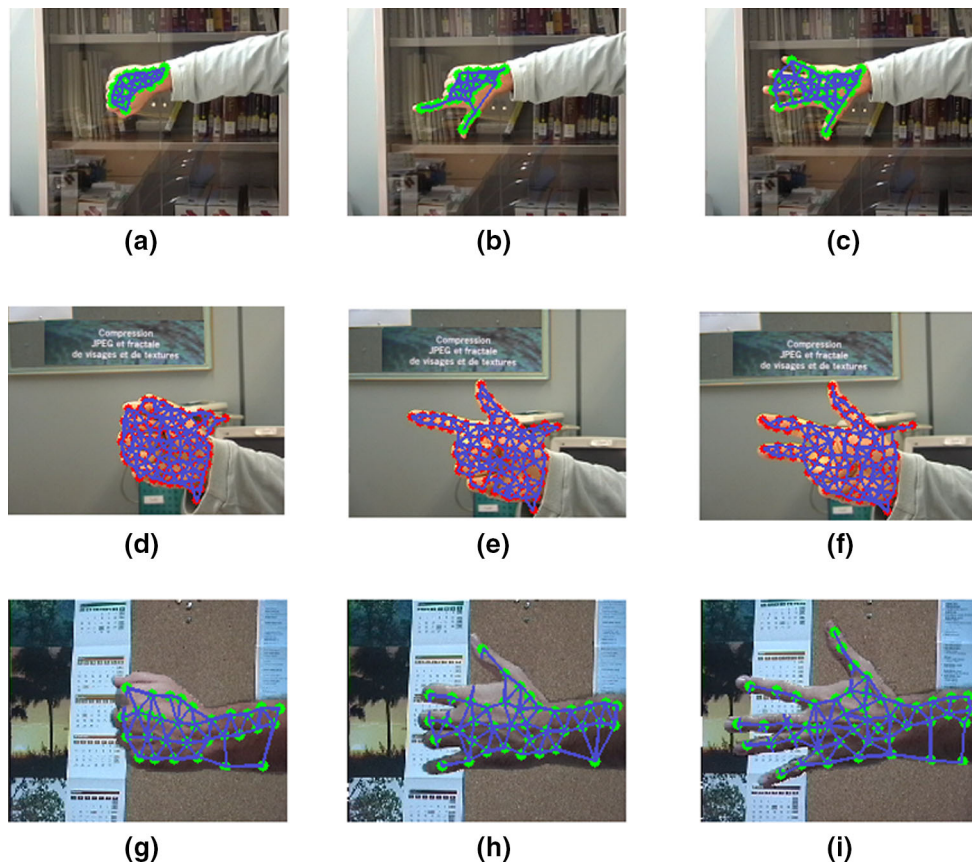

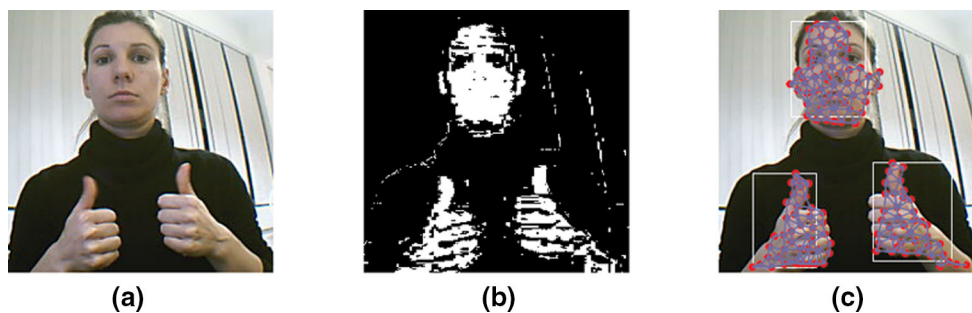**Fig. 8** Examples of gestures in three different cluttered backgrounds

Fig. 9 Example of correctly detected hands and face based on the golden ratio regardless of the scale and the position of the hands and the face. **a** Original image, **b** after applying EM to segment skin region, and **c** hand and face detector taking into account the connected nodes in the networks as well as the percentage of skin in the rectangular area

$$\varphi \equiv \frac{\text{Height}}{\text{Width}} \equiv \frac{(1 + \sqrt{5})}{2} \qquad (12)$$

Table 2 shows topology preservation, execution time, and number of nodes when different variants in the λ and the K are applied in the gesture (d) from Fig. 7 as the input space. Faster variants get worse topology preservation but the network converges quickly. However, the representation is sufficient and can be used in situations where minimum time is required like online learning for detecting obstacles in robotics where you can obtain a rough representation of the object of interest in a given time and with minimum quality.

Figure 10 shows the distribution of two different hand shapes and the plots of the MDL(K) cluster centres within the range of $(1 < K < 18)$. The optimum cluster is achieved at $K = 9$ (circled point).

Table 3 shows the topology preservation error for a number of nodes. We can see that the insertion of more nodes makes no difference to the object's topology. Based on the maximum size of the network, an optimum result is achieved when at least half of the network is developed.

