Pattern Recognition.   Accepted and in press

Parameterization of point-cloud freeform surfaces using adaptive sequential learning RBF networks

Qinggang Meng[a], Baihua Li[b,*], Horst Holstein[c], Yonghuai Liu[c]

[a]*Department of Computer Science, Loughborough University,UK.*
[b]*School of Computing, Mathematics & Digital Technology, Manchester Metropolitan University, UK*
[c]*Dept. of Computer Science, Aberystwyth University, UK*

Abstract

We propose a self-organizing Radial Basis Function (RBF) neural network method for parameterization of freeform surfaces from larger, noisy and unoriented point clouds. In particular, an adaptive sequential learning algorithm is presented for network construction from a single instance of point set. The adaptive learning allows neurons to be dynamically inserted and fully adjusted (e.g. their locations, widths and weights), according to mapping residuals and data point novelty associated to underlying geometry. Pseudo neurons, exhibiting very limited contributions, can be removed through a pruning procedure. Additionally, a Neighborhood Extended Kalman Filter (NEKF) was developed to significantly accelerate parameterization. Experimental results show that this adaptive learning enables effective capture of global low-frequency variations while preserving sharp local details, ultimately leading to accurate and compact parameterization, as characterized by a small number of neurons. Parameterization using the proposed RBF network provides simple, low cost and low storage solutions to many problems such as surface construction, re-sampling, hole filling, multiple level-of-detail meshing and data compression from unstructured and incomplete range data. Performance results are also presented for comparison.

Keywords: Surface parameterization, point clouds, adaptive sequential learning.

1. Introduction

Laser scanners are routinely used for model acquisition. They can obtain point clouds of surfaces more quickly and with greater accuracy compared to other digitization techniques. A point-cloud range scan, such as shown in Fig. 1, typically contains huge numbers of unstructured, densely and non-uniformly distributed

*Corresponding author
Email address: `b.li@mmu.ac.uk` (Baihua Li)

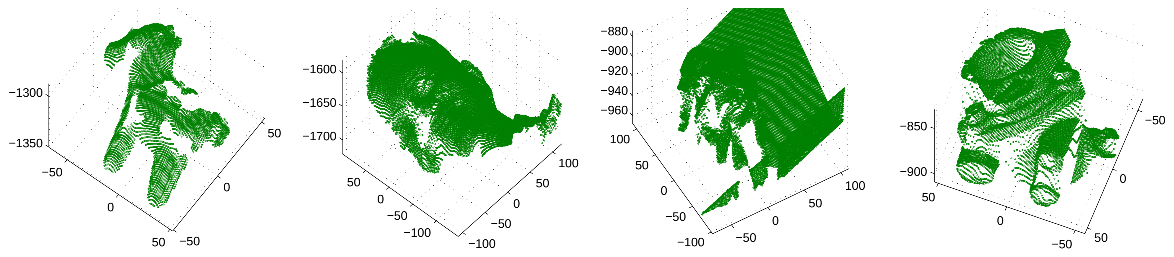


Figure 1: Unstructured noisy point clouds.

points. Measurement errors and occlusions during digitization can make range data noisy and incomplete, with "holes". Direct meshing and manipulation of such point clouds can be inefficient and difficult with regard to computational cost, memory overhead and robustness to data noise. Modeling from an intermediary parametric domain could significantly improve data representation and manipulation flexibility using only a small set of control parameters and mathematical calculations. Parameterization of point clouds in a low dimensional space, and consequently, with manageable computational cost, good compactness and desired accuracy, would therefore provide an alternative and more preferable solution to many problems in freeform surface modeling, including remeshing, multi-resolution analysis, level-of-detail (LOD), morphing, texture transfer, and geometry manipulation [1, 2].

While much research has been conducted on surface parameterization, the majority has focused on complete mesh surfaces with known vertex connections, or aimed at surfaces with lower spatial complexity represented by small data sets [1, 3, 4, 5]. Such parameterizations make useful tools for remeshing or triangulating clean data, but they are not suitable for noisy and unstructured point clouds. Direct parameterization from point clouds would involve less error compared to parameterization from intermediary meshed surfaces. However, a robust method which directly transforms noisy point clouds into a compact, unified parametric domain (as opposed to piecewise approaches) is still an open problem.

