EG UK Theory and Practice of Computer Graphics (2010)
John Collomosse, Ian Grimstead (Editors)

Implicit surface reconstruction and feature detection with a learning algorithm

D. Kaye and I. Ivrissimtzis

School of Engineering and Computing Sciences, Durham University, UK

Abstract

We propose a new algorithm for implicit surface reconstruction and feature detection. The algorithm is based on a self organising map with the connectivity of a regular 3D grid that can be trained into an implicit representation of surface data. The implemented self organising map stores not only its current state but also its recent training history which can be used for feature detection. Preliminary results show that the proposed algorithm gives good quality reconstructions and can detect various types of feature.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling I.6.5 [Simulation and modeling]: Model Development

1. Introduction

The goal of *surface reconstruction* algorithms is the creation of a surface that models a given set of point data. The input point set is typically acquired through the use of physical devices, such as optical scanners, and must therefore be assumed to contain noise. One of the main challenges in surface reconstruction is the detection of surface features, such as spikes, creases, corners, or separate surface sheets lying close to each other, in the presence of data noise.

The ill-posed nature of the surface reconstruction and the feature detection problems means that the use of machine learning techniques can be advantageous as they can handle the uncertainty of the data better than their equivalent geometry based techniques. In particular, different variants of self-organising maps (SOMs) have been successfully been used [Yu99, BF01]. In this paper, we use a 3D SOM with the connectivity of a regular grid, which is trained to implicitly represent the reconstructed surface [YIL08].

In [YIL08], and all other previously proposed SOM-based surface reconstruction algorithms, the SOM learns the shape of the input data through a training process that alters the values stored at the SOM's nodes and, sometimes, its connectivity. At each training step, only the curent state of the SOM is to be stored. Of course, as the evolution of the trained SOM is gradual, the current state does contain information related to previous states, however, in general, the previous states of the SOM can not be fully retrieved.

In contrast, the SOM based algorithm proposed in this paper explicitly stores information not only on its current state but on previous states as well. That is, it stores the *training history* of the SOM. This training history can be used to infer surface feature information under the assumption that the well-defined flat areas of the surface are likely to have a stable training history. Flat areas are expected to exhibit low variance of the SOM node value between different states, whereas the less well-defined feature parts of the surface are expected to have a more unstable training history, that is a higher variance of the SOM node value between different states.

As the implementation stores not only the current state of the SOM but also some of its training history, memory efficiency becomes a primary concern. To solve this problem, the implemented SOM does not have the shape of a full 3D grid, but considers only nodes that are near the training samples and thus near to the reconstructed surface. Other differences between the implemented implicit SOM and the one proposed in [YIL08] are discussed in Section 3.

Contribution: The two main contributions of the paper can be summarised as follows. First, we propose the use of SOMs that store information on their training history and

DOI: 10.2312/LocalChapterEvents/TPCG/TPCG10/127-130

apply them to the problem of surface reconstruction. Second, we implement a memory efficient implicit SOM which stores its training history and shows that the training history can be used for feature detection.

2. Related work

Surface reconstruction has a wide range of applications. As a result, the problem has received a considerable amount of research interest and a multitude of approaches have been proposed. Geometric techniques, such as the Voronoi based algorithms [ABK98, DG03] and statistical surface fitting techniques such as the MLS based algorithms [Lev98, FCOS05, LCOL07] are two of the mainstream approaches taken to the problem.

A third mainstream approach is implicit surface reconstruction, where the surface is represented by the zero level set of a function $f : \mathbf{R}^3 \rightarrow \mathbf{R}$, where f is usually thought of as an approximation to the signed distance to the surface. The technique was pioneered in [HDD*92] and has been proven to be particularly robust in the presence of data noise. The most well-known examples of the technique, proposed in [CBC*01, OBA*03], compute the signed distance function f as a partition of unity, or as a linear combination of radial basis functions.

Self Organising Maps were introduced in [Koh82] as a special type of neural network that is trained through a competitive learning process, adapting itself to the data. In [HV98, Yu99, BF01, IJS03], surface reconstruction algorithms based on SOMs, or similar types of neural networks, are proposed. They all use neural networks with 2D connectivity and the result of the training is an explicit model of the surface data, such as a triangle mesh or the control grid of a uniform bivariate spline. Implicit SOM methods for surface reconstruction were introduced in [YIL08]. The algorithm proposed in this paper is also an implicit SOM surface reconstruction, however, in a major difference with [YIL08] and all the other previously proposed SOM methods, we store not only the current state of the SOM but also a portion of its recent training history.

