

Neural-Network-Based Models of 3-D Objects for Virtualized Reality: A Comparative Study

Ana-Maria Cretu, *Student Member, IEEE*, Emil M. Petriu, *Fellow, IEEE*, and Gilles G. Patry

Abstract—The paper presents a comprehensive analysis and comparison of the representational capabilities of three neural architectures for three-dimensional (3-D) object representation in terms of purpose, computational cost, complexity, conformance and convenience, ease of manipulation, and potential applications in the context of virtualized reality. Starting from a pointcloud that embeds the shape of the object to be modeled, a volumetric representation is obtained using a multilayer feedforward neural network (MLFFNN) or a surface representation using either the self-organizing map (SOM) or the neural gas network. The representation provided by the neural networks (NNs) is simple, compact, and accurate. The models can be easily transformed in size, position, and shape. Some potential applications of the presented architectures in the context of virtualized reality are for the modeling of set operations and object morphing, for the detection of object collision, and for object recognition, object motion estimation, and segmentation.

Index Terms—Feedforward neural networks, geometric modeling, neural network applications, unsupervised learning, virtual reality.

I. INTRODUCTION

A VIRTUALIZED reality environment differs from a virtual reality environment in the manner that the virtual world models are constructed. While virtual reality is typically constructed using simplistic computer-aided design (CAD) models and lacks fine details, virtualized reality starts from the real-world scene and virtualizes it [1], [2]. In other words, a virtualized environment is a conformal representation of the mirrored real world based on information about the real-world objects and phenomena as captured by a variety of sensors [3] that preserve the visible detail of the described real-world scene [2]. The virtual navigation of such an environment requires a complete three-dimensional (3-D) model of the scene [4]. One of the most challenging steps in the process of building a virtualized reality environment is that of modeling the real objects for incorporation into the virtualized world.

Depending on the complexity of the environment and on the application, achieving realism and geometrical correctness for the modeled objects can become a very difficult task. There are several aspects that have to be considered when building an object model, including the methods to define and acquire or create the data that describes the object, the purpose of the model, its computational cost, the complexity of the resulting

model, its conformance and convenience, and the ease of model manipulation. Often, information from various types of sensors has to be integrated and registered. The resulting object models are not continuous and often too large for real-time interaction. The effectiveness of these models depends strongly on the accuracy of data and on the number of samples they use, and their quality and degree of the approximation can be only determined by validation against experimental results [3].

Much research effort has been made in the area of 3-D object modeling over the last decades. The most common approaches are surface modeling, solid modeling, and procedural modeling. Neural networks (NNs) are a relatively new research trend in the context of object modeling. As a complement to classical modeling or as a standalone tool, they have been employed to create and reconstruct surfaces from scattered data, range data, or shape from shading [5]–[9], to recover, reconstruct, and match shape [10], [11], to represent and model objects [12], [13], to estimate motion of objects and classify them [6], [7], and for object segmentation [14], [15]. One of the solutions to model 3-D objects is to use a NN that has the 3-D coordinates of a point as inputs and learns a volumetric representation of an object. In Hwang and Tseng [6], Hwang and Li [7], and Tseng *et al.* [13], objects are represented by a “continuous distance transform NN” that has the structure of a standard feedforward NN. The network maps any 3-D coordinate into a value that corresponds to the distance between the point and the nearest surface point of the object. Two regularization terms are used to remove the unbounded erroneous surfaces and to form a constant slope profile. After training, a mismatch between the reference object and an unknown object can be easily computed. When encountered with deformed objects, this mismatch information can be backpropagated through the network to iteratively determine the deformation in terms of affine transformations [13]. The model can also be used to estimate motion of 3-D objects [6]. A similar modeling approach is used by Kumazawa [11] and Piperakis and Kumazawa [12]. A feedforward multilayer backpropagation network outputs 1 if the point is inside the object and 0 if outside. An object can be considered an “AND” of edge description nodes (planes or curved surface patches). These nodes are then connected finally by an “OR” operator to create the final object from patches. The representation is accurate and compact and affine transforms can easily be applied using these models. Other possible applications for the model are suggested but not evaluated.

This paper discusses two categories of neural-network (NN)-based 3-D object representation, namely 1) a multilayer feedforward neural network (MLFFNN) architecture and 2) two self-organizing architectures, namely a) self-organizing map

Manuscript received October 30, 2004; revised May 3, 2005.

The authors are with the School of Information Technology and Engineering, University of Ottawa, Ottawa, ON K1N 6N5, Canada (e-mail: acretu@site.uottawa.ca; petriu@site.uottawa.ca).

Digital Object Identifier 10.1109/TIM.2005.860862

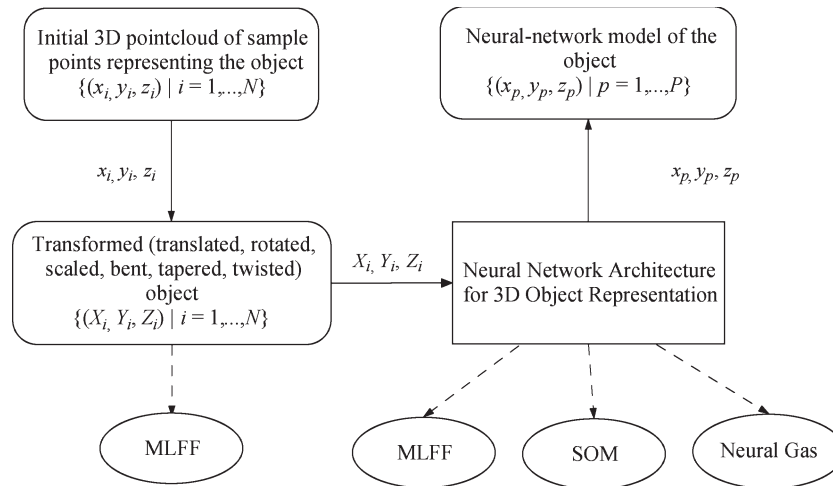


Fig. 1. Modeling framework.

(SOM) and b) neural gas network, as representatives of the two main learning categories—supervised and unsupervised learning. While the proposed MLFFNN model follows the ideas in [6], [7], and [11]–[13] and can be considered an extension to the 3-D case of the model proposed by Piperakis and Kumazawa [12], the contribution of this paper is to provide a comprehensive analysis and comparison of representational capabilities of this NN and of the self-organizing architectures in the context of 3-D object modeling: an aspect that has not been extensively investigated in the literature. The emphasis is on particular NN architectures, their characteristics (purpose, computational cost, complexity, conformance and convenience, and ease of manipulation) and potential applications for virtualized reality. This paper is an extension of our previous work in the context of NN-based models for 3-D objects [16], [17].

II. NEURAL ARCHITECTURES FOR 3-D OBJECT REPRESENTATION

The architectures used in this paper consist of two cascaded modules (depicted in Fig. 1), namely 1) a transformation module and 2) a representation module. The representation module is either an MLFFNN or one of two self-organizing architectures, i.e., SOM and neural gas network.

Starting from a pointcloud that embeds the shape of the object to be modeled, a volumetric representation is obtained using the multilayer feedforward network. The surface of an object is described by a set of zeros in the output response of the network, while the interior or exterior of an object is mapped to a negative or positive value, respectively. The representation is simple, compact, and accurate (as long as the user is willing to pay the computational time for the training phase). It can offer theoretically infinite resolution, and it saves storage space. The representation is continuous and permits an extensive study not only of the modeled object but also of the entire object space. The proposed model solves the problem of large memory usage and rendering times of polygonal models, the problem of the large amount of storage of octrees, the incapability to specify a point on a surface of the implicit surfaces or the lack of representation of the object interior in the case of constructive

solid geometry (CSG) models, and the problems in modeling minute details of splines. It can be used to perform simple operations such as object morphing, set operations, and object-collision detection [18].

On the other hand, the representation offered by the two self-organized architectures has the characteristics of a surface model. The models are obtained faster and can achieve higher accuracy than the multilayer feedforward counterpart. Both can be used for object segmentation in complex scenes.

The representation module can be cascaded with a transformation module, modeled as an MLFFNN, whose role is to ensure an easy manipulation of the represented objects in size, position (it models affine transformations), and shape (it also models deformations). Moreover, the module can be used to learn transformation/deformation information, when it is provided with a reference object and a transformed/deformed object. This makes it useful in motion and alignment estimation. When cascaded with the transformation module, the multilayer feedforward architecture can be used for object recognition. In all of the three cases, the representation module cascaded with the transformation module can be used for motion estimation.

