Reconstruction of Freeform Objects with Arbitrary Topology from Multi Range Images

Jacob Barhak

Reconstruction of Freeform Objects with Arbitrary Topology from Multi Range Images

Research Thesis

Submitted in Partial Fulfillment of the
Requirements for the
Degree of
Doctor of Philosophy

Jacob Barhak

Submitted to the Senate of the Technion
Israel Institute of Technology

Kislev 5763 Haifa December 2002

The Research Thesis Was Done Under The Supervision of
Dr. Anath Fischer in the Faculty of Mechanical engineering

THE GENEROUS FINANCIAL HELP OF THE TECHNION IS GRATEFULLY ACKNOWLEDGED

I would also like to thank:
My Family that supported me throughout the way
And remarkable teachers throughout the way

I thank Prof. Alex Oron for training me in mathematics
I thank Prof. David Degani for aiming me to proper methodologies
I thank Prof. Gershon Elber for his insightful remarks and consultation
I thank Prof. Moshe Shpitalni for pointing out the direction

But first and foremost I thank my Instructor Dr. Anath Fischer for her guidance and constant and relentless support

Table of Contents

Table of Contents

List of Figures

List of Figures

List of Figures

List Of Tables

Abstract

Reverse Engineering (RE) is the process of reconstructing a computerized model from a digitized object. Recently, RE has begun to play an essential role in CAD systems due to its robustness with respect to real 3D complex objects. In Reverse Engineering, laser scanners are commonly used for RE since they can sample 3D range images fast and very accurately relative to other technologies. Several open problems in the literature are viewed as a bottleneck in the RE reconstruction process: (1) The data size is enormous (2) The data is noisy (3) The topology is unknown, and sampled point connectivity relations are undefined. Moreover, the reconstructed CAD model should be compatible with current CAD systems.

This research proposes a new approach utilizing neural network techniques to overcome these problems. Neural networks generalize information by learning from data examples, making them suitable for RE tasks. This research utilizes and extends neural networks that employ competitive learning techniques, such as *self-organizing maps* (SOM) and the *neural gas* neural network technique.

This work proposes two new approaches for reconstructing objects from multiple range images, both of which rely mainly on neural network algorithms.

In the first approach, a parametric surface is reconstructed from each range image. Then, surfaces are merged together in order to reconstruct the boundary surface of the volumetric object. The parameterization problem, defining the connectivity between sampled points is solved using PDE (Partial Differential Equation) and SOM (Self Organizing Map) Neural Network. Then, an extension to 3D SOM methods is proposed for a single range image. Efficient B-Spline surface fitting methods were developed: (1) Control Mesh Approximation (CMA), which creates a fast initial approximation; (2) Gradient Descent Algorithm (GDA), which is based on an iterative LSQ technique; (3) Random Surface Error Correction (RSEC), which is based on the SOM method.

The second reconstruction approach reconstructs a triangular mesh, approximating the geometry and deriving the topology from the cloud of points. This approach solves the difficult topology detection problem by extending the *neural gas* concept beyond the original algorithm.

For feasibility of the reconstruction process, examples of several freeform objects with arbitrary topology will be presented.

1 Introduction

Reverse Engineering (RE) is the process of reconstructing a computerized model from a digitized object. Recently, RE has begun to play an essential role in CAD systems due to its robustness with respect to real 3D complex objects. Laser scanners are commonly used in RE since they can sample 3D range images fast and very accurately relative to other technologies. However, a laser scanner system provides an enormous amount of digitized point data that is irregular and scattered and therefore it requires intensive processing in order to reconstruct the surface model of the object.

Several open problems in the literature are viewed as a bottleneck in the RE reconstruction process: (1) The data size is enormous and oversized for the shape they represent. (2) The data is noisy, leading to reconstruction of an oscillating surface. (3) The topology is unknown; therefore, point connectivity relations are undefined. Moreover, the reconstructed CAD model should be compatible with current CAD systems.

A comprehensive review that includes comparison of reverse engineering techniques is provided in [VAR97]. In the literature, reconstruction of an existing object is usually based on the following stages:

1. Data acquisition
2. Range image registration
3. Topology detection
4. Segmentation
5. 3D freeform surface reconstruction.

Some reconstruction techniques may omit some of the stages above. However, the stages described above appear in most methods. The stages are described in the following sections followed by a presentation of solution methods as they appear in the literature.

1.1 Data acquisition

The **data acquisition** stage deals with sampling of physical objects. The most common sampling method today is laser scanning, which generates range data. Other sampling techniques include (1) CMM machines [MEN96] that generate sets of points; (2) cameras that produce intensity images or a stereoscopic image (digital photogrammetry); (3) acoustic sampling devices, such as sonar, that generate height maps; and (4) medical scanning devices such as CT and MRI that generate volumetric information. The data acquired from these devices is noisy and has significant problems such as occlusion. An elaborate discussion on these problems, including the advantages and disadvantages of the methods, can be found in [VAR97, MAN98].

This research focuses on range data obtained from a laser scanner. This technology is preferred due to its speed, robustness and accuracy relative to other methods.

A range image obtained from a laser scanner contains large amounts of data, usually tens of thousands of points. The data is reduced for one of the following reasons: (1) the data contains redundant points; (2) some reconstruction methods are not suitable for large amounts of data; (3) sampling errors occur, making the data less reliable. After sampling, unneeded data is filtered either by user intervention or by automatic methods.

1.2 Registration

A range image obtained from a laser sampling device consists of a set of points in space (x,y,z) that produces a 2½D height map in the local coordinate system of the laser. The object has to be scanned from several directions to solve occlusion problems. The local coordinates of each view must be transformed into a global coordinate system, taking into account the laser scanner position and orientation compared to the object. This process of transforming the local coordinate system of the views to a global coordinate system is called registration.

The most common technique found in the literature for solving the registration problem is referred to as ICP (iterate closest point). An instance of this technique, is presented in [CHE92]. It was extended to produce better results for multiple views in [BER96]. Another registration method is mapping curvature information of views to a sphere, so that the transformation between spheres is obtained from two views [HIG95].

At the end of the registration phase, the data consists of a cloud of points that represents the boundary of the object. However, errors are added to the cloud during the registration process.

1.3 Topology detection

The scans produced by the laser scanner usually lack connectivity information between the points. This topological information is essential for correct representation of the CAD model. The CAD model should represent objects of different genus as well as approximate their geometry.

The most common method for topology detection is based on creation of a triangular mesh that represents the scanned object. There are several triangulation methods: (1) local projection techniques [MEE00]; (2) methods that use triangulation per view and merge the triangles after registration [TUR94]; (3) methods that use *Delaunay triangulation* as a base for the final triangulation [AME98, AME99, EDE94, GUO97].

Local projection techniques are fast but are based on greedy methods. Methods that use per-view triangulation are also fast. However, some information is lost during the merging process. Methods that employ *Delaunay triangulation* are more complex and time consuming. However, these methods are preferable since *Delaunay triangulation* is a dual to the *Voronoi diagram* [EDE87, PRE85] that represents neighborhood relations between points in space. Since 3D *Delaunay triangulation* does not always represent the boundary of the scanned object, usually a subset of this triangulation is chosen. This subset can be the crust of the object [AME98] or a more elaborate representation, such as alpha shapes that depends on a distance parameter [EDE94].

1.4 Segmentation

The **segmentation** stage separates the object into different regions. Each region contains common geometric features or is bounded by a curve possessing a certain feature. Each region is later reconstructed separately. These segments can either be specified by the user or automatically derived from the data by a segmentation method.

Segmentation is usually performed on an existing polygonal representation of the object. In methods where the final geometric model is polygonal, segmentation is usually unnecessary. However, when the final geometric model is composed of primitives or freeform surfaces, segmentation defines the surface patch boundary and connectivity information to neighboring surface patches.

There are two major categories of segmentation methods: (1) edge based segmentation methods [MIL97, WAN94]; (2) region growing methods [FIT97, NIC95].

1.5 3D surface reconstruction

In this stage of the reconstruction, freeform surfaces are applied. The surfaces are fitted onto an object in order to create a CAD model. The surface reconstruction process usually consists of two main phases: the parameterization stage and the surface fitting stage, both discussed in the following sections. But first B-Spline surface representation, which is common in CAD systems is discussed.

1.5.1 B-Spline surfaces

B-Spline curves and tensor product B-Spline surfaces are defined by the equations:

$$C(u) = \sum_{i=1}^{n} B_i(u) \cdot \overline{V_i} \qquad (1)$$

$$S(u,v) = \sum_{i=1}^{n} \sum_{j=1}^{m} B_i(u) \cdot B_j(v) \cdot \overline{V_{ij}} \qquad (2)$$

where u and v are location parameters that uniquely locate a point on a curve or surface, and $\{B_i(u)\}_{i=1}^{n}$, $\{B_j(v)\}_{j=1}^{m}$ are the B-Spline basis functions which are piecewise polynomial.

A B-Spline surface is completely defined by its order *(Kn,Km)*, an ordered set of *n* by *m* control points (vertices) $\overline{V_{ij}}$, and two knot vectors, $\xi = \{\xi_i / i = 1..n + Kn\}$ and $\eta = \{\eta_j / j = 1..m + Km\}$, that define the parametric domain of the basis functions.

For the sake of simplicity, in this research the B-Spline surfaces used have the same order and knot vector in both directions (*u*,*v*). We will refer to the number of control points of the surface (*n*x*m*) as the size of the surface. And unless stated otherwise the knot vectors of the B-spline will be uniform with multiplicity at the boundary. That is, the surface interpolates the boundary to boundary control points.

Trimmed B-Spline surfaces are B-Spline surfaces that are defined only in part of their parametric domain, as in the following equation:

$$T(u,v) = S(u,v) \forall (u,v) \in \Omega \qquad (3)$$

where Ω is the domain in which the trimmed surface is defined.

A B-spline surface defines the shape of a surface, while trimming curves define the areas inside the parametric domain that are part of the surface. Trimming curves are closed directed curves that enclose areas containing the parametric domain. Curves advancing in a counter-clockwise direction define the area enclosed inside the curve, while clockwise curves define the area enclosed outside the curve. To avoid ambiguities the trimming curves should not self-intersect.

1.5.2 Parameterization

The parameterization stage establishes the intrinsic 2D parametric order and connectivity between the sampled points. The problem of assigning parameterization to range data is addressed in the literature for simple boundaries. The problem increases when the boundary shape is concave or curved, as discussed in [FIS99].

Parameterization assigns *(u,v)* parameters to each sampled point. For an ideally fitted surface, these parameters interpolate the sampled points. The parameterization problem is similar to the chicken and the egg problem. In order to fit a parametric surface to the points, the parameterization of the points should be known, however, in order to determine their exact parameterization, the surface shape should be known. Therefore, surface prediction methods need to be devised. In the literature, there are several parameterization schemes. However, different parameterizations will lead to different surface characteristics, such as orientation of its iso-curves. A good initial parameterization can be defined as a parameterization that satisfies the following conditions:

Neighborhood preservation – neighboring sampled points will be neighbors in the *(u,v)* parametric domain.

Boundary interpolation – boundary points will have boundary parametric values.

Non-self-intersection – the surface that is fitted according to the parameterization will be non self-intersecting.

Uniform parametric density – the sampled points will be spread uniformly over the parametric domain and thus satisfy a fairness criterion.

The parameterization techniques found in the literature fulfill only two to three of these criteria [DEL97, HOS93, MA95]. A summary of the main parameterization methods is given in table 1.

1.5.2.1 Initial parameterization

The main parameterization characteristics are depicted by the initial parameterization. Different initial parameterization criteria can be used to assign the parametric values: equally spaced, chordal, centripetal, geometric and affinity invariant parameters. These strategies are based on the assumption that the points are arranged in an ordered grid and therefore are not suitable for scattered digitized points.

A common method for initial parameterization is based on projecting the points onto a plane [PAR95, HOS93] and performing the parameterization in 2D. Selecting the optimal projection plane can improve results, as seen in [CHA99]. However, 2D parameterization methods lose information in the projection stage. On the other hand, 3D parameterization techniques are usually accomplished by projecting the sample points directly onto an initial 3D base surface. By that, the parameterization is significantly improved [MA95]. Several types of construction methods of initial 3D base surfaces are common: (1) four-corner points or two boundary curves; (2) four boundaries represented as a Coon's patch; (3) section curves; (4) boundaries and interior sections. For each type, a B-spline implementation is fitted. However, all the methods reviewed [HOS93, MAN98, MA95] may suffer from self-intersecting parameterization. For example, concave boundaries of a Coons surface might produce a parameterization that maps points inside the boundary to the exterior of the surface. Moreover, it may also connect the points in a faulty topology, which leads to a self-intersecting surface. [DEL97] confronts this self-intersection problem using conformal mapping. However, conformal mapping is time consuming and suffers from numerical problems.

Reference	General Method			Initial parameterization Characteristics			Global and Optimization characteristics				Notes
	Initial parameterization	Alternating optimization	Parametric optimization & surface fitting	Non self intersecting	Fast initial parameterization	Uniform parametric density	Fast overall convergence	Global optimization	Low memory complexity	Applicable for surfaces	
ALH99		+					+	+	–		Optimization by implicit cost function.
CHA99	+				–					–	Chooses best projection plane
DEL97	+			+	–	–				+	Conformal mapping
HOS93 HOS89 HOS88	+	+		–	+	–			+	+	Uses Coons Patch representation
LAU93			+						–	–	Optimization by global cost function.
MA95	+	+		–		–			+	+	Uses base surface
ROG89		+								+	Point projection
SAR91		+					+	+	–		Optimization by explicit cost function
SPE98			+						–	–	Optimization by global cost function
VAI97	+				–			+			Uses linear programming and is reported slow
Proposed approach PDE	+	+		+	+		+		+	+	See section 3.1.1.1.1
Proposed approach SOM	+	+			+	+	+		+	+	See section 3.1.1.1.2

Table 1: Parameterization techniques comparison.
טבלה 1: השוואה של שיטות פרמטריזציה.

1.5.2.2 Parametric optimization

Parametric optimization methods can be divided into two main categories: (1) global parametric optimization performed during surface fitting; and (2) alternating method, in which parametric optimization is performed separately from surface fitting.

In global parametric optimization techniques, parameterization is part of the surface fitting process [SPE98, LAU93]. In those methods, the distance of the points from the surface is minimized according to a cost function using the Conjugate-Gradient algorithm [LAU93] or the Levenberg-Marquardt algorithm [SPE98]. This minimization changes the point parameterization and fits the surface simultaneously. These methods converge for curves, but are not efficient for surfaces due to the large number of sampled points.

Most parametric optimization methods are based on the alternating method [MA95, HOS88, HOS89, ROG89, ALH99, MAN99]. In these methods, parameterization and surface fitting are applied in a recurring procedure until convergence. An initial parametric surface is used as a base for parametric optimization. The points are then projected onto the surface and are re-parameterized according to its projected value [HOS88, HOS89, ROG89, MA95]. The method is simple and has small memory complexity. However, it is optimal only locally and not globally. In [SAR91], an implicit global cost function is defined and then minimized using the Levenberg-Marquardt optimization method. In [ALH99], an optimal control formulation is used to explicitly calculate the gradients. Moreover, other cost functions are introduced that minimize curve length or curve bending energy.

The alternating method used by [HOS89] can efficiently handle large-scale data according to its low space complexity and simplicity. Therefore, it was chosen as the base for our method despite its local nature.

1.5.3 Surface fitting

A fitted surface must satisfy global and local constraints, such as shape preservation, continuity conditions, boundary conditions, and geometric constraints, as well as smoothness and accuracy according to a given tolerance. In order to fit a B-Spline parametric surface to the sampled points, all the following parameters should be determined: mesh size, control points, degree of basis functions, knot vectors and weights.

Generally, the mesh size, degree and knot vectors are predicted, while point parameterization is calculated in the parameterization stage and the control points are calculated in the surface fitting stage [HOS88, HOS89, ROG89]. [MA95] expands these methods by giving a relatively fast, but non-optimal, algorithm for calculating an average knot-vector. [SPE98] calculates the point parameterization and control points during the surface fitting stage. The method presented in [CHA99] uses cross-validation techniques in order to determine mesh size of the B-Spline; this ,however, requires recalculation. In [LAU93], only the size and the order of the B-Spline are predetermined, while all other variables are calculated during the surface fitting stage.

Most techniques use least squares (LSQ) minimization algorithms that minimize the distance from the sampled points to the approximated surface [MA95, HOS89, ROG89, SPE98, LAU93, ALH99]. Some LSQ methods satisfy additional geometric constraints that are formulated in the minimization function [ROG89, ALH99]. Some surface fitting techniques have a local nature [LEV] using moving least squares and do not use B-Spline formulation. The LSQ methods can be divided into iterative and non-iterative methods. Common iterative methods include gradient methods, Newton and Quasi-Newton methods and the Levenberg-Marquardt method [HOS89, ROG89, MA95, SPE98, LAU93, ALH99]. Non-iterative LSQ methods solve an over-constrained equation set by using the Householders reduction [MA95], by calculating the pseudo inverse matrix, or by other methods such as SVD [MAN99, FIS99]. However, only iterative methods can handle large-scale data (table 2) due to their small memory consumption.

Reference	Parameterization method			Surface parameters that are calculated					Efficiency		Notes
	Optimization & surface fitting	Alternating	Other	Parameterization	Control points	Knot vector	Point weights	Size	Low memory complexity	Efficient for surfaces	
ALH99		+							–	–	Gradient descent method
CHA99			+	+				+	–	–	Size determined by cross validation
HOS93 HOS89		+		+	+				+	+	Alternating method
LAU93	+			+	+	+	+		–	–	Conjugate gradient method
MA95		+		+	+	+			+	+	Non-iterative solution using matrix sparsity
SAR91		+		+	+				+	+	Levenberg Marquard method
SPE98	+			+	+				–	–	Levenberg Marquard method
Proposed approach		+		+	+			+	+	+	See section 3.1.2

Table 2: Comparison of B-Spline surface fitting techniques.
טבלה 2: השוואה של שיטות התאמת משטחי B-Spline.

1.6 Reconstruction techniques

There are very few methods that are fully automated and suitable for objects with arbitrary topology. These methods are described in this section and summarized in table 3.

Most methods described are based on an approach, in which an initial triangular mesh is created to represent the topology, usually by Delaunay triangulation. Then, where freeform surfaces are used, the mesh is simplified and surface patches are reconstructed over the simplified mesh. Some of the methods reconstruct 3D objects in the form of polygonal meshes and others in the form of as polynomial meshes like B-Spline surfaces , Bezier patches or subdivision surfaces. Polynomial patches are more suitable as CAD models since they can represent the surface of the scanned object compactly and provide information that is important for tasks like analysis or manufacturing. Following is a short description of the above methods.

Reference	Object representation					Intermediate reconstruction representation			Characteristics		
	Tensor product B-Spline surfaces	Polygons	Sub-division surfaces	Bezier patches	Other	α-shapes	Triangular mesh	Other	Model enables incremental adding of range images	Smooth surfaces with sharp edges	Continuity
AME99		+					+		–	–	C^0
BAJ95				+		+			–	–	C^1
CUR96		+						+	+	–	C^0
DUA01			+				+		–	–	C^1
ECK96	+						+	+	–	–	G^1
GUO97					+	+			–	–	C^2
HOP92		+						+	–	–	C^0
HOP94			+					+	–	+	C^{2*}
MEE00		+					+		–	–	C^0
RUT94		+					+		+	–	C^0
TUR94		+					+		+	–	C^0
CAR01		+						+	–	–	C^0
Proposed approach A section 2.1	+							+	+	–	C^{2*}
Proposed approach B section 2.2		+						+	–	–	C^0

*This continuity is not always preserved at patch boundaries

Table 3: Characteristic of automatic reconstruction methods for arbitrary topology.

טבלה 3: תכונות של שיטות שחזור אוטומטיות לגופים בעלי טופולוגיה כלשהי.

In [AME99] Voronoi filtering is used for reconstruction of the crust surface representing the volumetric object. The method works on input data that is sufficiently dense compared to the feature size of the scanned object. However, this is not always the case in practice.

In [BAJ95] alpha shapes provide support for triangular Bezier patches fitted later. The tetrahedrons are adaptively split until a certain tolerance is met. However, the specification of the alpha value presents a problem.

In [CUR96] a volumetric representation can store a cumulative signed distance function that is later interpreted to a triangular mesh. The method processes one range image at a time and thus has a advantage of being able to present new data to the model incrementally. This task is important for correcting incomplete scans or for reducing the overall time for scanning and reconstruction by pipelining the range image data. The same advantage exists in [RUT94,TUR94]. However, in these methods range images from each view are triangulated and then added to the final model, which may result in data loss and inaccuracies at overlapping areas.

In [DUA01] objects are reconstructed by minimizing an energy function that governs the shape of a mesh. The mesh starts from a genus zero balloon shape and changes its topology when collisions occur. Vertices are added and deleted from the mesh during its evolution. The mesh forms a base for a subdivision surface constructed using the Loops subdivision scheme. The method relies on collision detection while growing and thus is not suitable for large models.

In [HOP92, HOP94, ECK96] a signed distance function is calculated using local planes. Then a mesh is reconstructed by extracting the zero set of this implicit function. In [ECK96] the mesh is simplified and used as a base for the Loops subdivision scheme, with modifications enabling representation of sharp features. In [HOP94] the mesh is simplified and transformed to a quadrilateral mesh that is later fitted with B-Spline surfaces. Although these methods are successful for complex objects, they still rely on heuristics and parameters such as grid density.

In [GUO97] alpha shapes are used for reconstruction of the initial mesh and polynomial manifolds [GRI95] to represent the surface. These manifolds converge to tensor product B-Splines in rectangular domains, and they can model arbitrary topology meshes easily while preserving continuity. However, an optimal alpha value for creating the mesh is usually unknown.

In [MEE00] local projection techniques are used to reconstruct a triangular mesh. The mesh is created by locally projecting points in the normal direction that is computed by the axis of a semi minimal bounding cone. The neighboring points to the boundary of the mesh are ordered by increasing angle. While the mesh grows, local candidate points are checked so as not to occlude the mesh or create thin triangles. The method is fast; however the algorithm is greedy in nature.

In [CAR01] a discrete signed distance function is constructed using information about the point of view that the points in the cloud were taken from. A radial basis function, which is usually used in the field of neural networks, is then used to fit the signed distance function. Then the zero set of the function is extracted to compute a polygonal mesh. The method is able to fill in holes in the model as well as reconstruct it. However, additional information about the scanning direction is required.

1.7 Neural networks and reverse engineering

Artificial neural networks (NN) are techniques suitable for reconstructing data from samples. Artificial neural networks are inspired by the human brain and the phenomena occurring within it. An artificial neural network is composed of neural units that autonomically perform simple mathematical computation and of connections, called synaptic links, between the neural units. Artificial neural networks are usually trained using a learning process over partial data called a training data set. After the training process, the neural network can be queried for information on data other than the training data. In other words, it can be said that neural networks learn from examples. Neural networks are used for tasks such as classification, control and recognition.

Several common models exist for the design of neural networks. Basic common categories consist of *feed forward neural networks*, also referred to as *multi-layer perceptron neural networks,* and of *self organizing neural networks* [HAY94]. The name “neural networks” is usually associatively interpreted as *feed forward neural networks* due to their common use. However, in this work *self organizing neural networks* are used.

Self-organizing neural networks usually employ *competitive learning* techniques in which the neural units compete for proximity to a sampled data vector. The data within the winner neural unit is then changed accordingly. While in *hard competitive learning*, only the winner neural unit is adapted, in *soft competitive learning*, the neighborhood of the winner neural unit is usually also adapted. More details on *competitive learning techniques* can be found in [FRI95].

Various soft competitive learning methods exist of which the Self Organizing Map (SOM) neural network is the most well known. It was proposed by Kohonen [KOH97]. The method adapts a graph of a given topology into the geometry represented by the training data set. As in this work, the graph is frequently chosen as a rectangular topology lattice to represent a surface. The basic SOM method was extended to the *growing grid* algorithm [VAR99, FRI97], where the number of neural units in the lattice may grow by adding rows and columns to the lattice during training.

The above neural networks adapt the shape for a given topology and do not alter its topology. The *neural gas* neural network [MAR91, MAR94] uses *soft competitive learning* principles to place the neural units in space according to the given data set. The method detects the connectivity between the neural units using the *Hebbian learning rule* [HAY94]. The topology is detected as a byproduct of the training process that positions the neural units in space. The number of neural units is set at the initialization of the algorithm. The method, however, cannot detect topological information such as faces of a triangular mesh.

Neural Network type	References	Number of neural units changes during training	Topology information is used during training	Detects topology	Represents faces in a mesh	Remarks
SOM	KOH97		+			Initial topology may be set and will not change.
Growing Grid	VAR99 FRI97	+	+			VAR99 and FRI97 differ in formulation of neural unit addition rule
Neural Gas	MAR91 MAR94 FRI97			+		Geometrical information is used to define topology
Growing Neural Gas	FRI95 FRI97	+	+	+		
Growing Cell Structures	FRI97	+	+		+	Initial topology may be set and will not change.
Extended Neural Gas	Section 3.2.1			+	+	Proposed approach for topology detection

Table 4: Comparison of self-organizing neural network types.

טבלה 4: השוואה של רשתות בינה מלאכותיות מסוג self-organizing.

In [FRI95,FRI97], the *growing neural gas* neural network is presented. This neural network extends the *neural gas* neural network. The extensions enable the neural network to use already detected topological information while training in order to conform to the geometry. This neural network is able to add neural units while preserving the detected topology. However, the method cannot represent faces in a triangular mesh.

A neural network that can represent faces in a mesh and grow neural units while learning the geometry is the *growing cell structures* neural network. However, the topology of the neural network has to be preset at the initialization stage.

Neural networks are used extensively within this research for RE tasks, as will be described in the following chapters. Various forms of neural networks were used for different tasks. A brief comparison of these and similar algorithms is presented in table 4.

The nature of neural networks makes them suitable for reverse engineering techniques. There have been several attempts to solve RE problems using artificial neural networks. *Feed forward neural networks* were used in [GU95] to represent B-Spline surfaces. Typically *self-organizing* methods were used to reconstruct the geometry of objects under a given topology, as in [CHE96, VAR99]. Other NN techniques that use self-organization methods in conjunction with topology detection include *neural gas* [MAR91, MAR94] and *growing neural gas* [FRI95, FRI97]. *Neural gas* was employed by [BOR00] in order reconstruct meshes; however, the topological issue was not discussed. In [CAR01] radial basis functions, which are often used in the NN field, are used to reconstruct arbitrary topology objects. This method requires a signed distance function extracted from the point cloud as input.

Neural network techniques have the potential for extensions in other research areas. Several examples are the SOM neural network resemblance to *Snakes* [KAS86] and *active contours* described in [ABR96] and the *Hebbian learning law* connection to *Voronoi diagram*s [MAR91,MAR94] used in the *neural gas* neural network. The following chapters will describe in detail how extended neural network techniques are applicable for RE problems.

1.8 Research Aim

The research aim is to develop reconstruction methods for arbitrary topology freeform objects from multiple range images. Two reconstruction approaches are presented that solve the bottleneck problems of the reconstruction process. The novelty of the approaches lie in the way that topology is detected. Both approaches take advantage of artificial learning techniques by employing neural networks that are able to deduce connectivity information while approximating the objects shape.

The major reconstruction problems addressed in this research are:

a. The connectivity between the sampled points is not defined explicitly. Therefore the topology of the object is not defined.

b. The noise contained in the range images interferes in the reconstruction process, making model interpolation unsuitable and motivating towards geometric approximation.

c. The enormous data size of the point cloud is too large for CAD systems to handle efficiently.

d. The representation of the model is not always compatible with current CAD systems.

This research proposes two reconstruction approaches for reconstructing the boundary representation from a scanned point cloud. The stated reconstruction problems are solved in this research by using neural network methodologies and other numerical and geometrical techniques. New methods were developed for are employed for the reconstruction approaches solving sub-problems such as topology detection, parameterization and surface fitting. Although not a major goal, existing neural network techniques are improved within this research and are oriented for the purpose of model reconstruction.

In the first approach, triangulation is avoided and a CAD model composed of trimmed B-Spline surfaces is reconstructed (section 2.1). Parameterization defines the connectivity of the points by using PDE and SOM neural network techniques. Surface fitting methods approximate the surface of the object compactly and by this reduce noise significantly. Merging of the reconstructed surfaces enables reconstruction of the object from multi-range images. The reconstructed CAD model is compatible with current CAD systems.

In the second approach, a triangular mesh is reconstructed using neural network techniques (section 2.2). The main advantage of the approach is its ability to detect the topology using an artificial learning mechanism. The geometry of the object is approximated to a predefined mesh size. The final model is compatible with current CAD systems.

In the following chapters, both approaches will be described in detail.

2 Approach

This chapter describes the methods used to solve various RE problems presented in the previous chapter. Within this research, artificial neural network methods were primarily applied to parameterization, fitting and topology detection problems. Parameterization and surface fitting were also solved using other numerical methods.

The reconstruction approaches will be described in the following sections, a description of the implementation is given in chapter 3. Results are displayed in chapter 4.

2.1 Approach A – reconstruction of surfaces from a cloud of points

This approach reconstructs freeform surfaces over a large domain of an object. Each range image is fitted separately with a B-Spline surface and the surfaces are trimmed and merged in order to reconstruct a boundary representation of the final object. This approach, captures both global and local shape characteristics. The approach consists of the following stages, as shown in figure 1: (1) parameterization (2) surface fitting (3) merging.

In most reconstruction techniques, range images are merged and then segmented. The segments are parameterized and fitted by surfaces. Adding a new range image requires recalculation of the entire model, where segments are fitted locally. In contrast to common methods, the approach presented in this section requires no segmentation and is adaptive. It allows incremental refinement of the geometric model by adding range images adaptively during or after object reconstruction.

The approach is based on the assumption that range images have already been registered to a global coordinate system. Figure 1 demonstrates the reconstruction stages ,while figure 2 demonstrates a flow diagram of the reconstruction approach.

As can be seen from figure 2, several possibilities exist for choosing a parameterization method and a surface fitting method. The following sections will briefly present the various parameterization and surface fitting methods developed in this research. The methods are discussed in details in chapter 3.

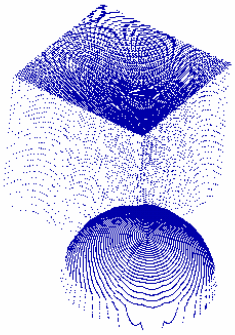
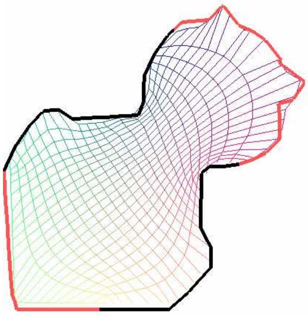
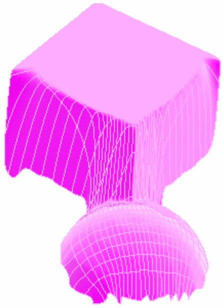
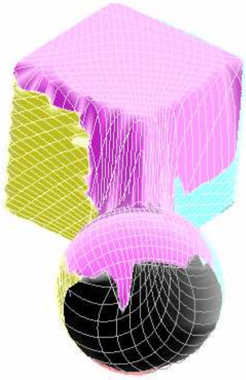
0. Data acquisition stage The range data is scanned from a single view of an object.	
1. Parameterization stage The 3D data is mapped to the 2D parametric domain while preserving point neighborhood relations.	
2. Surface fitting stage A B-Spline surface is fitted to the parameterized range image.	
3. Merging stage The surfaces from each view are trimmed and merged into a complete 3D object.	

Figure 1: Approach A - reconstruction stages.

איור 1: גישת שחזור א' - שלבי שחזור.