To address this problem, we introduce a neural network approach involving self-growing Radial Basis Functions (RBFs). Parameterization is achieved through adaptive sequential learning. The resulting network forms a parametric space, so that vertices or control points can be generated, from which a complete parametric surface exhibiting smoothness can be created. To our knowledge, the proposed RBF network for direct parameterization of point clouds has the following novel aspects:

- Parameterization is achieved in a *unified* self-organizing network space, superior to piecewise or spatial multi-partitioning representation.

- Our method is applicable to unoriented, noisy, incomplete and non-uniformly distributed point clouds, rather than only clean, regular or oriented point sets, or surface points generated from structured polygonal meshes. It can deal with freeform surfaces with real-world geometric complexities, such as sharp local details and low-frequency global variations.

- Parameterization can be conducted at a desired LOD, simplifying multi-resolution applications. It establishes a compact functional representation and finds applications on surface construction, re-sampling, mesh repair, LOD, and data compression from point sets with only coordinate information.

- The network is constructed through adaptive sequential learning using a single instance of range scan. Our proposed adaptive learning provides a general solution to the common problem of effective RBF fitting. Neurons are generated according to heuristic *novel inputs* rather than being randomly chosen from all points. They can be located, removed and adjusted in full dimensionality in terms of location, weight and width, thereby adapting to the distribution of underlying data. This adaptivity ultimately determines the effectiveness and compactness of RBF fitting, which is particularly important for handling large point clouds and complex spatial features.

- The development of the *Neighborhood Extended Kalman Filter* (NEKF) dramatically reduces the RBF network construction cost, enabling parameterizing large point sets within feasible time.

- Experimental results demonstrate that the proposed parameterization RBF network provides an efficient solution for many frequently encountered tasks that process point-sampled surfaces, such as surface reproduction, multiple LODs, mesh repair and data compression.

The rest of the paper is organized as follows: Section 2 reviews related work on surface parameterization. Section 3 presents point-cloud parameterization through adaptive learning. In Section 4, we provide experimental results and evaluate parameterization regarding accuracy, speed, compactness, adaptivity and multi-LOD ability. Section 5 discusses general issues such as network parameter definition and parameterization performance relative to other methods. We conclude our work in Section 6.

2. The research context and related work

Apart from the usual requirements concerning accuracy, speed, low memory overhead and compactness, an important criterion for sophisticated surface parameterization is the ability to deal with complex freeform surfaces, such as those containing highly variable and irregular spatial features, represented by unstructured noisy data. Numerous methods have been developed, but the majority have focused on mesh surface parameterization [1, 3]. In these works, the main purpose of parameterization was to obtain piece-wise linear mappings between a 3D mesh (represented by triangles or polygons) and a parametric space, such as a parametric plane [6], a parametric sphere [7, 8] or an intermediary parametric domain [9, 10], so as to minimize angular and area distortion for the whole mesh. To this end, an entire mesh was commonly partitioned into patches or charts according to certain feature curves, and then each patch was interpolated or approximated piecewise using, for example, polynomials, splines or radial basis functions. The whole parameter mapping was often obtained by linear assembly of a number of local functions based on least-square energy minimization. Consequently, piecewise approaches were likely to suffer from discontinuity and self-intersection over the cuts between patches. The computational cost for non-linear optimization could be tremendously high for large and complex surfaces. Most importantly, these parameterizations aimed to achieve an exact one-to-one mapping between each vertex and a point in the parametric domain, relying on information about vertex connections in the mesh. Such methods are suitable for structured mesh data, but not to unstructured and noisy point sets.

Surfaces represented by unorganized scattered point sets provide challenges of their own. Parameterization of point-sampled surfaces is particularly important for geometry formulation and data compression. To cope with unstructured point sets, a *characteristic shape* algorithm was proposed to generate simple polygons for a shape of a set of clean point data in the plane [4]. The algorithm was based on the Delaunay triangulation of the points, and a single normalized parameter was used to control the parametric shape. Floater and Hormann [5] have suggested that piecewise methods for convex combination mapping could be applicable when suitable neighborhood information was available. Several choices of such neighborhoods were proposed, but the most effective and widely adopted method was the use of nearest neighbors. To this end, *k*-means clustering or similar techniques have been extensively adopted to search locally for *k*-nearest neighbors, so as to assist partitioning a surface into a set of charts [11, 12, 13]. Due to searches being based on local surface attribute estimates, such as curvatures and differential features, the *k*-means clustering could be very sensitive to noisy and incomplete data. A neighborhood graph has also been introduced to preserve topological information [14]. However, it could be argued that the highly complex graph connectivity structure could become unmanageable in the case of noisy and extensive point sets.