A. Transformation Module

The transformation module implements transformations (rotation, translation, and scaling) and deformations (bending, tapering, and twisting). It receives as inputs the 3-D coordinates (x, y, z) of a point and their combinations and returns the 3-D coordinates of the corresponding transformed point. As a novelty, the introduced NN deformation module allows not only for generation but also for the recuperation of deformation parameters of deformed 3-D objects represented by NNs, as it will be further shown. The network has no hidden layer and the three outputs have linear activation functions, as depicted in Fig. 2. The scaling and rotation will be encoded in the weight matrix, which will implement the generalized matrix according to the translation–roll–pitch–yaw (TRPY) convention used in robotics.

The same idea is exploited to model the operations of tapering, bending, and twisting. Knowing the equations, the

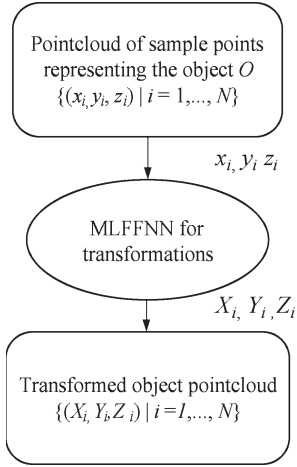


Fig. 2. Transformation framework.

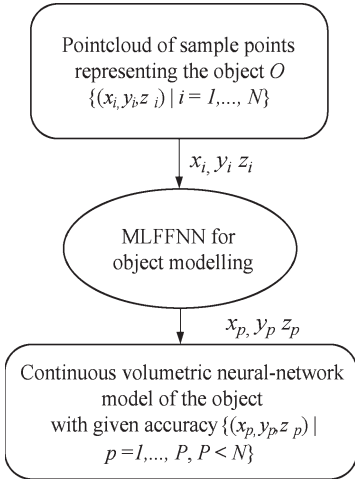


Fig. 3. MLFFNN framework.

network weights are set such that the desired deformation is performed. When provided with corresponding points from a given reference model as inputs and those of a transformed model as targets, the transformation module can learn and store in its weights information on how these points have been transformed and/or deformed. This information can be used for object classification and motion estimation.

B. Multilayer Feedforward Representation Module

The purpose of the representation module is to build a volumetric description of a 3-D object. One network is used for each represented object. The network has as inputs the 3-D coordinates x , y , and z of a point and their combinations (x^2 , y^2 , z^2 , xy , yz , and xz) and outputs a value proportional to the distance between the point given as input and the modeled object surface. The surface of an object is, thus, described by a set of zeros in the output response of a network, while the interior or exterior of an object are mapped to a negative or positive value, respectively. Therefore, the NN model is, in fact, an implicit function for continuous volumetric representations of objects, as shown in Fig. 3.

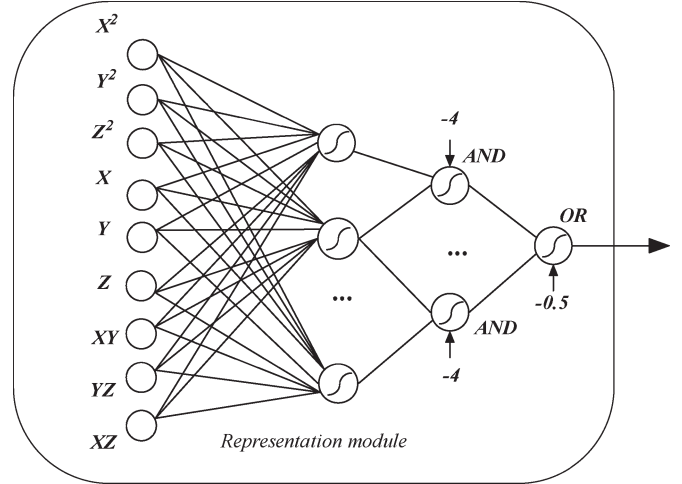


Fig. 4. MLFFNN representation module with activation functions.

A three-layered architecture is used to model a 3-D object, as depicted in Fig. 4. The input layer has the sole purpose of transmitting input patterns. In the first hidden layer, each neuron describes the shape of an object patch (plane or curved surface) that approximates the object surface. Using the sigmoid activation function, for a sufficiently large value of σ , the regions outside or inside of a patch can be represented [11], [18]. Each sigmoid in the second hidden layer combines the patches represented in the first layer with an ‘‘AND’’ operator. These neurons will share the space in regions inside or outside given patches.

The ‘‘AND’’ operator is implemented in such a way that it can describe plane and curved surfaces, either concave or convex. A minimum of six patches is considered, since this is the minimum number of patches that can describe a 3-D volume of whatever complexity (concave or convex). To ensure that the ‘‘AND’’ operation is performed correctly using a sigmoid, a sufficiently large value for σ is needed ($\sigma = 5$ is used during training) to create a quite steep increase, and a bias of -4 is used to direct the output into the correct value. The outputs are then connected with an ‘‘OR’’ operator to obtain the final shape of the object. Again, to ensure that the ‘‘OR’’ operation is performed correctly by a sigmoidal function, a bias of -0.5 is used. This representation is, in fact, an analogy with classical modeling approach with patches on one side and with the neural trees [19], [20] and trees of cascaded sigmoid functions [11] on the other side.

The representation module can be used in ‘‘generation’’ mode to build simple objects whose equations are known [12], [18]. The weights and biases are set such that the output neuron computes the equation that represents the desired object. For example, in order to model an ellipsoid whose equation is given as

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} - 1 = 0 \quad (1)$$

the x^2 , y^2 , and z^2 inputs of the representation module will be used, with weights $1/a^2$, $1/b^2$, and $-1/c^2$ and bias -1 . When tested for a given set of points in 3-D space, such a network

will reply with a positive value if the given point is outside the ellipsoid and with a negative value if the point is contained in the ellipsoid. The use of this modeling is mainly to provide for means of building simple objects without the need of having a 3-D pointcloud model, therefore, without scanning them.

The same architecture can be used in “training” mode to learn volumetric representations of objects given in form of pointclouds. Such data can be obtained from image acquisition systems (range data in the context of this paper) or using 3-D modeling tools. In this case, the user has to decide upon which inputs to use, the number of neurons in the hidden layer, and the values for the training parameters. Intuitively, an object with only flat surfaces will use x , y , and z inputs and one with curved surface will use x^2 , y^2 , and z^2 inputs and the combinations of x , y , and z . When no *a priori* knowledge is available on the shape of the object, at least x , y , and z and x^2 , y^2 , and z^2 should be used in order to ensure that an object with variable complexity can be represented.

As using only the points of the given object for training has proven inappropriate, in order to avoid the trivial solution—where the output should be zero for all points on the surface of the object—extra surfaces are added in the interior and exterior of the object at an equal given distance from the object surface [9], [13]. All the interior points will be assigned a negative value proportional to the distance and all the exterior ones a positive value. Thus, the entire object space is modeled.

All the points of a given object are normalized in the $[-1 \ 1]$ interval. The network has tangent sigmoid activation functions in all the neurons of the hidden layer and in the output layer. The module is trained in a supervised manner, using scaled conjugate gradient backpropagation. Testing is performed by evaluating it for points in the 3-D object space $[-1 \ 1 \ -1 \ 1 \ -1 \ 1]$. To visually assess the results, only those points that belong to the interior of the object (with values less than or equal to 0.1—considering a 0.1 tolerance) are plotted. In order to estimate the generalization error, cross-validation is used.

C. Self-Organizing Representation Modules

As opposed to the solution presented in the previous section, which was based on supervised learning, both architectures proposed here are unsupervised ones.

The Kohonen SOM [21] is one of the most important unsupervised network architectures that uses soft competition and at the same time belongs to the vector quantization methods. As in adaptive vector quantization methods, the input space is clustered by assigning each neuron to a defined region in the input space. The number of inputs to each external neuron being equal to the dimension of the input space, the weight vectors can be interpreted as locations in this space. Each region is a set of locations that are closer according to a distance measure to the corresponding neuron than any other one. After the learning procedure, two vectors belonging to the same cluster will be projected on two close neurons in the output space. Usually, the defined topology of an SOM is a two-dimensional (2-D) grid.