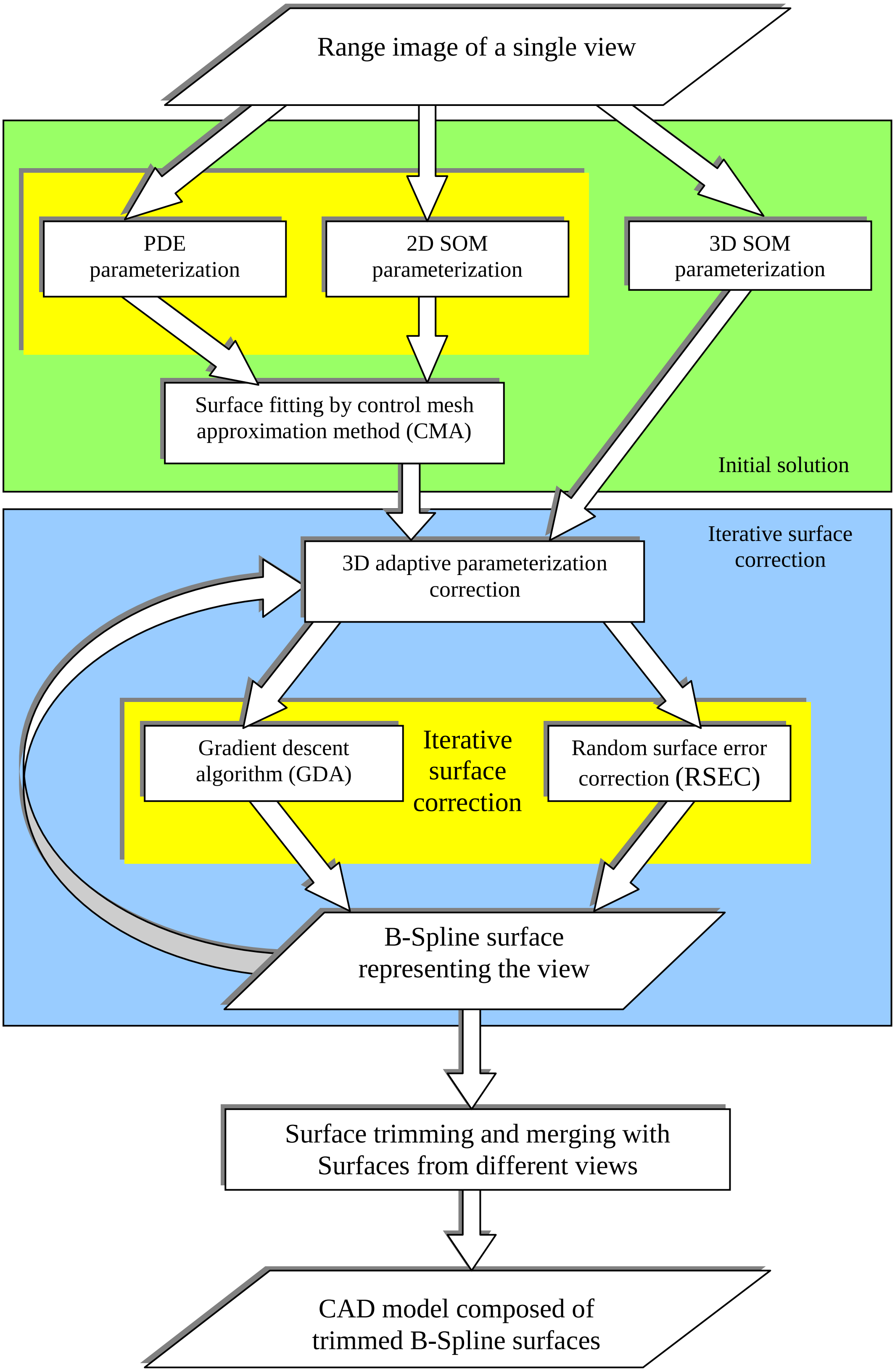

Range image of a single view
PDE parameterization
2D SOM parameterization
3D SOM parameterization
Surface fitting by control mesh approximation method (CMA)
Initial solution
Iterative surface correction
3D adaptive parameterization correction
Gradient descent algorithm (GDA)
Iterative surface correction
Random surface error correction (RSEC)
B-Spline surface representing the view
Surface trimming and merging with Surfaces from different views
CAD model composed of trimmed B-Spline surfaces

Figure 2: Approach A - Flow diagram.
איור 2 : גישת שחזור א' - תרשים זרימה

2.1.1 Parameterization

In the proposed reconstruction method, parameterization is accomplished in two main stages: (1) initial parameterization and (2) parametric optimization.

Initial parameterization can be performed by a 2D parameterization method, where the sampled points are projected onto a plane in the scanning direction. Initial parameterization can also be performed by a 3D parameterization method where a 3D grid with rectangular topology approximates the range image.

For the proposed approach, new 2D initial parameterization methods, PDE (Partial Differential Equation) and neural network SOM (Self Organizing Map), were developed for initial parameterization. Those methods are described in detail in sections 3.1.1.1.1 and 4.1.1.1.2.

A 3D SOM initial parameterization method is also presented in which a 3D rectangular grid is created that can be viewed as an approximation of the surface. 3D parameterization is useful when the projection direction is unknown. The SOM method is extended using the *growing grid* algorithm to aid in convergence and to detect the aspect ratio of grid. 3D parameterization is described in detail in section 3.1.1.1.3.

3D base surface parameterization is used for parametric optimization in order to improve surface fitting results. It is part of the alternating method and can be applied only when an approximating surface exists. The method is described in section 3.1.1.2.

2.1.2 Surface Fitting

After the parameterization stage, the neighborhood relations between the points are known. The goal of the fitting process is to find the best approximation with respect to the sampled points. In this work, B-Spline parametric surfaces are used as the geometric representation due to their natural compatibility in CAGD. B-Spline surfaces are preferred due to their intuitive control, smoothness, subdivision characteristics and local support.

When point parameterization is set, surface-fitting problem can be formulated as a linear over-constraint set of equations. However, traditional direct methods like pseudo inverse and SVD were found unsuitable for practical reasons. Instead, iterative methods were developed that require a fast initial approximation.

Three algorithms are proposed for the surface fitting process:

(1) Control Mesh Approximation (CMA), which creates a fast initial approximation and is described in detail in section 3.1.2.1.

(2) Gradient Descent Algorithm (GDA), which is an iterative LSQ technique that corrects the initial approximation and is described in detail in section 3.1.2.2.

(3) Random Surface Error Correction (RSEC), which corrects the initial surface approximation and is based on the ideas of the SOM neural network. The method is presented in section 3.1.2.3.

2.1.3 Merging surfaces from multi range images

Previous stages created B-Spline surfaces representing the object from different views. Surface merging is based on trimming the surfaces at a common curve and then creating a reconstructed 3D envelope from trimmed surfaces. This approach enables adaptive reconstruction from multi range images. The local merging of a new surface to the 3D model has significantly lower computational complexity than global fitting methods. Surface merging consists of the following main stages:

1. Detecting 3D proximity between 3D surfaces.
2. Creating distinction curves between surface pairs.
3. Trimming the surfaces.

The stages are presented in detail in section 3.1.3.

2.2 Approach B – reconstruction of a triangular mesh from a cloud of points

The second reconstruction approach reconstructs an approximating triangular mesh from a cloud of points. The cloud is composed of several range images registered and merged together. The cloud can represent objects with arbitrary topology.

The base for the reconstruction method is the *neural gas* neural network that is able to detect the geometry and topology of the point cloud. In order to reconstruct the faces of the mesh, the *neural gas* algorithm was extended as part of this research.

The *extended neural gas* algorithm produces a mesh with normal information at the vertices. The mesh is not a manifold mesh. A manifold mesh creation algorithm follows the *extended neural gas* algorithm. The following sections will present the *extended neural gas* algorithm and the *manifold mesh creation* method. Figure 3 displays the stages in the approach.

2.2.1 Extended neural gas

The *neural gas* algorithm is based on a neural network technique. The neural network is defined by neural units representing positions in space and synaptic links representing connections between the neural units (see figure 4). The *neural gas* algorithm is usually used for minimizing the quantization error (distortion) between the sampled points and the neural unit positions; this process is referred to as *vector quantization*. The synaptic links between the neural units are formed by using a *Hebbian learning* process while training the neural network. The result is a *topology preserving map*, where the neural units represent the map vertices and the synaptic links represent the map edges.

An approximating mesh which is a *topology preserving map* holds a desirable property for reconstruction. This property can be shortly explained as follows: if two sample points are close in space, the vertices representing them in the mesh will be connected with an edge. Furthermore, if two vertices of the approximating mesh are close in space, they will be connected with an edge. This property was proved in [MAR94] and is elaborated in section 3.2.3.1.

Another desirable property of the *neural gas* algorithm is that the mesh created by it is a subset of the *Delaunay triangulation* as shown in [MAR94]. The *Delaunay triangulation* [is dual to the *Voronoi diagram* that represents neighborhood relations among the vertices of the mesh (see [EDE87]).

The mesh created by the *neural gas* algorithm is not a well defined mesh in the sense that it is composed only of vertices and edges, while faces are not defined. The *neural gas* algorithm was extended in order to reconstruct a triangular mesh with normal information at the vertices. The *extended neural gas* algorithm simultaneously accomplishes the following tasks:

- Geometric approximation of the point cloud by *vector quantization*.
- Detection of edges and triangular faces that are a subset of the *Delaunay triangulation*.
- Approximation of normals to the mesh vertices.

The extensions to the algorithm are elaborated in section 3.2.1.

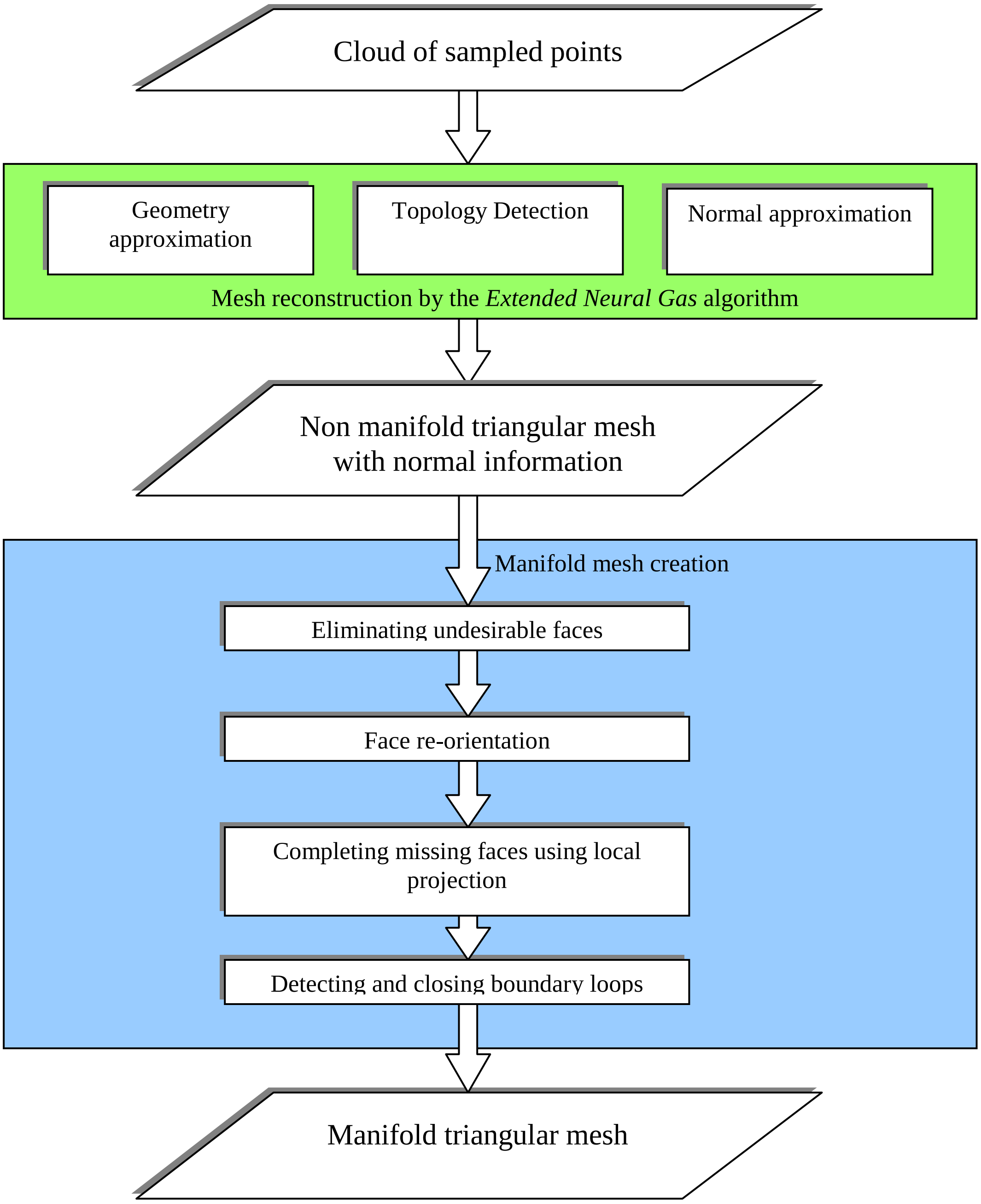

Cloud of sampled points
Geometry approximation
Topology Detection
Normal approximation
Mesh reconstruction by the *Extended Neural Gas* algorithm
Non manifold triangular mesh with normal information
Manifold mesh creation
Eliminating undesirable faces
Face re-orientation
Completing missing faces using local projection
Detecting and closing boundary loops
Manifold triangular mesh

Figure 3: Reconstruction approach B - flow diagram.
איור 3 : גישת שחזור ב' – תרשים זרימה

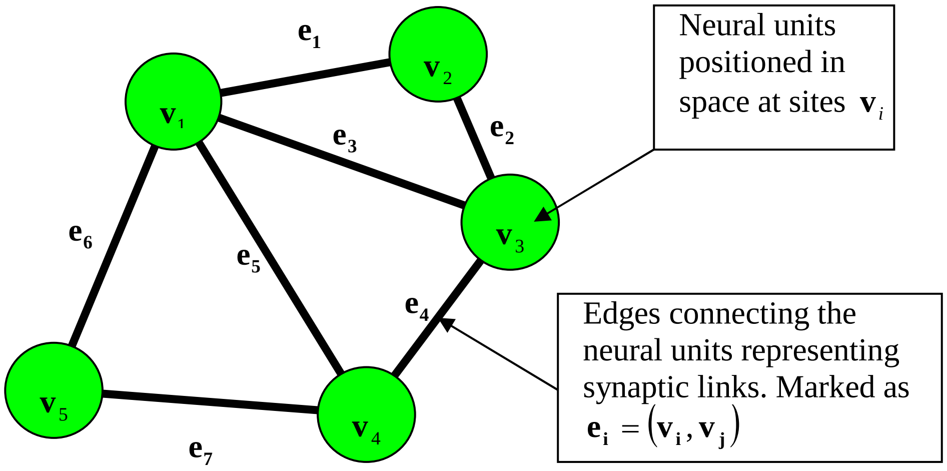

e1
v2
v1
e2
e3
e6
v3
e5
e4
v5
v4
e7
Neural units positioned in space at sites $\mathbf{v}_i$
Edges connecting the neural units representing synaptic links. Marked as $\mathbf{e}_\mathbf{i} = (\mathbf{v}_\mathbf{i}, \mathbf{v}_\mathbf{j})$

Figure 4: The structure of a *neural gas* neural network.
איור 4: מבנה רשת הבינה המלאכותית *neural gas*

2.2.2 Manifold Mesh Creation

The *extended neural gas* algorithm creates a mesh that approximates the surface. This boundary representation is not a manifold surface, since holes exist in it and the faces are not oriented in the same direction. A non-manifold mesh is not usable as a CAD model.

A completion algorithm composed of four stages is used to create a manifold mesh. The stages are:

Stage 1: Eliminating undesirable faces.

Stage 2: Face re-orientation.

Stage 3: Completing missing faces using local projection.

Stage 4: Detecting and closing boundary loops.

A detailed implementation of those stages is briefly presented in section 3.2.2.

3 Implementation

The previous chapter described two reconstruction approaches. This chapter describes the methods used in those approaches in detail. The chapter is divided into two main sections: section 3.1 describes the implementation of methods used in approach A where a CAD model composed of B-Spline surfaces is reconstructed and section 3.2 describes implementation of the methods that are used in approach B where a triangular mesh is reconstructed.

3.1 Approach A – reconstruction of surfaces from a cloud of points

Reconstruction of B-Spline surfaces relies on parameterization methods, surface fitting methods and surface merging methods. Parameterization methods are descried in details in section 3.1.1, surface fitting methods are described in section 3.1.2 and the surface merging method is described in section 3.1.3.

3.1.1 Parameterization

Parameterization of sampled points for reconstruction of B-Spline surfaces is accomplished in two main stages: initial parameterization and parametric optimization.

Section 3.1.1.1 describes the implementation of initial parameterization methods and section 3.1.1.2 describes the implementation of parametric optimization that is used to improve parameterization.

3.1.1.1 Initial parameterization

The purpose of initial parameterization is to define the neighborhood relations for the sampled points in the range image by assigning them parametric values. Since the surface that is described by the range image is unknown, initial parameterization is accomplished using PDE and neural network methods.

In this research, the initial parameterization methods create a rectangular topology grid that is used to establish the neighborhood relations. A distinction is made between 2D initial parameterization and 3D initial parameterization. In 2D parameterization, the sampled points are projected onto a plane and the grid is constructed around the projected points. In 3D parameterization the grid resides in 3D and approximates the surface. First 2D parameterization is discussed.

The initial 2D parameterization is based on the following stages: (1) projection, (2) 2D parametric grid generation, and (3) point parameterization by grid. In the projection stage, 3D data is mapped into 2D data, since it is simpler to find a neighborhood in 2D. During the mapping process, the point neighborhood should be preserved. The reference plane is obtained from the sampling direction of the view (see figure 5a). After the 3D points *(x,y,z)* are projected onto a plane. Each point is assigned with 2D *(p,q)* values on the coordinate system of the plane. Then a rectangular quadrilateral grid is created. Finally, the parameterization is established in two steps: (1) mapping a point *(p,q)* to a quadrilateral cell *(i,j)* and (2) calculating its *(u,v)* parameterization value. For that, a reverse bi-linear interpolation is applied on the quadrilateral element as described in section 3.1.1.1.4.

As part of this research two parameterization methods were developed:

PDE parameterization – based on the Laplace Partial Differential Equation.

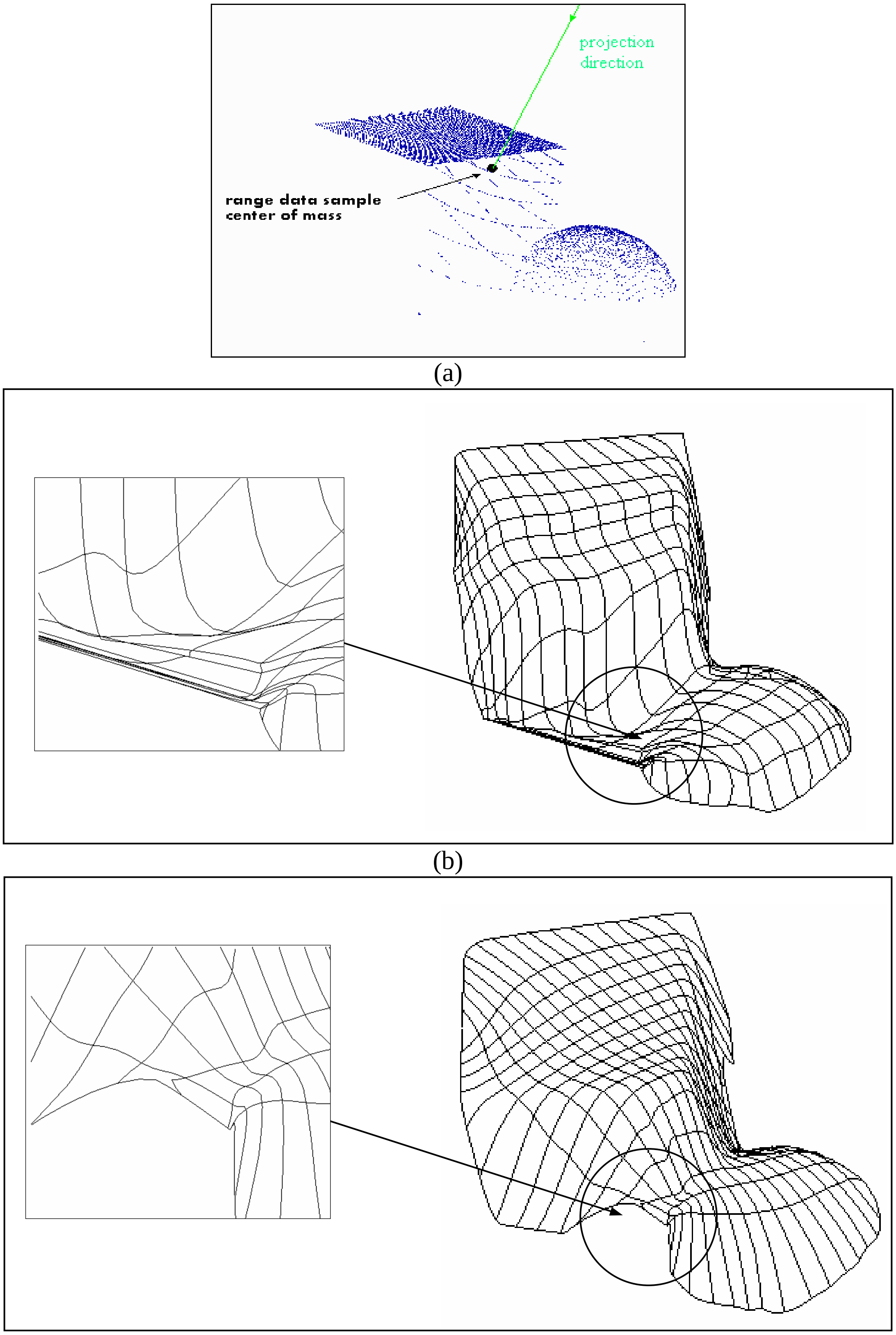
SOM parameterization – based on the Self Organizing Map neural network.

Both methods fulfill the first two grid requirement criteria of neighborhood preservation and boundary interpolation that were described in section 1.5.2.

The PDE method fulfills the third criteria of non self-intersection. Surfaces fitted over sampled points parameterized with this method are less prone to self-intersection as can be seen in figure 5. The method is based on the idea that iso-temperature curves on a plate heated at its boundaries do not intersect. Boundary conditions have to be defined in order to solve the Laplace equations. The method is described in detail in section 3.1.1.1.1.

The SOM method strives to fulfill the requirement of uniform parametric density by minimizing the quantization error. The method detects the orientation and the grid without assigning boundary conditions. However, the boundary of the object is not interpolated with the SOM algorithm. Two correction algorithms were developed to cope with this problem: *Boundary first Algorithm* and *Boundary Last Algorithm*. These extensions were implemented and described in section 3.1.1.1.2.

The SOM parameterization method has the inherent capability of dimensionality reduction. Meaning that 3D data can be represented with a grid that has 2D rectangular topology residing in 3D space. Therefore, the SOM method can be used as a base for 3D parameterization. Parameterization in 3D is more difficult than in 2D. In order to aid in convergence and detect the aspect ratio of the range image in 3D, the *growing grid* extension to the SOM algorithm was implemented. 3D SOM parameterization and the *growing grid* extension are presented in section 3.1.1.1.3.



(a)

(b)

(c)

Figure 5: (a) Sampling of a sphere and a cube. (b) A self intersecting surface fitted after bilinear parameterization. (c) The non self intersecting surface fitted after PDE parameterization.

איור 5: (a) דגימה של גוף הבנוי מקוביה וכדור. (b) משטח עם התנגשות עצמית שהותאם אחרי פרמטריזציה
בילינארית. (c) משטח בלי התנגשות עצמית שהותאם לאחר פרמטריזציה בשיטת PDE.

3.1.1.1.1 PDE parameterization

The Laplacian PDE was chosen because it produces non-self-intersecting iso-curves. Such a second order PDE formulation is frequently used as a model to formulate heat transfer problems and to compute iso-potential curves.

In the proposed method, the non-intersecting iso-parametric curves are computed by applying the boundary conditions on the four sub-boundaries and numerically solving the Laplacian PDE:

$$\frac{\partial^2 T}{\partial p^2} + \frac{\partial^2 T}{\partial q^2} = 0 \qquad p,q \in \Omega \tag{4}$$

where Ω describes the area of the projected points and T is the potential function inside the domain Ω. A mapping from *(p,q)* to *(u,v)* is calculated by applying the non-linear transformation from the domain Ω to the rectangular domain [(0,1),(0,1)]. The domain Ω is enclosed by the four boundary curves Γ_i i=1,..4 on which initial boundary conditions need to be imposed. The PDE equation is solved once for the *u* iso-curves and once in the *v* iso-curves with the following boundary conditions:

For the *u* iso-curves:

$$\begin{aligned}
T(p,q) &= 0 && p,q \in \Gamma_1 \\
T(p,q) &= f_1(L_2(p,q)) && p,q \in \Gamma_2 \\
T(p,q) &= 1 && p,q \in \Gamma_3 \\
T(p,q) &= f_2(L_4(p,q)) && p,q \in \Gamma_4
\end{aligned} \tag{5}$$

and for the *v* iso-curves:

$$\begin{aligned}
T(p,q) &= f_3(L_1(p,q)) && p,q \in \Gamma_1 \\
T(p,q) &= 0 && p,q \in \Gamma_2 \\
T(p,q) &= f_4(L_3(p,q)) && p,q \in \Gamma_3 \\
T(p,q) &= 1 && p,q \in \Gamma_4
\end{aligned} \tag{6}$$

where f_i are monotonically ascending / descending functions in the domain [0,1] that are initially picked as linear functions. L_i are the arc length functions of the sub-boundary curves in the domain Γ_i respectively.

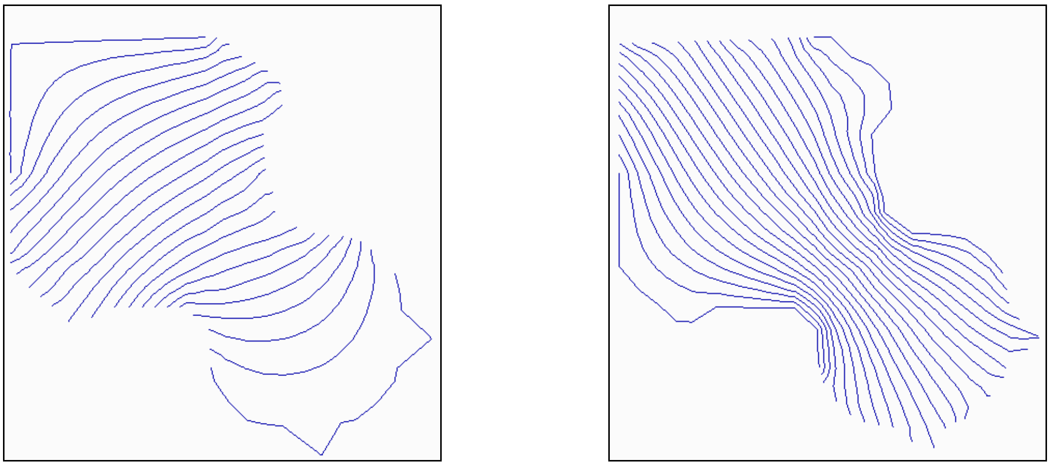
In order to numerically solve the Laplacian PDE, an intermediate grid is created in the domain Ω with node spacing *(Δp, Δq)* in the *p* and *q* directions, respectively. The solution is based on the following discrete finite difference scheme:

$$T_{p+\Delta p,q} + T_{p-\Delta p,q} + T_{p,q+\Delta q} + T_{p,q-\Delta q} - 4T_{p,q} = 0 \quad \forall \quad p,q \in \Omega \tag{7}$$

Writing the scheme for each node leads to a set of linear equations. The right side of the equation set is obtained by substituting boundary values into equation 7. In the proposed method, the Gauss-Seidel iterative method was used to solve the equation set, where each node inside the boundary is assigned the average value of its adjacent nodes. The process converges to an equilibrium state. The method is described in detail in [PRE89]. As a result, each node in the intermediate grid has a solution value. Then, the u and *v* iso-curves of the solution are extracted (figure 6) by applying a 2D version of the Marching Cube Algorithm [LOR87]. Other methods based on neighborhood scanning can be applied.

The main advantage of using the Laplacian PDE is that the parametric grid is non-self-intersecting in the sense that the iso-parametric curves do not split; rather, they start from one side of the boundary and end at the opposite side without crossing each other or the boundary (figure 6).

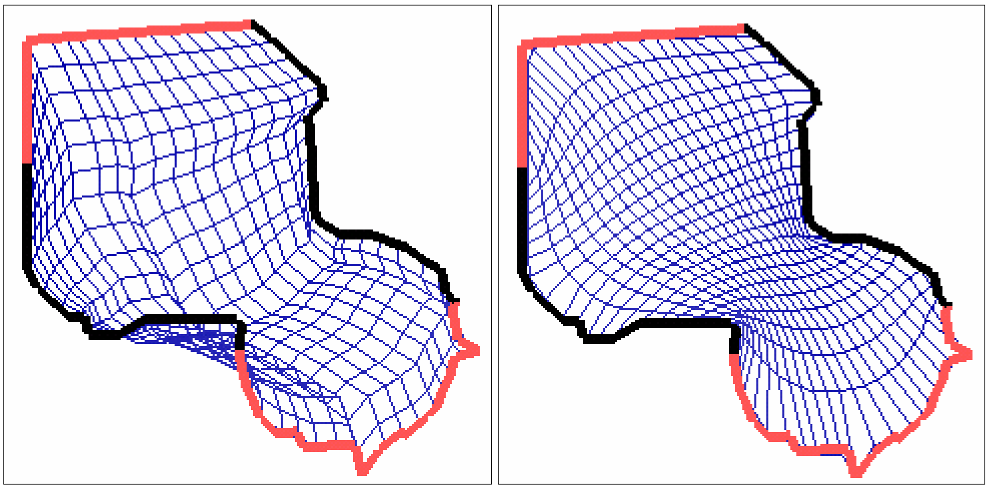
Next, the 2D grid is created by intersecting the *u* and *v* iso-parametric curves, producing a grid of Nu x Nv cells (figure 7). The time complexity of the entire algorithm is O(Np*Nq) per Gauss Seidel iteration. The number of iterations depends on the convergence tolerance.



(a) (b)

Figure 6: The iso-curves of a projected object: (a) iso-u parametric curves. (b) iso-v parametric curves.

איור 6: עקומות שוות פרמטר של גוף מוטל: (a) עקומות שוות פרמטר בכוון u. (b) עקומות שוות פרמטר בכוון v.



(a) (b)

Figure 7: 2D parameterization grid: (a) Self-intersecting grid. (b) PDE non-self intersecting grid.

איור 7: סריג פרמטריזציה דו ממדית (a) סריג עם התנגשות עצמית. (b) סריג ללא התנגשות עצמית שחושב בשיטת PDE.

The PDE parametric grid does not fulfill the fourth criterion of uniform parametric density. That is, the number of points in a cell is not uniformly distributed because boundary sample points, and not internal points, affect the creation of the parametric grid. An adaptive grid correction method can be added, which: (1) changes the f_i functions shapes from linear to non-linear monotonic functions that take into account the number of points per row/column; (2) resolves the PDE with the new boundary conditions. The process is repeated until convergence. Such a correction process was implemented and significantly improved parametric density in most cases. Since the orientation of the grid was not changed, however, the process was limited. On the other hand, the neural network SOM was able to produce uniform parametric density since it determines grid orientation. This method is described in the following section.

3.1.1.1.2 SOM parameterization

SOM grids are based on the self-organizing map proposed by Kohonen in [KOH97]. The advantage of a SOM grid is that all the sampled points, and not only the boundary points, participate in grid creation, producing a grid that is sensitive to sampling density. In this paper, the concept of Self-Organizing Maps (SOM) is proposed as the basis for the neural network SOM parameterization method.

SOM consist of grid nodes represented by neurons. Each neuron contains geometrical information about the node coordinates and neighborhood relations. The entire neural map is trained by the sampled data in random order and is modified into the shape of the projected sampled points. The connections between the neurons do not change during training; instead the neurons get different geometrical values. Implementation of the basic SOM algorithm, which can be found in [KOH97], it is simple and quick. For the parameterization process, however, some modifications were required:

- Each neuron can be either mobile or static.
- Each neuron can be either active or inactive.
- The training function is both bubble and Gaussian in nature (see definition below).

SOM Algorithm:

The SOM algorithm is described as follows:

1. Initialize the position of all the neurons $\{\mathbf{N}(i,j)\ /\ i=0..Nu\ ,\ j=0..Nv\}$. Neuron coordinates are usually set to random numbers within the bounding box of the sampled points.
2. Pick a sample $\mathbf{P}_s$ in random order.
3. Find (i^*,j^*) parametric coordinates of the winner neuron $\mathbf{N}(i^*,j^*)$ among all the *active* neurons. The winner neuron is the closest active neuron to the sample point $\mathbf{P}_s$.
4. Update the position of each *active* and *mobile* neuron in the neighborhood of the winner neuron towards the sample point (see update rules below).
5. Increment *s*; if *s* is smaller than the *run length*, go to step 2, otherwise stop.