Table 2 Topology preservation and processing time using the quantisation error and the topology preservation error for different variants

Variant	Number of nodes	Time (s)	QE	TE
$GNG_{\lambda=100,K=1}$	23	0.22	8.932453	0.4349
$GNG_{\lambda=100,K=9}$	122	0.50	5.393949	−0.3502
$GNG_{\lambda=100,K=18}$	168	0.84	5.916987	−0.0303
$GNG_{\lambda=300,K=1}$	23	0.90	8.024549	0.5402
$GNG_{\lambda=300,K=9}$	122	2.16	5.398938	0.1493
$GNG_{\lambda=300,K=18}$	168	4.25	4.610572	0.1940
$GNG_{\lambda=600,K=1}$	23	1.13	0.182912	−0.0022
$GNG_{\lambda=600,K=9}$	122	2.22	0.172442	0.3031
$GNG_{\lambda=600,K=18}$	168	8.30	0.169140	−0.0007
$GNG_{\lambda=1000,K=1}$	23	1.00	0.188439	0.0750
$GNG_{\lambda=1000,K=9}$	122	12.02	0.155153	0.0319
$GNG_{\lambda=1000,K=18}$	168	40.98	0.161717	0.0111

Table 3 shows that for the different type of gestures, this optimum number is in the range >90 and <130. Furthermore, the more nodes added during the learning process, the more time it takes for the network to grow (Fig. 11).

Finally, we added different levels of Gaussian noise to three different gestures to test the validity of the modified GNG in comparison with Kohonen map and the growing cell structures (GCS). The results of applying different levels of noise to the gestures are shown in Fig. 12, and error measurements for all methods are calculated in Table 4.

4.2 Variability and comparison with the snake model

Our modified GNG network has been compared to the methodology of the active snake model. The snake converges when all the forces achieve an equilibrium state. The drawbacks with this method are that the snake has no a priori knowledge of the domain, which means it can deform to match any contour; this attribute is not desirable if we want to keep the specificity of the model or preserve the physical attributes such as geometry, topological relations, etc., and that the *active* step is performed globally even if parts of the snake have already converged. Figure 13 shows the tracking of a hand gesture using the modified GNG in the outline of the hand.

Figure 14 shows the fitting results of a snake applied to the same gesture. Figure 14a is the original state of the snake after manually locating an area around the hand. The closer we allocate landmark points around the hand the faster the convergence of the snake. The snake after a number of iterations converges to the palm of the hand but fails to converge around the thumb.

The parameters for the snake are summarised in Table 5. The execution time for modified GNG is approximately 4 times less compared to the snake. The computational and convergence results are summarised in Table 6.

4.3 3D reconstruction

This section shows the result of applying an existing approach proposed by Orts-Escolano et al. [25] for performing 3D surface reconstruction using the GNG algorithm. In this work, we focused on the application of the above-mentioned method for performing reconstruction of human hands and faces that were acquired using the Kinect sensor. Moreover, some experiments were performed using synthetic data.

In [25], the original GNG algorithm is extended to perform 3D surface reconstruction. Furthermore, it considers surface normal information during the learning process. It modifies original Competitive Hebbian Learning process, which only considered the creation of edges between neurons, producing wire-frame 3D representations. Therefore, it is necessary to modify the learning process in order to create triangular faces during network adaptation.

The edge creation, the neurons insertion and the neuron removal stages were extended considering the creation of triangular faces during this process. Algorithm 5 describes the extended CHL to produce triangular faces during the adaption process.

```
input  : A point cloud
output : 3D mesh
1   For each input pattern presented to the network, the two nearest neurons to the input pattern are
    selected as winning neurons s₁ and s₂
2   if s₁ and s₂ are already connected by an edge then
3       if s₁ and s₂ have one or two common neighbours then
4           foreach common neighbour nᵢ do
5               Create a face f using s₁, s₂ and nᵢ;
6           end
7       end
8       if s₁ and s₂ have one or zero common neighbours then
9           if There exist two neighbours n₁ and n₂ of s₁ and s₂ respectively that are connected and
            are not common to s₁ and s₂ then
10              Triangulate rectangular hole: Create two faces using s₁, s₂, n₁ and s₂, s₁, n₂;
11          else if There exist two neighbours n₁ and n₂ of s₁ and s₂ respectively that are not
            connected between them and are not common to s₁ and s₂ and. n₁ and n₂ have a common
            neighbour n₃ then
12              Triangulate pentagonal hole: Create three faces using s₁, s₂, n₂; s₁, n₂, n₃ and s₁, n₁,
                n₃;
13          end
14      end
15      if s₁ and s₂ have two common neighbours n₁ and n₂ then
16          if n₁ and n₂ are already connected then
17              Edge between n₁ and n₂ is removed;
18              Create an edge between s₁ and s₂;
19              Faces coincident to n₁ and n₂ are removed;
20              Create two faces using n₁, s₁, s₂ and n₂, s₁, s₂;
21          else
22              Create an edge between s₁ and s₂;
23              Create two faces using n₁, s₁, s₂ and n₂, s₁, s₂;
24          end
25      else
26          Create edge between s₁ and s₂;
27          if s₁ and s₂ have one common neighbour n₁ then
28              Create a face f using s₁, s₂ and n₁;
29          end
30      end
31  end
```