The low-dimensional mapping topology of SOM is removed in the neural gas network [22]. The neural gas network consists

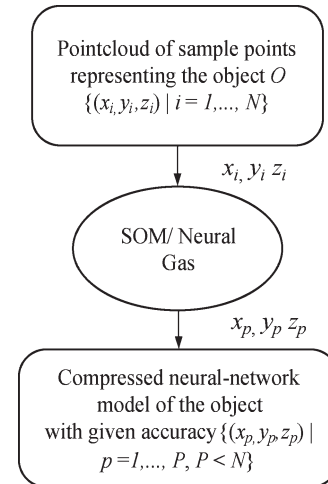


Fig. 5. Self-organizing framework.

of nodes, which independently move over the data space while learning. The name emphasizes the similarity to the movement of gas molecules in a closed container [23]. Neural gas is a vector quantizer that converges quickly, reaches lower distortion error after convergence than other methods, and obeys a gradient descent on an energy function [22], which is not the case of the Kohonen map [21]. It presents the identical vector quantization property of Kohonen’s algorithm but does not hold the topology preservation property.

In the context of this paper, these architectures are employed to obtain a compressed model for the 3-D pointcloud that represents an object, as shown in Fig. 5. The weight vector will consist of the 3-D coordinates of the object’s points. During the learning procedure, the model will contract asymptotically towards the points in the input space, respecting their density and, thus, taking the shape of the object encoded in the pointcloud.

Data is normalized such that it has a mean of 0 and a variance of 1. The weights are initialized linearly along the two principal eigenvectors of the data set, and a hexagonal lattice is used. The networks are trained using the batch version of the training algorithm. The considered neighborhood function is a Gaussian for the SOM.

III. COMPARISON OF RESULTS

We built a Matlab-based software platform to test the cascaded architecture composed of the transformation MLFFNN-based module and the three NN-based architectures (MLFFNN, SOM, and neural gas) of the representation module. A set of real range data and another set of synthetic data were used for experiments. We compared the results in terms of the methods to define the objects, purpose, cost and complexity, performance and convenience, and ease of model manipulation. The specific procedure is explained for each possible application and the benefits and problems are discussed.

A. Methods to Define the Objects

From the point of view of the methods used to define the objects, all resulting models (representations of 3-D models)

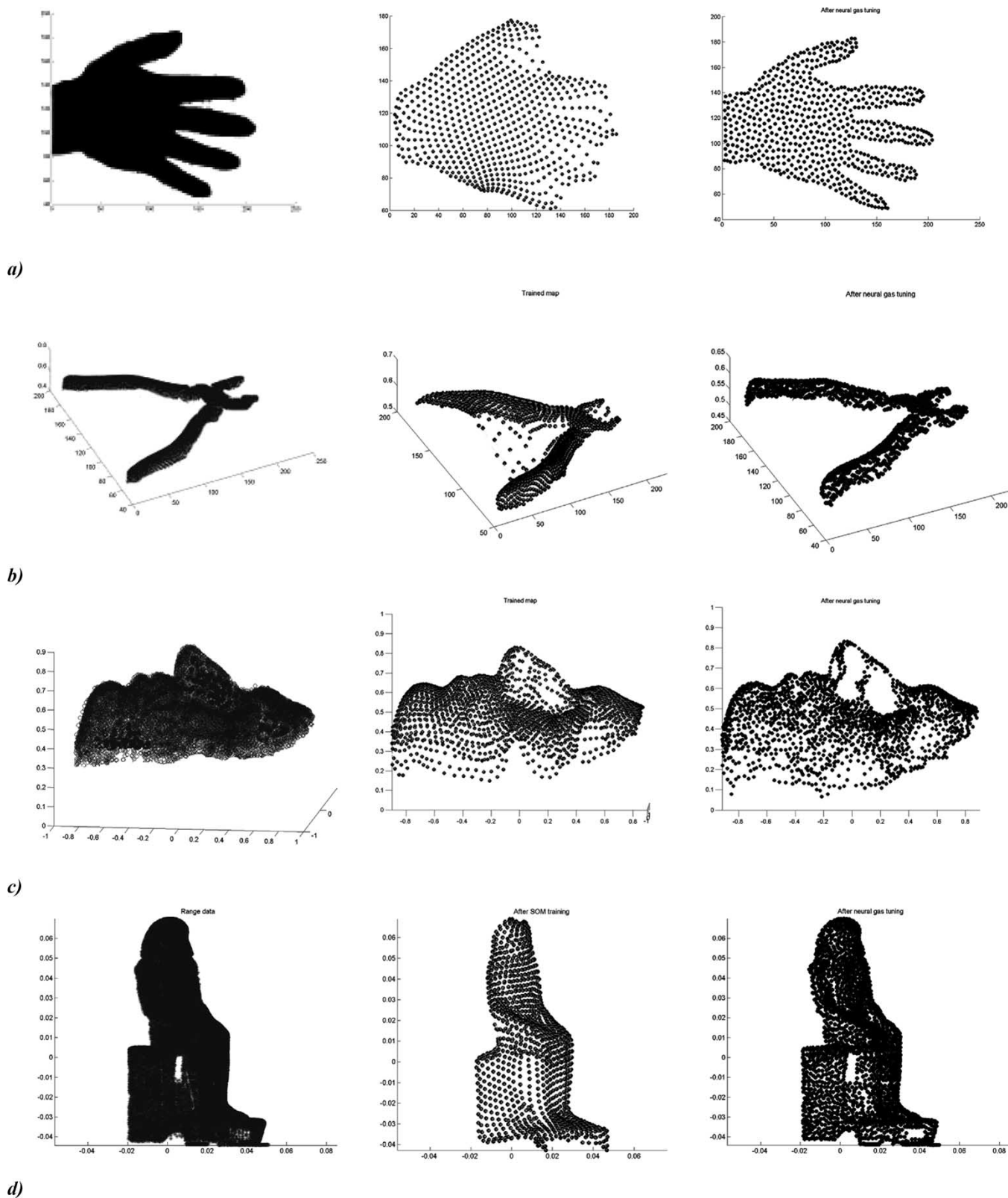


Fig. 6. MLFFNN, SOM, and neural gas comparison in training time and visual quality. (a) Hand model—19 280 points: MLFFNN 2.8 h—column 1; SOM 18.2 min—column 2; neural gas 28 min—column 3. (b) Pliers model—7440 points: MLFFNN 1 h; SOM 3 min; neural gas 9 min. (c) Face model—19 080 points: MLFFNN—3.3 h; SOM 26 min; neural gas 42 min. (d) Statue model—51 096 points: MLFFNN 5.2 h; SOM 30 min; neural gas 3.5 h.

are defined by a matrix of weights. The initial pointcloud of range data or synthetic data is first normalized and supplied to the inputs of the neural architecture. After the learning

procedure, the weights of the NN embed the representation of the object. To allow the reconstruction of an object defined by an MLFFNN, this representation has to be accompanied by

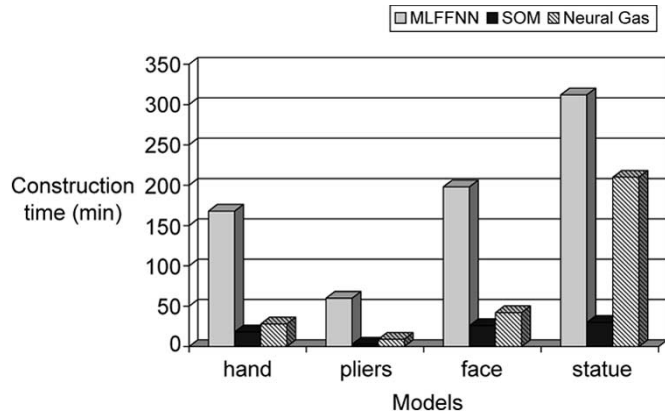


Fig. 7. Comparison of construction time.

the network architecture (two digits, representing the number of neurons in each hidden layer). For the case of SOM and neural gas, the object representation is inherent in the matrix of weights, which, in this case, contains the coordinates of points in space where the neurons that represent the objects are located.

B. Purpose of the Object Models

Regarding the purpose of the object models, there are two main reasons for building 3-D objects, namely 1) “image making” and 2) simulation [24]. While “image making” requires that the model looks good (is realistic, sufficiently complex, and conforms to the geometry of the model), simulation requires accurate models. While the first one is described as interactive and approximate, the second one is generally slow but precise.