The neuron update rules are as follows:

- Projected sampled points $\mathbf{P_s}$ are represented in *(p,q)* coordinates.
- The weight of each sample point is W_s, where 1 is the default.
- Initial neighborhood radius is *R*.
- Initial learning rate is α_0
- *Run length* of the algorithm is *L*.
- The bubble neighborhood of the winner neuron is defined as follows: All the **N**(i,j) that satisfies: $|i-i^*|+|j-j^*| < \frac{R\bullet(L-s)}{L}$.
- The neuron displacement is defined as: $\mathbf{N}(i,j)=\mathbf{N}(i,j)+hc_S\bullet(\mathbf{P_S}-\mathbf{N}(i,j))$ where hc_s is the Gaussian training function: $hc_S=\alpha_S\cdot e^{-\frac{(i-i^*)^2+(j-j^*)^2}{2}}$ and α_s is the learning rate for point *s*: $\alpha_S = 1 - \left(1 - \frac{\alpha_0\bullet(L-s)}{L}\right)^{Ws}$

The time complexity of the algorithm for practical cases is $O(L^*R^2+L^*Nu^*Nv)$.

The SOM algorithm creates a 2D SOM grid based on the projected points. This grid reflects the distribution of the projected sampled points. There is no known proof for preserving uniform 2D parametric density. A proof does exist for the 1D case and can be found in [KOH97].

The grid created by SOM suffers from boundary problems: the boundary of the neural network does not coincide with the boundary of the projected points.

The "*boundary last*" and "*boundary first*" algorithms are proposed in order to cope with the boundary problem.

"Boundary last" method

With the "*boundary last*" method, the SOM boundary neurons are drawn toward the projected points outside the grid by increasing the weight of the sampled points. Thus, the SOM is inflated iteratively like a balloon until very few instances fall outside its boundary. As a result, the boundary points converge to the SOM boundary (figure 19), thus overcoming the boundary problem in the original SOM algorithm. After the "*boundary last*" algorithm has been implemented, a small number of sample points still lie outside the grid; this, however, is later corrected in the parametric optimization stage.

This algorithm has the advantage of detecting the orientation of the grid without user intervention. Moreover, the "*boundary last*" method is suitable for samples with convex boundaries. It is easy to implement, but remains problematic for concave boundaries (figure 19).

"Boundary first" method

With the "*boundary first*" method, the 3D sampled points lying close to the boundary are detected via an edge detection algorithm. The SOM boundary neurons undergo training, while the SOM interior neurons do not participate in this training. The SOM internal neurons are then subsequently trained, while the SOM boundary neurons remain immobile. Figure 20 shows the progress of the algorithm.

With the “*boundary first*” method, the boundary is trained separately from the whole grid. Thus the boundary, with an intrinsic dimension of one, is trained by data having the same intrinsic dimension and is not affected by inner points. This sort of training yields a better boundary than the “*boundary last*” algorithm, especially for concave curves. As with the “boundary last” method, this method independently determines grid orientation and the four grid corner positions.

3.1.1.1.3 3D SOM parameterization

As stated previously, the characteristics of the SOM neural network enable it to capture a 2D topological shape in 3D. This trait is sometimes referred to as dimensionality reduction and it enables a range image to be represented by a rectangular topology grid residing in 3D.

Thus a 3D initial approximation of the surface can be achieved using the SOM method. And together with parametric optimization, this method enables 3D parameterization, and thus the projection stage of 2D parameterization can be omitted. Using this method, merged range images, that cannot be projected onto a plane with no self intersection, can be parameterized.

Moreover, the growing grid algorithm [VAR99,FRI97] offers better convergence to the shape of the object while determining the size of the B-Spline surface control mesh and determining its aspect ratio. Although the algorithm is addressed as an individual neural network in the literature, it is actually an extension to the SOM algorithm described previously. The *growing grid* extension allows the grid size to increase during the SOM training. The changes at the SOM algorithm are as follows:

Growing Grid Extension to the Algorithm

Extensions to the initialization of the SOM algorithm:

1. The initial SOM grid size is defined as n=3, m=3.
2. Every neuron retains a win counter that accumulates the number of times it was chosen as the neuron closest to a sampled point. This number is reset to 0 at initialization.

Extensions applied to the SOM algorithm:

The grid size is increased by adding one row/column to the grid, whenever the number of processed points - s is a λ multiple of the neurons number ($s=\lambda*n*m$), as follows:

1. Find the neuron N_B with the largest number of wins.
2. Find the direct neighbor N_N with the largest number of wins for the neuron N_B
3. If N_B and N_N are on the same row, add a column of neurons between the columns of N_B and N_N:
   - Increase the network size m by 1 and modify the indices of all the neurons accordingly.
   - Interpolate the position of the new column neuron positions with respect to the neighboring columns.
4. If N_N and N_B are on the same column, add a row of neurons between the rows of N_B and N_N:
   - Increase the network size n by 1 and modify the indices of all the neurons accordingly.

- Interpolate the position of the new row neuron positions with respect to the neighboring rows.

5. Reset the win counter of all the neurons to zero and continue the SOM algorithm.

Other modifications of the SOM algorithm are possible. Within this research a few more modifications are used. For example the SOM algorithm can be trained using either a linearly decaying learning rate as described above or using an exponentially decaying learning rate as described in [FRI97]. This modification aids in the process of ordering the grid during training and not as a special detached phase of training. Another modification is to the growing grid algorithm. In [VAR99] a row/column is added using the criterion of maximum number of wins alone, without any geometrical considerations. This modifications was not used, however, in this research.

The boundary problem is much more severe in 3D than in 2D. The modifications used for the 2D case are not suitable since the boundary cannot be detected in 3D as easily as in 2D. Another modification to the SOM algorithm that involves increasing the weights of sampled points outside the grid can solve this problem [BAR3].

3.1.1.1.4 Sampled point parameterization.

After a parameterization grid is constructed, parametric values can be assigned to the sampled points. If the grid was created using 3D parameterization then it can be used as the control mesh for the B-Spline surface. In this case, 3D base surface parameterization can be performed and it is described in section 3.1.1.2. However, if a 2D parameterization grid was constructed by the PDE or SOM methods then point parameterization is calculated using the method described below.

The sampled points can be parameterized by assigning (*u,v*) parametric values to each sampled point (*x,y,z*) as follows:

1. Each point $\mathbf{P_s}$=(x,y,z) from the N sampled points is projected on the parametric grid plane to (X,Y) and its cell (i,j) is detected.
2. Each cell (i,j) is represented as a 2D Coons parametric surface. The value of (*u,v*) is determined by the relative location of (X,Y) on the grid cell (i,j) using the equations 8-10.

The time complexity of the entire parameterization stage after grid creation is: O(N*Nu*Nv)

Equations 8-10 calculate the parameterization of a projection point (X,Y) that falls in a cell (i,j) on the parameterization grid (figure 8). The corners of the grid cell and the projected sampled point (X,Y) are given in the (p,q) coordinate system of the projection plane. Again, Nu and Nv represent the size of the parameterization grid in the *u* and *v* parametric directions respectively. A quadratic equation is solved (equation 10) in order to calculate the Δu and Δv values that represent the relative position of the sampled point (X,Y) inside the grid cell (i,j). Two solutions exist, one is inside the grid cell while the other is outside the cell. The solution that falls within the grid cell is retained.

If the grid cell is concave, the results obtained by the formula is ambiguous. In that case, the sampled point is not parameterized at this stage. Instead it will be parameterized in the parametric optimization stage described in the next section.

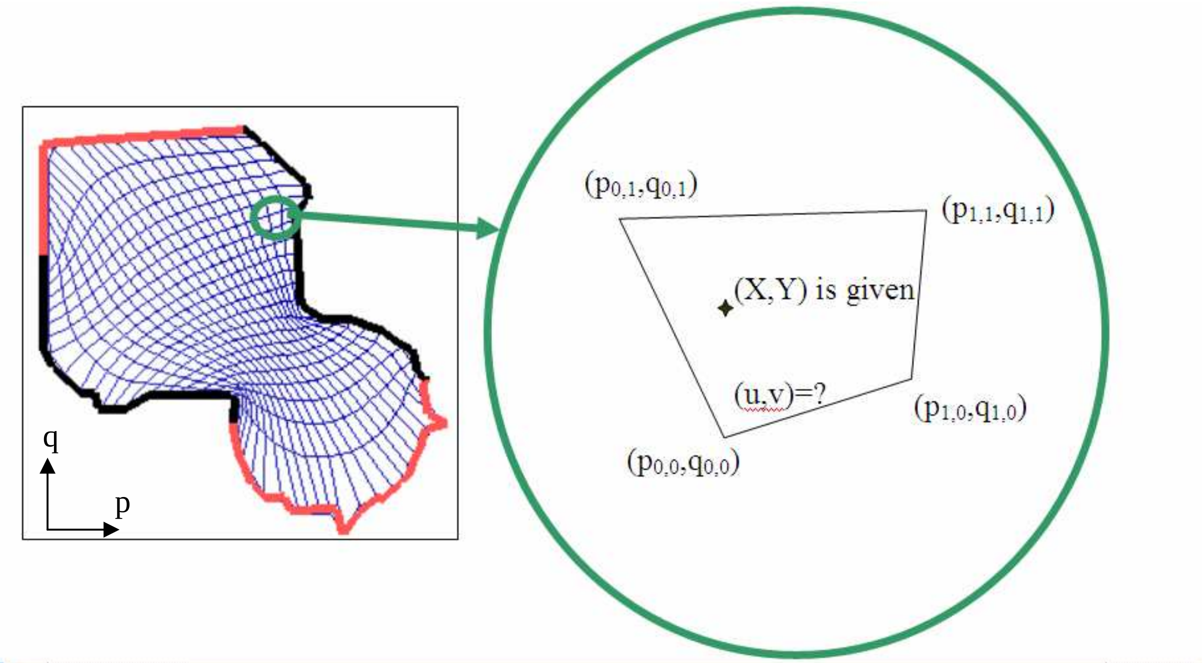


Figure 8: Parameterization within a single parameterization grid cell. Point (X,Y) is the projected sampled point in the (p,q) coordinate system.

איור 8: פרמטריזציה בתוך תא יחיד בסריג הפרמטריזציה. נקודה (X,Y) היא נקודת הדגימה המוטלת
בקואורדינטות (p,q).

$$u = \frac{i+\Delta u}{Nu} \quad i = 0 \ldots Nu-1$$
$$v = \frac{j+\Delta v}{Nv} \quad j = 0 \ldots Nv-1 \tag{8}$$

$$\Delta u = -\frac{p_{0,0} - \Delta v p_{0,0} + \Delta v p_{0,1} - X}{-p_{0,0} + p_{1,0} + \Delta v p_{0,0} - \Delta v p_{1,0} - \Delta v p_{0,1} + \Delta v p_{1,1}} \tag{9}$$

$$A\Delta v^2 + B\Delta v + C = 0 \qquad where:$$

$$\begin{cases} A = q_{0,0}p_{1,0} - q_{0,0}p_{1,1} - q_{1,0}p_{0,0} + q_{1,0}p_{0,1} - q_{0,1}p_{1,0} + q_{0,1}p_{1,1} + q_{1,1}p_{0,0} - q_{1,1}p_{0,1} \\ B = q_{0,0}p_{1,1} + 2q_{1,0}p_{0,0} - 2q_{0,0}p_{1,0} - q_{1,1}p_{0,0} + q_{1,1}X - Yp_{0,0} + Yp_{1,0} + \\ \qquad + Yx_{0,1} + q_{0,0}X - y_{1,0}X - q_{1,0}p_{0,1} + q_{0,1}p_{1,0} - q_{0,1}X - Yp_{1,1} \\ C = q_{0,0}p_{1,0} - q_{0,0}X - q_{1,0}p_{0,0} + q_{1,0}X - Yp_{1,0} + Yp_{0,0} \end{cases} \tag{10}$$

3.1.1.2 Parametric optimization

The alternating method [HOS89, MA95] was chosen based on its convergence characteristics to optimize the parametric values of the sampled points. In our approach, the initial approximation of the 3D base surface is calculated by surface fitting (section 3.1.2). The role of 3D base surface parameterization is to correct the initial parameterization. This correction accomplishes the following: (1) the parameter values of sampled points are corrected and the fitted surface is improved, and (2) parameters are assigned to sampled points that were discarded during the 2D parameterization process due to their position beyond the grid boundary.

Once an approximated surface has been computed (section 3.1.2), it can be used as the new parametric base surface for another iteration. This is accomplished by projecting the 3D sample points onto the base surface, as follows:

1. The initial point on the base surface closest to the sampled point is found by discretely searching the base surface parametric domain in Δu, Δv intervals. The (u,v) parameters of the closest point are assigned to the sampled point and retained for the next step.
2. These parameters are corrected iteratively. Equation 11 is used to calculate the change in the parameter values up to a given tolerance or until the number of iterations exceeds a given limit, implying diversion or very slow convergence. The development of the equation can be found in [PIE97].

$$\begin{bmatrix}\Delta u\\ \Delta v\end{bmatrix}=\begin{bmatrix}\left|\mathbf{S_u}\right|^2+\mathbf{r}\bullet\mathbf{S_{uu}} & \mathbf{S_u}\mathbf{S_v}+\mathbf{r}\bullet\mathbf{S_{uv}}\\ \mathbf{S_u}\bullet\mathbf{S_v}+\mathbf{r}\bullet\mathbf{S_{vu}} & \left|\mathbf{S_v}\right|^2+\mathbf{r}\bullet\mathbf{S_{vv}}\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{r}\bullet\mathbf{S_u}\\ \mathbf{r}\bullet\mathbf{S_v}\end{bmatrix} \tag{11}$$

where :

- $\mathbf{S_u}, \mathbf{S_v}, \mathbf{S_{uu}}, \mathbf{S_{vv}}, \mathbf{S_{uv}}$ – The derivatives of the B-spline surface at the parameters *(u,v).*
- $\mathbf{r}$ – The distance of the sample point from the point on the surface that is: $\mathbf{r}=\mathbf{S}-\mathbf{P_s}$ at parameters *(u,v).* Step 1 may be omitted, and the original parametric values of the point can be used as an initialization for step 2. In this work, however, we found that step 1 improves results even though the process is very slow.

After the parametric correction has been applied, a new surface is fitted to the reparameterized points. The process is then repeated, usually converging adaptively to the desired surface. The final surface satisfies convergence tolerance while capturing the original shape of the sampled surface [PIE97]. The complexity of one iteration of the reparameterization stage is given by O(N*n*m), if the degree of the B-Spline surface is considered a constant.

3.1.2 Surface Fitting

After the parameterization stage, the neighborhood relations between the points are known. The goal of the fitting process is to find the best approximation with respect to the sampled points. In this work, B-Spline parametric surfaces are used as the geometric representation due to their natural compatibility in CAGD. B-Spline surfaces are preferred due to their intuitive control, smoothness, subdivision characteristics and local support. These surfaces have the following parameters: mesh size, control points, degree of basis functions and knot vectors. In this work, mesh size, degree, order and knot vectors of the surface are defined apriori. The surface fitting algorithm calculated the *(x,y,z)* coordinates of the control points.

Using the above definitions the surface-fitting problem can be formulated as a linear over-constraint set of equations. However, traditional direct methods like pseudo inverse and SVD were found unsuitable for practical reasons. Instead, iterative methods were developed that require a fast initial approximation.

Three algorithms are proposed for the surface fitting process:

(1) Control Mesh Approximation (CMA), which creates a fast initial approximation and is described in detail in section 3.1.2.1.

(2) Gradient Descent Algorithm (GDA), which is an iterative LSQ technique. The technique is also referred to in the literature as steepest descent. The method formulated for B-Spline surfaces is presented in detail section 3.1.2.2.

(3) Random Surface Error Correction (RSEC), which is based on the ideas of the SOM method. Modifications to the SOM algorithms were implemented so as to conform with the formulation of B-Spline surfaces. The method has no formal proof, however it has succeeded in practice as can be seen in section 3.1.2.3.

3.1.2.1 Control Mesh Approximation (CMA)

Once the scattered measured points have been parameterized (section 3.1.1), the surface is fitted to the data points by iterative methods. These methods are faster than non-iterative LSQ because they utilize the B-Spline local support characteristic, and they are less prone to numerical problems. An initial approximation of the control polygon created by the CMA algorithm was implemented in order to shorten the fitting time. This approximation is not as accurate as was desired, but provides good initial conditions for the subsequent correction algorithms (GDA or RSEC).

The control polygon of the approximated B-Spline surface is calculated as follows:

1. A 2D n x m grid is created in the parametric domain u,v (not to be confused with the size of the parametric grid).
2. For each 3D point $\mathbf{P}_s$ (s=1..N) with normalized parameterization (*u*,*v*), a cell (i,j) is determined according to the following mapping: $(i-1)/n < u < i/n$ and $(j-1)/m < v < j/m$
3. The control point V_{ij} is calculated as the average of all the points $\mathbf{P}_s$ that belong to the same cell (i,j).
4. Given the control polygon and a knot vector, the surface is determined (equation 2).

Note that the surface does not interpolate the control polygon or the sampled points. However, the convex hull property guarantees convergence of the control mesh to the surface as control mesh density increases. This comes from the B-Spline characteristic in which the control-polygon is converged to the surface points during refinement. The maximum density of the control mesh is limited to the density of the sampled points. In addition, the approximation was improved by modifying the position of the control points. In this paper, two iterative control mesh correction methods were implemented GDA and RSEC and are described in the following sections.

3.1.2.2 Gradient Descent Algorithm (GDA)

The GDA is a correction algorithm that was developed in order to overcome the inaccuracies in the CMA algorithm. It also corrects the boundary of the surface where the error can grow. This algorithm is based on an iterative LSQ technique. The surface error is defined in equation 12. The error for each control point $\mathbf{V_{ij}}$ is reduced by changing $\mathbf{V_{ij}}$ in small steps, in the direction of the error gradient:

1. For a point $\mathbf{P_s}$ with parameters *(u_s.v_s)*, s=1..N :

   Evaluate the distance of the point $\mathbf{P_s}$ from the surface $S(u_s.v_s)$:

$$\mathbf{Err_s} = \mathbf{P_s} - \sum_{i=1}^{n}\sum_{j=1}^{m} B_i(u_s)\cdot B_j(v_s)\cdot \mathbf{V_{ij}} \qquad (12)$$

2. Correct the surface control points as follows:

$$\mathbf{V_{ij}} = \mathbf{V_{ij}} + \eta\sum_{s=1}^{N} B_i(u_s)\cdot B_j(v_s)\mathbf{Err_s} \qquad (13)$$

   where η determines the step size in the error gradient direction. The step size η can be controlled and usually determines the accuracy of the generated surface. In our examples, it was chosen empirically as $O(10^{-3}\text{-}10^{-5})$, about 5 times greater than the convergence tolerance - ε.

3. Step 2 is iterated until the change in all $\mathbf{V_{ij}}$ is smaller than a convergence tolerance ε.
4. The control points are reparameterized by projecting them onto a base surface (section 3.1.1.2). The error is measured, and stage 2 is repeated until the global error converges to a given convergence tolerance - σ.

For computation simplification, each component *x(u,v)*, *y(u,v)* and *z(u,v)* of the B-Spline is fitted separately. The value of η should be sufficiently small in order to converge to a local minimum and sufficiently large to convergence fast.

The GDA can be efficiently implemented, while using the local support property of B-Splines, with a complexity of $O(N^*k^2)$ per iteration, where *k* is the order of the B-Spline surface and N is the number of sampled points.

3.1.2.3 Random Surface Error Correction (RSEC)

In order to improve the control mesh of the surface, another algorithm (RSEC) is proposed. In this algorithm, correction is according to a local error rather than a global one. The random concept is again based on the SOM neural network. In this algorithm, the B-Spline surface is considered as a neural network, and the control points are the neurons, but instead of a winner neuron, we use a winner parameter, and instead of a Gaussian neighborhood training function, we use the B-Spline basis functions for control point position update.

The algorithm is based on the following stages:

1. Re-order the sampled points randomly. Without reordering the sample points randomly, the algorithm will not converge to the desired surface.
2. For a point $\mathbf{P_s}$ with parameters $(u_s.v_s)$, s=1..N :

   a. Evaluate the distance of the point $\mathbf{P_s}$ from the surface $\mathbf{S}(u_s.v_s)$:

$$\mathbf{Err_s} = \mathbf{P_s} - \sum_{i=1}^{n}\sum_{j=1}^{m} B_i(u_s)\cdot B_j(v_s)\cdot \mathbf{V_{ij}} \qquad (14)$$

   b. Correct the surface control points as follows:

$$\mathbf{V_{ij}} = \mathbf{V_{ij}} + \eta B_i(u_s)\cdot B_j(v_s)\mathbf{Err}_s \qquad (15)$$

3. The control points are reparameterized by projecting them onto a base surface (section 3.1.1.2). The error is measured, and stage 2 is repeated until the global error converges to a given convergence tolerance σ.

Figure 9 displays the effect of a single sample point on a B-Spline curve control polygon. The complexity of using the RSEC algorithm is O(N*n+N*m), if the degree of the B-Spline surface is considered as a constant due to the local support property of B-Splines. Although the algorithm was successful in practice as can be seen at section 4.1.2 there is no proof of its convergence.

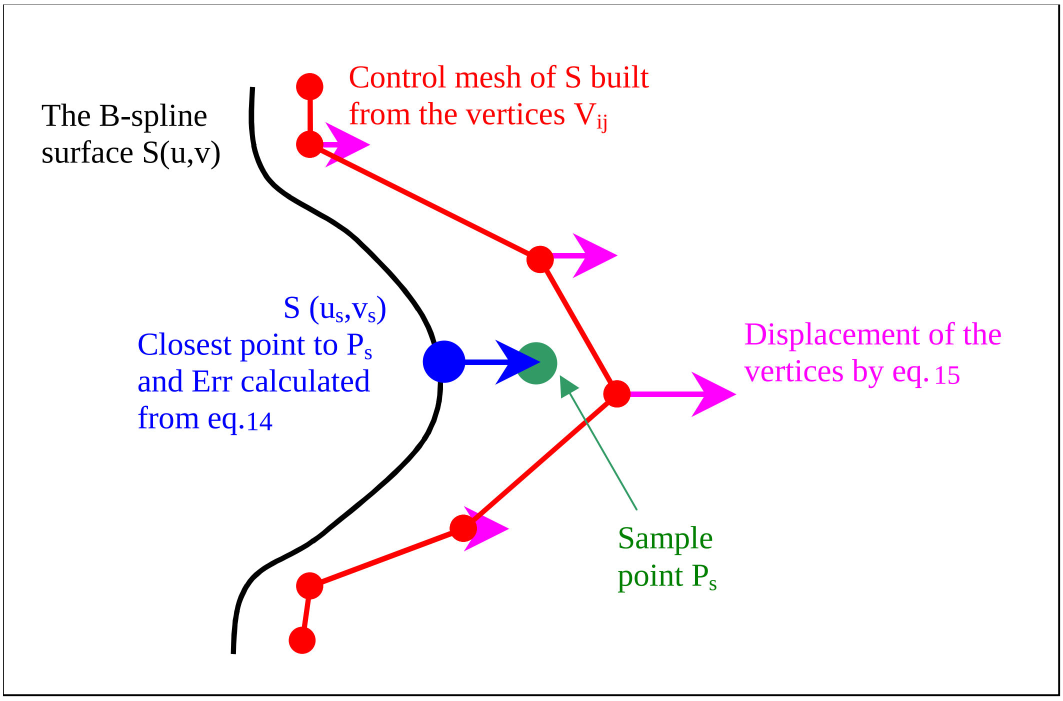

The B-spline surface S(u,v)
Control mesh of S built from the vertices V_{ij}
$S(u_s,v_s)$
Closest point to P_s and Err calculated from eq.14
Displacement of the vertices by eq. 15
Sample point P_s

Figure 9: The effect of a single sampled point on the control mesh of a B-Spline surface iso-curve.

איור 9: ההשפעה של נקודת דגימה בודדת על סריג נקודות הבקרה של עקומה פרמטרית השייכת למשטח
.B-Spline

3.1.3 Merging surfaces from multi range images

Previous stages created B-Spline surfaces representing the object from different views. Surface merging is based on trimming the surfaces at a common curve and then creating a reconstructed 3D envelope from trimmed surfaces. This approach enables adaptive reconstruction from multi range images. The local merging of a new surface to the 3D model has significantly lower computational complexity than global fitting methods. Surface merging consists of the following main stages:

1. Detecting 3D proximity between 3D surfaces.
2. Creating distinction curves between surface pairs.
3. Trimming the surfaces.

3.1.3.1 Proximity Detection

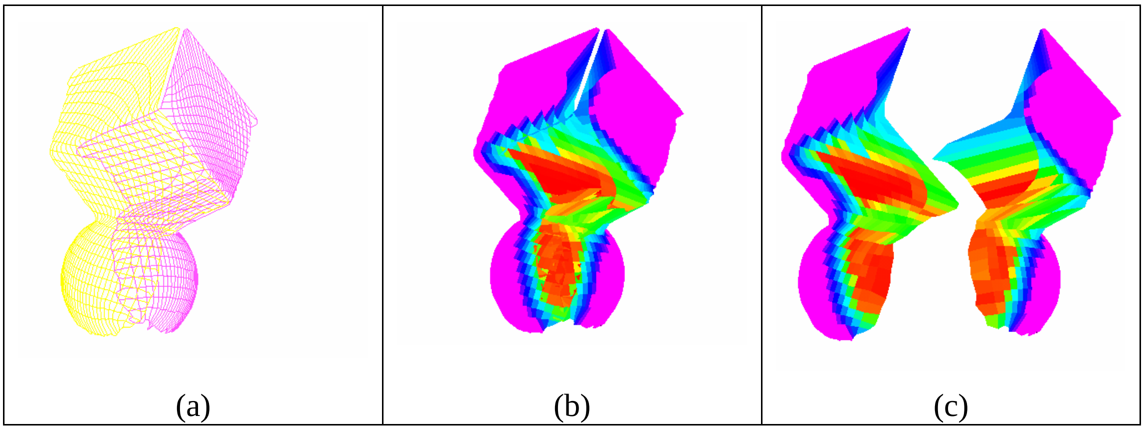
For two surfaces, a distance map is created in each surface parametric domain. Overlapping areas are detected on the distance map. In our case, the surfaces are almost overlapping, since they are reconstructed from overlapping images and therefore only the distance map at the boundary regions is calculated (figure 10).

3.1.3.2 Distinction Curve Creation

The overlapping area of any two surfaces can be divided by various types of distinction curves (figure 11). A distinction curve is used as a trimming curve for each surface. A distinction curve separates the parametric domain of any two surfaces A,B to two distinct parametric domain areas (Ab,Ba) where (1) $A\cup B=Ab\cup Ba$; and (2) $Ab\cap Ba=\phi$. For simplicity, we trim only one surface at the other surface boundary (figure 11-b). Other approaches were applicable but had integration problems in the case of more than two surfaces. It is important to note that discontinuity anomalies remain between both surfaces due to fitting errors.

3.1.3.3 Surface trimming

Previous stages of proximity detection and distinction curve creation are used on surface pairs. However, The boundary of a volumetric object can consist of more then two surfaces. For three surfaces A, B, C, the following trimmed surfaces are created by intersecting untrimmed parametric domains: $Ab\cap Ac$, $Ba\cap Bc$, $Ca\cap Cb$. This can be extended for any number of surfaces, as long as there is no overlap area among three or more surfaces ($A\cap B\cap C=\phi$). To avoid the problem of overlapping area among three or more surfaces, the distinction curve is created using the method described in the previous section. Results obtained using the above integration method are demonstrated in section 4.1.3.



(a) (b) (c)

Figure 10: proximity between surfaces. (a) overlapping area; (b) distance maps; (c) proximity of each surface with respect to the other.

איור 10: זיהוי קרבה בין משטחים (a) איזור חופף (b) מפת המרחקים (c) הקרבה של כל משטח כלפי המשטח השני.

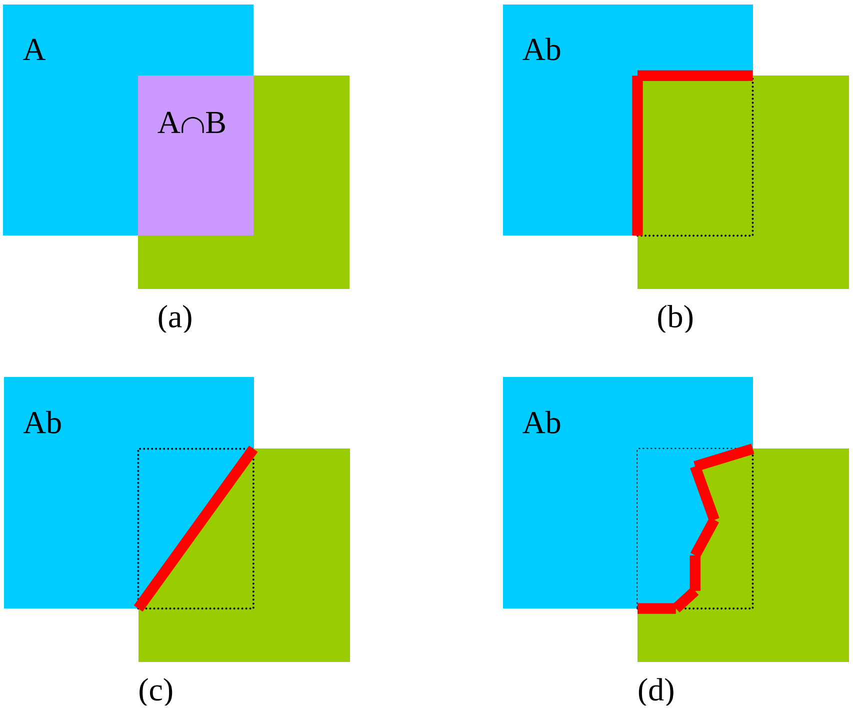


Figure 11: (a) two simple flat surfaces A and B along with the overlapping area A∩B. (b-d) various types of distinction curves.

איור 11: (a) שני משטחים שטוחים A,B יחד עם איזור החפיפה A∩B. (b-d) מספר דוגמאות לעקומות הפרדה.

3.2 Approach B – reconstruction of a triangular mesh from a cloud of points

Reconstruction of a triangular mesh from a cloud of points is composed of two main stages. The first stage employs an *extended neural gas algorithm* that its implementation is discussed in section 3.2.1. The second stage creates a manifold mesh and its implementation is discussed in section 3.2.2.

The *extended neural gas* algorithm is novel in the way it detects the topology of point cloud. The topological issue is elaborated in section 3.2.3.

3.2.1 Extended neural gas

The *neural gas* algorithm creates an mesh that approximates the cloud of points. The edges of the approximating mesh are a subset of the *Delaunay triangulation* as shown in [MAR94]. However, the mesh is not well defined in the sense that it is composed only of vertices and edges, while faces are not defined. In this section we present an extension to the *neural gas* algorithm that defines faces. Moreover, in this extension the normals are evaluated using the map structure.

We denote the sampled points $\mathbf{p}_s \in P$ where $s = 1..n$ and n is the number of sampled points of the object boundary denoted by S. The approximating mesh created by the *neural gas* method is denoted by M and is composed of *vertices* $V = \{\mathbf{v_i} \mid \mathbf{v_i} \in R^3\}$, edges $E = \{\mathbf{e_i} = (\mathbf{v_i}, \mathbf{v_j}) \mid i \neq j \wedge \mathbf{v_i}, \mathbf{v_j} \in V\}$ and faces $F = \{\mathbf{f_i} = (\mathbf{e_i}, \mathbf{e_j}, \mathbf{e_k}) \mid i \neq j \wedge i \neq k \wedge j \neq k \wedge \mathbf{e_i}, \mathbf{e_j}, \mathbf{e_k} \in E\}$, where $M = \{V, E, F\}$. Each *neural unit* holds the position of a mesh vertex and connectivity information to other *neural units*.

Before the learning process begins, the sampled points are reordered randomly so that they will be presented in random order to the learning algorithm. During the learning process, the neural units are ordered according to their distance from the current sampled point $\mathbf{p}_s$. The closest *neural unit* is called the *winner neural unit*, and its index is denoted here by s_0; the second closest *neural unit* will be denoted by s_1 (winner of degree 2); and so on. That is $\|\mathbf{p}_s - \mathbf{v}_{s_0}\| < \|\mathbf{p}_s - \mathbf{v}_{s_1}\| < \cdots < \|\mathbf{p}_s - \mathbf{v}_{s_k}\|$ where $k = 0..|V| - 1$. It is also said that sampled point $\mathbf{p}_s$ is classified to its *winner neural unit* $\mathbf{v}_{s_0}$.

The *neural gas* learning process starts with the *neural unit* positioned randomly in R^n. During the learning process, the *neural unit*s change their position in order to minimize the *quantization error*. The learning process is competitive in nature since the neural units compete for positions in space that are close to the current sampled point. The problem of positioning the *sites* is not addressed here. Rather, the topological face creation extension is presented by the extension of the *Hebbian learning law*.