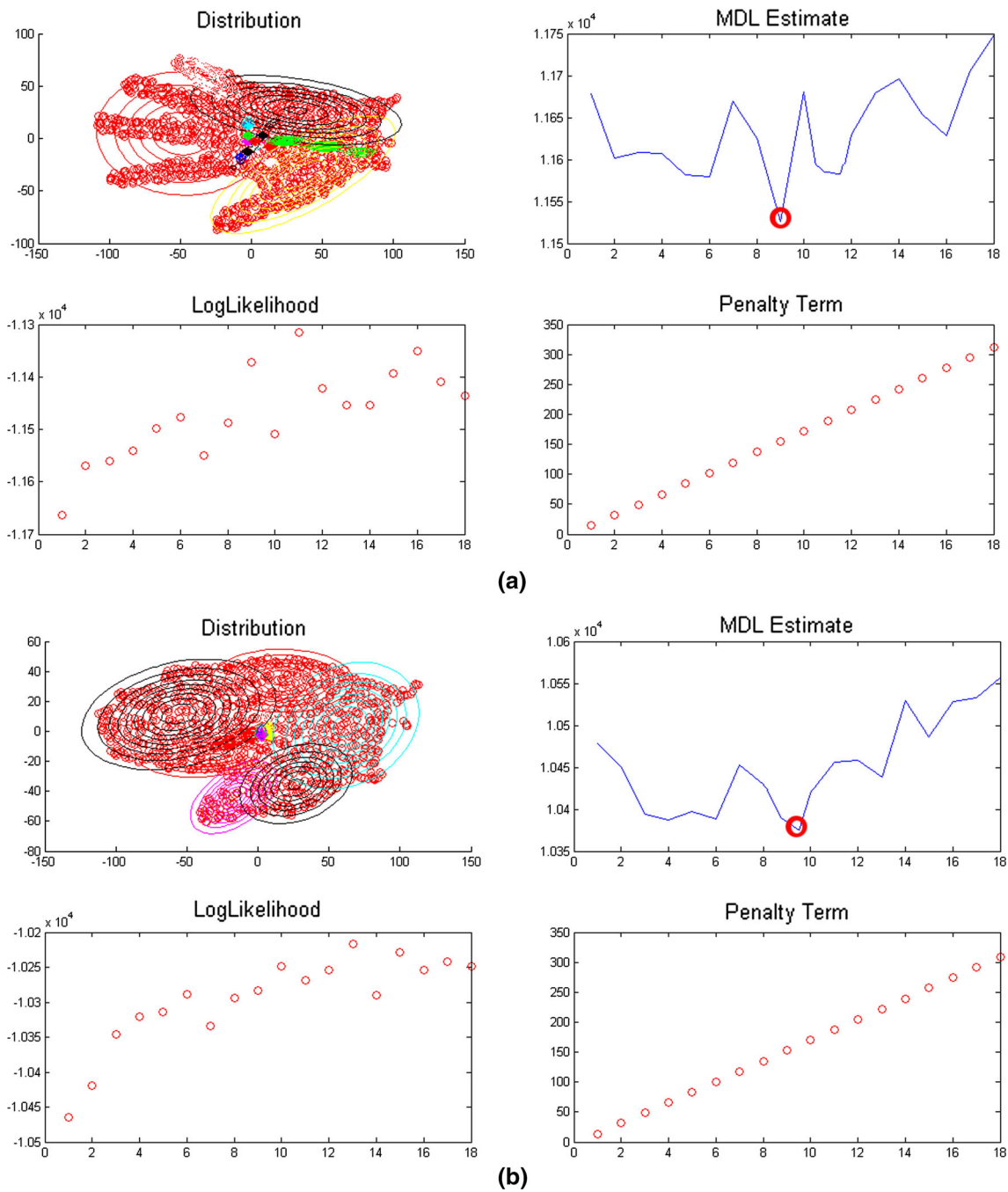**Algorithm 5:** Pseudo-code of the extended CHL stage.

Fig. 10 a, b Distribution of two different hand shapes with plotted MDL(K) values within the range of $(1 < K < 18)$ and a global minimum at 9 (*circled point*). **a, b** also show the likelihood term that measures the model fit and the penalty; both of which grow with the number of used clusters

Figure 15 shows the model created by applying the original GNG algorithm using as an input data a point cloud obtained using the Kinect sensor. It can be appreciated how the GNG produces a wire-frame representation of the input data, but no information about 3D surfaces is provided.

Figure 16 shows the 3D mesh created using the method mentioned above. It can be seen how this extended algorithm is able to create a coloured 3D mesh, surface information, that represents the input data. Since point clouds obtained using the Kinect are partial 3D views, the mesh

Table 3 The topology preservation error for gestures (a–d)

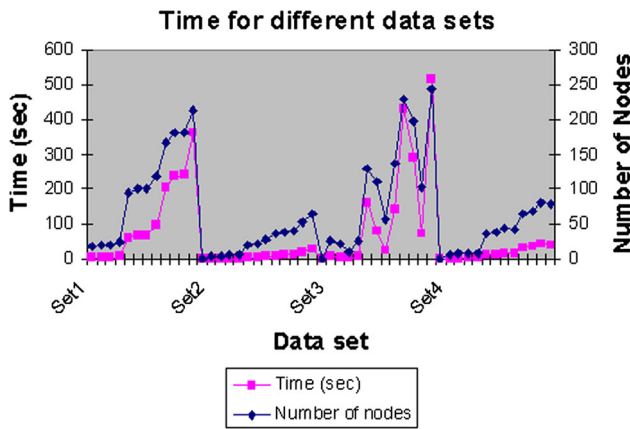| Image (a)		Image (b)		Image (c)		Image (d)	
Nodes	TE	Nodes	TE	Nodes	TE	Nodes	TE
26	−0.0301623	26	−0.021127	24	−0.017626	19	−0.006573
51	−0.030553	51	−0.021127	47	−0.047098	37	−0.007731
77	0.04862	77	0.044698	71	0.046636	56	0.027792
102	0.048256	102	0.021688	95	0.017768	75	0.017573
128	0.031592	128	0.011657	119	0.014589	94	0.018789
153	0.038033	153	0.021783	142	0.018929	112	0.016604
179	0.047636	179	0.017223	166	0.017465	131	0.017755
205	0.038104	205	−0.013525	190	0.017718	150	0.007332
230	0.037321	230	0.017496	214	−0.007543	168	0.007575