As shown in Fig. 6, the MLFFNN models can be used for “image making,” since they look good, are sufficiently complex, and conform to the geometry of the modeled object. They also reveal to be appropriate for simulation, since they can be accurate if the user is willing to pay the computational cost. SOM models are more suitable for simulation, as they present good topology preservation and are accurate even in the presence of noise, but are not suitable for “image making,” since they suffer from the boundary problem (the boundary of the NN does not coincide with the real boundary of the object), as depicted in Fig. 6(a) and (b) (the middle column).

The neural gas models are, again, more suitable to simulation, since they are very accurate (as long as the training data is not too noisy), but a supplementary technique like point-based rendering is needed to make them look better and avoid the point-like aspect of the modeled objects.

C. Computational Cost and Complexity

With regards to computational cost matters, there are three aspects to be considered, namely 1) the construction (training) time; 2) the display cost (generation time); and 3) the storage space. First, considering the training time, the cheapest in terms of computational time is the SOM followed by the neural gas network, as seen in Fig. 7. The MLFFNN is the most expensive. The relatively long training time is not such an

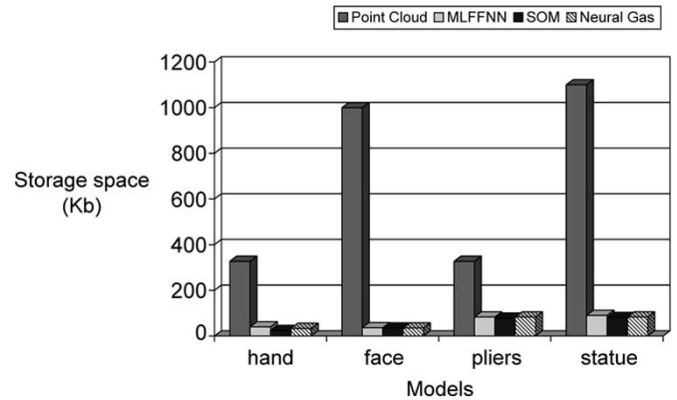


Fig. 8. Comparison of required storage space.

important disadvantage, as the training procedure is performed only once for each object. The resulting average processing time per model point is lower than that reported in the only other available reference [12] discussing MLFFNN modeling. The 2250-point hand model reported in [12] has requested 3 h and 21min for training, while it took only 2.8 h to train our 19 080-point hand model.

Besides the training time, the MLFFNN also needs a quite long generation time (up to 10 min), unlike the two self-organized architectures. The generation time is the time required for the network to build the 3-D model embedded in its weights. The network is evaluated for points in the $[-1 \ 1 \ -1 \ 1 \ -1 \ 1]$ range and all the points inside the object displayed, while for the other two architectures, the generation time is equivalent to the display time.

As for the storage space aspect, all three models reduce significantly the space needed to store the representation of the model in comparison with the initial pointcloud. For some models, the space required to store the MLFFNN representation is a bit larger than the one required by the self-organizing architectures, but, generally, there is no significant difference between the compared neural architectures, as depicted in Fig. 8.

In terms of complexity, the neural representations provided by the proposed neural architectures are simple and compact. They can be seen as a matrix or string that contains the values of the weights and the neural architecture in case of the MLFFNN.

D. Conformance and Convenience

Conformance measures the degree to which the model actually represents the desired objects and provides a realistic geometrically correct representation for the modeled object [3]. Due to its volumetric properties and to its property of theoretically infinite resolution (only limited by the number of points used to test the model), from the conformance viewpoint, the MLFFNN deserves the first position. Although it does not have the same accuracy as the neural gas network, the objects do not present the point-like aspect. However, when supplemented by point-based rendering, the neural gas takes the leading position. On the other hand, neither the neural gas nor the

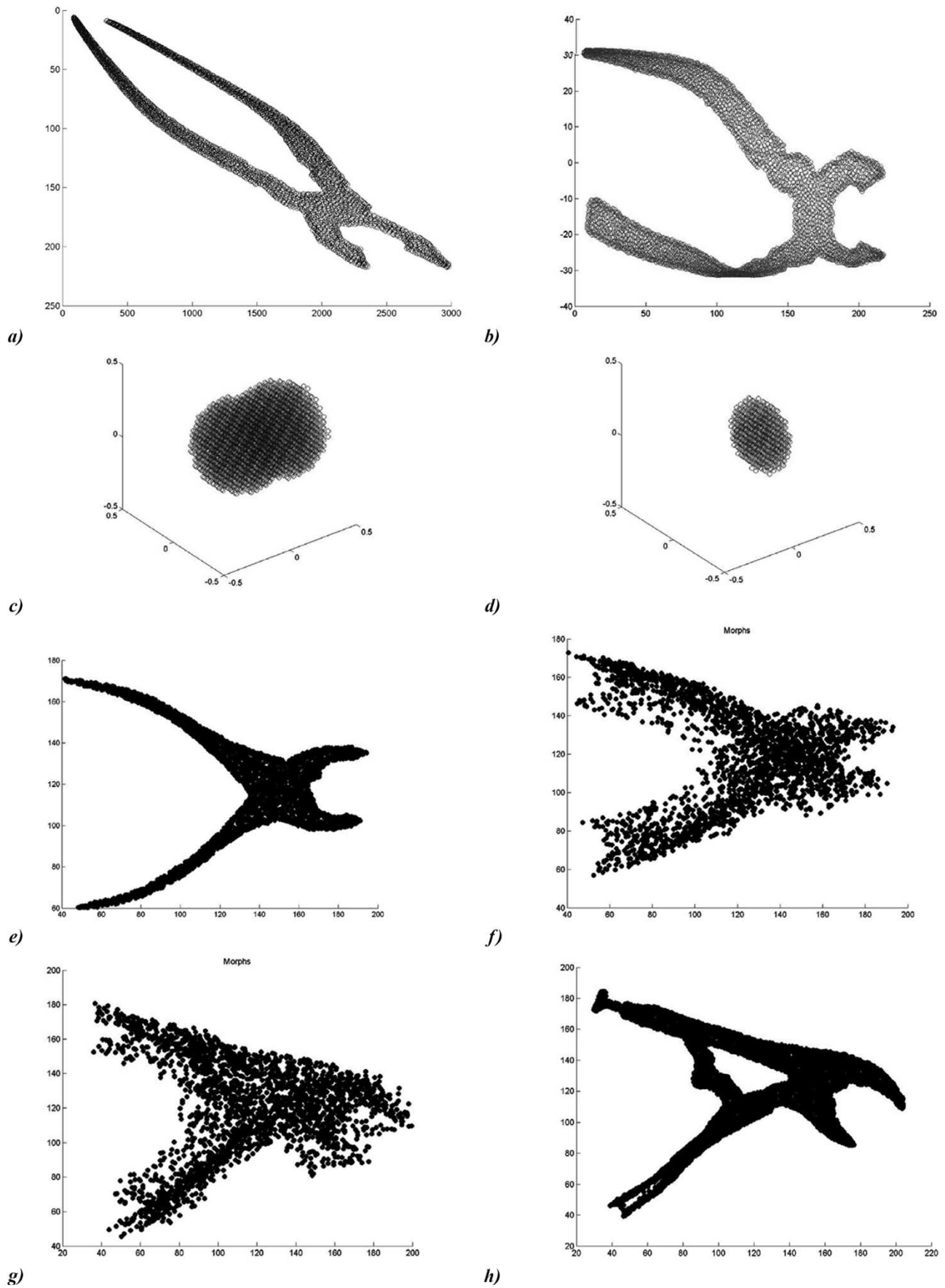


Fig. 9. (a) and (b) Tapered and bend pliers model. (c) and (d) Union and intersection of two spheres of equal radius 0.3, one centered at (0, 0, 0) and one centered at (0.3, 0, 0). (f) and (g) Intermediate morphing steps of the pliers in (e) to the pliers in (h).