The *Hebbian learning law* was first suggested in the neurobiological context and was adapted and extended for use in artificial neural networks [HAY94]. Its interpretation in the *neural gas* neural network (see [MAR94]) controls the creation of edges between neural units in the following manner: *Neural units* are connected with an edge if they are excited together meaning that they are closest to a sampled point. The edges decay and disappear if the neural units are not excited for a while during the training process.

The interpretation of the *Hebbian learning law* in the *neural gas algorithm* [MAR94] can be formalized geometrically: An edge forms between two *neural units* if a sampled point exists such that those two *neural units* are the *winner* and *second winner* for that sampled point.

The newly developed *extended Hebbian learning law* can be formalized as follows: A face is formed to connect three neural units if a sampled point exists such that those three *neural units* are the *winner* and *second winner* and *third winner* of that sampled point and if three edges exist connecting those three *neural units*.

The extension to the *Hebbian learning law* for faces enables calculation of the approximated normals to the surface. The same mechanism that approximates the positions of neural units can be used to approximate the normals. For that,, the normal to the most recently created triangle face $\mathbf{n}^*$ is considered as an input vector to the neural network. Normalization of the normal vectors is performed during the learning process to keep the normal size as a unit. The orientation of the normal is chosen to be in the same direction as the sampled point $\mathbf{p}_\mathbf{s}$ from its *winner neural unit* $\mathbf{v}_{\mathbf{s}_0}$.

Figure 12 presents the *extended neural gas* algorithm emphasizing the extensions. The layout of the algorithm can be expressed in words in the following manner:

Extended neural gas algorithm stages

a. Specify the number of desired neural units $|V|$ and training parameters $\varepsilon_i, \varepsilon_f \lambda_i, \lambda_f, a_i, a_f, L$.

b. Initialize neural units positions $\{\mathbf{v}_k\}$and normals $\{\mathbf{n}_k\}$ to random values.

c. Repeat the following steps for L iterations:

   1. Randomly pick a sampled point $\mathbf{p}_\mathbf{s}$
   2. Order the neural units $\{\mathbf{v}_k\}$ by increasing distance from the sampled point$\mathbf{p}_\mathbf{s}$: $\mathbf{v}_{\mathbf{s}_0}, \mathbf{v}_{\mathbf{s}_1}, \mathbf{v}_{\mathbf{s}_2} ... \mathbf{v}_{\mathbf{s}_{|V|}}$
   3. Connect the two nearest neural units with an edge $\mathbf{e}_1^*$.
   4. If all three nearest neural units $\mathbf{v}_{\mathbf{s}_0}, \mathbf{v}_{\mathbf{s}_1}, \mathbf{v}_{\mathbf{s}_2}$ are connected, create a face $\mathbf{f}^*$.
   5. Adapt the positions of all the neural units $\mathbf{v}_{\mathbf{s}_k}$ towards the sampled point $\mathbf{p}_\mathbf{s}$.
   6. If a face was created, adapt and normalize the normals $\mathbf{n}_{\mathbf{s}_\mathbf{k}}$ towards the face normal $\mathbf{n}^*$.
   7. increase the age a_k of all the edges $\{\mathbf{e}_k\}$and delete “old” edges and faces that their edge is higher than $a_{\max}$.

```
Procedure $ExtendedNeuralGas(|V|, S, \varepsilon_i, \varepsilon_f \lambda_i, \lambda_f, a_i, a_f, L)$ {
        $E \leftarrow \phi, F \leftarrow \phi$
        $V \leftarrow CreateRandomSitesInBoundingBox(|V|, S)$
        $S \leftarrow OrderRandomly(S)$
        For ( $s$ =0 To $L$ ){
                $V \leftarrow SortByWinners(V, \mathbf{p_s})$   //sort all neurons by increasing distance from $\mathbf{p_s}$
                // create or refresh the edge between the two closest winners
                $\mathbf{e_1^*} = (\mathbf{v_{s_0}}, \mathbf{v_{s_1}}), E \leftarrow \mathbf{e_1^*} \cup E$
                // check if the two other edges exist in the mesh. if so, a face is created
                $\mathbf{e_2^*} = (\mathbf{v_{s_0}}, \mathbf{v_{s_2}}), \mathbf{e_3^*} = (\mathbf{v_{s_1}}, \mathbf{v_{s_2}})$
                If ( $\mathbf{e_2^*} \in E \wedge \mathbf{e_3^*} \in E$ ) Then{
                        $\mathbf{f}^* = (\mathbf{e_1^*}, \mathbf{e_2^*}, \mathbf{e_3^*}), F \leftarrow \mathbf{f}^* \cup F$
                        $\mathbf{n}^* \leftarrow (\mathbf{v_{s_1}} - \mathbf{v_{s_0}}) x (\mathbf{v_{s_2}} - \mathbf{v_{s_0}})$   // normal to the face is calculated
                        $\mathbf{n}^* \leftarrow \mathbf{n}^* / \|\mathbf{n}^*\|$ // normalize
                        $d^* = \mathbf{n}^* \bullet (\mathbf{p_s} - \mathbf{v_{s_0}})$
                        If ( $d^* < 0$ ) Then $\mathbf{n}^* \leftarrow -\mathbf{n}^*$ // set direction to the sampled point
                }
                Else // if a face was not created, no normal is calculated
                        $\mathbf{n}^* \leftarrow \mathbf{0}$
                //learning parameters are set using exponential decay
                $\varepsilon = \varepsilon_i (\varepsilon_i / \varepsilon_f)^s$ , $\sigma = \sigma_i (\sigma_i / \sigma_f)^s, a_{\max} = a_i (a_i / a_f)^s$
                For ( $k = 0$ To $|V| - 1$ ){ //adapt positions of each site and each normal
                        $\mathbf{v_{s_k}} \leftarrow \mathbf{v_{s_k}} + \varepsilon(\mathbf{p_s} - \mathbf{v_{s_k}}) e^{(-k/\lambda)}$          // adapt site position
                        $\mathbf{n_{s_k}} \leftarrow \mathbf{n_{s_k}} + \varepsilon(\mathbf{n}^* - \mathbf{n_{s_k}}) e^{(-k/\lambda)}$          // adapt site normal
                        $\mathbf{n_{s_k}} \leftarrow \mathbf{n_{s_k}} / \|\mathbf{n_{s_k}}\|$
                }
                For ( $k = 1$ to $|E|$ ){ // refresh and delete old edges
                        If ( $\mathbf{v_{s_0}} \in \mathbf{e}_k$ ) Then $a_k \leftarrow 1$      // reset the “age” of the edge
                                Else $a_k \leftarrow a_k + 1$       // otherwise age edges
                        If ( $a_k > a_{\max}$ ) Then { //delete old edges and faces connected to them
                                $F \leftarrow F - \{f \mid \mathbf{e}_k \in f\}, E \leftarrow E - \mathbf{e}_k$
                        }
                }
        }
        Return $(V, E, F)$  //return the created mesh
}
```

Extensions to original neural gas algorithm

Figure 12: The *extended neural gas* algorithm, extensions are marked in the figure.

איור 12: אלגוריתם ה-*extended neural gas* ההרחבות מסומנות באיור.

3.2.2 Manifold Mesh Creation

The *extended neural gas* algorithm creates a mesh that approximates the surface. This mesh is composed solely of a subset of the Delaunay triangles and edges. This boundary representation is not a manifold surface, since holes exist in it and the faces are not oriented in the same direction. A non-manifold mesh is not usable as a CAD model.

In a manifold surface without a boundary, each *regular edge* has two coincident faces and each *regular vertex* has the same number of faces and edges incident to it. If a manifold surface has a boundary, a *boundary edge* has only one coincident face. In a *boundary vertex,* the number of faces attached to it is one less than the number of edges emanating from it.

The anomalies in the *extended neural gas* mesh consist of: (1) edges coincident to three or more faces; (2) edges not connected to any face; (3) vertices that are neither *boundary vertices* nor *regular vertices* of a manifold; (4) faces not oriented in the same direction. The manifold-creation algorithm described in this section solves these anomalies using a projection technique similar to [MEE00]. The manifold mesh creation algorithm is greedy in nature and is not optimal. It is presented here to show that a final mesh can be reconstructed by filling in the missing information.

Mesh creation consists of the following stages:

Stage 1: Eliminating undesirable faces.

Stage 2: Face reorientation.

Stage 3: Completing missing faces using local projection.

Stage 4: Detecting and closing boundary loops.

3.2.2.1 Eliminating undesirable faces

The purpose of the first stage is to eliminate faces that do not represent the volume boundary. Examples for such faces are displayed in figure 13. Faces that are candidates for elimination are characterized by:

1. Being incident to more than one face over the same edge, i.e. , the surface splits at that edge.
2. Having a small dihedral angle with another face, i.e. , the surface folds over that common edge.

Based upon these two rules, the edges are traversed. For each edge that has more than two faces connected to it, all faces are eliminated except from the two faces that have the maximal dihedral angle between them. Next, the dihedral angle between the two remaining faces is checked against a given tolerance. If the angle is small, one of the faces is deleted. The criterion for elimination is minimum connectivity to other faces; thus the face connected via edges to fewer faces is eliminated. If the faces have the same connectivity to other faces, theorem 3 from [AME99] is chosen as an elimination criterion. Thus, the face with the highest dot product of the face normal and the vertex normal (calculated using the *extended neural gas* algorithm) at its maximal angle vertex is eliminated.

Before reconstructing missing faces, those faces that will cause problems in local projection are also deleted (faces that will cause local surface folds). The vertices are traversed and the faces incident to them are projected onto a plane in the direction of the vertex normal. If two faces overlap on the plane, the older face created in the *extended neural gas* algorithm is eliminated.

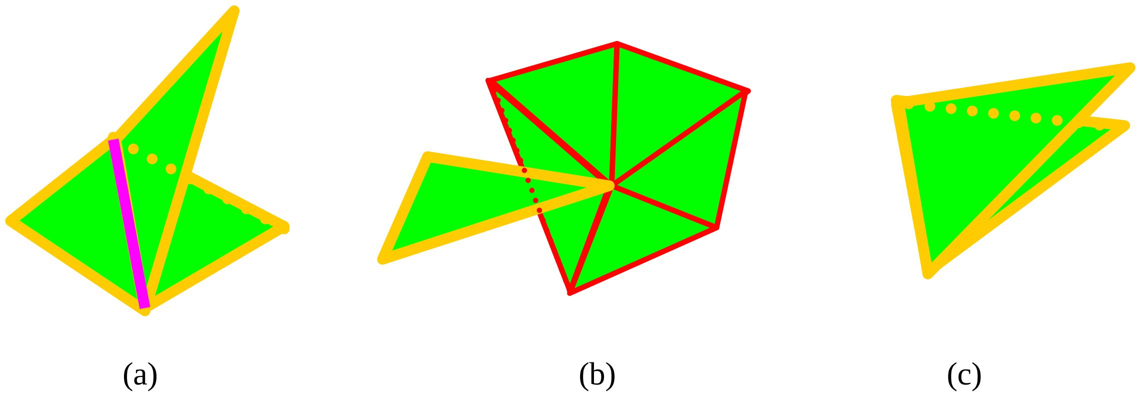


Figure 13: Undesirable face formations created by the extended neural gas algorithm: (a) More then two faces connected by an edge. (b) A face connected to a vertex with a triangle fan around it. (c) A fold created by two faces with a small dihedral angle.
איור 13: מבנים לא רצויים של פאות שנוצרו על ידי אלגוריתם ה extended neural gas. (a) יותר משתי פאות
המחוברות על ידי צלע אחת. (b) פאה מחוברת לקודקוד עם מניפת משולשים מסביבו. (c) קפל הנוצר ע"י שתי
פאות עם זוית קטנה בניהן.

3.2.2.2 Face re-orientation

The resulting mesh is reoriented in the same direction by traversing the faces and orienting all faces connected by an edge to the same direction (clockwise/counterclockwise orientation of vertices defines the front/back of the face). The face normals are flipped to agree with the front side of the face (figure 14). The orientation procedure separates the mesh to one or more surface patches not connected by an edge, where each patch is oriented in the same direction.

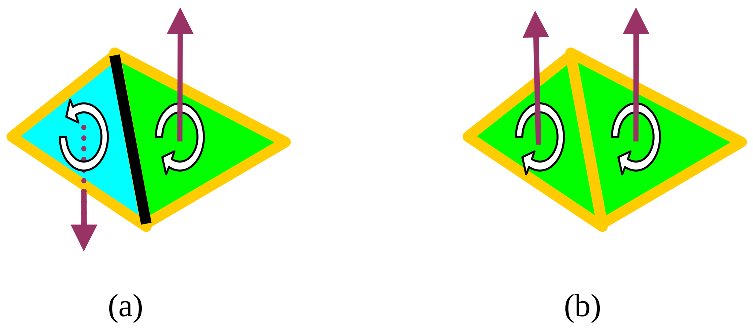


Figure 14: Face re-orientation. (a) Faces and normals not oriented in the same direction. (b) Faces and normals after re-orientation.
איור 14: כיוון מחדש של הפאות (a) פאות ונורמלים לא מכוונים בכוון אחיד (b) פאות ונורמלים לאחר כיוון
מחדש.

3.2.2.3 Completing missing faces using local projection

The normal information at the mesh vertices can now be used to define the local projection direction. The main idea is that if the edges emanating from the vertex are projected to a plane, they can be ordered by increasing angle, and this order defines the triangular face connectivity around the vertex.

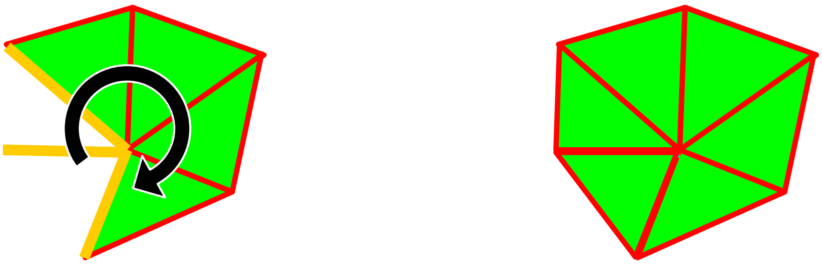
At this stage, the vertices are traversed, and a fan of triangles is constructed around each vertex. When a triangle fan is built, missing edges are added to the mesh. Also, new triangular faces are not allowed to overlap any other triangles already in the mesh.

Due to the greedy nature of the algorithm, the following precautions are taken during reconstruction.

1. The normal of each triangle to be created is checked against the normals of its vertices, and if a certain threshold of agreement is not passed, a triangle is not created.
2. The created face will not fold over one of its edges.
3. The created face will be able to be oriented with all surface patches it connects.

The order of traversing the vertices is defined by a confidence criterion of the vertex normal and the edge directions. The traversal is repeated several times and each time the threshold for normal agreement is broadened.

At the end of this stage, the surface created is a manifold surface with or without a boundary (figure 15).



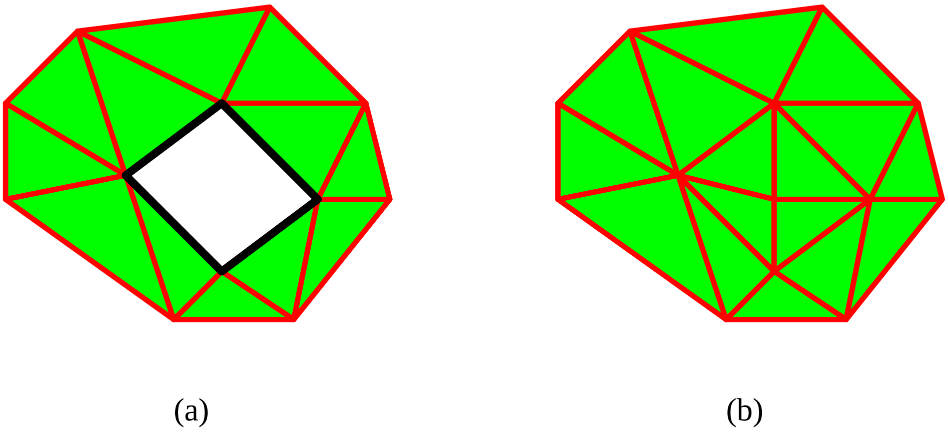
(a) (b)

Figure 15: The local projection of a mesh: (a) before completing missing faces (b) after completing missing face.

איור 15: הטלה מקומית של רשת (a) לפני השלמת פאות חסרות (b) לאחר השלמת פאות חסרות.

3.2.2.4 Detecting and closing boundary loops

This stage closes boundary loops in the manifold that may exist after the previous stage. Minimal boundary loops are detected and triangulated using a triangle fan. At the end of this stage, the surface is a manifold triangular mesh without a boundary (figure 16). Results employing the entire algorithm are presented in the next chapter.



(a) (b)

Figure 16: Closing boundary loops. (a) Part of a mesh with a boundary loop. (b) The same part of the mesh after the loop is closed.

איור 16: סגירת לולאות גבול ברשת (a) חלק מהרשת עם לולאת גבול (b) אותו חלק מהרשת לאחר סגירת הלולאה.

3.2.3 Topology detection by the extended neural gas algorithm

The ability of the *extended neural gas* algorithm to detect topology was previously presented. The *neural gas* algorithm creates a mesh which is a *topology preserving map* of the cloud of points. The latter term is explained in section 3.2.3.1. The *extended neural gas* algorithm is able to detect triangular faces that are a subset of the *Delaunay triangulation*. An elaborate explanation and a proof to this statement is provided in section 3.2.3.2.

From this point on, we refer to the positions held as information in the *neural units* as *sites*, which are the vertices of the approximating mesh . This naming is significant since it conforms with the name used in the field of computational geometry that is used to analyze the topological issue.

3.2.3.1 Topology Preserving Map

The *sites* of the mesh divide R^3 into convex polyhedrons (which are not necessarily closed) called *Voronoi regions* [PRE85, EDE87, BOI93]. The set of these polyhedrons is named the *Voronoi diagram*, that in turn defines neighborhood relationships between *sites* and points in space. All the points in R^3 closest to *site* $\mathbf{v_i}$ form the *Voronoi region*, denoted here by $Vor(i)$. Sampled points of the surface that are classified to a *site* $\mathbf{v_i}$ are part of $Vor(i)$, meaning that $\mathbf{p_s} \in Vor(s_0)$. This defines a mapping φ from sampled points on the surface S to *sites* V on the mesh M (defined by their index) $\varphi : S \to V$ such that $\varphi(\mathbf{p_s}) = s_0$. In other words, the mapping returns the index of the nearest *site* to the sampled point. An inverse mapping φ^{-1} exists between the vertex and positions in space $\varphi^{-1} : V \to S$ such that $\varphi^{-1}(i) = \mathbf{v_i}$ (the assumption is that the *site* is positioned on the surface of the object). In other words, the inverse mapping is from an index of the *site* to its position.

The edges and vertices (*sites*) of the mesh M define a graph $G(E,V)$. Keeping this in mind, we can assign neighborhood criteria to the mappings φ and φ^{-1}. The word “neighborhood” will be used to describe *sites* connected with an edge on the graph G. The word “adjacent” will define vertices in 3D which share a face in their Voronoi region (meaning that they are close in space). The mapping φ is said to be neighborhood preserving if the following condition holds: if two points are adjacent (close on the sampled surface), their closest *sites* are connected neighbors on the approximating mesh M . The inverse mapping φ^{-1} is said to be neighborhood preserving if the complementary condition holds: if two *sites* are connected neighbors, they are also adjacent (close in 3D).

The learning process of the *neural gas* algorithm creates a *topology preserving map*. This means that both the mapping φ and the mapping φ^{-1} are neighborhood-preserving. In other words, a map is said to be topology preserving if each two adjacent points on the sampled surface are classified to neighboring *sites* on the mesh and if each two neighboring *sites* on the mesh are adjacent in space. The subject of topology preserving maps is elaborated with a few examples in [MAR94].

3.2.3.2 Mesh Connectivity Definition

In the *neural gas* algorithm [MAR94] the extracted mesh is defined by vertices (*sites*) and edges. The edges are a subset of the *Delaunay triangulation* of the *sites*. The *Delaunay triangulation* [EDE87] is dual to the *Voronoi diagram* and the following dualities hold:

- Every neural unit (*site*) defines a vertex of a *Delaunay triangle* (simplex of dimension 0).
- Every face of the *Voronoi polyhedron* defines an edge of a *Delaunay triangle* (simplex of dimension 1).
- Every edge of a *Voronoi polyhedron* defines a *Delaunay triangle* (simplex of dimension 2).

The purpose of the *extended neural gas* algorithm is to define the faces of the mesh M . The mechanism that defines the connectivity between neural units in the original *neural gas* algorithm is *Hebbian learning*.

The *Hebbian learning law* was first suggested in the neurobiological context and was adapted and extended for use in artificial neural networks [HAY94]. Its interpretation in the *neural gas* neural network (see [MAR94]) controls the creation of edges between neural units in the following manner: Neural units are connected with an edge if they are excited together meaning that they are closest to a sampled point. The edges decay and disappear if the neural units are not excited for a while during the training process. In the following paragraphs the *Hebbian learning* law will be extended to define faces that are a subset of the *Delaunay triangulation*. Note that this extension digresses from the original *Hebbian learning law* and its existing interpretations as presented in [HAY94] since it defines a connection for more then two neural units.

The interpretation of the *Hebbian learning law* in [MAR94] can be formalized geometrically: An edge between neural units (*sites*) $\mathbf{v_i}$ and $\mathbf{v_j}$ is created if a sample point $\mathbf{p_s}$ exists on the surface such that $\mathbf{v_i}$ and $\mathbf{v_j}$ are its *winner* and *second winner*. That is, for point $\mathbf{p_s}$ the edge $\mathbf{e}=(\mathbf{v_{s_0}},\mathbf{v_{s_1}})$ is created. This rule can be explained using second-order Voronoi diagrams [PRE85, EDE87, BOI93].

A *second order* Voronoi region of *site* $\mathbf{v_i}$ and $\mathbf{v_j}$ is denoted by $Vor(i,j)$. Each point $\mathbf{p}\in Vor(i,j)$ is closer to $\mathbf{v_i}$ and $\mathbf{v_j}$ than to any other neural unit $\mathbf{v_k}$. That is $Vor(i,j)=\{\mathbf{p}\in R^3 \mid \|\mathbf{p}-\mathbf{v_i}\|\le\|\mathbf{p}-\mathbf{v}_k\|\wedge\|\mathbf{p}-\mathbf{v_j}\|\le\|\mathbf{p}-\mathbf{v}_k\|\forall k\ne i,j\}$. A Delaunay edge exists between two neural units $\mathbf{v}_\mathbf{i}$ and $\mathbf{v}_\mathbf{j}$ if $Vor(i) \cap Vor(j) \neq \phi$, meaning that the two Voronoi regions are neighboring regions and share a common face in 3D. In order to explain the *Hebbian learning law*, we will repeat the proof given by Martinez and Schulten in [MAR94]. It is presented here for the sake of completeness, with clarifications and extensions that will later be used.

Theorem 1 :

$Vor(i) \cap Vor(j) \neq \phi \Leftrightarrow Vor(i, j) \neq \phi$

That is, two *sites* i,j are neighboring in space if and only if there is at least one point in space that is closest to both *sites*.

Proof :

We first prove direction 1: if $Vor(i) \cap Vor(j) \neq \phi$ then $Vor(i, j) \neq \phi$.

If a point $\mathbf{p} \in Vor(i) \cap Vor(j)$, then $\|\mathbf{p} - \mathbf{v}_\mathbf{i}\| = \|\mathbf{p} - \mathbf{v}_\mathbf{j}\| \leq \|\mathbf{p} - \mathbf{v}_k\| \forall k \neq i, j$ and thus q.e.d. direction 1.

Direction 2 is more complicated, if $Vor(i, j) \neq \phi$ then $Vor(i) \cap Vor(j) \neq \phi$. Without loss of generality we assume that $\mathbf{p} \in Vor(i)$. This means that $\|\mathbf{p} - \mathbf{v}_\mathbf{i}\| \leq \|\mathbf{p} - \mathbf{v}_\mathbf{j}\| \leq \|\mathbf{p} - \mathbf{v}_k\| \forall k \neq i, j$. We pick a point $\mathbf{q}$ on the line segment $\mathbf{p}, \mathbf{v}_\mathbf{j}$ and write it as $\mathbf{q} = u\mathbf{p} + (1-u)\mathbf{v}_\mathbf{j}, 0 \leq u \leq 1$, In order to proceed, we need to prove that there is no *site* $\mathbf{v}_\mathbf{k}$ closer to point $\mathbf{q}$ (other than $\mathbf{v}_\mathbf{j}$ and $\mathbf{v}_\mathbf{i}$) while changing u from 0 to 1. This will be proved in Lemma 1.

Lemma 1:

$\|\mathbf{q} - \mathbf{v}_\mathbf{j}\| \leq \|\mathbf{q} - \mathbf{v}_k\| \forall 0 \leq u \leq 1 \wedge k \neq i, j$.

Proof:

We write the distance between $\mathbf{q}$ and $\mathbf{v}_\mathbf{j}$

(1) $\|\mathbf{q} - \mathbf{v}_\mathbf{j}\| = \|u\mathbf{p} + (1-u)\mathbf{v}_\mathbf{j} - \mathbf{v}_\mathbf{j}\| = u\|\mathbf{p} - \mathbf{v}_\mathbf{j}\|$

Now we write the following inequality using the norm characteristics:

(2) $\|\mathbf{p} - \mathbf{v}_\mathbf{k}\| = \|\mathbf{p} - \mathbf{v}_\mathbf{k} - \mathbf{q} + \mathbf{q}\| \leq \|\mathbf{p} - \mathbf{q}\| + \|\mathbf{q} - \mathbf{v}_\mathbf{k}\|$

And thus:

(3) $\|\mathbf{q} - \mathbf{v}_\mathbf{k}\| \geq \|\mathbf{p} - \mathbf{v}_\mathbf{k}\| - \|\mathbf{p} - \mathbf{q}\|$

but $\|\mathbf{p} - \mathbf{v}_\mathbf{k}\| \geq \|\mathbf{p} - \mathbf{v}_\mathbf{j}\|$ from the definition of $Vor(i, j)$ so we can replace this term in (3) and write:

(4) $\|\mathbf{q} - \mathbf{v}_\mathbf{k}\| \geq \|\mathbf{p} - \mathbf{v}_\mathbf{j}\| - \|\mathbf{p} - \mathbf{q}\|$

Since $\mathbf{q}$ is on the line segment $\mathbf{p}, \mathbf{v}_\mathbf{j}$ we can write

(5) $\|\mathbf{p}-\mathbf{q}\| = \|\mathbf{p}-\mathbf{v_j}\| - \|\mathbf{q}-\mathbf{v_j}\|$

we replace (5) into (4) and get

(6) $\|\mathbf{q}-\mathbf{v_k}\| \geq \|\mathbf{p}-\mathbf{v_j}\| - \|\mathbf{p}-\mathbf{v_j}\| + \|\mathbf{q}-\mathbf{v_j}\|$

Eliminating similar terms we get :

$\|\mathbf{q}-\mathbf{v_k}\| \geq \|\mathbf{q}-\mathbf{v_j}\|$

Q.E.D. *Lemma 1*

Lemma 1 suggests that while advancing towards *site* $\mathbf{v_j}$ the distance to it decreases while the distance to other *sites* may increase or decrease. And no site that was farther than $\mathbf{v_j}$ becomes closer while moving.

We continue with the proof of direction 2 in theorem 1. If we decrease u continuously from 1 to 0, the distance of $\mathbf{q}$ to $\mathbf{v_j}$ decreases continuously towards zero while the distance of $\mathbf{q}$ to $\mathbf{v_i}$ increases. Since for $u=1$ $\|\mathbf{q}-\mathbf{v_i}\| \geq \|\mathbf{q}-\mathbf{v_j}\|$ there must be a parameter u^* for which $\|\mathbf{q}-\mathbf{v_i}\| = \|\mathbf{q}-\mathbf{v_j}\| \leq \|\mathbf{q}-\mathbf{v_k}\| \forall k \neq i, j$, (the inequality at the end is derived from *Lemma 1*) and thus for parameter u^* $\mathbf{q} \in Vor(i) \cap Vor(j) \neq \phi$ Q.E.D. *Theorem 1.*

When the edges of the mesh are established, the faces of the mesh can be created using an extension to the *Hebbian rule*. For explanation of this extension to the rule we introduce the third order Voronoi region. The third order Voronoi region of *sites* $\mathbf{v_i}$, $\mathbf{v_j}$ and $\mathbf{v_k}$ is denoted here by $Vor(i, j, k)$. Each point in the third order Voronoi region $\mathbf{p} \in Vor(i, j, k)$ is closer to $\mathbf{v_i}$, $\mathbf{v_j}$ and $\mathbf{v_k}$ than to any other *site* $\mathbf{v_m}$. This relation can be written formally as $Vor(i, j, k) = \{\mathbf{p} \in R^3 \mid \|\mathbf{p}-\mathbf{v_i}\| \leq \|\mathbf{p}-\mathbf{v_m}\| \wedge \|\mathbf{p}-\mathbf{v_j}\| \leq \|\mathbf{p}-\mathbf{v_m}\| \wedge \|\mathbf{p}-\mathbf{v_j}\| \leq \|\mathbf{p}-\mathbf{v_m}\| \forall m \neq i, j, k\}$.

An elaborate definition of higher Voronoi diagrams in R^n is given in [BOI93]. We present a short interpretation to the three dimensional case for Voronoi diagrams of higher orders.

Definition of higher order Voronoi diagrams in R^3

The Voronoi diagram of order h in R^3 is composed of convex Voronoi regions. The vertices of the Voronoi regions are defined by the center $\mathbf{b}$ of ball B passing through four *sites* $\mathbf{v_i}$, $\mathbf{v_j}$, $\mathbf{v_k}$, $\mathbf{v_m}$. Let R be a set of all the *sites* residing inside ball B. For $|R| = l$ the center of the ball $\mathbf{b}$ is a vertex of the Voronoi diagram of order h where $l+1 \leq h \leq l+3$.

If ball B is empty, the Voronoi vertex $\mathbf{b}$ is a vertex of the first order Voronoi diagram and is the intersection of the following four Voronoi regions $Vor(i)$, $Vor(j)$, $Vor(k)$, $Vor(m)$. That vertex will persist in the second order Voronoi diagrams as the common vertex of six Voronoi regions $Vor(i, j)$, $Vor(i,k)$, $Vor(j,k)$, $Vor(i,m)$, $Vor(j,m)$, $Vor(k,m)$. It will also persist in the third order Voronoi diagram as the common vertex of four Voronoi regions

$Vor(i,j,k)$, $Vor(i,j,m)$, $Vor(j,k,m)$, $Vor(i,k,m)$. This type of vertex is called a close type vertex.

If ball B contains one *site* $\mathbf{v_n}$, then the Voronoi vertex $\mathbf{b}$ is a vertex of the second order Voronoi diagram and is the intersection of four Voronoi regions $Vor(n,i)$, $Vor(n,j)$, $Vor(n,k)$, $Vor(n,m)$. It will persist in the third order Voronoi diagram and will be the intersection of the six Voronoi regions $Vor(n,i,j)$, $Vor(n,i,k)$, $Vor(n,i,m)$, $Vor(n,j,k)$, $Vor(n,j,m)$, $Vor(n,k,m)$.

If ball B contains two *sites* $\mathbf{v_n}$ and $\mathbf{v_z}$ then the Voronoi vertex $\mathbf{b}$ is also a medium type vertex of the third order Voronoi diagram and it is the intersection of the following four Voronoi regions $Vor(n,z,i)$, $Vor(n,z,j)$, $Vor(n,z,k)$, $Vor(n,z,m)$.

Before proceeding, we reiterate the duality connections between the Voronoi diagram (of order 1) and the Delaunay triangulation. A Delaunay face is dual to an edge of a Voronoi polyhedron. An edge of a Voronoi polyhedron in R^3 is created by the intersection of three first order Voronoi polyhedrons. Thus, a Delaunay face exists between three *sites* $\mathbf{v_i}$, $\mathbf{v_j}$ and $\mathbf{v_k}$ if $Vor(i) \cap Vor(j) \cap Vor(k) \neq \phi$. This means that the three first order Voronoi regions are neighboring regions and share a common edge.

In *Theorem 1* the existence of second order Voronoi diagrams was used to detect Delaunay edges. The analogy tying third order Voronoi diagrams to Delaunay faces is not trivial and is presented in theorem 2.