Fig. 11 Time taken to insert the maximum number of nodes per dataset

obtained is not complete and therefore the model generated by the GNG is an open coloured mesh.

Moreover, it can also be appreciated that the generated representation is accurate and implicitly it performs some typical computer vision preprocessing steps such as filtering, downsampling and 3D reconstruction.

Figure 17 shows the result of applying the GNG-based method for surface reconstruction applied to complete hand 3D models. These models were synthetically generated using 3D CAD software.

Figure 18 shows the mean square error of different representations of the hand obtained with different numbers of neurons. In addition, the graph shows that with approximately 180 neurons, the adaption error obtained is satisfactory and provides an adequate representation of the input data. We chose the minimum number of neurons with an acceptable quality as it allows real-time processing.
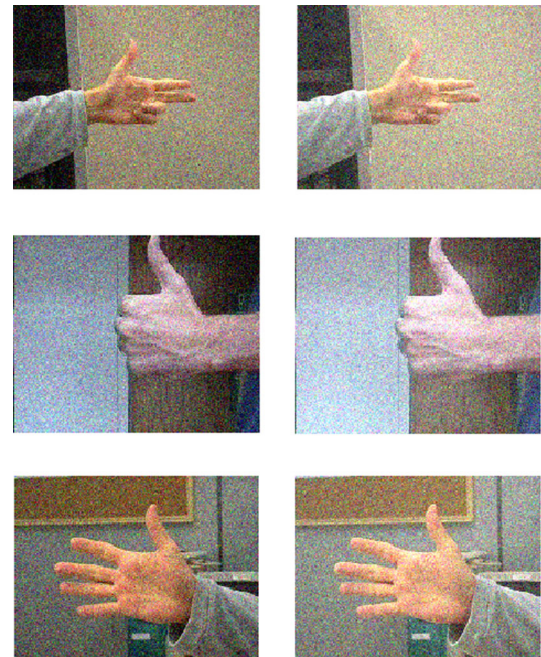

Fig. 12 Gestures with different levels of Gaussian noise. From *left* to *right* mean = 0, sigma = 0; mean = 0, sigma = 0.25
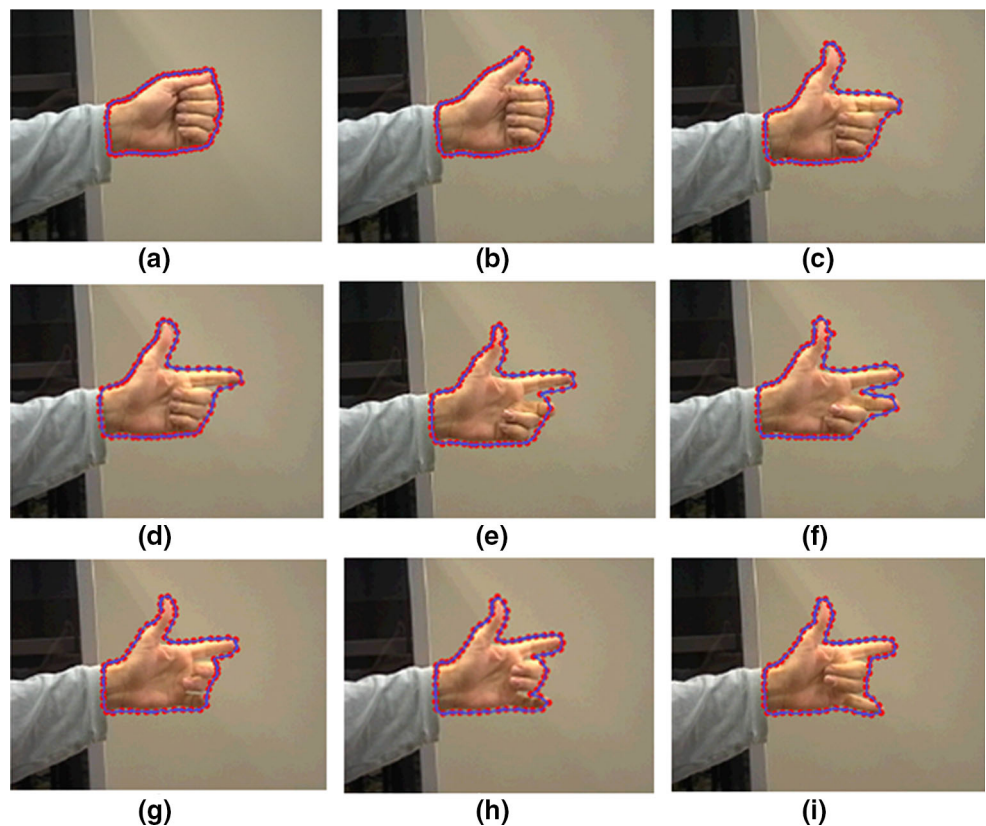
Finally, we performed some experiments using 3D human faces instead of hands to demonstrate that the method can also deal with different shapes. Figure 19 shows the 3D reconstruction of a human face acquired using the Kinect sensor (top) and the 3D reconstruction of a synthetically generated human face (bottom). Both faces were reconstructed using the GNG for 3D surface reconstruction. Synthetic data were generated using the Blensor software [12], for simulating a virtual Kinect sensor (noise-free).