Feature detection is a problem closely related to surface reconstruction. In many cases, the two problems are solved concurrently by a feature preserving surface reconstruction algorithm [FCOS05]. However, features can also be detected on the input point set as a pre-processing analysis of the data [PKG03], or on the reconstructed surface as a post-processing analysis of the obtained model [YBS05].

3. The Main Algorithm

As input, the algorithm takes a set of 3D points with normals, either from a static file or a stream source. The output is a triangle mesh with any potential features highlighted. First we describe the *basic step* of the algorithm, that is, the training of the SOM by a single input point. Next we compute the *separation* of a node, that is, an estimate of its value based on its training history. Next we describe the smoothing step, which increases the quality of the reconstruction, and finally, we describe the extraction of the triangle mesh from the trained grid.

3.1. Basic step

A training point s is uniformly randomly sampled from the input point cloud (or obtained from the input stream), s has a position p and a normal $\vec{\mathbf{n}}$. Twelve training points are then created equidistantly along the line segment $[-3\vec{\mathbf{n}}, +3\vec{\mathbf{n}}]$, see Figure 1 (left). For each training point t we compute is its position, its signed distance from p (denoted by x) and its weight;

$$w(x) = 1/(1+x^2). \tag{1}$$

The weighting means that the greater the distance between a training point and the input point from which it is derived, the less confidence we have in its reported distance from the surface. This becomes more relevant near features, where two areas of the surface lie close to each other, and so the farther reaches of the training data may interfere.

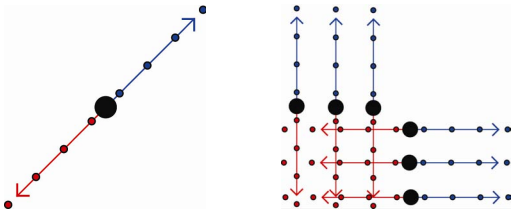

Figure 1: ***Left:*** *The construction of the training data.* ***Right:*** *Training data interference near a feature (corner).*

For each training point t, the nearest SOM node (the *winner*) is found and the weighted distance of t is added to the node's list of training data. If this causes the node's list of training data to exceed 20 entries, the oldest training data is discarded. Due to the regular arrangement of the SOM nodes, finding the winning node is a computationally inexpensive process. The algorithm runs for a fixed number of basic steps.

3.2. Separation Calculation

After every 5% of the total number of steps have been completed, the SOM nodes have their separations calculated and smoothed. First, the separation of each node is computed by calculating the weighted mean of all the distances in its training history.

$$\text{sep} = (\sum_i w_i d_i)/(\sum_i w_i). \tag{2}$$

If the node has had its separation calculated previously, its separation is updated by

$$\sigma_u = a\sigma_o + (1-a)\sigma_n \tag{3}$$

Where σ_o is the node's current separation, σ_n is the newly-computed separation and σ_u is the node's updated separation. If the node has not had its separation computed previously, then it is just set to the value computed in equation 2. The speed of learning is controlled by a, the learning rate parameter ($0 < a \leq 1$). A higher value trains the SOM quickly, but makes it more susceptible to corruption by noise. A lower value trains the SOM more slowly and favours slow convergence to a single value.

3.3. Smoothing

The smoothing takes the form of a new distance being appended to the training history of each node. First, for each node, the separation σ_1 of the L_0 radius 1 neighbourhood (direct neighbour nodes) is calculated as the mean of all neighbours that have been trained. If none of its neighbours have been trained then σ_1 will not be used for smoothing. Second, the separation σ_1 of the L_0 radius 2 neighbourhood is calculated (this includes only those nodes have an L_0 distance of exactly two), again, as the mean of the separations of all the trained neighbours. If fewer than three of these nodes have been trained then σ_2 will not be used for smoothing. Finally, the weighted sum

$$d_n = \frac{2}{3}\sigma_1 + \frac{1}{3}\sigma_2. \tag{4}$$

is computed, and the new distance is added to the node's training history with a weight of 1.