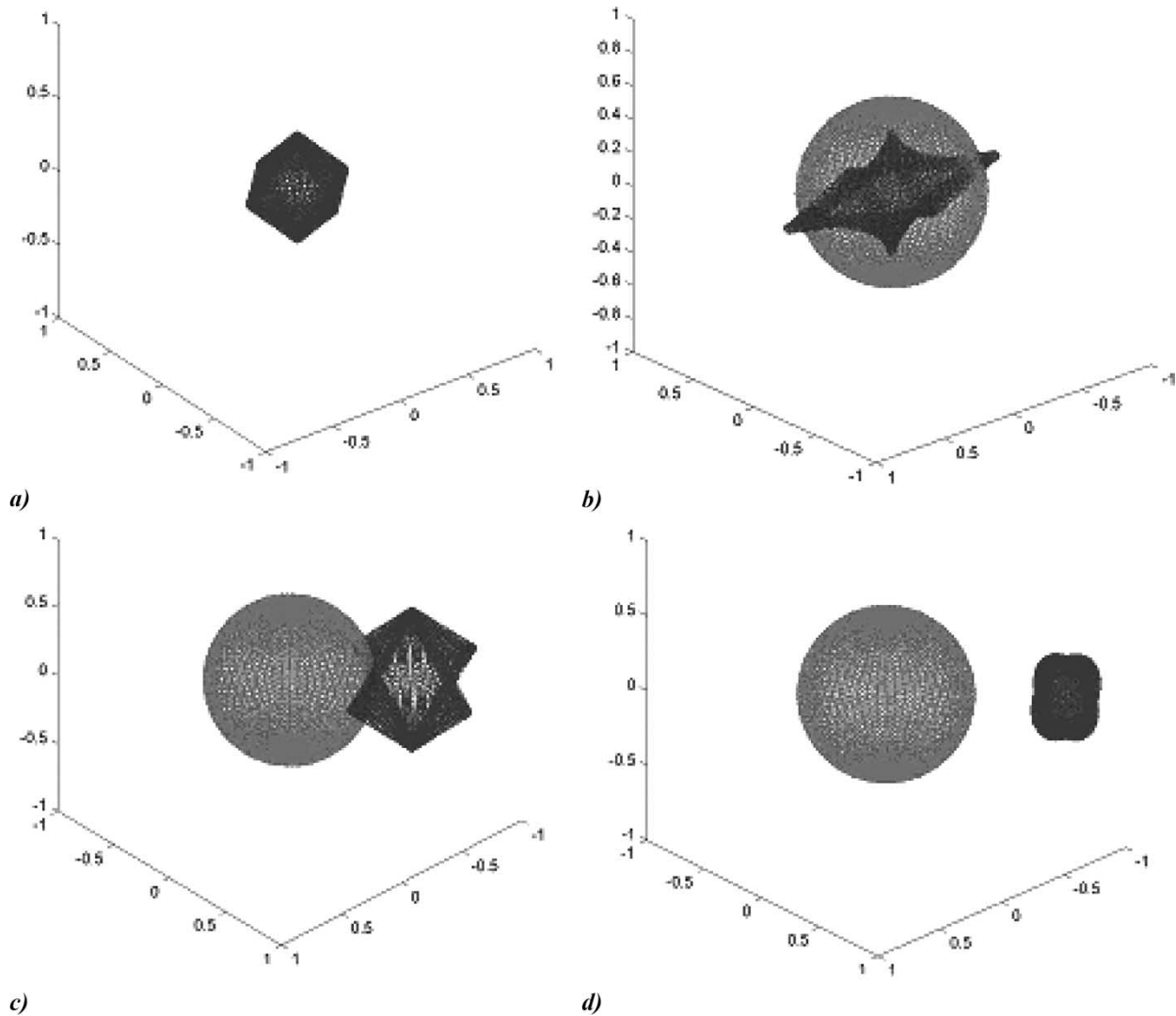


Fig. 10. Object collision detection testing cases.

SOM models offer the property of infinite resolution. In spite of its topological preservation capability, the SOM gets the last position. It suffers from the boundary problem, and the modeled objects do not look good. All these aspects are revealed by the results in Fig. 6 (column 2).

Regarding convenience, neural gas and SOM do not permit an extensive study on the modeled objects. They build a sparse model of an object and do not offer means to test if a point is inside or outside the model. In contrast, the MLFFNN representation is continuous and due to the volumetric and implicit properties, an MLFFNN permits an extensive study not only of the modeled object, but also of the entire object space.

E. Ease of Model Manipulation

As our experiments prove, cascaded with the transformation module, all models can be easily transformed in size, position, and shape. Affine transformation, deformations, and morphing can be easily performed regardless of the representation, as seen in the examples in Fig. 9. The capability of building new objects by means of set operations [Fig. 9(c) and (d)] is restricted to the

MLFFNN models due to their implicit volumetric representation capability. Using the property that, given the coordinates of a 3-D point, the trained model of each object replies with a negative value if the point is inside the object or with a positive value if the point is outside, a point will belong to the union of two objects if it is inside the first object or inside the second object. In a similar manner, a point will belong to the intersection if it is inside the first object and inside the second object. A point will belong to the difference of two objects if it is inside the first object and outside the second (or outside the first and inside the second). Points from the $[-1 \ 1 \ -1 \ 1 \ -1 \ 1]$ object space are evaluated with the two NNs corresponding to the two objects, and an “AND” or “OR” operation is performed between the outputs to determine if the points belong or not to the union/difference/intersection of the two objects. To assess visually the models resulting from set operations, only those points that belong to the desired operation are plotted. It takes approximately 15 min on our software platform to build the model of a set operation. The modeling of set operations does not require any conditions for the two objects, as long as their NN representation exists.

TABLE I
COLLISION DETECTION IN TEST SCENES (PERCENTAGE OF POINTS DETECTED INSIDE THE REFERENCE MODEL)

	<i>No noise</i>	<i>Random noise level between 0 – 0.04</i>	<i>Random noise level between 0 – 0.05</i>	<i>Random noise level between 0 – 0.1</i>
<i>An object inside the reference model (a)</i>	100%	99.5%	95%	58%
<i>An object almost contained in the reference model (b)</i>	96.56%	92.4%	88.7%	50%
<i>An object barely touching the reference model (c)</i>	2.3%	1.8%	0.15%	2%
<i>An object not touching the reference model (d)</i>	0%	0%	0%	0%

In order to perform object morphing [Fig. 9(e)–(h)], two assumptions have to be satisfied. First, the reference and target objects must be aligned (if not, the transformation module can be used to recuperate the displacement information). Second, a 3-D neural model must exist for each of the objects. The objects will be represented using the same architecture (to ensure the weight matrix has the same size). Given these, the morphing is performed by modifying the weights of the reference model to match the weights of the target model with a linear interpolation. As the weights for each morphed object are known, the morphed object models are visually built using the representation module. Points in the $[-1 \ 1 \ -1 \ 1 \ -1 \ 1]$ cube are used as inputs, and only those whose output is smaller than 0 (meaning the points are inside the object) are plotted.

F. Applications

The inherent volumetric and implicit representation provided by the MLFFNN makes it also easy to be used for the detection of object collisions in virtualized reality scenes. As the 3-D neural representation is an implicit function, in order to detect a collision, it is sufficient to sample a set of points from the tested object surface and substitute them in the NN model of the reference object. If a negative value is returned, it means that the tested points are inside the reference object, and, thus, a collision occurred between the two objects. The tests are run with/without influence of noise for four scenes, for different situations of collision of a given object with a reference spherical object, as depicted in Fig. 10.

In most cases, the collision can be detected even in presence of random noise. Problems appear in the case of barely touching objects, as shown in Table I. Because of the random error, the user is no longer able to decide if the objects are collided or not [case of noise 0–0.05, scene from Fig. 10(c)].

The number of sampled points also has an influence on the recognition rate. The more sample points used, the biggest the chance is that the collisions are detected correctly. However, this comes at the cost of computational time. For too many sample points, the detection can no longer be done in real time, as shown in Table II.

Cascaded with the transformation module, the MLFFNN module can also be used for object motion estimation [10], [11], [18]. The motion of a 3-D object is described in terms

TABLE II
INFLUENCE OF THE NUMBER OF SAMPLE POINTS OVER COLLISION DETECTION TIME

<i>Number of sample points</i>	<i>Detection time (s)</i>
18	0.79
60	0.99
1200	1.8
1600	2.3
5000	21
10000	124

of rotation, translation, and scaling, as given in the TRPY generalized motion matrix. The reference object is assumed centered in the origin, and an initial guess is made for all the implied parameters: The translation and rotation parameters are initially set to 0, and the scaling factor to 1. The motion parameters are computed using the transformation module at each time step. As the motion and the measurements are considered continuous, the next translation parameter is always initialized with the position of the object's centroid in the previous step. The rotation angles are set also to their previous value (as it is unlikely that abrupt changes occurred, since the last movement step), and the scaling factor to 1. The procedure is repeated until there are no more movements in the scene. It is considered that no more movements have been made when, for four consecutive steps, there has been no change in any of the parameters. Two test scenarios are used, as shown in Fig. 11(a) and (c). Tests are run for 60 time steps over which the spherical object and the square object move from one end to the other end of the trajectory for different levels of noise.