Theorem 2:

$$Vor(i) \cap Vor(j) \cap Vor(k) \neq \phi \Leftrightarrow Vor(i,j,k) \neq \phi \wedge Vor(i,j) \neq \phi \wedge Vor(i,k) \neq \phi \wedge Vor(j,k) \neq \phi$$

Proof:

We first prove direction 1: if $Vor(i) \cap Vor(j) \cap Vor(k) \neq \phi$ then $Vor(i,j,k) \neq \phi \wedge Vor(i,j) \neq \phi \wedge Vor(i,k) \neq \phi \wedge Vor(j,k) \neq \phi$.

If a point $\mathbf{p} \in Vor(i) \cap Vor(j) \cap Vor(k)$, this means that $\|\mathbf{p}-\mathbf{v_i}\| = \|\mathbf{p}-\mathbf{v_j}\| = \|\mathbf{p}-\mathbf{v_k}\| \leq \|\mathbf{p}-\mathbf{v_m}\| \forall m \neq i,j,k$

and thus Q.E.D. direction 1.

Direction 2 is again more complicated:

if $Vor(i,j,k) \neq \phi \wedge Vor(i,j) \neq \phi \wedge Vor(i,k) \neq \phi \wedge Vor(j,k) \neq \phi$ then $Vor(i) \cap Vor(j) \cap Vor(k) \neq \phi$.

Without loss of generality we assume that: $\|\mathbf{p}-\mathbf{v_i}\| \leq \|\mathbf{p}-\mathbf{v_j}\| \leq \|\mathbf{p}-\mathbf{v_k}\| \leq \|\mathbf{p}-\mathbf{v_m}\| \forall m \neq i,j,k$.

We pick a moving point $\mathbf{q}$ on the line segment $\mathbf{p},\mathbf{v_k}$. We reuse *Lemma 1* to establish that while moving $\mathbf{q}$ from $\mathbf{p}$ towards *site* $\mathbf{v_k}$ the distance $\|\mathbf{q}-\mathbf{v_k}\|$ decreases, whereas the distance to other *sites* either increases or decreases while keeping the following relation $\|\mathbf{q}-\mathbf{v_k}\| \leq \|\mathbf{q}-\mathbf{v_m}\| \forall m \neq i,j,k$. Thus, while advancing towards $\mathbf{v_k}$ one of the following distinct events occurs first:

Case 1 : $\|\mathbf{q}-\mathbf{v_i}\| = \|\mathbf{q}-\mathbf{v_k}\|$

Case 2 : $\|\mathbf{q}-\mathbf{v_j}\| = \|\mathbf{q}-\mathbf{v_k}\|$

Case 3 : $\|\mathbf{q}-\mathbf{v_i}\| = \|\mathbf{q}-\mathbf{v_j}\|$

Case 4 : $\|\mathbf{q}-\mathbf{v_i}\| = \|\mathbf{q}-\mathbf{v_j}\| = \|\mathbf{q}-\mathbf{v_k}\| \le \|\mathbf{q}-\mathbf{v_m}\|$

Case 4 means that $Vor(i) \cap Vor(j) \cap Vor(k) \ne \phi$ and thus Q.E.D.

We will now establish what happens in the other cases:

Case 1: Due to *Lemma 1* and to the initial *site*-naming assumption that $\|\mathbf{p}-\mathbf{v_i}\| \le \|\mathbf{p}-\mathbf{v_j}\| \le \|\mathbf{p}-\mathbf{v_k}\| \le \|\mathbf{p}-\mathbf{v_m}\| \forall m \ne i, j, k$, either case 2 or case 3 were passed before getting to case 1, or case 4 was reached. Thus, it is impossible to get to case 1 before other cases. We continue examining the other cases.

Case 2: We can write that $\|\mathbf{q}-\mathbf{v_i}\| < \|\mathbf{q}-\mathbf{v_j}\| = \|\mathbf{q}-\mathbf{v_k}\| \le \|\mathbf{q}-\mathbf{v_m}\| \forall m \ne i, j, k$. Since point $\mathbf{q}$ satisfies $\|\mathbf{q}-\mathbf{v_j}\| = \|\mathbf{q}-\mathbf{v_k}\|$, we deduce that point $\mathbf{q}$ is on the bisection plane between *site* $\mathbf{v_j}$ and *site* $\mathbf{v_k}$ denoted by $Bis(\mathbf{v_j}, \mathbf{v_k})$. We define a new moving point $\mathbf{t}$ on the line connecting point $\mathbf{q}$ with the projection of *site* $\mathbf{v_i}$ on the bisection plane $Bis(\mathbf{v_j}, \mathbf{v_k})$. Point $\mathbf{t}$ moves on the line in the direction increasing $\|\mathbf{t}-\mathbf{v_i}\|$. One of the following cases will occur first:

Case 2.1: $\|\mathbf{t}-\mathbf{v_i}\| = \|\mathbf{t}-\mathbf{v_j}\| = \|\mathbf{t}-\mathbf{v_k}\| \le \|\mathbf{t}-\mathbf{v_m}\| \forall m \ne i, j, k$

Case 2.2: $\|\mathbf{t}-\mathbf{v_i}\| < \|\mathbf{t}-\mathbf{v_j}\| = \|\mathbf{t}-\mathbf{v_k}\| = \|\mathbf{t}-\mathbf{v_m}\| \le \|\mathbf{t}-\mathbf{v_n}\| \forall n \ne i, j, k, m$

If case 2.1 was reached, then Q.E.D.

If we reached case 2.2, we define a new moving point $\mathbf{r}$ moving continuously away from $\mathbf{t}$ on the line that satisfies $\|\mathbf{r}-\mathbf{v_j}\| = \|\mathbf{r}-\mathbf{v_k}\| = \|\mathbf{r}-\mathbf{v_m}\|$ towards the direction that decreases the distance from *sites* $\mathbf{v_j}$ and $\mathbf{v_k}$. One of the following cases will occur first:

Case 2.2.1 $\|\mathbf{r}-\mathbf{v_i}\| = \|\mathbf{r}-\mathbf{v_j}\| = \|\mathbf{r}-\mathbf{v_k}\| = \|\mathbf{r}-\mathbf{v_m}\| \le \|\mathbf{r}-\mathbf{v_n}\| \forall n \ne i, j, k, m$

Case 2.2.2 $\|\mathbf{r}-\mathbf{v_i}\| < \|\mathbf{r}-\mathbf{v_j}\| = \|\mathbf{r}-\mathbf{v_k}\| = \|\mathbf{r}-\mathbf{v_m}\| = \|\mathbf{r}-\mathbf{v_n}\| \le \|\mathbf{r}-\mathbf{v}_z\| \forall z \ne i, j, k, m, n$

If case 2.2.1 was reached then Q.E.D .

We will show that case 2.2.2 is a recurring case that leads back to case 2.2.1. Notice that ball B, which is centered at $\mathbf{r}$ and whose boundary passes through $\mathbf{v_j}$, $\mathbf{v_k}$, $\mathbf{v_m}$, $\mathbf{v_n}$, contains one site $\mathbf{v_i}$. Thus, by definition, point $\mathbf{r}$ is a vertex of the second order Voronoi diagram, which is the common point of Voronoi regions $Vor(i, j)$, $Vor(i,k)$, $Vor(i,m)$, $Vor(i,n)$. These Voronoi regions are partially bounded by the intersection of three bisection planes. Notice that the planes do not separate *site* $\mathbf{v_i}$ from the other sites mentioned since case 2.2.1 was not reached. Several cases exist for the arrangement of those bisection planes:

Case 2.2.2.1 $Bis(\mathbf{v_j},\mathbf{v_k}), Bis(\mathbf{v_j},\mathbf{v_m}), Bis(\mathbf{v_k},\mathbf{v_n})$.

Case 2.2.2.2 $Bis(\mathbf{v_j},\mathbf{v_k}), Bis(\mathbf{v_j},\mathbf{v_n}), Bis(\mathbf{v_k},\mathbf{v_m})$.

Case 2.2.2.3 $Bis(\mathbf{v_j},\mathbf{v_k}), Bis(\mathbf{v_j},\mathbf{v_n}), Bis(\mathbf{v_j},\mathbf{v_m})$.

Case 2.2.2.4 $Bis(\mathbf{v_j},\mathbf{v_k}), Bis(\mathbf{v_k},\mathbf{v_n}), Bis(\mathbf{v_k},\mathbf{v_m})$.

In all those cases, moving on the plane $Bis(\mathbf{v_j},\mathbf{v_k})$ to the other side of any other bisection plane will result in a point that is closer to $\mathbf{v_m}$ or $\mathbf{v}_n$ than to $\mathbf{v_i}$ or $\mathbf{v_j}$ (depending on the plane passed). Thus the point a satisfying $a \in Vor(j) \cap Vor(k)$ does not exist on the other side of any of those bisection planes. The given implies that such a point does exist, due to *Theorem 1* and the fact that $Vor(j,k) \neq \phi$. The possibility remains not to pass one of those planes and to continue moving along the lines created by those planes – for example continue moving on the line $\|\mathbf{r}-\mathbf{v_j}\| = \|\mathbf{r}-\mathbf{v_k}\| = \|\mathbf{r}-\mathbf{v_n}\|$. This possibility brings us back to cases 2.2.1 or 2.2.2 replacing $\mathbf{v_m}$. Since the number of *sites* creating the Voronoi diagram is finite, the number of cases to be checked is finite. In each such case we either reach case 2.2.1 and thus Q.E.D or we reach case 2.2.2 and continue moving along the boundary of $Vor(i,j,k)$ on $Bis(\mathbf{v_j},\mathbf{v_k})$. Summarizing case 2.2.2 we conclude that it recurs a finite number of times until we reach case 2.2.1 or the given that $Vor(j,k) \neq \phi$ is contradicted.

Case 3: We can write that $\|\mathbf{q}-\mathbf{v_i}\| = \|\mathbf{q}-\mathbf{v_j}\| < \|\mathbf{q}-\mathbf{v_k}\| \leq \|\mathbf{q}-\mathbf{v_m}\| \forall m \neq i,j,k$.

We continue moving $\mathbf{q}$ towards $\mathbf{v_k}$. In this case, however, we can write $\|\mathbf{q}-\mathbf{v_j}\| < \|\mathbf{q}-\mathbf{v_i}\| \leq \|\mathbf{q}-\mathbf{v_k}\| \leq \|\mathbf{q}-\mathbf{v_m}\| \forall m \neq i,j,k$ and since we cannot pass the bisection plane $Bis(\mathbf{v_i},\mathbf{v_j})$ again while moving $\mathbf{q}$ towards $\mathbf{v_k}$ the following event occurs: $\|\mathbf{q}-\mathbf{v_j}\| < \|\mathbf{q}-\mathbf{v_i}\| = \|\mathbf{q}-\mathbf{v_k}\| \leq \|\mathbf{q}-\mathbf{v_m}\| \forall m \neq i,j,k$

Interchanging $\mathbf{v_i}$ and $\mathbf{v_j}$, this event is similar to case 2 which we already proved.

Summarizing, it was shown that a path to a point on $Vor(i) \cap Vor(j) \cap Vor(k)$ exists without leaving $Vor(i,j,k)$ and thus Q.E.D. *Theorem 2*.

From *Theorem 2* one may notice that the existence of a sample point in the third order Voronoi region is not sufficient to indicate that a Delaunay face exists on the mesh. However, it is a necessary condition along with the condition that the edges providing support for that face exist on the mesh.

The extension to the *Hebbian learning law* for creating a Delaunay face can be formalized as follows: for a sampled point $\mathbf{p_s}$ whose winner *sites* are $\mathbf{v}_{s_0}, \mathbf{v}_{s_1}, \mathbf{v}_{s_2}$, a Delaunay triangle $\mathbf{f} = (\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ will be created if $\mathbf{e}_1 \neq \phi \wedge \mathbf{e}_2 \neq \phi \wedge \mathbf{e}_3 \neq \phi$ where $\mathbf{e}_1 = (\mathbf{v}_{s_0}, \mathbf{v}_{s_1})$, $\mathbf{e}_2 = (\mathbf{v}_{s_0}, \mathbf{v}_{s_2})$, $\mathbf{e}_3 = (\mathbf{v}_{s_1}, \mathbf{v}_{s_2})$.

In other words, each Delaunay face can be defined by three sampled points. Each sampled point defines an edge between two closest sites as with the *Hebbian learning law*. One of the sampled points defines the face by being closer to all three *sites* creating the face.

4 Results

This chapter presents the results of the reconstruction approaches described in the previous chapter. The results are presented in the same order of presentation as the approaches.

4.1 Approach A – reconstruction of surfaces from a cloud of points

The approach consists of several parameterization methods and surface fitting methods as can be seen in figure 2.

The feasibility of the proposed reconstruction methods is demonstrated on several freeform objects. The development environment consisted of a CyberWare 3D laser scanner and a 350 Mhz Pentium II PC with 128M RAM memory and Windows NT 4.0 operating system. The software was written in the C Programming Language, using the Irit software package [ELB96].

The objects were sampled at high density, O(10^4) points (figure 17). Next, the proposed PDE or neural network SOM parameterization methods were applied (see section 4.1.1). Finally, several surface approximation methods were tested and compared (see section 4.1.2). The results of the overall reconstruction process are presented in section 4.1.3.

4.1.1 Parameterization

The following sections will present several examples for parametric grids constructed in order to parameterize the range images in figure 17. Section 4.1.2 will present surface fitting over the points parameterized using the grids displayed in this section.

4.1.1.1 2D initial Parameterization using PDE and SOM

2D initial parameterization is performed using PDE or SOM. Figure 18 shows grids created by the PDE parameterization method. The corner points for the grid were selected by extreme coordinate values. The ability of the PDE method to create a grid where the iso-curves do not intersect is demonstrated on the plane model where the boundary is highly concave. Figures 19 and 20 show the SOM parameterization method with the "*boundary last*" and "*boundary first*" corrections.

The knife sample demonstrates how the orientation of the grid can change qualitatively. Figure 21 shows the parameterization grids and the point distribution of both grids. In the PDE parameterization, the sub-boundaries were picked arbitrarily (as extreme coordinate values). This causes a highly non-uniform point distribution in the parametric domain during initial parameterization. However, with the neural network SOM parameterization method, the points are distributed more fairly in the parametric domain.

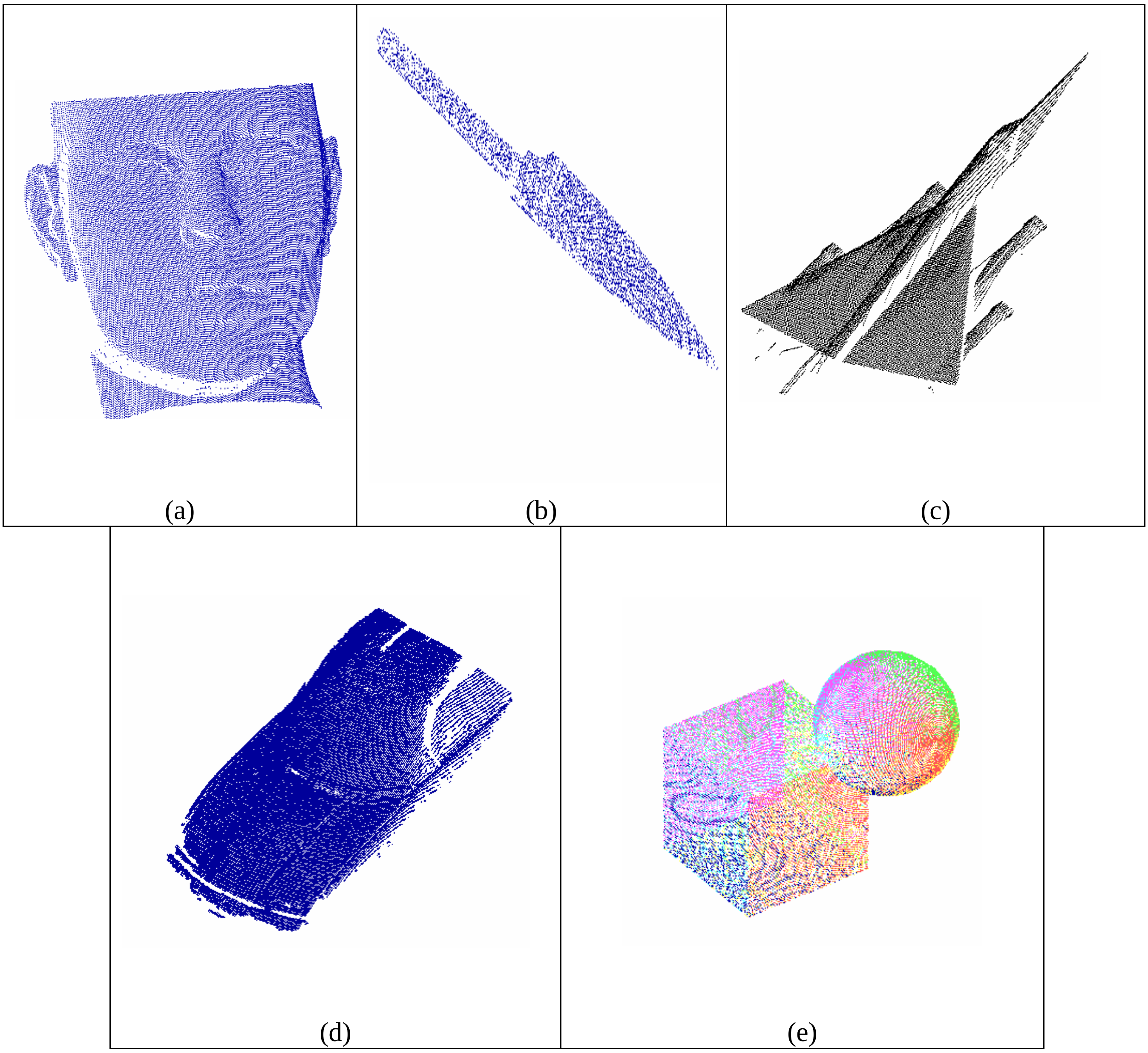


Figure 17: Sampled models: (a) A range image of a mask. (b) A range image of a knife. (c) A range image of a plane. (d) A range image of a toy car. (e) A cloud of points that consists of several range images of joined sphere and cube.

איור 17: גופים דגומים (a) תמונה תלת ממדית של מסיכה. (b) תמונה תלת ממדית של סכין. (c) תמונה תלת ממדית של מטוס. (d) תמונה תלת ממדית של מכונית צעצוע. (e) ענן נקודות של כדור וקוביה הבנוי ממספר תמונות תלת ממדיות.

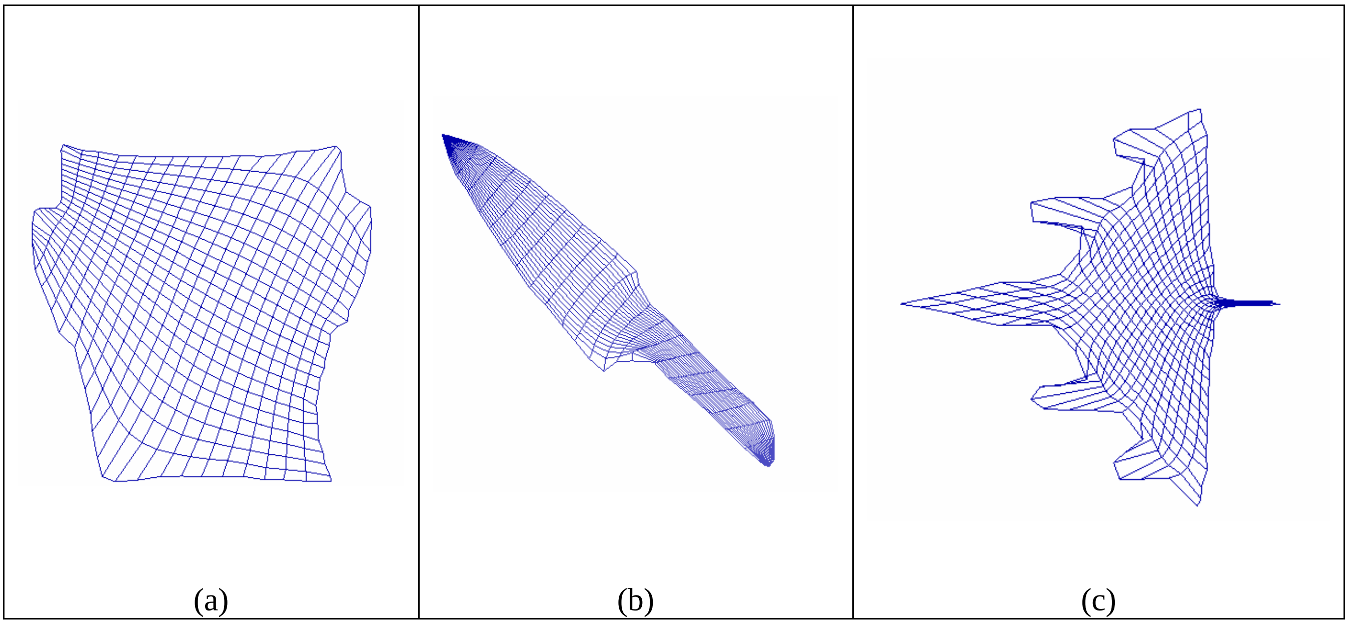


Figure 18: PDE parameterization grid for the: (a) mask sample (b) knife sample (c) plane sample.

איור 18: סריג פרמטריזציה שחושב בשיטת PDE עבור: (a) דוגמת מסיכה (b) דוגמת סכין (c) דוגמת מטוס.

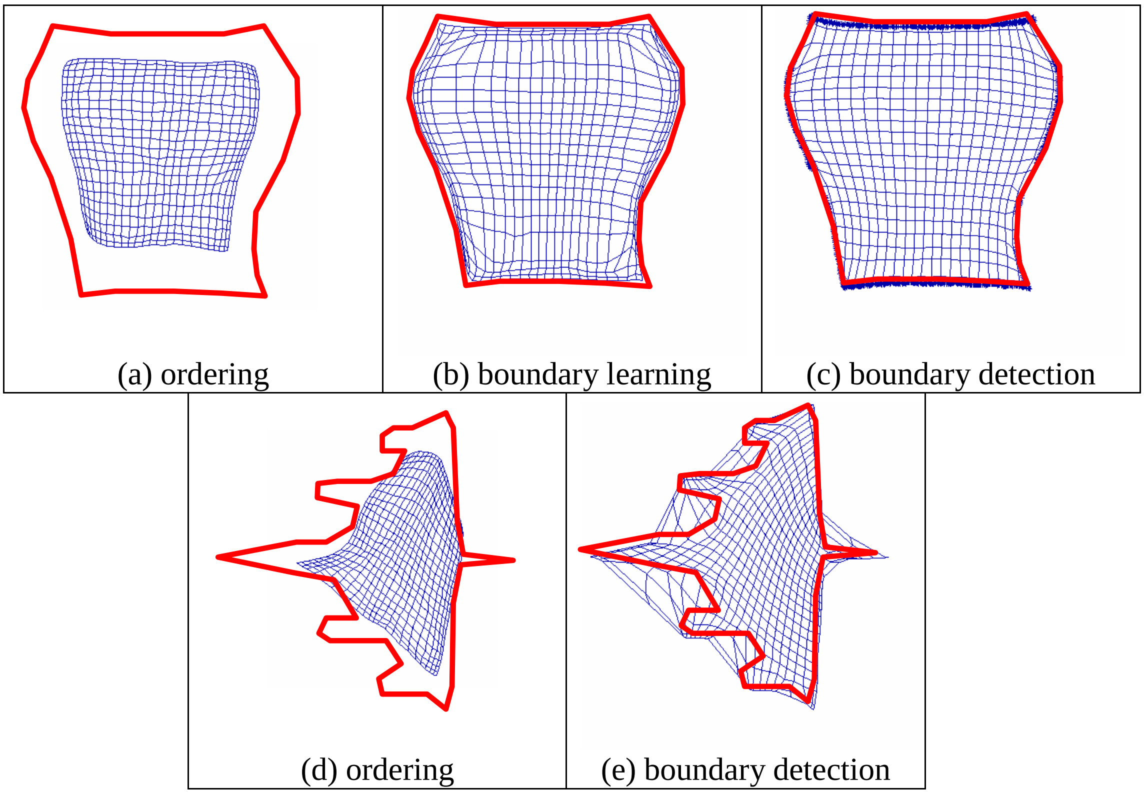
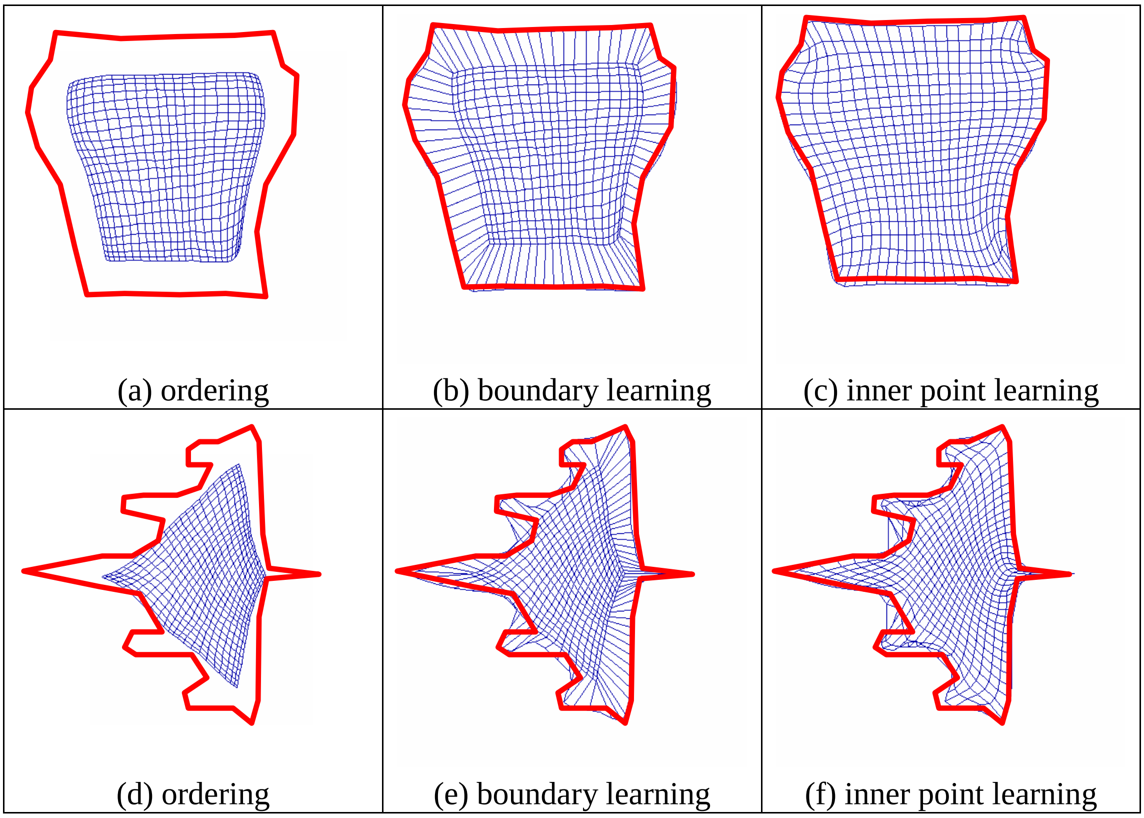


Figure 19: Different phases in the SOM boundary last method for a mask and an airplane. (a-c) mask model. (d-e) airplane model.

איור 19: שלבים שונים בתהליך יצירת סריג באמצעות שיטת SOM boundary last (a-c) מודל המסיכה.
(d-e) מודל המטוס.



(a) ordering

(b) boundary learning

(c) inner point learning

(d) ordering

(e) boundary learning

(f) inner point learning

Figure 20: Different phases in the SOM boundary first method applied on a mask sample and on an airplane. (a-c) mask model. (d-f) airplane model.

איור 20: שלבים שונים בתהליך יצירת סריג באמצעות שיטת SOM boundary first (a-c) מודל המסיכה.
(d-f) מודל המטוס.

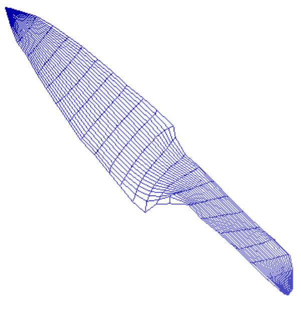
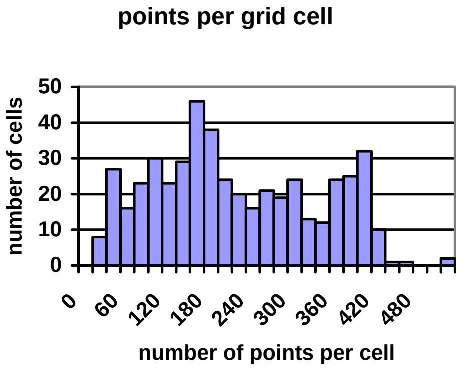
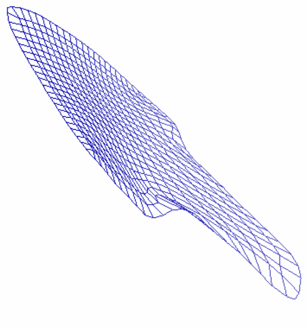
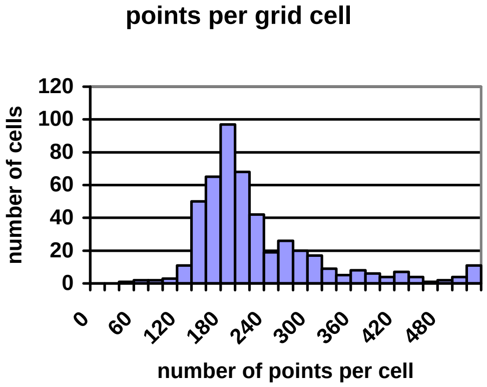
Parameterization method	2D parametric grid	Point density in grid histogram
PDE		
Neural Network SOM		

Figure 21: Parameterization grid of the knife model using PDE and neural network SOM parameterization, and a point distribution diagram.

איור 21: סריג הפרמטריזציה של סכין תוך שימוש במשוואה דיפרנציאלית חלקית ועל ידי שימוש ברשת בינה מלאכותית מסוג SOM ודיאגרמת פיזור נקודות בתאי הסריג.

4.1.1.2 3D SOM Parameterization

Using the SOM parameterization method in 3D can result in many orientations of the created grid as can be seen in figure 22. Moreover, due to non uniform distribution of the points, the aspect ratio of the grid may change. Figure 23 demonstrates the phases of the 3D SOM algorithm with the *growing grid* extension. The resulting grid has the size of 35x48 nodes. The size of the bounding box of the toy car range image is: 105x75x27 mm thus the aspect ratio of 1.4 is approximated by a grid aspect ratio of 1.37.

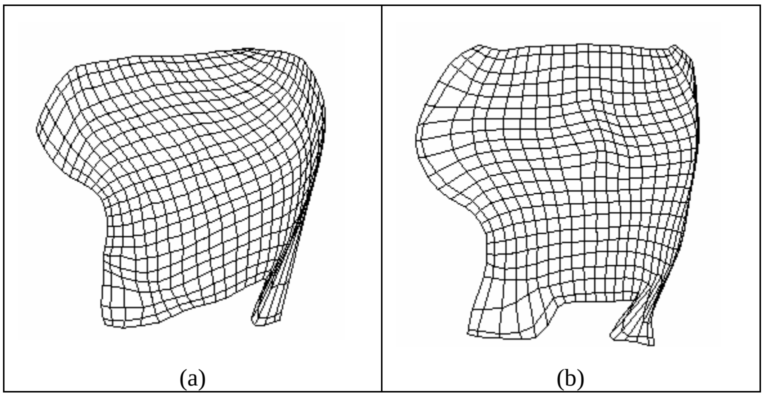


Figure 22: Mask model: two orientations of the 3D SOM grid.
איור 22 : מודל המסיכה, שתי אוריינטציות לסריג המסיכה הנוצר בשיטת ה-SOM התלת ממדי.