To make the algorithm more efficient (both in terms of speed and memory), nodes had their separation checked prior to being smoothed. If their current estimated distance from the surface was larger than $\sqrt{3}$, they were left unsmoothed. $\sqrt{3}$ is the length of the diagonal of the unit cube of the the SOM grid, therefore if a node is farther than $\sqrt{3}$ from the surface then it it cannot lie in an intersected cube. Consequently, there is little value in smoothing it. This gave a 10% increase in speed, and saved memory by not increasing the size of their training history. The requirement to have several trained neighbours is also beneficial as it results in higher-quality meshes and a faster run-time. Nodes are only smoothed if they are in a sufficiently well-trained neighbourhood, and time is not wasted by smoothing nodes that lie far from the surface.

3.4. Isosurface Extraction

After the training is complete, we cycle through the list of SOM nodes and examine their training history. The weighted variance β of the node's training history is calculated as

$$\beta = (\sum_i \gamma_i^2)/(\sum_i w_i) \quad \text{with} \quad \gamma_i = w_i(d_i - s) \tag{5}$$

where w_i is the weight of distance d_i, and the sums run over all the weighted distances in the node's training history.

If the β is above a predefined threshold then the node is flagged as having a high distance variance and being close to a suspected surface feature. A high value of β could be caused by features such as a spike, a crease, a corner, or two parts of the surface lying sufficiently close so that the training data for each part interferes with the other, see Figure 1 (right). It could also be caused by inaccurate training data caused by spatial and normal noise, or by wrongly oriented normals.

At the final step of the algorithm, the surface is extracted using a variant of the marching cubes algorithm [LC87], for which the regular arrangement of the SOM nodes is ideal. If a vertex is created between two nodes that are both flagged as having a high distance variance, then it flagged up as a suspected feature vertex. The requirement that both nodes have the flag set ensures that fewer areas are falsely marked as potential features.

4. Results

For a point set containing N points, good results were obtained by processing $10N$ training samples. The only exception was the very dense Neptune set, where the processing of $3N$ training samples was sufficient. Different values of the learning rate parameter a in Eq. 3 were tested and a value of 0.9 was found to give a good combination of adaptivity and numerical stability.

Figure 2 shows reconstruction of the Cube, Horse, Stanford Bunny and AIM@SHAPE Neptune point sets. The reconstructions of the Cube and the Horse show the ability of the algorithm to detect features in an clean point sets. In the reconstruction of the Bunny, the input data are the original raw data from the Stanford 3D scanning repository, not the vertices of an already reconstructed model. We notice that both features and spots of noise have been correctly identified. Finally, the reconstruction of the Neptune highlights the inherent ability of the algorithm to handle very large point sets, given that the input set is not globally processed but only randomly sampled. The thresholds for the weighted variance β were manually chosen at 1, 2, 1 and 0.3, respectively.

Training the SOM with the Horse dataset (100K points, 1M training samples) took 51 seconds, the Stanford Bunny (360K points, 4M training samples) took 164 seconds, whilst Neptune (3.2M points, 10M training samples) took 19 minutes. The timings are comparable to those reported in [YIL08]. In both cases the main bottleneck is the smoothing step. Notice, that in our case the theoretical complexity of the smoothing step is quadratic rather than cubic, however, each operation is more expensive as it processes a larger neighborhood of the smoothed node.

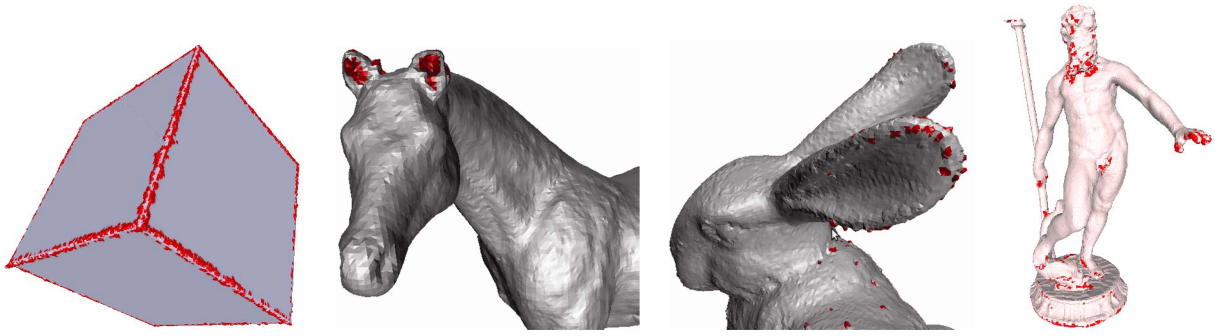

Figure 2: *Implicit SOM reconstructions. The detected feature points are drawn in red.*