As discovered by recent advances in function approximation and pattern classification using Radial Basis Functions [15, 16, 17, 18], RBFs possess many useful properties such as good generalization, continuity and stability [19, 20]. These abilities make RBFs well suited for accommodating scattered data without relying on *prior* information about the connectivity and topology of underlying data. Meanwhile, the extraordinary interpolation and extrapolation capabilities of the RBFs allow smooth approximation and repair of noisy range data. Fitting RBFs to scattered points as implicit surface [21, 22, 23, 24] has been proposed in computer graphics as modeling methods.

It has been noticed that using all the data to interpolate RBF centers could result in a poorly conditioned matrix, producing unmanageable computational costs and wasted RBFs on large and dense data sets [21]. As an improvement, multi-scale [25] or multi-level spatial partitioning approaches [26, 27, 28] have been developed. A common strategy in these works has been first to fit the surface with basis functions of large support, followed by fitting the residuals with basis functions of diminishing support. To accommodate RBF fitting, the entire shape was decomposed into subdivisions iteratively according to local errors. For example, regions with large fitting residuals were hierarchically partitioned into small cells using Octrees [26], or support centers were iteratively chosen by spatially uniform, random sampling of the point set [27]. RBF support centers were restricted on regular cells, or on a subset of the scattered points. This is not optimal with regards to adaptivity of spatial features of underlying points and robustness to data noise. In addition, oriented point sets (coordinates and normals of all points) were required to analyze sharp geometric features when choosing an appropriate approximation type [26, 27]. RBF scales were often fixed at a same partition level or determined requiring additional information (e.g. acquisition confidence of scan points). For RBF based modeling, fitting effectiveness, namely using the least number of RBFs to best fit underlying data, remains largely unsolved.

As a new genre of computational infrastructure, neural network based methods have been reported. In particular, a Self-Organizing Map (SOM) [29] has been introduced for forming a quadrilateral control grid from scattered point sets, thus allowing surface fitting by Bezier-surface or NURBS [30, 31]. Barhak and Fischer [32] used this network map to parameterize small sets of clean points with low frequency spatial variations. They reported that the SOM outperformed the traditional Partial Differential Equation (PDE) method, leading to smooth approximation of surfaces. Motivated by the similar idea to [31] and [32], a self-organizing feature map (SOFM) has also been proposed [33]. However, the structure of these network maps was fixed, and the scale of all the neurons was at a constant pre-defined value. A more effective network approach was one employing a multi-layer hierarchical RBF structure as proposed in [34, 35]. Layers in this network were represented by partition grids at increasing resolutions. Although neurons were still located on these regular grids, the neuron scales were allowed to be halved at every higher layer, allowing coarse-to-fine RBF fitting to the mapping residuals.

Despite the various advantages of RBF based methods, problems remain with adaptive fitting of RBFs to the underlying data. Due to the lack of flexibility on RBF distribution and scaling in the aforementioned works, a large number of fine-scale RBFs were inevitably required to absorb local residuals. This could result in the required number of RBFs being even higher than the number of original points. For a RBF network approach, a large portion of neurons may be unnecessarily fitted in low frequency regions, while resolution could fall short for sharp features and details [31, 32]. Consequently, the computational cost of dense RBFs can be unmanageable, particularly for data with highly variable spatial frequencies. Solving the problem of fitting adaptivity is crucial and would be of benefit to both RBF based modeling and parameterization.

As an solution to effective RBF fitting of noisy unoriented point clouds, we propose a self-organizing RBF network approach. Specifically, a fast adaptive learning algorithm is presented. In particular, neuron choice is fully dynamic and adjustable according to the novelty and distribution of underlying data. These attributes make our approach preferable to those that use RBF fittings at fixed locations, network structure, or pre-defined scales. As demonstrated by the experimental results, parameterization using the proposed adaptive sequential RBF (ASRBF) network is highly adaptive to complex spatial features, robust to data noise, and is compact and efficient for handling extensive dense point sets.