The results show that the reconstruction of the trajectory is almost perfect for a reasonable noise level (0–0.2) but becomes very difficult for high levels of noise (0–1). However, even in this case, the reconstructed trajectory gives a hint on the shape of the real trajectory [Fig. 11(b)]. The prediction of rotation angles [as shown in Fig. 11(d), representing, for example, the real and estimated rotations around ψ] and of the scaling factors (as depicted in Table III, which shows that even in the presence of noise, the returned value for the scaling factor is closed to the value 0.1, the real value of the scaling factor parameter) is also correct, even in the presence of noise.

In order to perform recognition, an NN representation of all the objects must exist and must be stored in a database

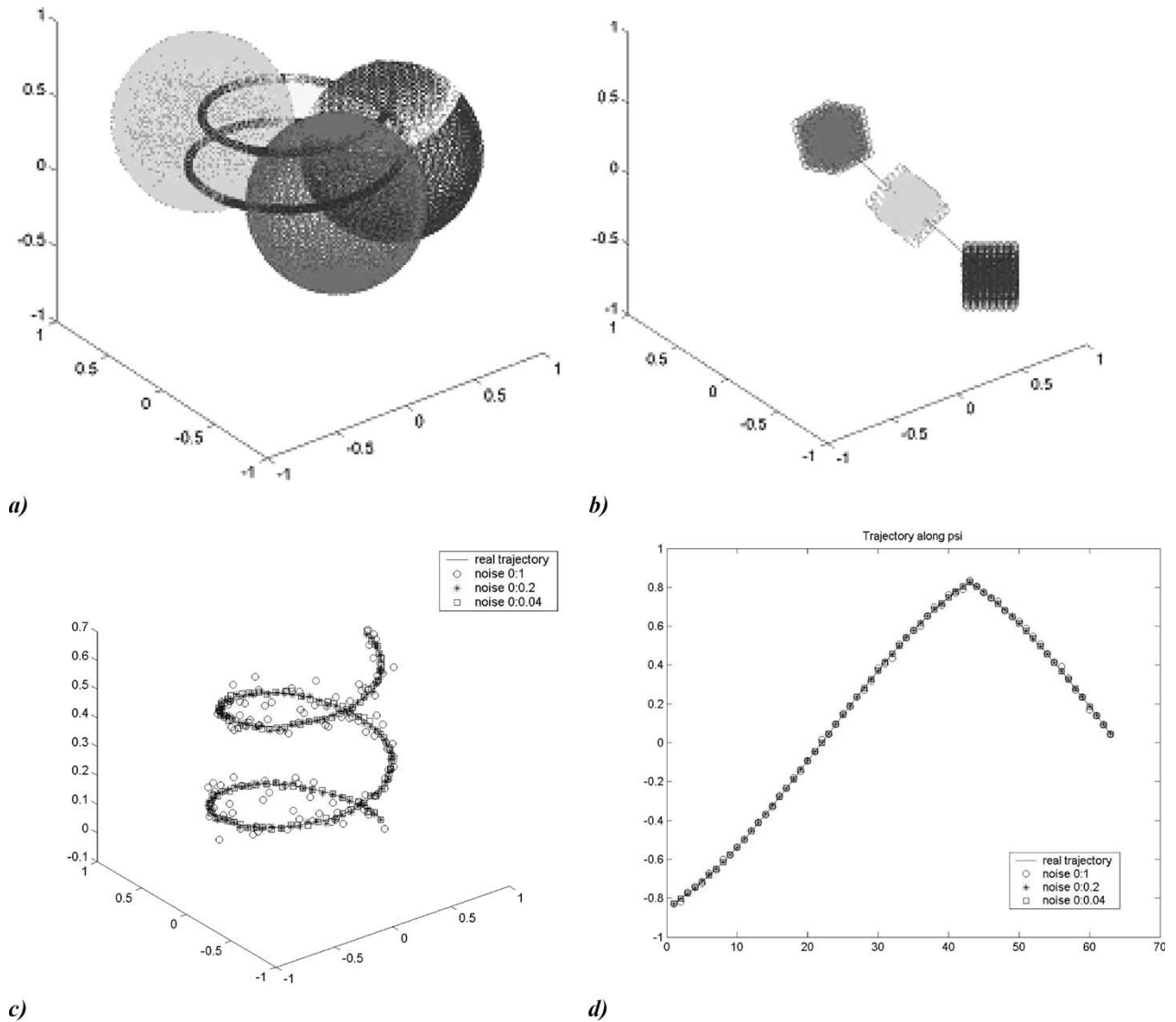


Fig. 11. Framework for translation, scaling, and rotation estimation. (a) Spherical object (b) moving along the real trajectory and (c) the estimated trajectory with/without the presence of noise; (d) square object moving along the trajectory and (e) the estimated rotation angle around ψ .

TABLE III
ESTIMATION OF SCALING FACTOR

	Scaling along x	Scaling along y	Scaling along z
<i>Real value</i>	0.1	0.1	0.1
<i>Noise 0-1</i>	0.18	0.2	0.18
<i>Noise 0-0.2</i>	0.12	0.12	0.115
<i>Noise 0-0.04</i>	0.11	0.1	0.1

of reference models. The model-based matching is done by first aligning the transformed object with the reference model in the database using the transformation module. The NN of the transformation module is trained using the reference model points as training points and the points of the transformed model as targets. In this way, the transformation information is stored in the weights of the network. After the alignment procedure, a set of points is sampled from the transformed object, and the points are tested whether they belong to the reference model or not.

Four scenes (Fig. 12) containing synthetic data of four (nonnoisy) objects are used during tests. The representations of these objects in the database are built using inputs x , y , and z and x^2 , y^2 , and z^2 , with two extra surfaces situated at 0.05 from the objects' surfaces, and using the same number of neurons in the first and second hidden layers. The objects are transformed randomly (translated, rotated, scaled, bent, or tapered in the given example). Tests are run for different levels of noise and different numbers of samples. Table IV shows the average percentage of points recognized in each scene as belonging to one of the object models for the case in which a random noise with a magnitude between 0 and 0.04 is added to the sampled points.

The percentage of recognition is high in the scenes where objects are not collided and objects are not bent or tapered. The recognition is harder for objects with sharp edges in those cases in which they are rotated (the star in scene 2, for example). However, only 60% of the points are recognized for a random noise level between 0 and 0.1, and the network