Table 4 Error measurements for modified GNG, Kohonen and GCS

Gestures	Method	Nodes	RMS	TE
Gesture-three fingers (sigma = 0)	Modified GNG	21	**0.2558**	**0.055554**
Gesture-three fingers (sigma = 0)	Kohonen	25	1.6410	0.172629
Gesture-three fingers (sigma = 0)	GCS	30	0.5494	0.159913
Gesture-three fingers (sigma = 0.25)	Modified GNG	21	**1.4189**	**0.083485**
Gesture-three fingers (sigma = 0.25)	Kohonen	25	2.6578	0.237586
Gesture-three fingers (sigma = 0.25)	GCS	30	1.6134	0.241429
Gesture-thumb (sigma = 0)	Modified GNG	25	**0.2440**	**0.046621**
Gesture-thumb (sigma = 0)	Kohonen	30	0.5376	0.194685
Gesture-thumb (sigma = 0)	GCS	31	0.3144	0.176336
Gesture-thumb (sigma = 0.25)	Modified GNG	25	**0.3844**	**0.058153**
Gesture-thumb (sigma = 0.25)	Kohonen	30	0.6956	0.242131
Gesture-thumb (sigma = 0.25)	GCS	31	0.3956	0.239292
Gesture-open hand (sigma = 0)	Modified GNG	23	**0.9660**	**0.048011**
Gesture-open hand (sigma = 0)	Kohonen	25	3.4727	0.146884
Gesture-open hand (sigma = 0)	GCS	27	2.3790	0.150354
Gesture-open hand (sigma = 0.25)	Modified GNG	23	**1.4025**	**0.059658**
Gesture-open hand (sigma = 0.25)	Kohonen	25	3.5340	0.240014
Gesture-open hand (sigma = 0.25)	GCS	27	2.4599	0.112732

Bold numbers demonstrate the lowest errors for our modified GNG network

Fig. 13 Tracking a gesture. The images correspond from *left* to *right* and from top to bottom to every 10th frame of a 190 frame sequence. In each image the *red points* indicate the nodes and their adaptation after 4 iterations (colour figure online)
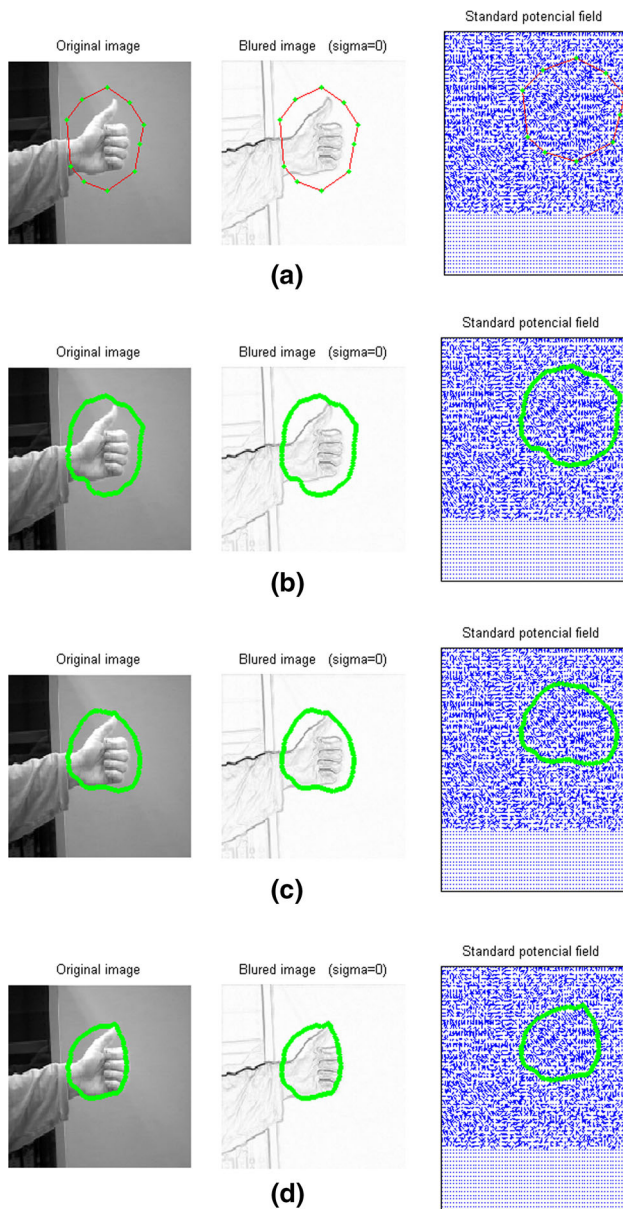

(a)　　　(b)　　　(c)

(d)　　　(e)　　　(f)

(g)　　　(h)　　　(i)

Fig. 14 a Manual initialisation of the snake. **b–d** Adaptation of the snake after a number of iterations

In all our experiments, the parameters of the network are as follows: $\lambda = 100$ to 1000, $\epsilon_x = 0.1$, $\epsilon_n = 0.005$, $\Delta x_{s_1} = 0.5$, $\Delta x_i = 0.0005$, $\alpha_{\max} = 125$.