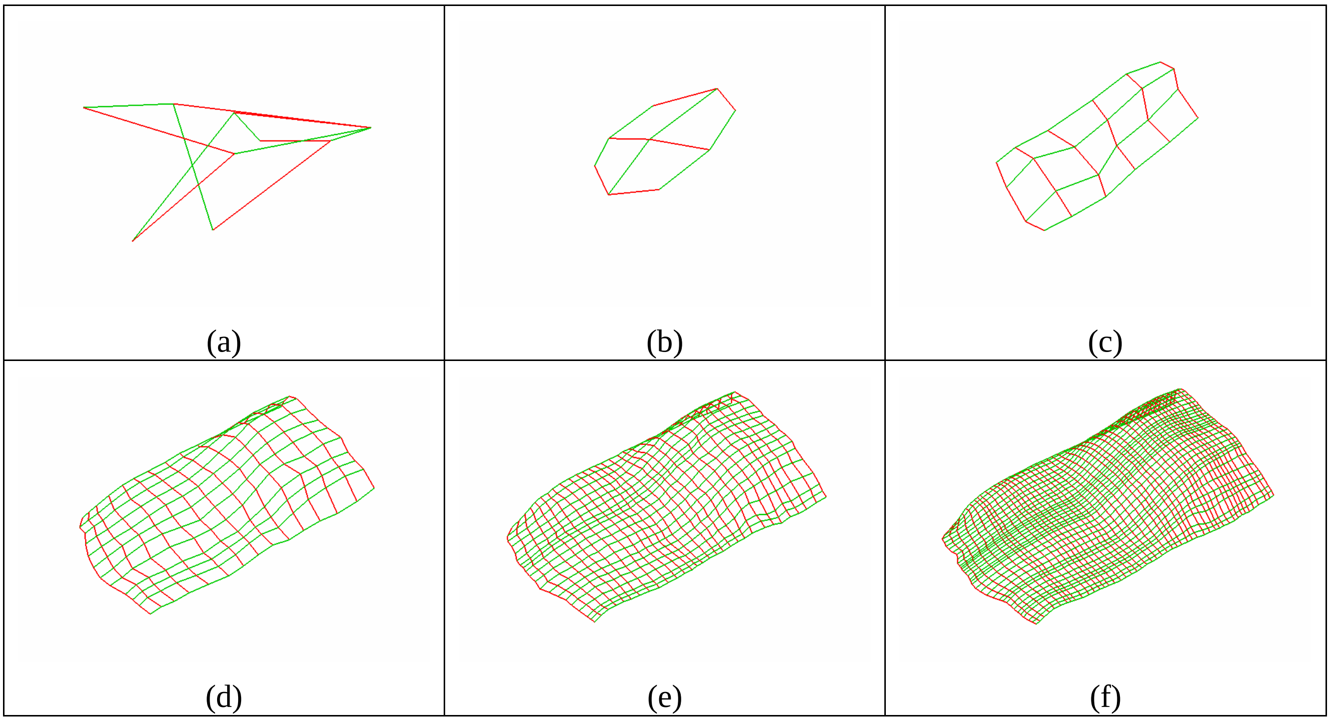


Figure 23: A toy car model: The progress of the SOM algorithm with the *growing grid* extension on the toy car range image. (a) Random initialization. (b-e) Intermediate stages. (f) The final grid.
איור 23: מודל מכונית צעצוע: התקדמות אלגוריתם הSOM המורחב על ידי *growing gri*d על תמונה תלת ממדית של מכונית צעצוע (a) אתחול אקראי. (b-e) שלבי ביניים. (f) הסריג הסופי.

4.1.2 Surface Fitting

This section compares the surface fitting methods described in section 3.1.2. The methods are applied to sampled points that were parameterized using various parameterization methods as described at section 3.1.1.

For 2D parameterization methods, the CMA method was used to rapidly approximate an initial B-Spline surface. For 3D parameterized points, the 3D grid was used as the control mesh of the approximating surface.

In order to improve the error both on the surface and on the boundary, two correction methods were tested: GDA and RSEC (sections 3.1.2.2 and 3.1.2.3). Adaptive base surface parameterization was applied for parametric optimization. Reparameterization of the base surface proved to be the time bottleneck; otherwise, the RSEC and GDA had similar execution times. Initial parameterization time using PDE and the SOM neural network was negligible compared to surface fitting time and did not take more than 2-3 minutes in all the examples.

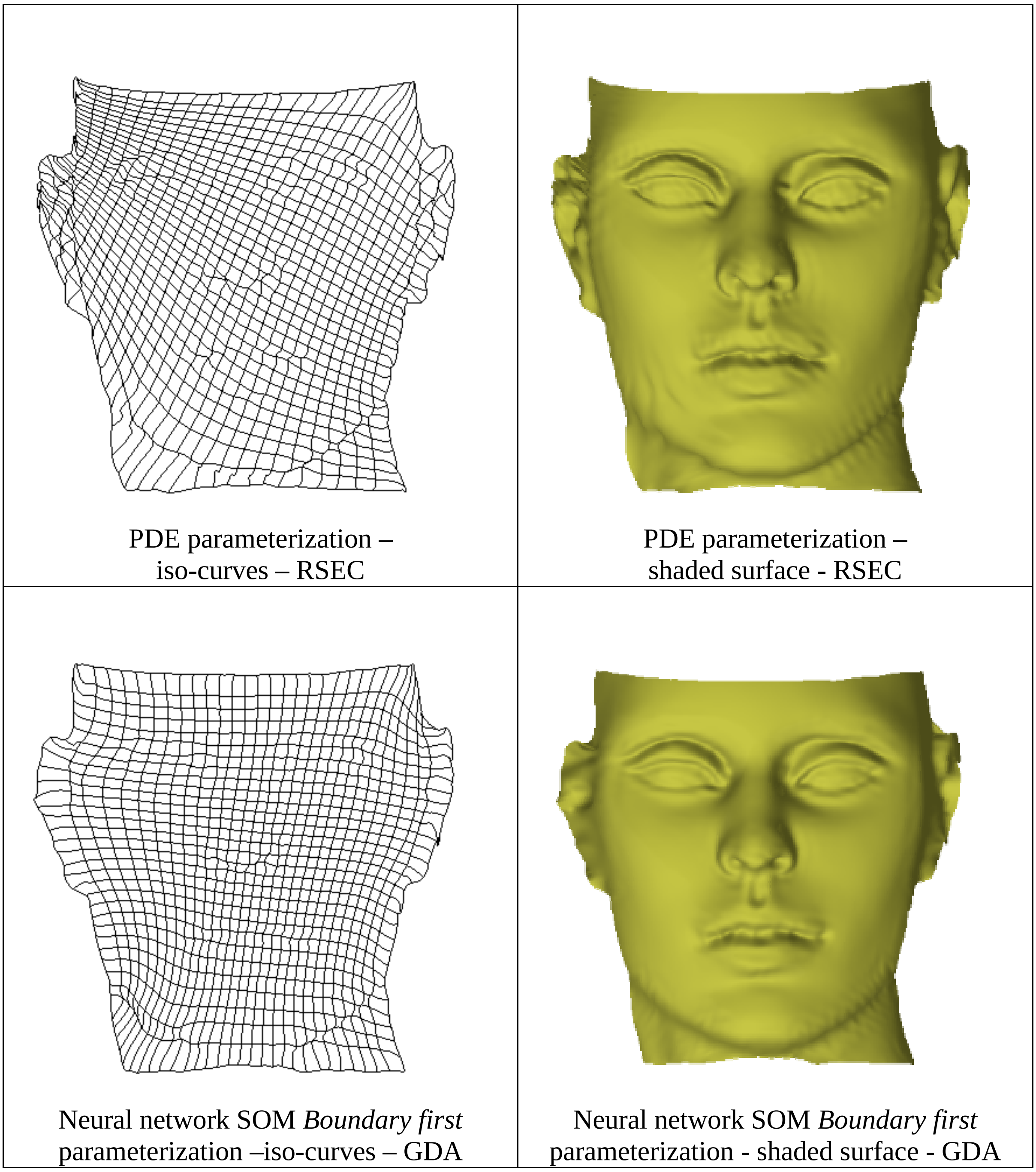
Following is a comparison between the approximation methods, according to performance parameters: size, global time, reparameterization time, surface error, boundary error. This comparison was carried out on mask, knife and airplane objects (figures 24-26 and tables 5-7). The mask object was sampled from a real object and demonstrates the ability of the methods to deal with complicated surfaces. The knife example was also sampled from a real object and illustrates the ability of the algorithms to deal accurately with a large number of sample points. The airplane example was synthetically calculated from an object and demonstrates the ability of the algorithms to deal with highly concave surfaces. In tables 5-7: iteration numbers refer to parametric optimization iterations (section 3.1.1.2); all size measurements are in centimeters; errors presented are maximum errors.

Method	Control Polygon size	Total Time	Repara-meterization Time	# of Iterations	Boundary Error	Surface Error
PDE Parameterization						
CMA	52x52	8 sec	-	1	1.202326	0.486495
GDA	52x52	62 min	~42 min	3	0.622919	0.226782
RSEC	52x52	65 min	~42 min	3	0.889290	0.168178
Neural network SOM “Boundary first” Parameterization						
CMA	52x52	8 sec	-	1	1.220696	0.441713
GDA	52x52	42 min	~28 min	2	0.831381	0.254475
RSEC	52x52	48 min	~28 min	2	1.121670	0.298538

Table 5: Comparison of approximation methods for mask (# of sampled points = 32626; bounding box size of the model =15 x 15 x 13 cm^3).

טבלה 5: השוואת שיטות שחזור עבור דגימת מסיכה. מספר הנקודות הדגומות הוא 32626 וגודל התיבה החוסמת הוא 15x15x13 סנטימטרים מעוכבים.

The mask example in figure 24 shows two results: neural network SOM parameterization was best fitted by GDA and the PDE parameterization was best fitted using RSEC. Although the error is smaller in the PDE-RSEC example, the mask looks worse because of the local nature of RSEC. The RSEC is greedier than the GDA and works per sampled point at a time. The error is smaller numerically in this case; however the surface seems less smooth.



PDE parameterization – iso-curves – RSEC

PDE parameterization – shaded surface - RSEC

Neural network SOM *Boundary first* parameterization –iso-curves – GDA

Neural network SOM *Boundary first* parameterization - shaded surface - GDA

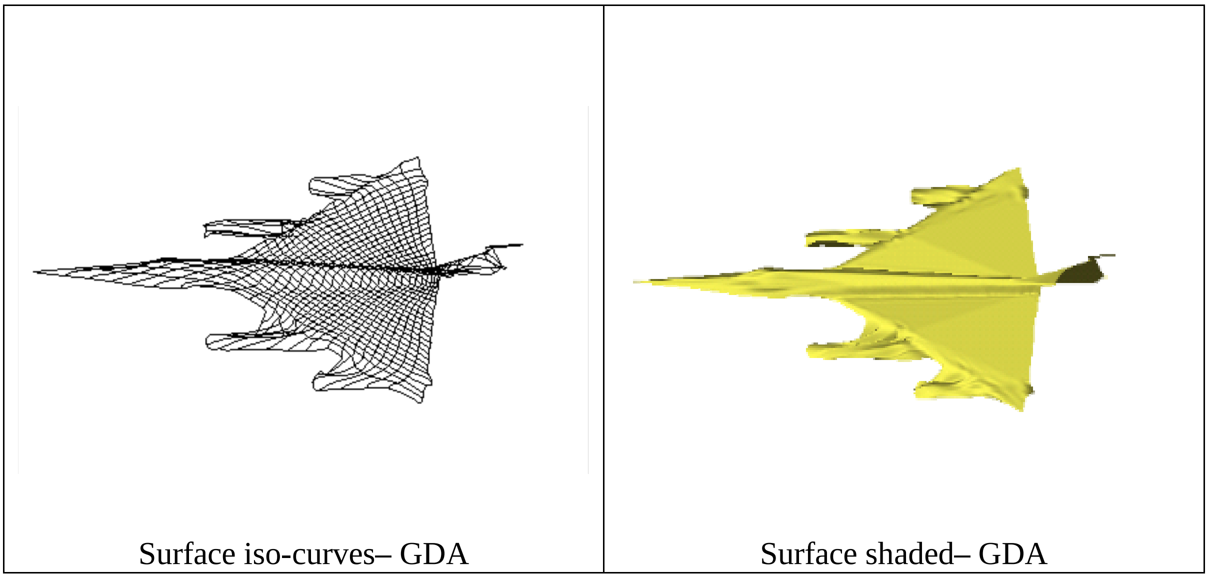
Figure 24: Example for mask surface fitting.

איור 24: דוגמא להתאמת משטחים למסיכה.

Method	Control Polygon size	Total Time	Repara-meterization Time	# of Iterations	Boundary Error	Surface Error
CMA	52x52	8 sec	-	1	1.038951	0.431203
GDA	52x52	340 min	~300 min	5	0.1222228	0.085556
RSEC	52x52	200 min	~160 min	3	0.3345834	0.077747

Table 6: Comparison of approximation methods for an airplane (# of sampled points = 37295; bounding box size of the model = 11x7x3 cm^3).

טבלה 6: השוואת שיטות שחזור עבור דגימת מטוס. מספר הנקודות הדגומות הוא 37295 וגודל התיבה החוסמת הוא 11x7x3 סנטימטרים מעוכבים.



Surface iso-curves– GDA

Surface shaded– GDA

Figure 25: Example for airplane surface fitting.

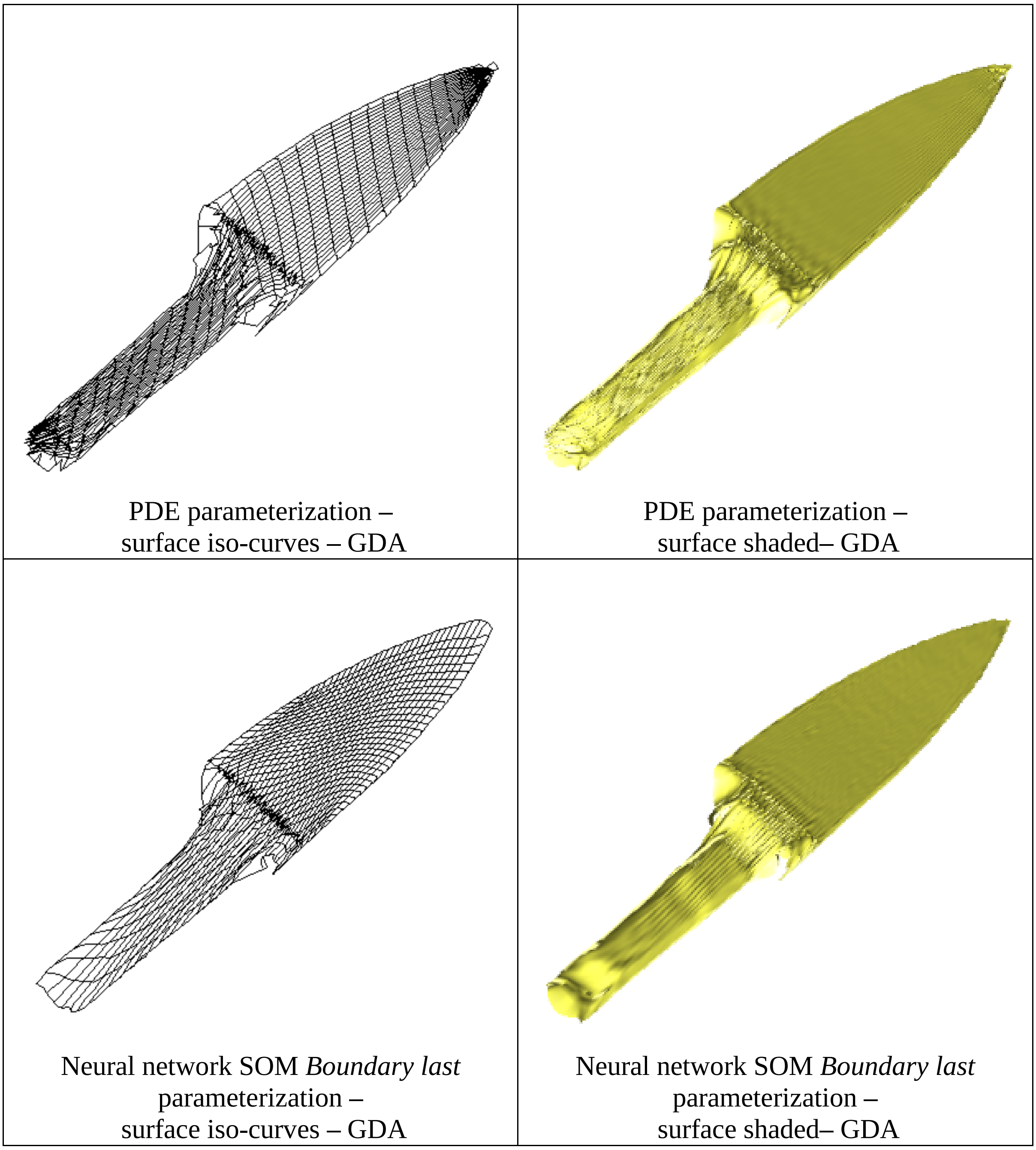
איור 25: דוגמא להתאמת משטחים למטוס.

Figure 25 and table 6 do not contain results from neural network SOM parameterization because SOM parameterization fails with concave boundaries, as can be seen in figures 19-20.

Method	Control Polygon size	Total Time	Repara-meterization Time	# of Iterations	Boundary Error	Surface Error
PDE Parameterization						
CMA	52x52	22 sec	-	1	0.3553214	0.196429
GDA	52x52	340 min	~300 min	5	0.1222228	0.085556
RSEC	52x52	200 min	~160 min	3	0.3345834	0.077747
Neural network SOM "Boundary last" Parameterization						
CMA	52x52	22 sec	-	1	0.4404472	0.1332435
GDA	52x52	235 min	~160 min	3	0.1419249	0.0651551
RSEC	52x52	195 min	~160 min	3	0.3996531	0.0610476

Table 7: Comparison of approximation methods for a knife
(# of sampled points = 106714; bounding box size of the model = 17 x 6 x 3 cm^3).
טבלה 7: השוואת שיטות שחזור עבור דגימת סכין. מספר הנקודות הדגומות הוא 106714 וגודל התיבה החוסמת
הוא 17x6x3 סנטימטרים מעוכבים.

The knife example (figure 26 and table 7) demonstrates how the orientation of the grid can change qualitatively. In the neural network SOM parameterization method, the points are distributed more fairly in the parametric domain, so that the resulting surface is much smoother (see the shaded handle area) as can be seen in figure 21. If the sampled points are distributed uniformly in the parametric domain, then each control point of the B-spline surface is influenced by a constant number of sampled points. As a result, areas with dense information will have many control points and will produce a surface with higher resolution in these areas. Areas with lower density will have fewer control points as support.



PDE parameterization –
– surface iso-curves – GDA

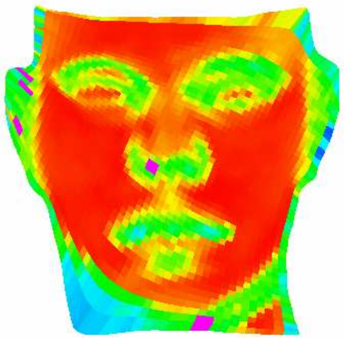
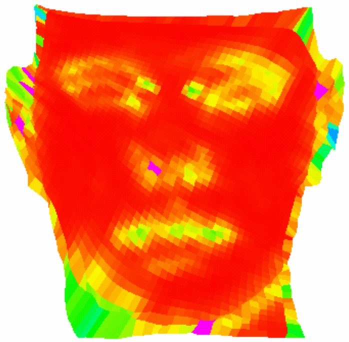
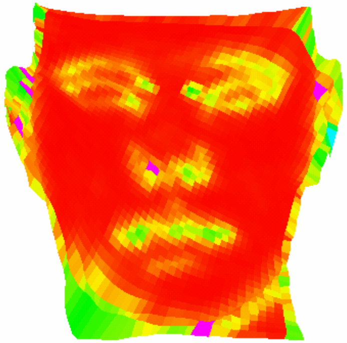
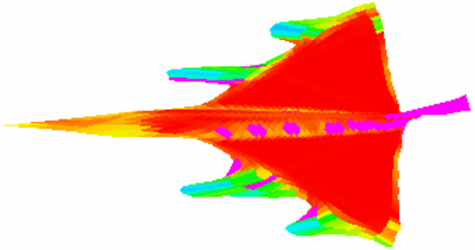
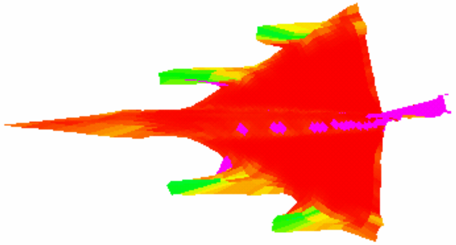
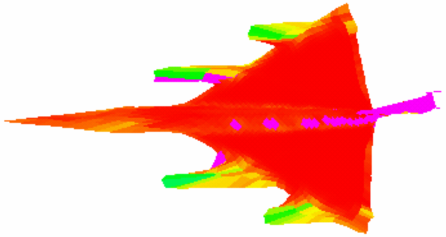
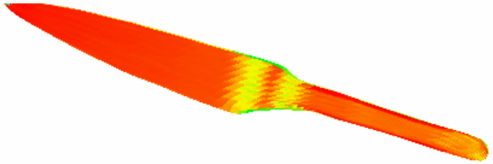
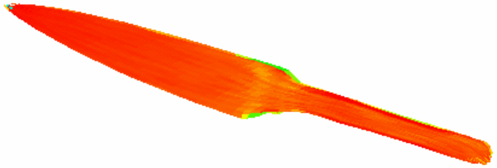
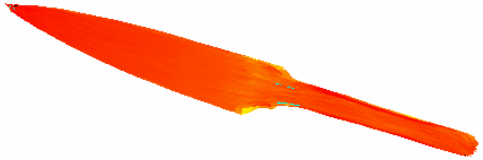
PDE parameterization –
– surface shaded– GDA

Neural network SOM *Boundary last*
parameterization –
surface iso-curves – GDA

Neural network SOM *Boundary last*
parameterization –
surface shaded– GDA

Figure 26: Example for knife surface fitting.
איור 26: דוגמא להתאמת משטחים לסכין.

The error convergence was illustrated on the different fitting methods relative to each object by error maps for PDE parameterization only (figure 27). The purple areas reflect regions with an error above a given threshold i.e., either areas with very large errors or areas with no sampled points projected on them.

	Control mesh approximation	Gradient descent	Random surface error correction
Mask			
Air-plane			
Knife			

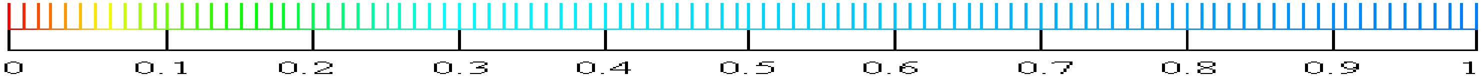
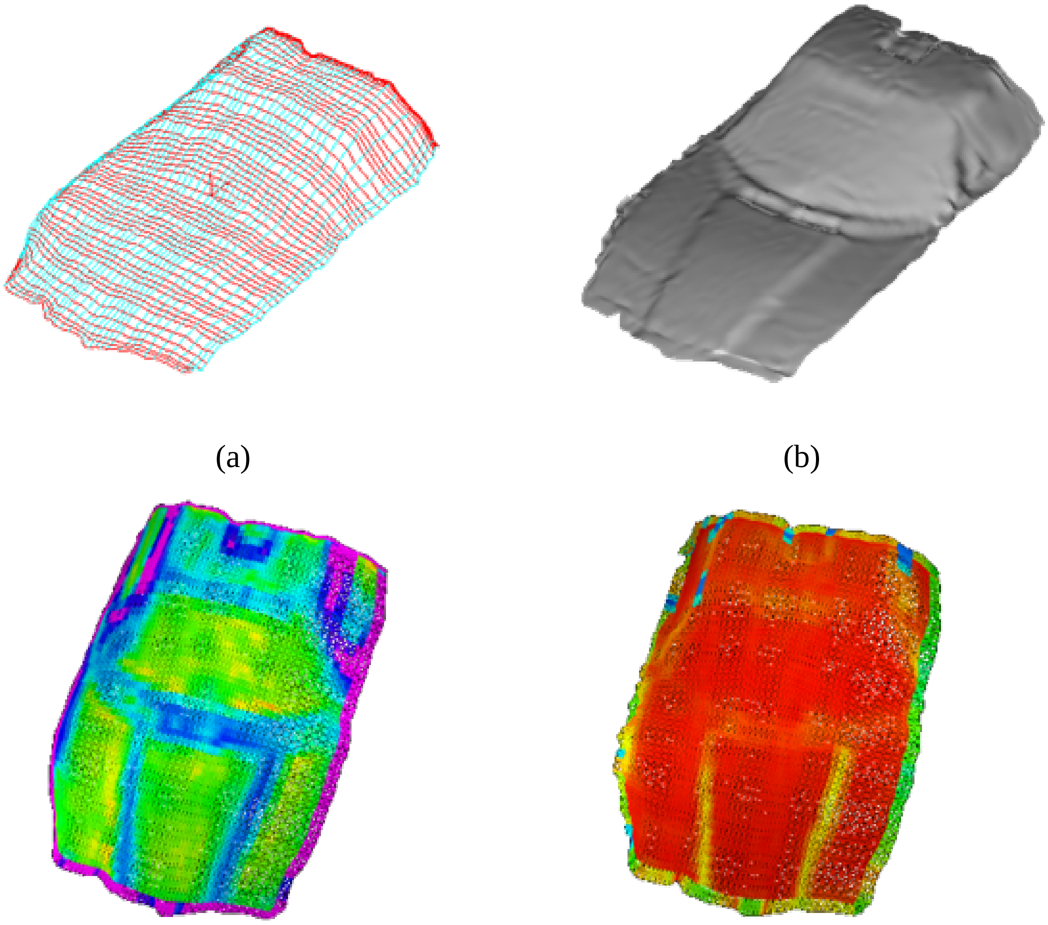


Figure 27: Error analysis of the models by the four fitting methods (error is in centimeters).

איור 27: אנליזת שגיאה של התאמת משטחי הגופים בשיטות שונות. (השגיאות מוצגות בסנטימטרים)

Although the total execution time may seem long, most of the time is spent in the optimization stage, which is the time bottleneck. Until this stage, the fitted surface is a good approximation of the original surface, and it would take no more than ten minutes. However, it is not optimal.

Previous examples demonstrated surface fitting over 2D parameterized points. Figure 28 demonstrates RSEC surface fitting over initial 3D parameterization using SOM with the *growing grid* extension. The 3D grid is used to represent the control mesh of the initial B-Spline surface. Thus, initial surface fitting by CMA is not needed. Notice that the whole reconstruction process is based solely on Neural network techniques.



(a) (b)

(c) (d)

Figure 28: The toy car range image that is composed of 40138 points is reconstructed using 3D parameterization and RSEC surface fitting. (a) The 3D grid of size 36x54 using SOM with the growing grid extension (b) The B-Spline surface reconstructed using RSEC. (c) The error map of the initial surface.(d) The error map of the surface after RSEC.

איור 28: התמונה התלת ממדית של מכונית הצעצוע מורכבת מ 40138 נקודות ומשוחזרת באמצעות
פרמטריזציה תלת ממדית עם SOM והתאמת משטחים מסוג RSEC : (a) הסריג התלת ממדי בגודל
36x54 שמשוחזר בשיטת SOM עם הרחבת growing grid. (b) משטח הB-Spline ששוחזר
באמצעות RSEC. (c) מפת השגיאות של משטח הקירוב הראשוני. (d) מפת השגיאות של המשטח
ששוחזר באמצעות RSEC.

4.1.3 Merging surfaces from multi range images

In the previous sections, reconstruction results for single range images were discussed. This section demonstrates the feasibility of the merging process described in section 3.1.3 and completes the reconstruction approach.

Figures 29a and 29b demonstrate the sampling from multiple directions of the joined sphere and cube object. Figure 29c shows the surfaces fitted to the range images. The different colors represent the scanning direction, both for the sample points and the fitted surfaces. Figure 30 demonstrates the reconstructed object after surface merging.

Figure 31 shows the merging process for the mask object that is reconstructed using two surfaces merged together. In this example the distinction curve is arbitrary and not a boundary distinction curve.

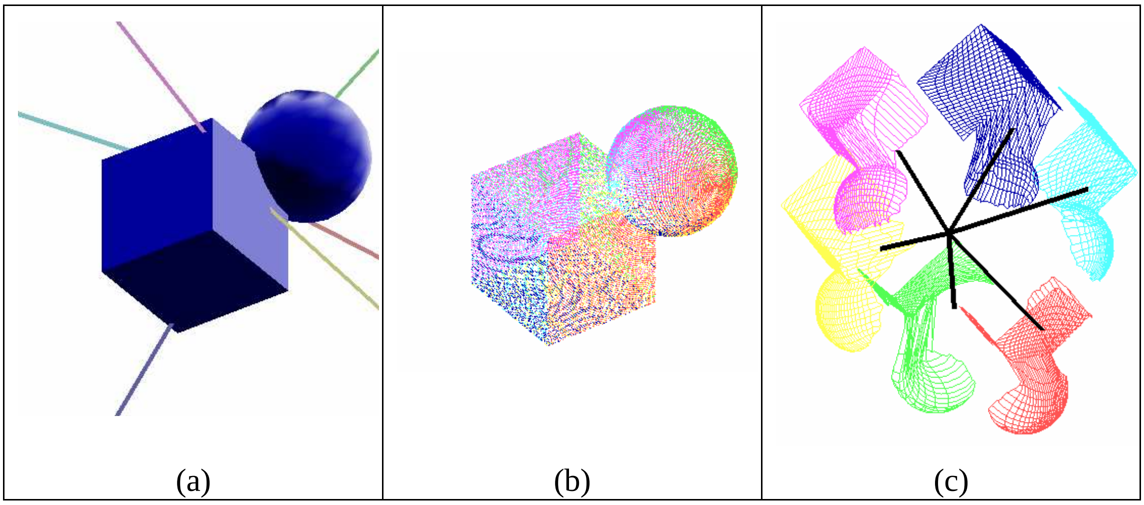


Figure 29: Multi sampling of an object: (a) The object and its sampling directions. (b) Merged range images from several directions. (c) Surfaces fitted for each range image.

איור 29: דגימה מרובת כיוונים של גוף. (a) כיווני הדגימה של הגוף. (b) איחוד תמונות תלת ממדיות ממספר כיווני דגימה. (c) משטחים משוחזרים עבור כל תמונה תלת ממדית.

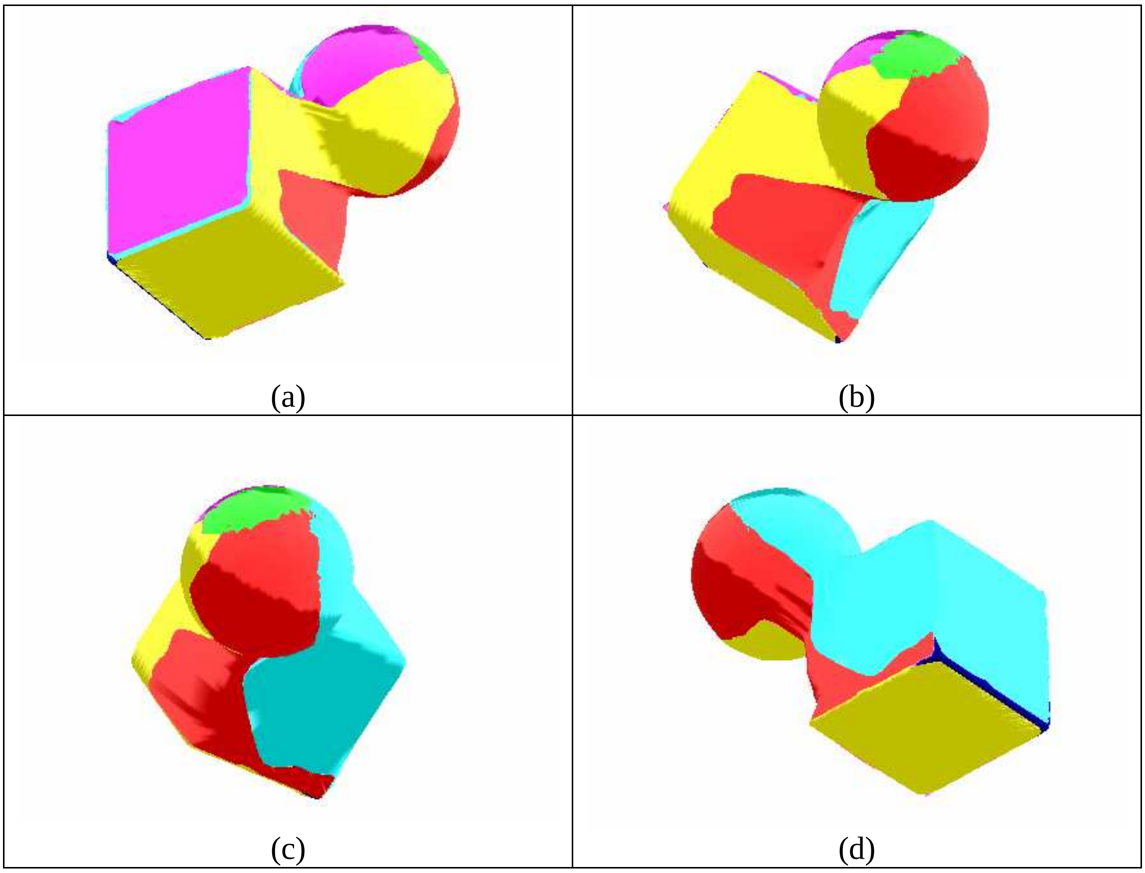


Figure 30: The sphere and cube model composed of trimmed surfaces. (a-d) Different views of the object.