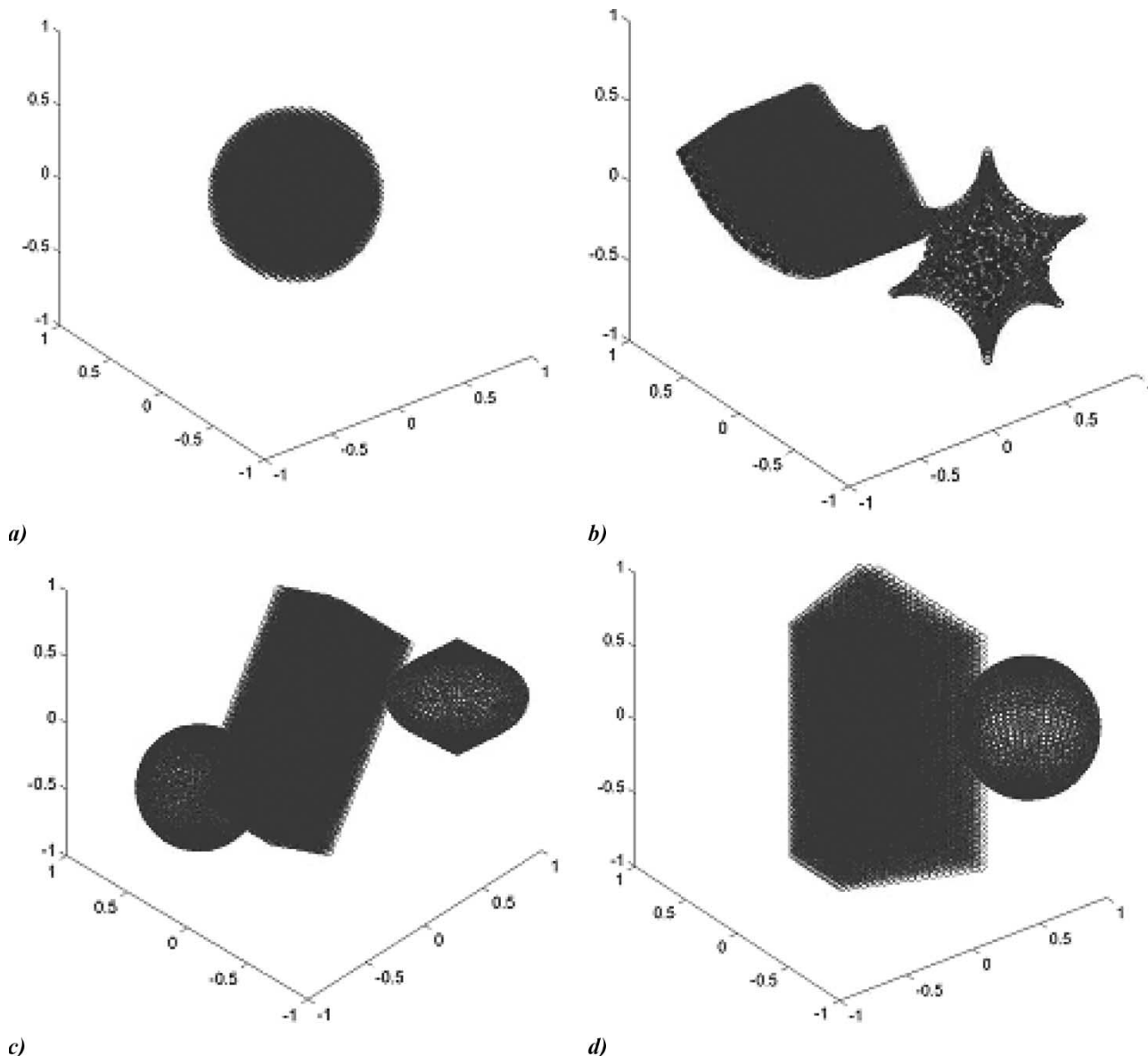


Fig. 12. Test scenes for object recognition.

TABLE IV
PERCENTAGE OF RECOGNIZED POINTS BELONGING TO MODELS IN THE DATABASE FOR EACH SCENARIO (CASE OF A NOISE LEVEL 0–0.04)

Recognized points	Object 1	Object 2	Object 3	Object 4
Scene 1	2.4%	4.9%	1.6%	99.1%
Scene 2	93.3%	3.3%	91.7%	3.1%
Scene 3	95%	99.1%	5.78%	98.3%
Scene 4	91.7%	6.6%	1.6%	92.5%

is no longer able to recognize an object corrupted with noise between 0 and 0.2. In case of a complex scene, with many objects, the segmentation of the scene is required prior to the recognition process. A higher number of misclassified points occurs in test scene 4, where the objects are collided. This is also due to the fact that the *k*-means clustering algorithm is used to divide the scene in the component objects. The

k-means clustering algorithm is an algorithm for partitioning (or clustering) data points into disjoint subsets such that within-cluster sum of squares is minimized. Therefore, clustering may be an indicator of similarity of regions, and may be used for segmentation purposes. For the testing scenes where objects are not collided, between 0.5% and 5% of the points are misclassified due to errors in the clustering algorithm and more in test scene 4.

For comparison with other published work, the average rate that we obtained for object classification and object motion is slightly above the one reported by Hwang and Tseng [6], Hwang and Li [7], and Tseng *et al.* [13], which used different NN architectures and scenarios to test the performance.

The self-organized architectures do not have volumetric properties; thus, their use is restricted to the modeling of morphing operation. Moreover, the representation is not implicit, and, therefore, there is no way to test if a point belongs to an object or not, offering no means to detect a match between two

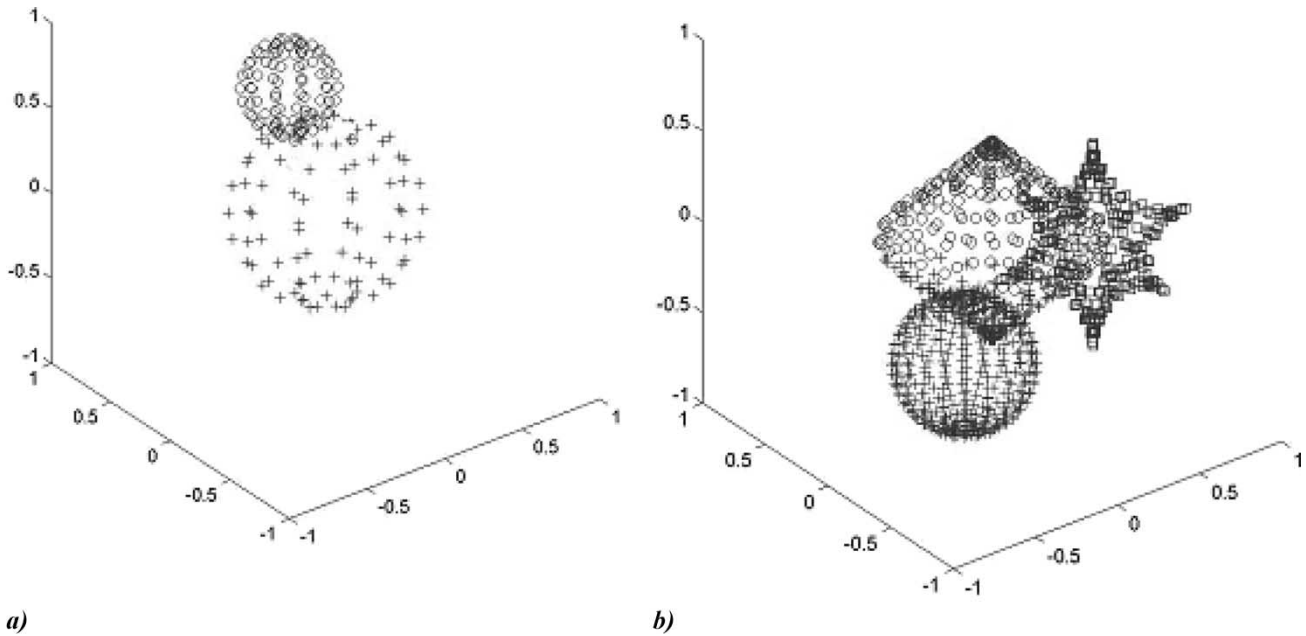


Fig. 13. Test scenes for segmentation.

TABLE V
SEGMENTATION RESULTS

Scene	Misclassified points (%)		
	<i>k</i> -means	<i>k</i> -means + neural gas	<i>k</i> -means + SOM
Scene 1	1	0.56	0.55
Scene 2	3.5	1.7	2.3

objects. This implies that they cannot be used for recognition either. They can be used, however, for motion estimation and also for object segmentation in complex scenes [18].

The motion estimation is done in the same manner as for the MLFFNN-based representation. Two segmentation scenarios (Fig. 13) are tested with just *k*-means clustering and with *k*-means clustering on the models obtained using the SOM and neural gas representations to elicit the improvements given by the neural representation in the clustering procedure. The first scenario contains synthetic data of two barely touching spheres. The second one contains three objects situated very close to each other. The results are presented in Table V. In both cases, the *k*-means clustering preceded by neural gas or SOM gives better results than in the case in which only *k*-means is applied alone. To our knowledge, no results are reported in the literature for the use of self-organizing architectures and their applications for 3-D object modeling.

IV. CONCLUSION

The paper presented a comprehensive analysis and comparison of the representational capabilities of three neural architectures in the context of 3-D modeling, taking into account the computational cost, complexity, conformance, convenience, and ease of manipulation. A study of some possible applications of the neural architectures in the virtualized reality framework is also presented.