While 3D downsampling and reconstruction methods like Poisson or Voxelgrid are not able to deal with noisy data, GNG method is able to avoid outliers and obtain an accurate representation in presence of noise. This ability is due to the Hebbian learning rule used and its random nature that update vertex location based on the average influence of a large number of input patterns.

Table 5 Parameters and performance for snake

Hand	Constants	Iterations	Time (s)
Sequence (a)	$\alpha = 0.05$	40	15.29
	$\beta = 0$		
	$\gamma = 1$		
	$\kappa = 0.6$		
	$D_{\min} = 0.5$		
	$D_{\max} = 2$		
Sequence (b)	$\alpha = 4$	50	15.20
	$\beta = 1$		
	$\gamma = 2$		
	$\kappa = 0.6$		
	$D_{\min} = 0.5$		
	$D_{\max} = 2$		
Sequence (c)	$\alpha = 4$	40	12.01
	$\beta = 1$		
	$\gamma = 3$		
	$\kappa = 0.6$		
	$D_{\min} = 0.5$		
	$D_{\max} = 2$		
Sequence (d)	$\alpha = 4$	20	5.60
	$\beta = 1$		
	$\gamma = 3$		
	$\kappa = 0.6$		
	$D_{\min} = 0.5$		
	$D_{\max} = 2$		

Table 6 Convergence and execution time results of modified GNG and snake

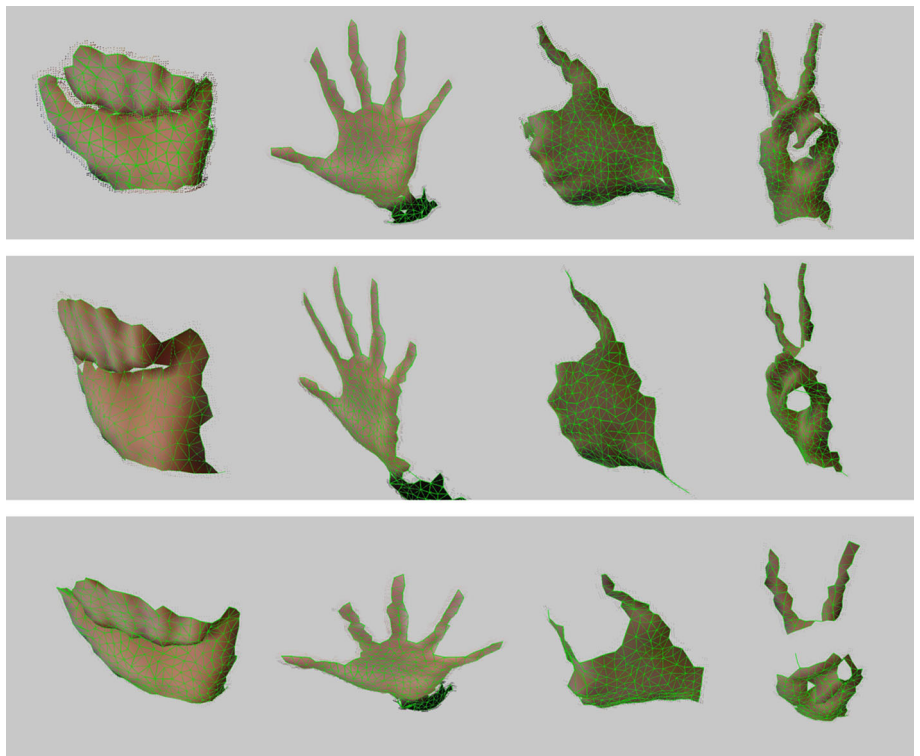| Method	Convergence (iteration times)	Time (s)
Snake	20	5.60
	40	12.01
	50	15.20
	40	15.29
Modified GNG	2	0.73
	2	1.22
	3	2.17
	5	4.88

5 Conclusions and future work

Based on the capabilities of GNG to readjust to new input patterns without restarting the learning process, we developed an approach to minimise the user intervention by utilising an automatic criterion for maximum node growth.

Fig. 15 The two images on the left represent the raw data obtained from the low-cost sensor Kinect. The wire-frame representation generated by the original GNG is shown on the right