3. Parameterization through adaptive sequential learning RBF networks

In this section, we present the network topology structure, the form of RBF kernel employed, and in particular, the three-stage adaptive sequential learning algorithm.

3.1. Network topology

A typical feed-forward network is shown in Fig. 2. It has a simple topology linking an input layer, a hidden layer and an output. The input layer has an i-dimensional input $\mathbf{x}(x_1,...,x_i)$, the hidden layer has K kernels $\phi_k(\mathbf{x}), k=1,2,...K$, and the network output $f(\mathbf{x})$ is a linear combination of kernels taking the form

$$f(\mathbf{x}) = a_0 + \sum_{k=1}^{K} a_k \phi_k(\mathbf{x}) \ , \tag{1}$$

where coefficient a_k is the real-valued *weight* of the kth kernel, and a_0 is the basis element.

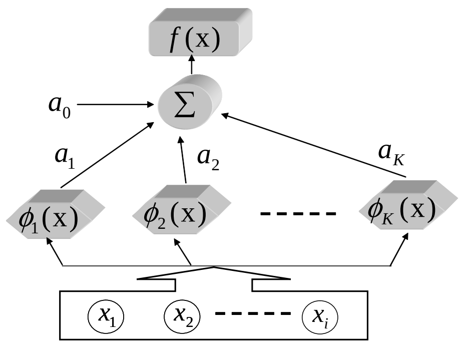


Figure 2: Feed-forward RBF network topology.

In this study, Gaussian Radial Basis Functions (RBFs), acting as the nonlinear kernels of the hidden layer, are used as the computational substrate of the network. A Gaussian RBF imposes no restriction on point location, and its response falls quickly with increasing distance from the kernel, allowing significant local influence and controllability. This quasi-locality makes Gaussian RBFs well suited for surface modeling from irregularly sampled points. Therefore Gaussian RBFs, of the form

$$\phi_k(\mathbf{x}) = \exp\left(-\frac{1}{\sigma_k^2}\|\mathbf{x}-\mathbf{u}_k\|^2\right) \, , \tag{2}$$

are chosen in this study. In Eq. (2), $\|.\|$ denotes the Euclidean norm, $\mathbf{u}_k$ indicates the *center* of the kth neuron kernel, and σ_k represents the *width* of its coverage.

3.2. The adaptive sequential learning

The adaptivity of the parameterization RBF network derives from the strategy of dynamic network construction through supervised sequential learning. The network starts from an empty space with no neurons and no pre-defined structures. Training data (a point set from a single instance of range scan) is sequentially fed into the network. For each input, three tasks will be carried out:

1. Network growth: the network may self-grow one neuron at the current input, if the input point satisfies the “novelty” criteria. (Section 3.3)
2. Optimization: for a fast optimization that minimizes parameterization residuals, a subset of neighbor neurons will be updated with full dimensionality using a Neighborhood Extended Kalman Filter (NEKF) algorithm; (Section 3.4)
3. Neuron pruning: finally, a pruning strategy is applied to remove network redundancy. *Pseudo* neurons, which consistently make little contribution to the parameterization process, will be discarded. (Section 3.5)

We present a three-stage parameterization process in the following sections and summarize the adaptive sequential learning algorithm in Algorithm. 1.

3.3. Network growth according to the novelty of point inputs

The training sequence of a point cloud can be denoted as $\mathscr{T} = \{\mathbf{x}_n, z_n\}|_{n=1,2,...\mathscr{N}}$. It consists of $\mathscr{N}$ independent observations of 3D points $\{x_n, y_n, z_n\}$ with random data orderings. Vector $\mathbf{x}_n = (x_n, y_n)$ is used as network input, and z_n is its associated measurement output, to be approximated by the network. The random ordering of the training set aims to obtain uniformly distributed points from the whole data domain. This is necessary for balanced network growth and pseudo neuron validation (Section 3.5). We do not assume any prior knowledge on the topology or dependencies within the point set.

The network starts with no neurons in its space. At each learning step n, if the current observation $\{\mathbf{x}_n, z_n\}$ satisfies the following three novelty conditions below, then a corresponding new neuron will be added into the network.