5. Discussion

We proposed a new SOM based algorithm for implicit surface reconstruction and feature detection. Its main novelty is that instead of only storing the current state of the SOM, the recent training history is also explicitly stored and used in the reconstruction and feature detection. One limitation of our current approach results from the fact that the distance that the training data extend from the sample point does not decrease as the algorithm progresses. Consequently, for a static point cloud, the changes of the SOM have fixed granularity, and so, after a certain time further training does not improve the mesh quality. An overfitting control method is currently being developed to catch this situation and force the isosurface extraction. The algorithm would be simple to modify to take a live input, and performance could be improved by taking a multithreaded approach.

In a second direction for the future development of the algorithm, we plan to use a more sophisticated statistical analysis of the separation of a single node, or the separations of a neighbourhood nodes, and extract more reliable feature information. Our ultimate goal is to produce an algorithm able to classify the features of a surface into few basic categories, such as spikes, creases, or data noise.

References

[ABK98] AMENTA N., BERN M., KAMVYSSELIS M.: A new voronoi–based surface reconstruction algorithm. In *SIGGRAPH* (1998), pp. 415–422. 2

[BF01] BARHAK J., FISCHER A.: Adaptive reconstruction of freeform objects with 3D SOM neural network grids. In *Pacific Graphics* (2001), pp. 97–105. 1, 2

[CBC*01] 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. In *SIGGRAPH* (2001), pp. 67–76. 2

[DG03] DEY T. K., GOSWAMI S.: Tight cocone: a water-tight surface reconstructor. In *Symposium on Solid Modeling and Applications* (2003), pp. 127–134. 2

[FCOS05] FLEISHMAN S., COHEN-OR D., SILVA C. T.: Robust moving least-squares fitting with sharp features. In *SIGGRAPH* (2005), pp. 544–552. 2

[HDD*92] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Surface reconstruction from unorganized points. *SIGGRAPH* (1992), 71–78. 2

[HV98] HOFFMANN M., VÁRADY L.: Free-form modelling surfaces for scattered data by neural networks. *Journal for Geometry and Graphics 1* (1998), 1–6. 2

[IJS03] IVRISSIMTZIS I., JEONG W.-K., SEIDEL H.-P.: Using growing cell structures for surface reconstruction. In *Shape Modeling International* (2003), pp. 78–86. 2

[Koh82] KOHONEN T.: Self-organized formation of topologically correct feature maps. *Biological Cybernetics 43* (1982), 59–69. 2

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3D surface construction algoritm. *SIGGRAPH* (1987), 163–168. 3

[LCOL07] LIPMAN Y., COHEN-OR D., LEVIN D.: Data-dependent mls for faithful surface approximation. In *Symposium on Geometry Processing* (2007), pp. 59–67. 2

[Lev98] LEVIN D.: The approximation power of moving least-squares. *Mathematics of Computation 67*, 224 (1998), 1517–1531. 2

[OBA*03] OHTAKE Y., BELYAEV A., ALEXA M., TURK G., SEIDEL H.-P.: Multi-level partition of unity implicits. In *SIGGRAPH* (2003), pp. 463–470. 2

[PKG03] PAULY M., KEISER R., GROSS M.: Estimation of planar curves. *Comp. Graph. Forum 22*, 3 (2003), 281–289. 2

[YBS05] YOSHIZAWA S., BELYAEV A., SEIDEL H.-P.: Fast and robust detection of crest lines on meshes. In *Symposium on Solid and Physical Modeling* (2005), ACM Press, pp. 227–232. 2

[YIL08] YOON M., IVRISSIMTZIS I., LEE S.: Self-organising maps for implicit surface reconstruction. In *Theory and Practice of Computer Graphics* (2008), EG Press, pp. 83–90. 1, 2, 3

[Yu99] YU Y.: Surface reconstruction from unorganized points using self-organizing neural networks. In *IEEE Visualization* (1999), pp. 61–64. 1, 2