Fig. 16 GNG 3D surface reconstructions. 3D reconstruction of different hand poses obtained using the Kinect sensor
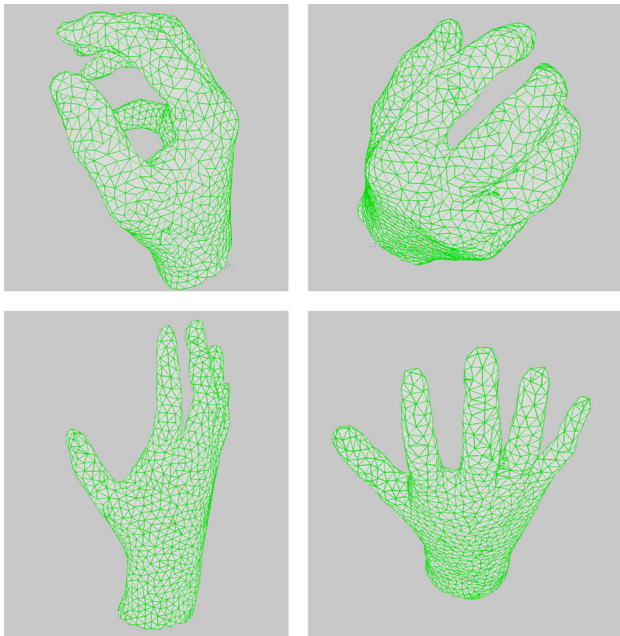
Fig. 17 GNG 3D surface reconstructions. 3D reconstruction of two 3D hand models generated using CAD software
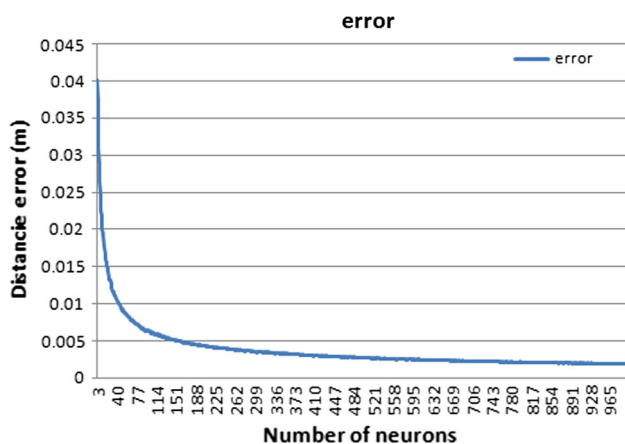

Fig. 18 Mean error based on the number of neurons that compose the network

This automatic criterion for GNG is based on the object's distribution and the similarity threshold (e_T) which determines the preservation of the topology. The model is then used for the representation of motion in image sequences by initialising a suitable segmentation. During testing we found that for different shapes there exists an optimum number that maximises topology learning versus adaptation time and MSE. This optimal number uses knowledge obtained from information-theoretic considerations. Furthermore, we have shown that the low dimensional incremental neural model (GNG) adapts successfully to the high dimensional manifold of the hand by generating 3D models from raw data received from the Kinect. Future work will
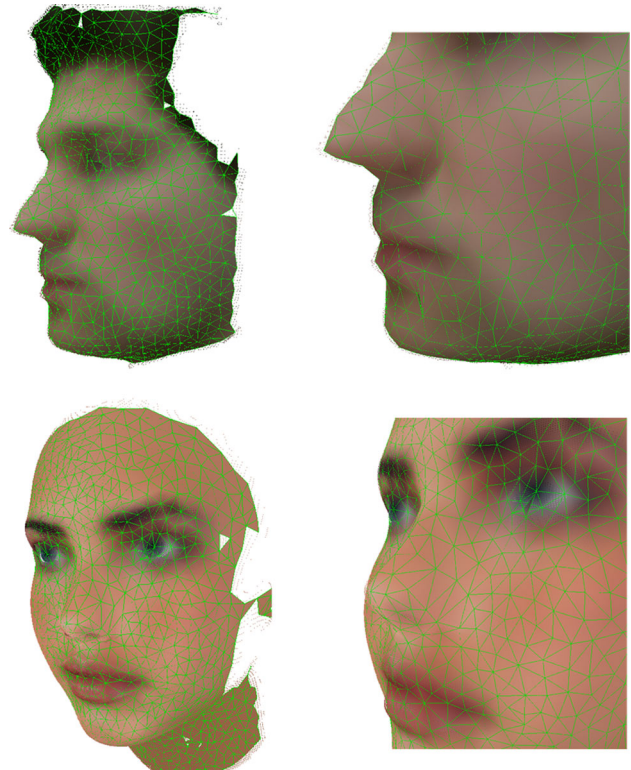

Fig. 19 GNG 3D reconstructions. *Top* 3D face reconstruction from data obtained using the Kinect sensor. *Bottom* 3D face reconstruction from data synthetically generated using the Blensor software

aim at improving system performance at all stages to achieve a natural user interface that allows us to interact with any object manipulation system. Likewise, the acceleration of the whole system should be completed on GPUs.

References

1. Albrecht I, Haber J, Seidel H (2003) Construction and animation of anatomically based human hand models. In: Proceedings of the 2003 ACM SIGGRAPH/eurographics symposium on computer animation, pp 98–109
2. Angelopoulou A, García J, Psarrou A, Gupta G, Mentzelopoulos M (2013) Adaptive learning in motion analysis with self-organising maps. In: The 2013 international joint conference on neural networks (IJCNN), pp 1–7

3. Brown G, Pocock A, Zhao M-J, Luján M (2012) Conditional likelihood maximisation: a unifying framework for information theoretic feature selection. J Mach Learn Res 13:26–66

4. Cretu A, Petriu E, Payeur P (2008) Evaluation of growing neural gas networks for selective 3D scanning. In: Proceedings of IEEE international workshop on robotics and sensors, environments, pp 108–113

5. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, vol I, pp 886–893

6. Eddy S (1996) Hidden Markov models. Curr Opin Struct Biol 6(3):361–365

7. Fernandez A, Ortega M, Cancela B, Penedo MG (2012) Contextual and skin color region information for face and arms location. In: Proceedings of the 13th international conference on computer aided systems theory—EUROCAST 2011, vol 6927/2012, pp 616–623

8. Fritzke B (1994) Growing cell structures—a self-organising network for unsupervised and supervised learning. J Neural Netw 7(9):1441–1460

9. Fritzke B (1995) A growing neural gas network learns topologies. In: Advances in neural information processing systems 7 (NIPS'94), pp 625–632

10. García-Rodríguez J, Angelopoulou A, García-Chamizo JM, Psarrou A, Escolano SO, Giménez VM (2012) Autonomous growing neural gas for applications with time constraint: optimal parameter estimation. Neural Netw 32:196–208

11. Govindaraju V (1996) Locating human faces in photographs. Int J Comput Vision 19(2):129–146

12. Gschwandtner M, Kwitt R, Uhl A, Pree W (2011) BlenSor: blender sensor simulation toolbox advances in visual computing, volume 6939 of lecture notes in computer science, chap 20. Springer, Berlin

13. Gupta G, Psarrou A, Angelopoulou A, García J (2012) Region analysis through close contour transformation using growing neural gas. In: Proceedings of the international joint conference on neural networks, IJCNN2012, pp 1–8

14. Holdstein Y, Fischer A (2008) Three-dimensional surface reconstruction using meshing growing neural gas (MGNG). Vis Comput Int J Comput Graph 24(4):295–302

15. Kakumanu P, Makrogiannis S, Bourbakis N (2007) A survey of skin-color modeling and detection methods. Pattern Recogn 40(3):1106–1122

16. Koike H, Sato Y, Kobayashi Y (2001) Integrating paper and digital information on enhanced desk: a method for real time finger tracking on an augmented desk system. ACM Trans Comput Hum Interact 8(4):307–322

17. Kruppa H (2004) Object detection using scale-specific boosted parts and a Bayesian combiner. PhD Thesis, ETH Zrich

18. Kruppa H, Santana C, Sciele B (2003) Fast and robust face finding via local context. In: Proceedings of the IEEE international workshop on visual surveillance and performance evaluation of tracking and surveillance, pp 157–164

19. Lee M, Nevatia R (2005) Integrating component cues for human pose estimation. In: Proceedings of the IEEE international workshop on visual surveillance and performance evaluation of tracking and surveillance, pp 41–48

20. Leibe B, Seemann E, Sciele B (2005) Pedestrian detection in crowded scenes. In: Proceedings of the IEEE conference on computer vision and pattern recognition, vol I, pp 878–885

21. Levin A, Viola P, Freund Y (2003) Unsupervised improvement of visual detectors using co-training. In: Proceedings of the IEEE international conference on computer vision, vol I, pp 626–633

22. Martinez T, Schulten K (1994) Topology representing networks. J Neural Netw 7(3):507–522

23. Mignotte M (2008) Segmentation by fusion of histogram-based k-means clusters in different color spaces. IEEE Trans Image Process 5(17):780–787

24. Nair V, Clark J (2004) An unsupervised, online learning framework for moving object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, vol II, pp 317–324

25. Orts-Escolano S, Garcia-Rodriguez J, Morell V, Cazorla M, Perez J, Garcia-Garcia A (2015) 3D surface reconstruction of noisy point clouds using growing neural gas: 3D object/scene reconstruction. Neural Process Lett 43:1–23

26. Papageorgiou C, Oren M, Poggio T (1998) A general framework for object detection. In: Proceedings of the IEEE international conference on computer vision, pp 555–562

27. Rêgo R, Araújo A, de Lima Neto F (2007) Growing self-organizing maps for surface reconstruction from unstructured point clouds. In: Proceedings of the international joint conference on artificial neural networks, IJCNN'07, pp 1900–1905

28. Rissanen J (1978) Modelling by shortest data description. Automatica 14:465–471

29. Rusu R, Cousins S (2011) 3D is here: point cloud library (PCL). In: Proceedings of the IEEE international conference on robotics and automation, ICRA, Shanghai, China, May 9–13, 2011

30. Sivic J, Everingham M, Zisserman A (2005) Person spotting: video shot retrieval for face sets. In: International conference on image and video retrieval, pp 226–236

31. Stergiopoulou E, Papamarkos N (2009) Hand gesture recognition using a neural network shape fitting technique. Eng Appl Artif Intell 22(8):1141–1158

32. Sui C (2011) Appearance-based hand gesture identification. Master of Engineering, University of New South Wales, Sydney

33. Uriarte EA, Martn FD (2005) Topology preservation in SOM. Int J Appl Math Comput Sci 1(1):19

34. Vamplew P, Adams A (1998) Recognition of sign language gestures using neural networks. Aust J Intell Inf Process Syst 5(2):94–102

35. Wong S, Ranganath S (2005) Automatic sign language analysis: a survey and the future beyond lexical meaning. In: IEEE transactions on pattern analysis and machine intelligence, pp 873–8912005

36. Yang J, Bang W, Choi E, Cho S, Oh J, Cho J, Kim S, Ki E, Kim D (2009) A 3D hand-drawn gesture input device using fuzzy ARTMAP-based recognizer. J Syst Cybern Inform 4(3):1–7