Novelty criterion 1: the input $\mathbf{x}_n$ of the current nth observation is far away from the centers of all existing K neurons in the network,

$$\|\mathbf{x}_n - \mathbf{u}_{n\_near}\| > \mathscr{D}_K \ , \tag{3}$$

where $\mathbf{u}_{n\_near}$ represents the neuron center nearest to the current input $\mathbf{x}_n$.

This criterion aims to ensure that neurons are inserted at a distance of at least $\mathscr{D}_K$ from each other, so as to guarantee a well spread and balanced neuron distribution in the network space. This separation distance $\mathscr{D}_K$ is initially set to a maximum $\mathscr{D}_{\max}$, allowing a sparse neuron insertion. During the network growing process, $\mathscr{D}_K$ is made to decay exponentially with the increasing number of neurons K involved at that stage in the network,

$$\mathscr{D}_K = \max\{\mathscr{D}_{\max}\gamma^K, \mathscr{D}_{\min}\},\ 0 < \gamma < 1, \tag{4}$$

until a pre-defined minimum $\mathscr{D}_{\min}$ is reached. The minimum separation distance $\mathscr{D}_{\min}$ actually indicates an overall *neuron separation level* in the network which can be used to control the detail level of parameterization and network compactness. A *decay factor* γ is used to control the decline speed. Obviously, the consistent decay on the separation distance enforces the tendency of sparse-to-dense RBF fitting.

Novelty criterion 2: the parameterization error e_n between the network output $f(\mathbf{x}_n)$ and the measurement value z_n at current observation $\{\mathbf{x}_n, z_n\}$ is significant,

$$\|e_n\| > E \ , \text{ where } \ e_n = f(\mathbf{x}_n) - z_n \ . \tag{5}$$

Algorithm 1 : Parameterization by adaptive sequential RBF network.

```
1. Initialization: network parameter initialization and sequential training data generation in random point ordering
2. Iteration: parameterization starts with no neuron in the network $K = 0$
for each point input $(\mathbf{x}_n, z_n)$, $n = 1, 2, \ldots, \mathcal{N}$ do
    Stage 1: Network growth (Section 3.3)
    (1) define the neuron separation distance: $\mathscr{D}_K = \max\{\mathscr{D}_{\max}\gamma^K, \mathscr{D}_{\min}\}$
    (2) calculate network output and error residuals at current network input $\mathbf{x}_n$:
          $f(\mathbf{x}_n) = a_0 + \sum_{k=1}^{K} a_k \phi_k(\mathbf{x}_n)$ ,
          $\|e_n\| = \|f(\mathbf{x}_n) - z_n\|$ ,
          $e_n^{\omega} = \sqrt{\left\| \frac{\sum_{i=n-(\omega-1)}^{n} e_i^2}{\omega} \right\|}$ .
    (3) apply novelty criteria to add a new neuron:
          if $(\|\mathbf{x}_n - \mathbf{u}_{n\_near}\| > \mathscr{D}_K) \wedge (\|e_n\| > E) \wedge (e_n^{\omega} > E_{\omega})$
              add a new $K+1$th neuron;
              set neuron parameters as: $\mathbf{u}_{K+1} = \mathbf{x}_n$ , $a_{K+1} = e_n$ , $\sigma_{K+1} = \psi \|\mathbf{u}_{K+1} - \mathbf{u}_{n\_near}\|$ .
          end if
    Stage 2: Optimization (Section 3.4)
    (1) select neighbor neurons
    (2) update neuron parameters by NEKF.
    Stage 3: Neuron pruning (Section 3.5)
    for each neuron $k$ in the network, $k = 1, 2, \ldots, K$
    (1)    calculate the neuron output $\Pi_k = a_k \exp\left(-\frac{\|\mathbf{x}_n - \mathbf{u}_k\|^2}{\sigma_k^2}\right)$ at current observation
    (2)    calculate its contribution ratio $r_k = \frac{|\Pi_k|}{\sum_{k=1}^{K} |\Pi_k|}$
           if $r_k < \mathscr{P}$ for $\omega$ observations
               remove the $k$th neuron from the network.
           end if
    end for
end for
```

This criterion is used to verify whether the desired *network accuracy* E has been achieved locally at the current input. Obviously, this criterion attempts a closest fit at local points. It is effective when a point measure is accurate. However this assumption usually does not hold. Point clouds can be severely corrupted due to various measurement errors, data holes and outliers. To tolerate possible data noise, a "closeness" measure in the "mean" sense would be preferable, as described in Criterion 3.