All things considered, it can be concluded that, because of the boundary problem, the SOM models are to be avoided for nonnoisy data. Depending on the application requirements, one should choose between the MLFFNN and the neural gas network representation supplemented, for example, with a point-based rendering technique. While providing a quite accurate model, with theoretically infinite resolution and with facilities to perform volumetric and implicit operations, the MLFFNN representation implies a computationally expensive training phase and also some time investment for the generation of the modeled object. Neural gas offers very good accuracy and a reduced computation cost. However, its range of use in applications is reduced to segmentation, morphing, and motion estimation.

Further developments of the work will examine some more alternative NN architectures that could improve the learning time and ensure a real-time behavior for at least the model generation. Even though it possesses a high topology preservation and very good accuracy, the SOM network's modeling capability is limited in 3-D object modeling due to its boundary problem. Thus, a possible improvement could be the use of dedicated algorithms to treat the boundary distinctly from the entire object surface and, thus, avoid the holes created in the modeled objects' surfaces.

REFERENCES

- [1] T. Kanade, P. J. Narayanan, and P. W. Rander, "Virtualized reality: Concepts and early results," in *Proc. IEEE Workshop Representation Visual Scenes*, Cambridge, MA, Jun. 1995, pp. 69–76.
- [2] T. Kanade, P. Rander, and P. J. Narayanan, "Virtualized reality: Constructing virtual worlds from real scenes," in *Proc. IEEE Visualization*, Phoenix, AZ, Jan. 1997, pp. 277–283.
- [3] E. M. Petriu, "Neural networks for measurement and instrumentation in virtual environments," in *Neural Networks for Instrumentation, Measurement and Related Industrial Applications*, vol. 185, S. Ablameyko, L. Goras, M. Gori, and V. Piuri, Eds. Amsterdam, The Netherlands: IOS, 2003, pp. 273–290.

- [4] P. Boulanger, J. Taylor, S. El-Hakim, and M. Rioux, "How to virtualize reality: An application to the re-creation of world heritage sites," in *Proc. Virtual Systems and Multimedia (VSMM)*, Gifu, Japan, H. Thwaites, Ed. Amsterdam, The Netherlands: IOS Press, Nov. 18–20, 1998, vol. I, Int. Soc. Virtual Systems and Multimedia, pp. 39–45.
- [5] J.-H. Hsu and C.-S. Tseng, "Application of three-dimensional orthogonal neural network to craniomaxillary reconstruction," *Comput. Med. Imaging Graph.*, vol. 25, no. 6, pp. 477–482, Nov./Dec. 2001.
- [6] J.-N. Hwang and Y.-H. Tseng, "3D motion estimation using single perspective sparse range data via surface reconstruction neural networks," in *IEEE Int. Conf. Neural Networks*, San Francisco, CA, 1993, vol. 3, pp. 1696–1701.
- [7] J.-N. Hwang and H. Li, "A translation/rotation/scaling/occlusion invariant neural network for 2D/3D object classification," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, San Francisco, CA, 1992, vol. 2, pp. 397–400.
- [8] J. Barhak and A. Fischer, "Parameterization and reconstruction from 3D scattered points based on neural network and PDE techniques," *IEEE Trans. Vis. Comput. Graphics*, vol. 7, no. 1, pp. 1–16, Jan.–Mar. 2001.
- [9] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, "Reconstruction and representation of 3D objects with radial basis functions," in *Proc. ACM SIGGRAPH*, Los Angeles, CA, 2001, pp. 67–76.
- [10] S.-Y. Cho and T. W. S. Chow, "A neural-learning-based reflectance model for 3-D shape reconstruction," *IEEE Trans. Ind. Electron.*, vol. 47, no. 6, pp. 1346–1350, Dec. 2000.
- [11] I. Kumazawa, "Compact and parametric shape representation by a tree of sigmoid functions for automatic shape modeling," *Pattern Recognit. Lett.*, vol. 21, no. 6/7, pp. 651–660, Jun. 2000.
- [12] E. Piperakis and I. Kumazawa, "Affine transformations of 3D objects represented with neural networks," in *Proc. Int. Conf. 3-D Digital Imaging and Modeling*, Quebec City, Canada, May 2001, pp. 213–223.
- [13] Y.-H. Tseng, J.-N. Hwang, and F. H. Sheehan, "Three-dimensional object representation and invariant recognition using continuous distance transform neural networks," *IEEE Trans. Neural Netw.*, vol. 8, no. 1, pp. 141–147, Jan. 1997.
- [14] A. Saalbach, G. Heidemann, and H. Ritter, "Representing object manifolds by parametrized SOMs," in *Proc. Pattern Recognition*, Quebec City, QC, Canada, 2002, vol. 2, pp. 184–187.
- [15] S. Loncaric, T. Markovinovic, T. Petrovic, D. Ramljak, and E. Sorantin, "Construction of virtual environment for endoscopy," in *Proc. IEEE Int. Conf. Multimedia Computing and Systems*, Florence, Italy, 1999, vol. 1, pp. 475–480.
- [16] A.-M. Cretu, E. M. Petriu, and G. G. Patry, "A comparison of neural network architectures for the geometric modeling of 3D objects," in *Proc. Computational Intelligence Measurement Systems and Applications (CIMSAS)*, Boston, MA, Jul. 2004, pp. 155–160.
- [17] —, "Neural network architecture for 3D object representation," in *Proc. IEEE Int. Workshop Haptic, Audio and Visual Environments and Their Applications*, Ottawa, ON, Canada, 2003, pp. 31–36.
- [18] A.-M. Cretu, "Neural network modeling of 3D objects for virtualized reality applications," M.S. thesis, School Inf. Tech. and Eng., Univ. Ottawa, ON, Canada, 2003.
- [19] C. Jutten, "Supervised composite networks," in *Handbook of Neural Computation*, E. Fiesler and R. Beale, Eds. Philadelphia, PA: IOP, 1997, pp. C1.6:1–C1.6:13.
- [20] J. A. Sirat and J. P. Nadal, "Neural trees: A new tool for classification," *Network*, vol. 1, no. 4, pp. 423–438, Oct. 1990.
- [21] T. Kohonen, *The Self-Organizing Map*. [Online]. Available: <http://www.cis.hut.fi/projects/somtoolbox/theory/somalgorithm.shtml>
- [22] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten, "Neural-gas network for vector quantization and its application to time-series prediction," *IEEE Trans. Neural Netw.*, vol. 4, no. 4, pp. 558–568, Jul. 1993.
- [23] M. Daszykowski, B. Walczak, and D. L. Massart, "On the optimal partitioning of data with k -means, growing k -means, neural gas and growing neural gas," *J. Chem. Inf. Comput. Sci.*, vol. 42, no. 6, pp. 1378–1389, Nov. 2002.
- [24] N. I. Badler and A. S. Glassner, "3D object modeling," in *Proc. SIGGRAPH Introduction to Computer Graphics Course Notes*, Los Angeles, CA, Aug. 1997, pp. 1–13.



Ana-Maria Cretu (S'04) received her M.A.Sc. degree in electrical engineering from the School of Information Technology and Engineering, University of Ottawa, Ottawa, ON, Canada. She is currently working toward the Ph.D. degree in electrical engineering at the University of Ottawa.

Her research interests include neural networks, tactile sensing, 3-D object modeling, and multisensor data fusion.



Emil M. Petriu (M'86–SM'88–F'01) received the Dipl. Eng. and Dr. Eng. degrees from the Polytechnic Institute of Timisoara, Romania, in 1969 and 1978, respectively.

He is a Professor and University Research Chair in the School of Information Technology and Engineering at the University of Ottawa, Ottawa, ON, Canada. His research interests include robot sensing and perception, intelligent sensors, interactive virtual environments, soft computing, and digital integrated circuit testing. During his career, he has published

more than 200 technical papers, authored two books, edited other two books, and received two patents.

He is a Fellow of the Canadian Academy of Engineering and a Fellow of the Engineering Institute of Canada. He is a Corecipient of the IEEE's Donald G. Fink Prize Paper Award for 2003 and recipient of the 2003 IEEE Instrumentation and Measurement Society Award. He is currently serving as Chair of TC-15 Virtual Systems and Cochair of TC-28 Instrumentation and Measurement for Robotics and Automation and TC-30 Security and Contraband Detection of the IEEE Instrumentation and Measurement Society. He is an Associate Editor of the IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT and a member of the Editorial Board of the IEEE I&M Magazine.

Gilles G. Patry, photograph and biography not available at the time of publication.