איור 30: חיבור של גוף הבנוי מכדור וקוביה מחוברים הבנוי ממספר משטחים חתוכים.
(a-d) מספר מבטים שונים של הגוף.

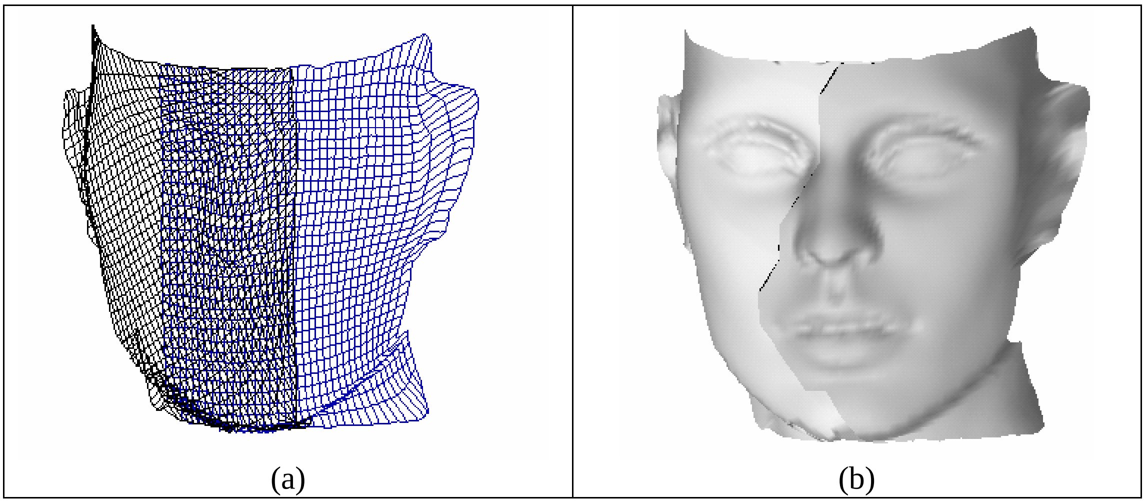


Figure 31: A mask model (a) iso-lines of two fitted surfaces. (b) merging the two surfaces.

איור 31: מודל המסיכה (a) עקומות שוות פרמטר של שני משטחים מותאמים. (b) מיזוג של שני
משטחים.

4.2 Approach B – reconstruction of a triangular mesh from a cloud of points

This section demonstrates the feasibility of the reconstruction approach discussed in sections 2.2 and 3.2 for reconstruction arbitrary topology objects as a triangular mesh.

The extended algorithm was implemented on a PC platform in visual C++ using the MFC and OpenGL libraries. The algorithm was tested on the following objects:

1. A computed cloud of points of a joined sphere and cube.
2. A computed cloud of points of a torus.
3. A sample of a toy helmet scanned by a Cyberware scanner.
4. A computed cloud of points of loops joined.

Clouds of points were computed using the IRIT solid modeler [ELB96].

The results are displayed in figures 32-35, and the properties of the samples and the *extended neural gas* algorithm are displayed in table 8. The objects were trained using the *extended neural gas* algorithm with the following parameters $\varepsilon_i = 0.3 \quad \varepsilon_f = 0.05 \quad \lambda_i = 30 \quad \lambda_f = 0.05 \quad a_i = 20 \quad a_f = 200$. Reconstruction times are presented on a computer with 1.13Ghz CPU and 256M RAM.

Object	Number of sampled points $\lvert S \rvert$	Genus	Number of neural units $\lvert V \rvert$	Run Length L	*Extended neural gas* neural network training time in seconds	Execution time of the manifold creation algorithm in seconds	Total reconstruction time in seconds
Ball & Cube	3435	0	100	40000	15	10	25
Torus	50604	1	100	80000	85	8	93
Toy Helmet	14372	1	200	80000	76	15	91
Joined Loops	72396	3	400	200000	464	40	504

Table 8: Reconstructed objects with reconstruction parameters.

טבלה 8 : גופים משוחזרים עם פרמטרי השחזור.

In figures 32-35, the range image is shown in (a) , the random initialization is shown in (b), and the progress of the *extended neural gas* algorithm is shown in (c) to (f). The mesh correction algorithm is shown afterwards with the final object.

In all the figures, the normals at the neural units are shown as yellow lines, the faces are shown lighted using the normal information derived during training, front faces are shown in green while back faces are shown in orange after reorientation. The edges of the mesh are also shown in various colors: green edges have no faces incident to them; cyan edges have one face incident to them and are boundary edges of the manifold; red edges are manifold edges that have two faces incident to them; thick purple edges are non manifold edges that have three or more faces incident to them and are deleted during mesh correction.

The sphere and cube object in figure 32 was sampled synthetically. The sample is quite sparse, and thus not many faces are created using the *extended neural gas* algorithm; however the mesh correction algorithm completes the mesh and reports a correct genus. The range image is shown in all the sub-figures for comparison.

In the torus object (figure 33), the sample is quite sparse and little work is left for the mesh correction algorithm. Actually, the mesh is complete after the face completion using local projection method, and no boundary exists. A correct genus is reported.

The toy helmet object in figure 34 is an object that was scanned and reconstructed using the suggested approach and it demonstrates the applicability of the suggested method for physical objects.

The joined loops object in figure 35 is a complex genus-3 object, where the object loops pass through the holes of the other loops. This object was successfully reconstructed with the correct genus, and it demonstrates the applicability of the algorithm for complex topological shapes.

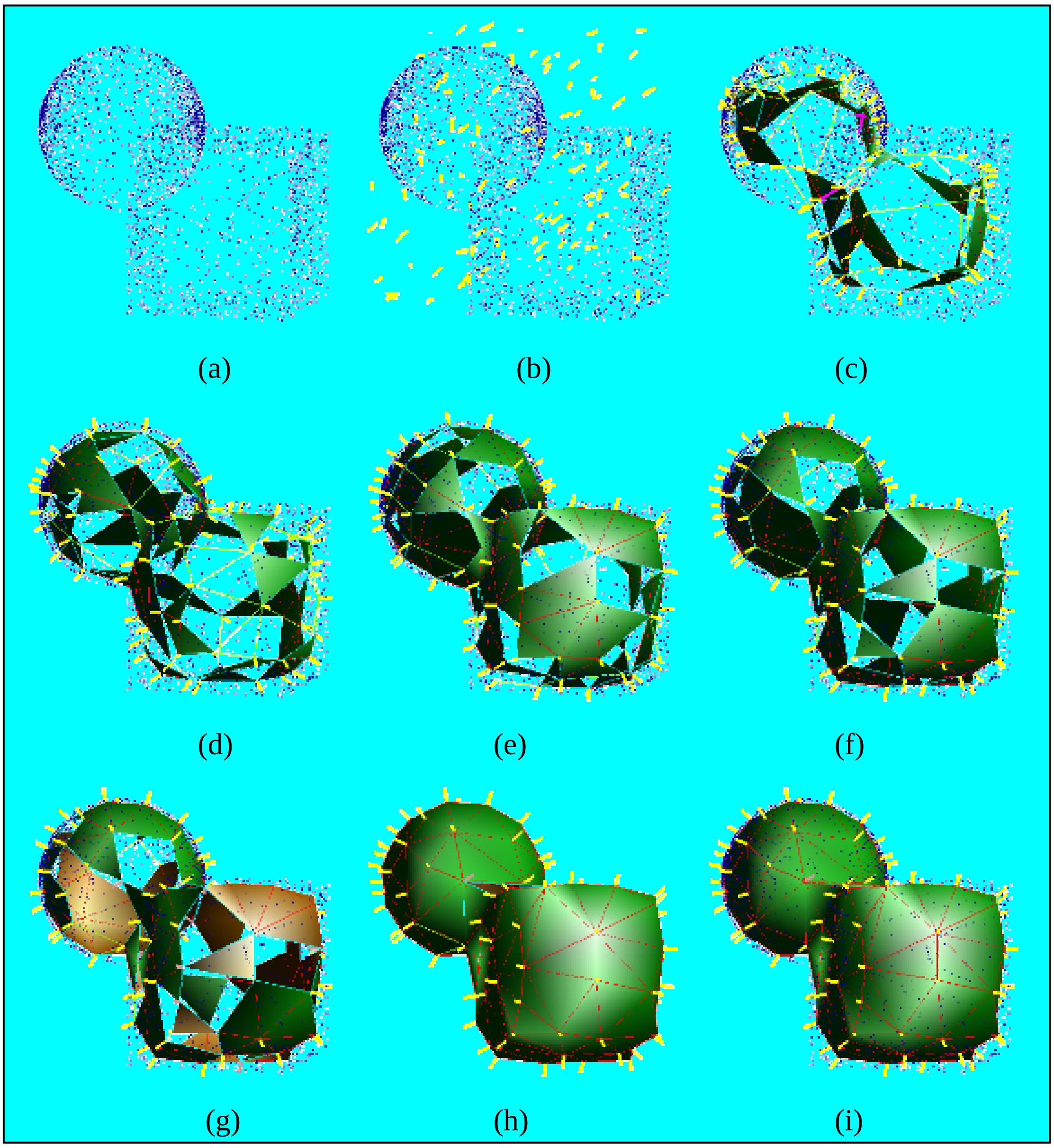


Figure 32: Reconstruction stages of a sphere connected to a cube: (a) range image; (b) initialization of the neural net; (c) after 10000 iterations; (d) after 20000 iterations; (e) after 30000 iterations; (f) after 40000 iterations; (g) after eliminating of faces not representing the volume boundary and face reorientation; (h) after completing missing faces using local projection; (i) after detecting and closing boundary loops.

איור 32: שלבי שחזור של כדור מחובר לקוביה : (a) תמונה תלת ממדית סרוקה. (b) אתחול רשת הבינה המלאכותית. (c) לאחר 10000 איטרציות. (d) לאחר 20000 איטרציות. (e) לאחר 30000 איטרציות. (f) לאחר 40000 איטרציות. (g) לאחר הסרת פאות שלא מייצגות את נפח הגוף ולאחר התאמת האוריאנטציה של הפאות. (h) לאחר השלמת פאות חסרות תוך שימוש בהטלה מקומית. (i) לאחר זיהוי וסגירת לולאות גבול.

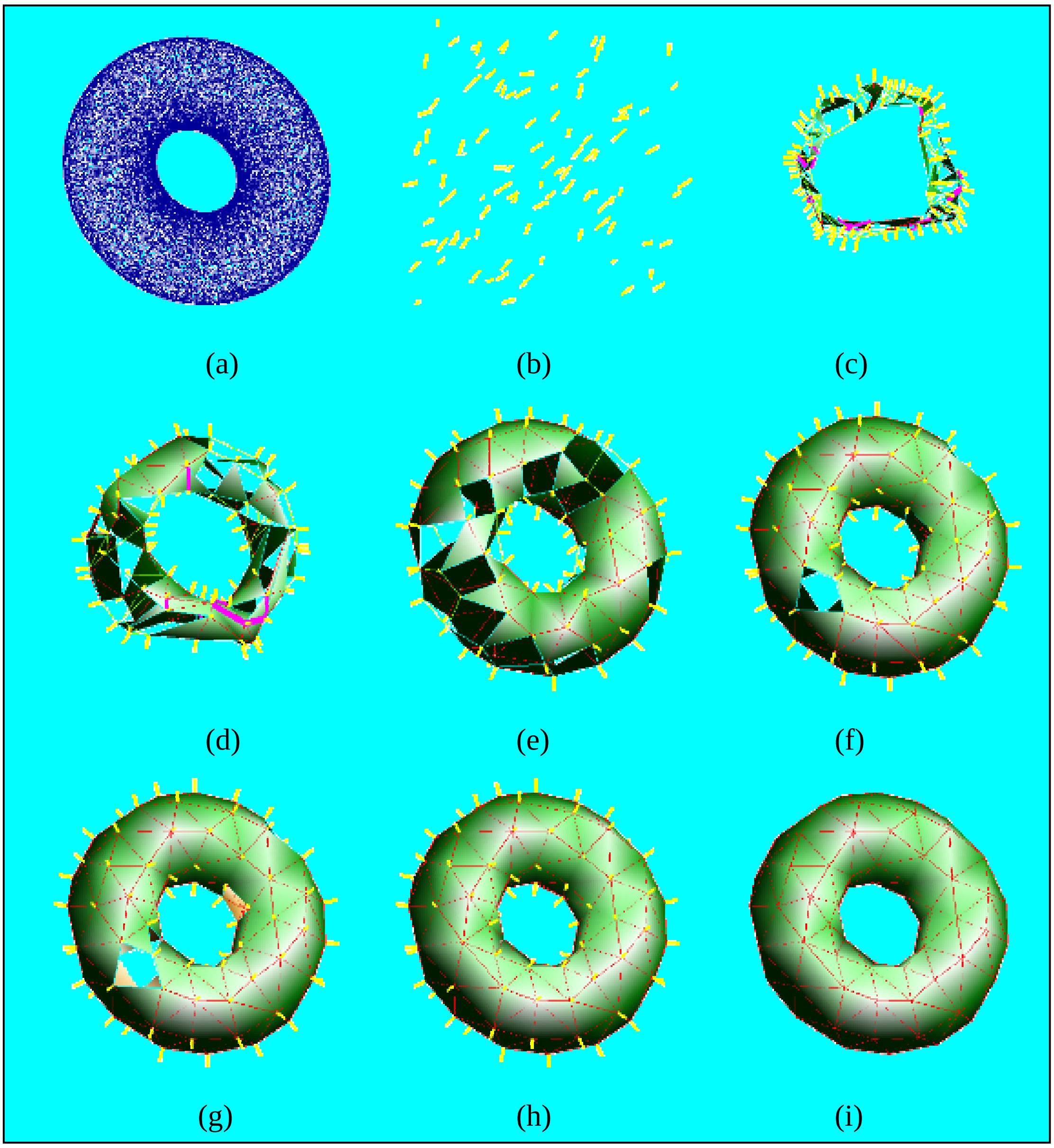


Figure 33: Reconstruction stages of a torus: (a) range image; (b) initialization of the neural net; (c) after 10000 iterations; (d) after 20000 iterations; (e) after 50000 iterations; (f) after 80000 iterations; (g) after eliminating of faces not representing the volume boundary and face reorientation; (h) after completing missing faces using local projection; (i) final object without normals.

איור 33: שלבי שחזור של טורוס : (a) תמונה תלת ממדית סרוקה. (b) אתחול רשת הבינה המלאכותית.
(c) לאחר 10000 איטרציות. (d) לאחר 20000 איטרציות. (e) לאחר 50000 איטרציות. (f) לאחר
80000 איטרציות. (g) לאחר הסרת פיאות שלא מייצגות את נפח הגוף ולאחר התאמת האוריאנטציה של
הפאות. (h) לאחר השלמת פאות חסרות תוך שימוש בהטלה מקומית. (i) גוף משוחזר ללא נורמלים.

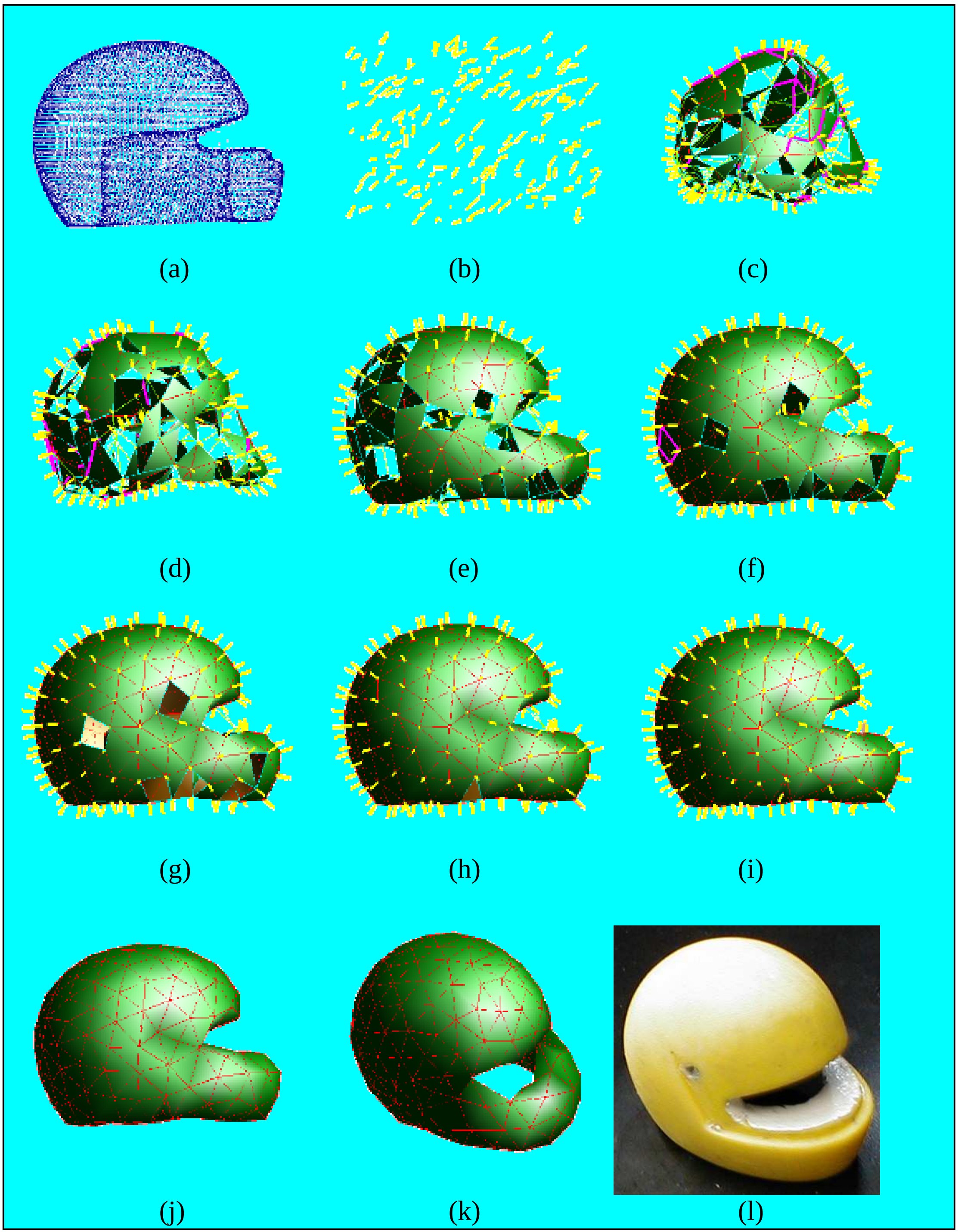


Figure 34: Reconstruction stages of a toy helmet: (a) range image; (b) initialization of the neural net; (c) after 10000 iterations; (d) after 20000 iterations; (e) after 50000 iterations; (f) after 80000 iterations; (g) after eliminating of faces not representing the volume boundary and face reorientation; (h) after completing missing faces using local projection; (i) after detecting and closing boundary loops; (j,k) final object without normals from different views. (l) original scanned object.

איור 34 : שלבי שחזור של קסדת צעצוע: (a) תמונה תלת ממדית סרוקה. (b) אתחול רשת הבינה המלאכותית. (c) לאחר 10000 איטרציות. (d) לאחר 20000 איטרציות. (e) לאחר 30000 איטרציות. (f) לאחר 40000 איטרציות. (g) לאחר הסרת פאות שלא מייצגות את נפח הגוף ולאחר התאמת האוריאנטציה של הפאות. (h) לאחר השלמת פאות חסרות תוך שימוש בהטלה מקומית. (i) לאחר זיהוי וסגירת לולאות גבול. (j,k) גוף משוחזר ללא נורמלים משני נקודות מבט שונות. (l) הגוף המקורי שנסרק.

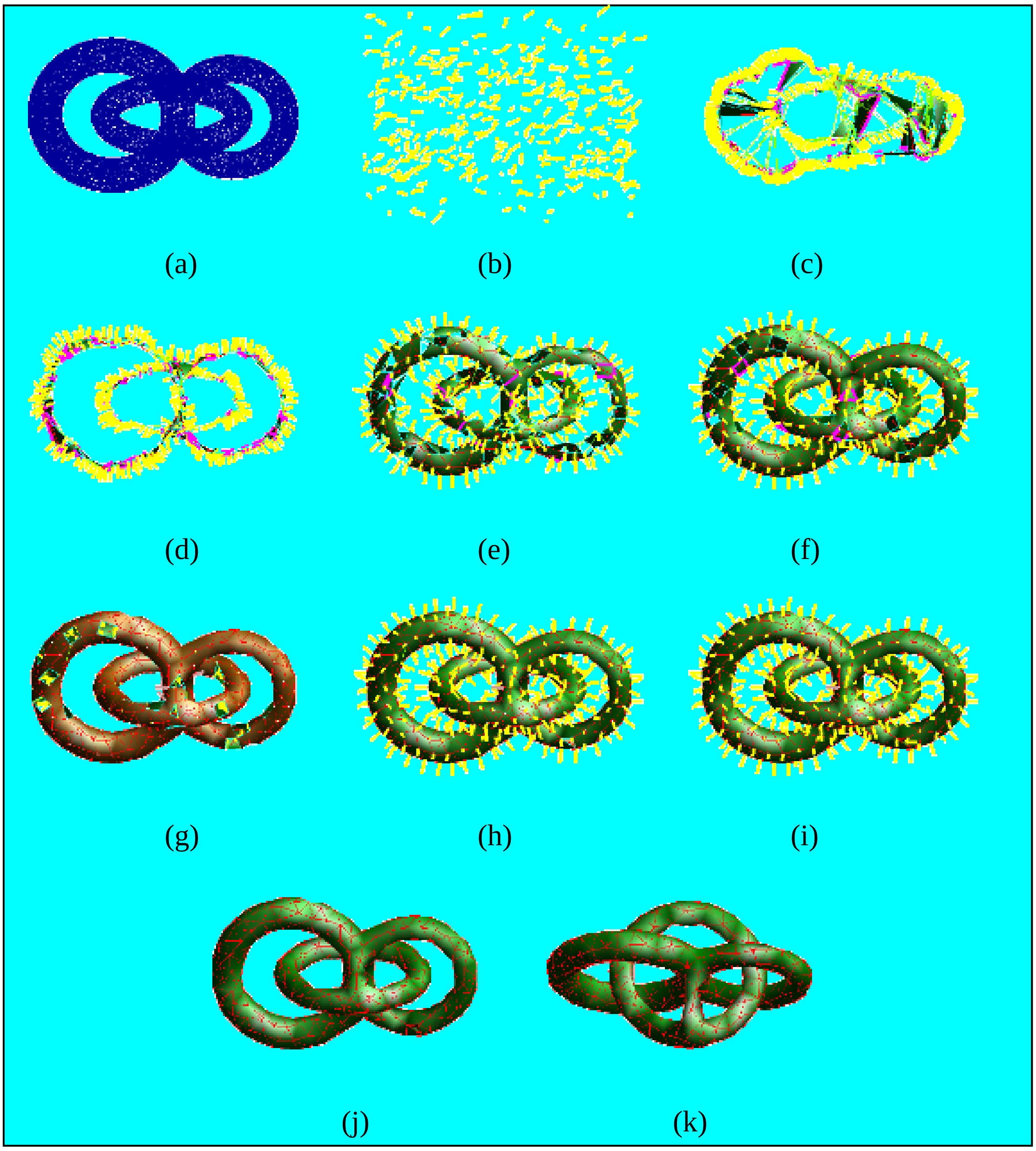


Figure 35: Reconstruction stages of joined loops: (a) range image; (b) initialization of the neural net, (c) after 10000 iterations; (d) after 50000 iterations; (e) after 100000 iterations; (f) after 200000 iterations; (g) after eliminating of faces not representing the volume boundary and face reorientation; (h) after completing missing faces using local projection; (i) after detecting and closing boundary loops; (j) final cad model; (k) final CAD model from a different view.

איור 35: שלבי שחזור של גוף הבנוי מלולאות מחוברות : (a) תמונה תלת ממדית סרוקה. (b) אתחול
רשת הבינה המלאכותית. (c) לאחר 10000 איטרציות. (d) לאחר 50000 איטרציות. (e) לאחר
100000 איטרציות. (f) לאחר 200000 איטרציות. (g) לאחר הסרת פאות שלא מייצגות את נפח הגוף
ולאחר התאמת האוריאנטציה של הפאות. (h) לאחר השלמת פאות חסרות תוך שימוש בהטלה מקומית.
(i) לאחר זיהוי וסגירת לולאות גבול. (j) גוף משוחזר. (k) גוף משוחזר מזווית אחרת.

5 Conclusions

The leap from 2D reverse engineering processes, embodied by desktop scanners and fax machines, to 3D processes is difficult. Mature methods and understanding of the problems involved are required in order to construct 3D applications. As part of this research, problems associated with reverse engineering were explored. New techniques, based on neural networks, were developed to overcome the problems.

The problems accompanying the reverse engineering process were described in previous chapters. Two complete reconstruction approaches were presented previously, one reconstructing a CAD model as B-Spline surfaces and the other as a triangular mesh. Problems encountered during the RE process, such as parameterization, surface fitting and topology detection were explained and solutions were proposed.

The neural network methodologies used here were suitable for reverse engineering tasks. In addition, this research bears a contribution to other applications by extending existing neural network methods.

This chapter discusses the advantages and shortcomings of each approach and draws conclusions.

5.1 Approach analysis

Two reconstruction approaches were presented in previous chapters. Both approaches are able to reconstruct a CAD model from multiple range images. However, they differ from each other in nature. Approach A reconstructs a model composed of trimmed B-Spline surfaces while approach B reconstructs a triangular mesh. A comparison between the proposed approaches and other methods found in the literature is summarized in table 3 in chapter 1.

Approach A employs several interchangeable methods for parameterization and surface fitting. These methods will be evaluated in section 5.2. This approach is more suitable for practical applications due to its characteristics discussed below.

The advantages of approach A can be summarized as:

Compatibility to current CAD systems – The fact that B-Spline surfaces are used makes the model highly compatible with current CAD systems.

CAD model simplicity – The CAD model is composed from a B-Spline surface for each view of the object. Thus large surfaces representing large areas of the surface are created. The CAD model will therefore be constructed with less surface patches.

Incremental addition of range images – Range images can be added incrementally to the model.

Parallelism potential – Reconstruction of the B-Spline surfaces in parallel can reduce computing time.

Pipelining potential – The whole process can be separated into small tasks that can be processed in a pipeline to reduce overall reconstruction time. For example, a pipeline can be composed of the following straightforward stations: scanning , surface reconstruction , merging. In such a pipeline three range images will be processed in parallel. Sub-stations for parameterization and surface fitting can also be added to regulate the pipeline.

The approach also has some shortcoming:

Discontinuity - Gaps left between the surfaces.

Dependence on range image order- Merging the surfaces in the order of scanning may lower accuracy.

Numerical anomalies – The process is difficult numerically, especially where surfaces with 'swimming gills' are introduced in the model.

Incomplete information - Overlapping areas between the range images are affected only by the last range image that was integrated.

Approach B introduces neural networks for solving the topology detection problem, and its main contribution is the explanation provided on the connection between the topology of the triangular mesh to the geometry of the approximated point cloud. An extension of the *Hebbian learning law* explains connections between three neural units and enables reconstruction of Delaunay triangular faces and approximation of surface normals.

This approach is more versatile in its characteristics, and paves the way for additional research such as further extension of the *Hebbian learning law* beyond 3D.

The advantages of the proposed approach consist of:

Manifold triangular mesh creation – The method constructs a triangular mesh that is manifold. The mesh can be used as a base for a freeform CAD model.

Simplicity – The algorithm is simple and easy to implement, requiring no special knowledge of neural networks.

Extendibility potential – The algorithm can be easily extended to other neural networks such as *growing neural gas* to enable changing CAD model size.

Modularity – The complete algorithm or parts of it can be used for different applications and different forms of data of higher or lower dimensions. For example, the neural gas algorithm without *Hebbian learning* can determine the density of a cloud of points in different areas. Moreover, *enhanced Hebbian learning* can be detached and applied on a sub-set of the point cloud. Also, the correction algorithm can be reformulated or changed.

The method has the following characteristics that can be viewed as shortcomings or advantages depending on the task required:

Fixed size of the reconstructed model – If the task involves compression of the model to a fixed size, this trait can be viewed as an advantage. However, the mesh may not represent geometrical or topological features adequately if the size of the mesh is too small or too large.

Local sensitivity to sampling density – when implementing the *neural gas* algorithm, the vertices of the mesh will be spread to accommodate minimal quantization error. Thus, higher density regions of the point cloud will be represented by more vertices and sparser regions with fewer vertices. This trait of fair representation can be seen as an advantage for some applications such as density detection. However, the density of the point cloud is sometime determined by the scanning and registration procedures and may not reflect meaningful information that should be preserved.

The drawbacks of the approach B are:

Non-determinism – The algorithm is random in nature, and different random seeds may produce different results and may cause the method to fail to reconstruct a manifold mesh.

Greedy nature – Both the *neural gas* implementation and the manifold creation algorithm applied are local and greedy and may cause failure of the methods.

Dependence on parameters – Optimal values for the required parameters are unknown and may vary from model to model. Rules of thumb need to be determined by empirical experimentation. However, most methods in the literature have this disadvantage.

5.2 Method analysis

Approach A is composed of several interchangeable algorithms at several stages (figure 2). In this section, parameterization is analyzed in section 5.2.1 and surface fitting is analyzed in section 5.2.2.

5.2.1 Analysis of parameterization method

The previous chapters of this work presented various parameterization approaches developed in this research. The main goal of the parameterization methods was to construct an initial parameterization that could be later optimized by parametric optimization. Three options for initial parameterization of range images were presented: PDE parameterization , 2D SOM parameterization and 3D SOM parameterization. The choice of the parameterization method may differ for different objects and requirements. A comparison of the developed approach to other approaches in the literature has already been presented in table 1. The following paragraphs will clarify the considerations for choice of the method by emphasizing the advantages and drawbacks of each method.

First, the choice between 2D and 3D parameterization approaches should be considered. The 3D parameterization approach presented is less robust than the 2D parameterization approaches. It should be chosen only when 2D parameterization is not possible due to lack of appropriate projection direction as in the case of already merged range images presented as input data or when the scanning direction is unknown. The next two sections will analyze the parameterization methods suggested and then the choice of the parameterization method will be clarified.

5.2.1.1 Analysis of PDE parameterization

The PDE approach presented is a 2D parameterization approach and is based on the Laplace equation. It has a robust mathematical explanation and a simple numerical approach.

PDE parameterization advantages:

Non self intersection – The iso-curves created by the solution of the Laplace PDE do not self-intersect. Thus, the method is suitable for range images with concave boundaries.

Determinism – The method will converge to the same solution for the same parameters.

Numerical stability – The solution of the Laplace equation using the Gauss-Seidel method is proven numerically stable due to the fact that the iterative method is employed on a weakly dominated matrix.

PDE parameterization shortcomings:

Dependence on boundary conditions – The method requires division of the boundary into four sub boundary curves.

5.2.1.2 Anaysis of SOM parameterization

When discussing SOM parameterization, some characteristics are present in both the 2D and the 3D approach. Both SOM methods are evaluated as a whole, including the extensions, such as boundary correction and growing grid.

SOM parameterization advantages:

Orientation detection – The grid orientation is determined during training and does not have to be set prior to execution of the algorithm.

Grid aspect ratio determination – The growing grid extension enables determination of the aspect ratio of the grid lattice.

Sensitivity to density – The method will strive towards uniform parametric density of the grid by minimizing quantization error.

SOM parameterization shortcomings:

Non determinism – The methods may produce different results, depending on the random seed chosen.

Boundary correction is required – The results of the basic algorithm need to be corrected at the boundary regions. This is accomplished by the boundary last algorithm and the boundary first algorithm in 2D; both methods, however, have their shortcoming, for example the tendency to protrude beyond concave boundaries for the *boundary last* algorithm and the need for predetermination of the boundary for the *boundary first* algorithm. In 3D, the boundary problem becomes more acute due to different densities and the inability to detect the boundary easily.

Sensitivity to training parameters – The SOM method requires many parameters, and although rules of thumb exists for setting those parameters, setting them requires user experience.

5.2.1.3 Choosing the parameterization method

Considering the methods advantages and shortcoming previously described, method choice criteria can be formulated. Figure 36 presents the considerations for choosing a parameterization method.

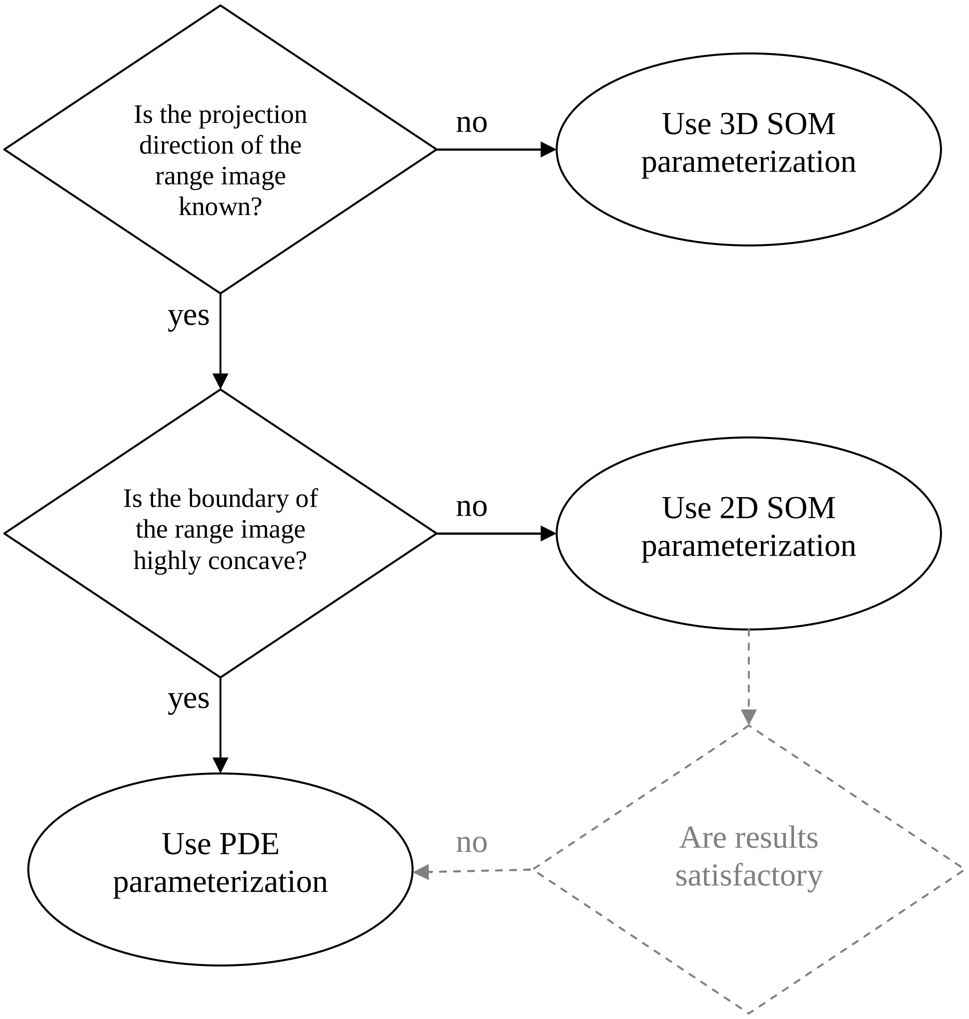

Is the projection direction of the range image known?
no
Use 3D SOM parameterization
yes
Is the boundary of the range image highly concave?
no
Use 2D SOM parameterization
yes
Use PDE parameterization
no
Are results satisfactory

Figure 36: A flow diagram for aiding the decision on a parameterization method

איור 36: תרשים זרימה המציג את תהליך הבחירה עבור שיטת הפרמטריזציה.

5.2.2 Analysis of surface fitting method

Approach A utilizes several surface fitting methods: the CMA, the GDA and the RSEC methods. The overall fitting approach was compared to other methods in the literature and is presented in table 2. As in other fast techniques, the fitting approach is built on iterative methods. A significant contribution in this research is the introduction of a good initial approximation that is achieved by 2D parameterization in conjunction with the CMA method or by the grid created with 3D parameterization.

The iterative methods used, however, compute only the positions of the vertices of the B-Spline control mesh. The fitting methods presented are applied in an alternating manner in order to optimize the fitted surface.

The CMA surface fitting method constructs a fast initial approximation of the B-Spline surface and although it is not accurate , especially at the boundaries, it provides a good approximation for the iterative methods that follow it.

The GDA algorithm uses an iterative LSQ approach in order to converge the surface to the range image. The contribution in this case is the formulation of the equations applied to the B-Spline representation.

The RSEC algorithm presented combines neural network techniques and B-Spline formulation in order to present a new surface fitting method. This method is very conservative in memory consumption. However, there is no formal proof for its convergence, and it has the drawbacks of the SOM algorithm such as non determinism. This technique is recommended in case GDA produces inadequate results, since its random nature can improve results.

Both iterative methods in conjunction with parametric optimization also provide an error estimation mechanism. The distance of the sampled points from the surface is calculated during the projection stage and convergence can be estimated.

5.3 General conclusions

The implications of 3D RE technology will provide significant aid to the field of engineering. The ability to reliably scan , digitally store , send and restore geometrical and topological information about 3D objects will enable closer interaction between designers, manufacturers and end users in industrial applications.

For this to occur, RE processes should be researched further. The difficulties encountered may change when different scanning techniques produce different kinds of data to be processed such as MRI data. However, the task will remain the same: to provide a machine with the ability to comprehend a physical object and reconstruct it as a digital model that can be later reproduced. Learning techniques similar to those of neural networks will probably assist the understanding process the machine undergoes.

6 Bibliography

AME98 Amenta N. ; Bern M. ; Kamvysselis M. : *A New Voronoi-Based Surface Reconstruction Algorithm*. Proc. SIGGRAPH 1998 : 415-421.

AME99 Amenta N. ; Choi S. : *One-Pass Delaunay Filtering for Homeomorphic 3D Surface Reconstruction*. TR99-08. University of Texas, 1999.

ABR96 Abrantes A.J.; Merques J.S.: *Unified Approach to Snakes, Elastic Nets and Kohonen Maps*. IEEE Int. Conf. Acoustics, Speech and Signal Proc., Detroit, EUA: 3427-3430 , 1995.

ALH99 Alhanaty M.; Bercovier M. : *Curve and Surface Fitting and Design by Optimal Control Methods*. Hebrew University, Leibniz center technical report, No. 99-29, 1999.

BAJ95 Bajaj C. ; Bernardini F. ; Xu. G. *: Automatic Reconstruction of Surfaces and Scalar Fields from 3D Scans*. SIGGRAPH 1995 pp. 109-118.

BAR00 J. Barhak, A. Fischer, *Multi - Reconstruction of a 3D Freeform Object From Range Images,* The International Journal for Manufacturing Science & Production Vol 3 Nos. 2-4, 2000.

BAR1 J. Barhak, A. Fischer, *Parameterization and Reconstruction from 3D Scattered Points Based on Neural Network and PDE Techniques*, IEEE Transactions on Visualization & Computer Graphics, Vol. 7, No.1, January-march, 2001 p.1-16.

BAR2 J. Barhak, A. Fischer, *Adaptive Parameterization for Reconstruction of 3D Freeform Objects from Laser-Scanned Data based on a PDE approach*, The Visual Computer Vol. 17 No. 6 , August 2001 p. 351-369.

BAR3 J. Barhak, A. Fischer, *Adaptive Reconstruction of Freeform Objects with 3D SOM Neural Network Grid* , Pacific Graphics 2001 October 16-18, 2001, Tokyo, Japan

BAR4 J. Barhak, A. Fischer, *Adaptive Reconstruction of Freeform Objects with 3D SOM Neural Network Grids*, to be published in: Special Issue on Geometrical Modeling and Computer Graphics, in Journal of Computers & Graphics, vol. 26, no. 5, 2002.

BER96 Bergevin R. ; Soucy M. ; Gagnon H. ; Laurendeau Denis : *Towards a General Multiview Registration Technique*. IEEE Transactions on pattern analysis and machine intelligence V. 18(5): 540-547, 1996.

BOI93 Boissonnat J. D. ; Devillers O. ; Teillaud M. : *A Semi-Dynamic Construction of Higher Order Voronoi Diagrams and its Randomized Analysis*. Algorithmica 9:329-356, 1993.

BOL91 Bolle R.M.; Vemuri B.C.; *On Three Dimensional Surface Reconstruction Methods*. IEEE PAMI V.13(1) 1-13, 1991.

BOR00 Borghese N.A. ; Ferrari S. : *Mesh Construction with Fast Soft Vector Quantisation*. Proc. Int. Joint Conference on Neural Networks, Como, 24-27 July 2000, Vol. 5, pp. 473-478.

BRO94 Bro-Nielsen M. : *Active Nets and Cubes*. http://www.imm.dtu.dk/documents/ftp/tr94/tr13_94.abstract.html , 1994.

CAR01 Carr J. C. ; Beatson R. K. ; Cherrie J. B. ; Mitchell T. J. ; Fright W. R. ; McCallum B.C. ; Evans T. R. : *Reconstruction and Representation of 3D Objects with Radial Basis Functions*. SIGGRAPH 2001, pp 67-76,

CHA99 Chalmond B.; Girard S.C. : *Nonlinear Modeling of Scattered Multivariate Data and its Application to Shape Change*. IEEE PAMI V. 21(8) 1999.

CHE92 Chen Y. ; Medioni G. ; *Object Modeling by Registration of Multiple Range Images*. Image and Vision Computing V. 10(3): 145-155, 1992.

CHE96 Chen S.W.; Stockman G.C.;Chang Kuo-En*: SO Dynamic Deformation for Building of 3-D Models*. IEEE Transactions on Neural Networks V.7(2), 374-387 :1996.

COH93 Cohen L.D.; Cohen I.*: Finite-Element Methods for Active Contour Models and Balloons for 2-D and 3-D images*. IEEE PAMI V.15(11): 1131-1147, 1993.

COM95 Comtat C.; Morel C.: *Approximate Reconstruction of PET data with a Self-Organizing Neural Network*. IEEE Transactions on Neural Networks V.6(3), 783-789 :1995.

CUR96 Curless B; Levoy M. : *A Volumetric Method for Building Complex Models from Range Images*. SIGGRAPH 1996:303-312.

DEL97 Dellahy I.O. ; Shariat B. ; Vandorpe D. *: Conformal Mapping for the Parameterization of Surfaces to fit Range Data*. Product modeling for computer integrated design and manufacture. Chapman & Hall 1997 pp.263-272.

DIC97 Dickinson S.J. ; Metaxas D. ; Pentland A. : *The Role of Model Based Segmentation in the Recovery of Volumetric Parts from Range Data*. IEEE transactions on pattern analysis and machine intelligence V. 19(3): 259-267, 1997.

DUA01 Duan Y. ; Hong Q : *Intelligent Balloon: A Subdivision-Based Deformable Model For Surface Reconstruction Of Arbitrary Topology*. Proceedings of the Sixth ACM Symposium on Solid Modeling and Applications, p. 47-58, Ann Arbor, Michigan, June 2001

ECK96 Eck M. ; Hoppe H. : *Automatic Reconstruction of B-spine Surfaces of Arbitrary Topological Type*. SIGGRAPH 1996: 325-334.

EDE87 Edelsbrunner H. : Algorithms in Combinatorial Geometry. Springer 1987

EDE94 Edelsbrunner H. ; Mucke E. P. : *Three-dimensional Alpha Shapes*. ACM Transactions on Graphics, 13(1):43-72 , 1994.

ELB96 Elber G.: *User's manual IRIT a solid modeling program version 7.0* . http://www.cs.technion.ac.il/~irit, 1996

FIS99 Fischer A.; Barhak J.: *B-spline Surface Reconstruction From a Single Range Image Using Parameterization Based on PDE Solution.* TME Report – 461,Technion Institute of technology, 1999.

FIS99 Fischer A.; Manor A.; and Barhak J.: *Adaptive Parameterization for Reconstruction of 3D Freeform Objects from Laser-Scanned Data.* IEEE Pacific Graphics '99, Seoul, Korea, October 5-7, 1999.

FIT97 Fitzgibbon A.W. ; Eggert D.W. ; Fisher R.B. : *High-Level CAD Model Acquisition from Range Images*. Computer-Aided Design V. 29(4): 321-330, 1997.

FOI97 Foigelman L. : *Parameterization and Approximation of a Set of Ordered Points in a Plane and in Space*. Research Thesis, Technion Institute of technology , 1997.

FRI95 Fritzke B. : *A Growing Neural gas Network Learns Topologies*. Advances in Neural Information Processing Systems 7. Editors G. Tesauro, D. S. Touretzky, T.K. Leen. MIT Press, 1995.

FRI97 Fritzke B.: *Some Competitive Learning Methods*. Institute for neural computation Ruhr-Universitat Bochum – http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn/JavaPaper/ report draft 1997.

GEO91 George P.L.: *Automatic Mesh Generation – Application to Finite Element Methods*. (trans) John Wiley & Sons 1991.

GER89 Gerald C.F. ; Wheatley P.O. : *Applied Numerical Analysis*, Addison-Wesley publishing Company, 1989.

GRI95 Grimm C.M; Huges J.F. *: Modeling Surfaces of Arbitrary Topology using Manifolds*. . SIGGRAPH 1995: 359-368.

GU95 Gu P., Yuan X.; *Neural Network Approach to the Reconstruction of Freeform Surfaces for Reverse Engineering*. Computer Aided Design V.27(1):59-69, 1995.

GUO97 Guo B. : *Surface Reconstruction from Points to Splines*. Computer-Aided Design V. 29(4): 269-277, 1997.

GUP95 Gupta K. ; Xu Z. : *Complete 3D Boundary Representation from Multiple Range Images: Exploiting Geometric Constraints*. Robotica V. 13: 339-349 1995.

HAY94 Haykin S. : Neural Networks - A Comprehensive Foundation. Prentice Hall, 1994.

HIG95 Higuchi K. ; Herbert M. ; Ikeuchi K. : *Building 3-D Models from Unregistered Range Images*. Graphical models and image processing V. 57(4): 315-333, 1995.

HOF87 Hoffman R. ; Jain A.K.; *Segmentation and Classification of Range Images*. IEEE PAMI V. 9(5): 608-620, 1987.

HOP92 Hoppe H. ; DeRose T. ; Duchamp T. ; McDonald J. ; Schweitzer J. ; Werner S. : *Surface Reconstruction from Unorganized Points*. SIGGRAPH 1992 :71-78.

HOP94 Hoppe H. ; DeRose T. ; Duchamp T. ; Halstead M. ; Hubert J. ; McDonald J. ; Schweitzer J. ; Werner S. : *Piecewise Smooth Surface Reconstruction*. SIGGRAPH 1994: 295-301.

HOS88 Hoscheck J. : *Intrinsic Parameterization for Approximation*. Computer Aided Geometric Design. V. 5. 27-31, 1988.

HOS89 Hoscheck J.; Schneider F.-J. ; Wassum P. *: Optimal Approximate Conversion of Spline Surfaces*. Computer Aided Geometric Design. V. 6. 293-306, 1989.

HOS93 Hoscheck J. ; Lasser D. ; Schumaker L. (trans) : *Fundamentals of Computer Aided Geometric Design*. A K Peters 1993.

JON74 Jones R.E.; *QMESH: Self –Organizing Mesh Generator Program.* SLA-73-1088 Report Sandia Labs. 1974.

KAS87 Kass M; Witkin A.; Terzopoulos D.: Snakes: *Active Contour Models*. Proceedings of the First International Conference on Computer Vision, London, UK, June 1987, IEEE Computer Society Press: 259-268.

KRI96 Krishnamurthy V.; Levoy M.: *Fitting Smooth Surfaces to Dense Polygon Meshes*. SIGGRAPH 1996:313-324.

KOH97 Kohonen T., *Self-Organizing Maps*, Springer, 1997.

LAU93 Laurent-Gengoux P. ; Mekhilef M. : *Optimization of a NURBS Representation*. Computer Aided Design. V. 25(11). 699-710, 1993.

LEE97 Lee S. ; Wolberg G. ; Shin S.Y. ; : *Scattered Data Interpolation With Multilevel B-Splines*. IEEE transactions on visualization and computer graphics, V. 3(3) : 228-244, 1997.

LEV Levin D. : *Mesh-independent surface interpolation* . To appear in Advances in Comp. Math.

LOR87 Lorensen W.E.; Cline H.E.; *Marching Cubes: A High Resolution 3D Surface Construction Algorithm.* ACM Computer Graphics, V. 21 , pp. 163-169, 1987.

MA95 Ma W.; Kruth J P. : *Parameterization of Randomly Measured Points for Least Squares Fitting of B-Spline Curves and Squares*. Computer-Aided Design V. 27(9): 663-775, 1995.

MAN98 Manor A. : *Reconstruction of 3D models through Integration of Image Processing Techniques in Reverse Engineering*. Research Thesis, Technion Institute of technology, 1998.

MAR91 Martinetz T. ; Schulten K. : *A “Neural-Gas” Network Learns Topologies*. Artificial Neural Networks. Editors: T. Kohonen; K. Makisara; O. Simula; J. Kangas. Elsevier, 1991.

MAR94 Martinetz T. ; Schulten K. : *Topology Representing Networks*. Neural Networks Vol 7, No. 3, pp. 507-522, 1994.

MEE00 Meenakshisundaram G. ; Krishnan S. : *A Fast and Efficient Projection-Based Approach for Surface Reconstruction*. International Journal of High Performance Computer Graphics, Multimedia and Visualization, Vol. 1, No. 1, pp. 1-12, 2000.

MEN96 Menuq C. ; Chen F.L. : *Curve and Surface Approximation from CMM Measurement Data*. Computers in engineering. V. 30(2): 211-225, 1996.

MIL97 Milroy M.j. ; Bradly C. ; Vickers G.W. : *Segmentation of Wrap Around Model Using an Active Contour*. Computer -Aided Design V. 29(4): 299-320, 1997.

NIC95 Nickolas S.; Sapidis N.S. ; Besl P.J. : *Direct Construction of Polynomial Surfaces from Dense Range Images through Region Growing*. ACM Transactions on graphics V.14(2) : 171-200 , 1995.

PAR95 Park H. ; Kim K. : *An Adaptive Method for Smooth Surface Approximation to Scattered 3D Points*. Computer-Aided Design V. 27(12): 929-939, 1995.

PIE97 Piegl L. ; Tiller W.: *The Nurbs Book*, Springer, 1997.

PRE85 Preparata F. P. ; Shamos M. I. :Computational Geometry an Intruduction. Springer 1985.

PRE89 Press W.H.; Flannery B.P.; Teukolsky S.A.; Vetterling W.T.: *Numerical Recipes*, Cambridge University Press, 1989.

PRI93 Pritschow G. ; Ioannides M. : *Volume oriented Digitalizing – Modeling and - Processing of Sculptured Surfaces*. Production Engineering V. I/1: 205-210, 1993.

RAN97 Randrup T. ; Pottmann H. : *Rotational and Helical Surface Approximation for Reverse Engineering*. Technical report No. 43 institute fur geometrie Technische universitat wien 1997.

ROG89 Rogers D.F.; Fog N.G. : Constrained B-Spline Curve and Surface Fitting. Computer-Aided Design V. 21(10):641-648 , 1989.

RUT94 Rutishauser M. ; Stricker M. Trobina M. : *Merging Range Images of Arbitrary Shaped Objects*. proc. 1994 IEEE PAMI: 573-580.

SAR91 Sarkar B. ; Menq C.H. : *Parameter Optimization in Approximating Curves and Surfaces to Measurement Data*. Computer Aided Geometric Design. V. 8. 267-290, 1991.

SIN93 Sinha S.S. ; Senevirante P. : Single Valuedness , *Parameterization and Approximating 3D Surfaces Using B-splines*. SPIE V. 2031 geometric methods in computer vision II (1993): 193-203.

SPE98 Speer T.; Kuppe M. ; Hoschek J.: *Global Reparameterization for Curve Approximation*. Computer Aided Geometric Design. V.15. 869-877, 1998.

SUZ99 Suzuki H.; Takeuchi S.; Kanai T.: Subdivision Surface Fitting to a Range of points. IEEE proc. Of the seventh pacific conference on computer graphics and applications: 158-167, 1999.

TUR94 Turk G. ; Levoy M. : *Zippered Polygon Meshes from Range Images*. Computer graphics proceedings, annual conference series, SIGGRAPH, 1994:311-318.

VAI97 Vainer M. : *Parameterization and Approximation of Spatially Scattered Point Set.* Research Thesis. Technion Institute of technology , 1997.

VAR97 Varady T. ; Martin R.R. ; Cox J. : *Reverse Engineering of Geometric Models – an Introduction*. Computer-Aided Design V. 29(4):255-268 , 1997.

VAR99 Várady L. , Hoffmann M. and Kovács E. : Improved Free-form Modelling of Scattered Data by Dynamic Neural Networks. Journal for Geometry and Graphics, 3(2):177-181, 1999.

WAN94 Wani M.A. ; Batchelor B.G. : Edge Region Based Segmentation of range images. IEEE PAMI Vol 16. No. 3 , March 1994.

WIL92 Williams D.J; Shah M.*: A Fast Algorithm for Active Contours and Curvature Estimation*. CVGIP: Image understanding. V.55(1): 14-26, 1992.

שיחזור גופים מפוסלים בעלי טופולוגיה כלשהי מתוך תמונות תלת-ממדיות

יעקב ברהק

שיחזור גופים מפוסלים
בעלי טופולוגיה כלשהי
מתוך תמונות תלת-ממדיות

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר
דוקטור לפילוסופיה

יעקב ברהק

הוגש לסנט הטכניון - מכון טכנולוגי לישראל

כסליו תשס"ג　　　חיפה　　　דצמבר 2002

המחקר נעשה בהנחיית ד"ר ענת פישר בפקולטה להנדסת מכונות

אני מודה לטכניון על התמיכה הכספית הנדיבה בהשתלמותי

בנוסף אני מודה
למשפחתי שתמכה בי לאורך כל הדרך
ולמורים מצויינים לאורך כל הדרך

אני מודה לפרופסור אלכס אורון על האימון המתמטי שהעניק
אני מודה לפרופסור דוד דגני שהדריך אותי למתודולוגיות מתאימות
אני מודה לפרופסור גרשון אלבר על עצותיו והערותיו המועילות
אני מודה לפרופסור משה שפיטלני על שהראה את הכיוון

ובראש ובראשונה אני מודה למנחתי, ד"ר ענת פישר על הנחייתה ותמיכתה הרציפה

תוכן עניינים

תוכן עניינים

רשימת איורים

רשימת איורים

רשימת איורים

רשימת טבלאות

תקציר

שחזור הנדסי (RE) הוא התהליך של שחזור מודל ממוחשב מתוך גוף סרוק. חשיבותו של תהליך השחזור ההנדסי למערכות תיב"מ עלה לאחרונה עקב יכולת לשחזר גופים תלת מימדים מסובכים. טכנולוגיית השחזור הנדסי נותנת מענה לדרישות העולות בתחום התיב"מ עבור שיפור תכנון של מוצרים קיימים.

בשחזור הנדסי, משתמשים בסורקי ליזר עקב יכולתם לסרוק תמונות תלת ממדיות בדיוק ובמהירות יחסית לטכנולוגיות אחרות. תמונה תלת ממדית שנלקחה ממבט מסויים של הגוף (range image) מורכבת ממספר רב של נקודות המתארות את משטח הגוף הסרוק במרחב התלת ממדי. ישנן מספר בעיות פתוחות בספרות שנחשבות לצוואר בקבוק בתהליך השחזור ההנדסי: (1) כמות המידע הנוצרת במהלך הסריקה היא גדולה. (2) הדגימה מכילה רעש. (3) הטופולוגיה אינה ידועה והקישוריות בין הנקודות הסרוקות אינה תמיד מוגדרת. בנוסף לכך, הגוף המשוחזר צריך להתאים למערכות תיב"מ עכשוויות.

בספרות קיים מספר רב של גישות שונות לפתרון הבעיה. שיטות אלו מסובכות לישום ולרב מבוססות על שיטות יוריסטיות. הפתרון מבוצע בדרך כלל במספר שלבים עיקריים והם : (1) סריקת הגוף ממספר כיוונים שונים. (2) התאמת מערכות הצירים של המבטים למערכת עולם (registration). (3) זיהוי הטופולוגיה של הגוף שמיוצגת בדרך כלל כרשת משולשים. (4) חלוקת הגוף לאיזורים שמייצגים משטחים עם מכנה משותף (segmentation). (5) התאמת משטחים למעטפת הגוף.

מחקר זה מציג גישה חדשה המשתמשת ברשתות בינה מלאכותיות בכדי להתגבר על בעיות השחזור. רשתות בינה מלאכותיות בנויות מיחידות חישוב פשוטות ואוטונומיות הנקראות נוירונים המקושרות זו לזו. המודל מהווה רשת חישוב מקבילית הלומדת מדוגמאות בדומה לתהליכים המתרחשים במוח האדם. רשתות בינה מלאכותיות לומדות ע"י תהליך אימון שבמהלכו מוצגות להן דוגמאות. בסוף תהליך הלימוד רשת הבינה המלאכותית מסוגלת לספק מידע על התופעה שיצרה את הדוגמאות. אחת משיטות האימון של רשתות בינה מלאכותיות הינה למידה תחרותית (competitive learning). בשיטה זו הנוירונים מייצגים מיקומים גיאומטריים במרחב ובזמן תהליך האימון, הנוירונים מסתדרים במרחב לפי הדוגמאות הנתונות ולפי המבנה הטופולוגיה של רשת הבינה המלאכותית.

תכונת הכללת הדוגמאות של רשתות בינה מלאכותיות הופך אותן מתאימות לתהליכי שחזור הנדסי. במחקר זה האינפורמציה על המשטח נתונה ע"י אוסף נקודות שמהוות דוגמאות שנלקחו מהמשטח. מחקר זה משתמש ומרחיב רשתות בינה מלאכותיות המשתמשות בלימוד תחרותי, כגון *self-organizing maps* (SOM) ו - *neural gas* .

בעבודה זו מוצגות שתי שיטות לשחזור גופים ממספר תמונות תלת ממדיות לאחר התאמה למערכת צירים אחת. שתי הגישות מבוססות על שימוש ברשתות בינה מלאכותיות.

בגישת השחזור הראשונה, משטח B-Spline פרמטרי משוחזר עבור תמונה תלת ממדית בנפרד לאחר מכן המשטחים מאוחדים בכדי לשחזר את המעטפת של הגוף הנפחי. לצורך שחזור המשטחים, פותחו שיטות חדשות הפותרות את בעיות הפרמטריזציה (Parameterization) והתאמת המשטחים (Surface fitting).

הפרמטריזציה מגדירה קשרי שכנות בין נקודות סרוקות תחת טופולוגיה של משטח. בכדי להתאים משטח פרמטרי לנקודות יש צורך בסידורן באמצעות פרמטריזציה, ואולם פרמטריזציה אופטימלית דורשת ידיעת המשטח. בעיה זו דומה לבעיית הביצה והתרנגולת ופתרון התחלתי עבור פרמטריזציה מסופק ע"י אחת מהשיטות הבאות:

(1) הנקודות מוטלות בכיוון הדגימה ונוצר סריג בעל מבנה טופולוגי ריבועי המגדיר קשרי שכנות בין הנקודות הדגומות. הסריג נוצר על ידי פתרון משוואת Laplace שהינה משוואה דיפרנציאלית חלקית. בסריג זהו עקומות הרשת האופקיות והאנכיות אינן נפגשות בינן ובין עצמן. סריג זה מתאים במיוחד לגופים בעלי גבול קעור במיוחד.

(2) הנקודות מוטלות בכיוון הדגימה ונעשה שימוש ברשת הבינה המלאכותית SOM בכדי לייצר סריג בעלת טופולוגיה ריבועית. הרשת מאתרת את האוריינטציה בצורה אוטומטית בדו מימד תוך כדי שמירה על ייצוג צפיפות המתאים לצפיפות הדגימה.

(3) שיטת ה SOM מאפשרת הרחבה לפרמטריזציה תלת ממדית שבה נוצר קירוב התחלתי למשטח הסופי. כמו כן הורחבה השיטה ל-Growing Grid שבה נוספים שורות וטורים לסריג הריבועי בזמן גדילת הסריג.

שיפור לפתרון ההתחלתי של הפרמטריזציה נעשה ע"י תהליך מחזורי של הטלת נקודות הדגימה על המשטח ההתחלתי המקורב והתאמת משטח מחדש.

התאמת משטחים מתבצעת ע"י חישוב קירוב ראשוני למשטח ושיפור הפתרון בשיטות איטרטיביות. פתרון ראשוני מהיר מתקבל ע"י שיטת Control Mesh Approximation (CMA) לאחר פרמטריזציה דו ממדית. הקירוב הראשוני כבר קיים אם נעשה שימוש בפרמטריזציה תלת ממדית ואין צורך בהתאמה משטח התחלתי. שיפור דיוק המשטח המקורב מתבצע באמצעות אחת מהשיטות הבאות: (1) שיטת Gradient Descent Algorithm (GDA) המבוססת על שיטת התאמת ריבועים פחותים איטרטיבית. (2) שיטת Random Surface Error Correction (RSEC) המבוססת על שילוב עקרונות הלמידה האקראית של SOM עם שילוב אלמנטים של הגדרת משטחי B-Spline.

איחוד המשטחים המשוחזרים מתבצע לפי השלבים הבאים: 1) מציאת אזורי חפיפה בין זוגות המשטחים השונים המתארים את הגוף מכיוונים שונים. 2) חיתוך המשטחים ע"י עקומות המבדילות בין האיזורים החופפים ויצירת Trimmed B-Spline Surfaces. 3) חיבור המשטחים למעטפת המתארת את הגוף הנפחי.

שיטת שחזור זו משתמשת בייצוג B-Spline שמקובל במערכות תיב"מ עכשוויות. השיטה מאפשרת הוספת תמונות תלת ממדיות לגוף המשוחזר בעת תהליך הבנייה עצמו ועל כן מאפשרת מקבול תהליך השחזור.

גישת השחזור השנייה משחזרת רשת משולשים המקרבת את הגיאומטריה ומזהה את הטופולוגיה של ענן הנקודות. גישה זו פותרת את בעיית זיהוי הטופולוגיה באמצעות הרחבת רשת הבינה המלאכותית *neural gas* מעבר ליכולותיה הבסיסיות. השיטה מורכבת משני שלבים עיקריים: שלב יצירת רשת משולשים ושלב התאמת רשת המשולשים ליריעה (manifold).

בשלב הראשון נוצרת רשת משולשים שמהווה מפה משמרת טופולוגיה (Topology Preserving Map) של ענן הנקודות הנתון. במפה משמרת טופולוגיה נשמרים קשרי שכנות בין הנקודות הדגומות ובין רשת המשולשים המקרבת כך שאם שתי נקודות דגימה הינן קרובות במרחב אזי הקודקודים המייצגים אותם ברשת המשולשים יהיו מחוברים באמצעות צלע. כמו כן, אם שני קודקודים קרובים במרחב, אזי הם יהיו מחוברים בצלע. המשולשים שנוצרים ברשת משולשים שהיא מפה משמרת טופולוגיה מהווים חלק משילוש Delaunay .

רשת המשולשים נוצרת רשת בינה מלאכותית שמייצגת את מיקומי הקודקודים של רשת המשולשים במרחב ע"י הנוירונים. בעת תהליך הלמידה של רשת הבינה המלאכותית, הנוירונים המלאכותיים מקרבים את הגיאומטריה של ענן הנקודות הדגום במטרה לצמצם את הרעש. הרעש מוגדר כמרחק בין מיקום הנוירונים לנקודות הדגימה הקרובות אליהם. במקביל לתהליך הקרוב מוגדרות צלעות ופאות המשולשים על ידי הרחבה של כלל הלימוד של Hebb כפי שהוא בא לידי ביטוי ברשתות בינה מלאכותיות. הרחבה זו שפותחה במהלך המחקר מאפשרת לקרב את הנורמלים למשטח המקורב בעת תהליך הלימוד.

בשלב השני מותאמת רשת המשולשים ליריעה ע"י: (1) הסרת פאות משולשים שלא מתחברים ליריעה. (2) כיוון מחדש של הפאות. (3) השלמת משולשים חסרים באמצעות הטלה מקומית בכוון הנורמלים וסידור הצלעות מסביב לקודקוד עם כיוון השעון. בשלב זה נשמר קריטריון כיווניות אחידה של המשולשים בעת תהליך השחזור בכדי למנוע תופעות של משטחים לא אוריינטביליים. (4) סגירת לולאות גבול במעטפת היריעה.

היתרון העיקרי של גישת השחזור השנייה הוא היכולת ליישמה בקלות. רשת המשולשים המקורבת מייצגת את הטופולוגיה של הגוף ומתאימה כבסיס להרחבות נוספות.

לשם הדגמת תהליך השחזור, מוצגות מספר דוגמאות של שחזור גופים מפוסלים בעלי טופולוגיה כלשהי.