Novelty criterion 3: the RMS parameterization error for the last ω consecutive inputs before the current nth input $\{\mathbf{x}_n, z_n\}$ is still significant,

$$e_n^{\omega} > E_{\omega} \text{ , where } e_n^{\omega} = \sqrt{\left\| \frac{\sum_{i=n-(\omega-1)}^{n} e_i^2}{\omega} \right\|} \ . \tag{6}$$

Satisfying this criterion could mean that an accuracy E_{ω} towards a smooth approximation in the whole data domain has not been achieved. Therefore, new neurons are still required to improve the parameterization accuracy.

In summary, the first criterion enforces a well separated and incrementally sparse-to-dense neuron distribution, guaranteeing balanced network growth and neuron coverage. The second criterion ensures that neurons are generated only in areas with larger local mapping residuals. The third criterion evaluates whether a global accuracy of the parameterization has been achieved.

If the current input point satisfies all these three novelty conditions, a new $(K+1)$th neuron will be inserted into the network. To best absorb mapping residuals, the new Gaussian neuron position $\mathbf{u}_{K+1}$ is placed at the same location as the input $\mathbf{x}_n$, its weight a_{K+1} is initialized to be the local mapping error e_n, and its width σ_{K+1} is scaled in proportion to the distance from its nearest neighbor, according to

$$\begin{aligned} \mathbf{u}_{K+1} &= \mathbf{x}_n \\ a_{K+1} &= e_n \\ \sigma_{K+1} &= \psi \|\mathbf{u}_{K+1} - \mathbf{u}_{n\_near}\| \ , \end{aligned} \tag{7}$$

where ψ is a user-defined *overlap factor* with a value between 0 and 1. Parameter ψ defines the overlap level between the new neuron and its neighbors. A higher value indicates a larger RBF support with more influence on its neighbors; while when a smaller value is chosen, the Gaussian will produce a more focused local response.

During network construction, parameterization error will consistently reduce. Accordingly, weights of newly added neurons tend to reduce in magnitude. Meanwhile, due to the decay policy applied to the separation distance $\mathscr{D}_K$, neuron density changes from sparse to dense, therefore, the neuron width σ associated with its nearest neighbor will decline as well. As an overall tendency, fewer neurons with significant widths and weights are initially generated to model low-frequency variations and form the smooth substrate surface, while neurons of smaller width and weight are subsequently recruited in areas with larger residuals to refine local details (see Section 4.4). This sparse-to-dense and coarse-to-fine adaptive RBF fitting gives the network high adaptivity to the underlying data.

If the current observation does not satisfy the three novelty conditions, no new neuron is added. In the next stage, a neighborhood EKF algorithm is employed to optimize neuron parameters, thereby minimizing parameterization error.

3.4. Optimization by fast neighborhood EKF

The least mean square (LMS), gradient descent (GD) and Extended Kalman filters (EKF) are commonly used methods to optimize neuron parameters in nonlinear networks and sequential learning [20, 36, 37]. EKFs have been reported to outperform LMS and GD on stability and accuracy, despite the high computational cost. The EKF method updates all neurons at each learning step, we therefore refer to it as *global EKF* (GEKF). The computational complexity of the GEKF is $O(A^2)$ per learning step, where A denotes the total number of neuron parameters to be updated [38]. In our case, if the network contains K neurons, $\mathbf{w}_n = \{a_0, \mathbf{w}_k \mid k = 1, 2, ..., K\}$, and each neuron is represented by four parameters as $\mathbf{w}_k = (a_k, \sigma_k, \mathbf{u}_k)$: two scalars for weight a_k and width σ_k, and one neuron center $\mathbf{u}_k$ in 2D, then the computational cost for updating all these K neurons will be $O((4K)^2)$. This cost can become unmanageable for networks with hundreds or thousands of neurons.

To reduce computational cost of the GEKF, we introduce a fast local approach, called *neighborhood EKF* (NEKF). At the nth learning step, only a subset of K' *neighbor neurons* of the current observation $\{\mathbf{x}_n, z_n\}$ are updated. Selection of neighbor neurons is based on:

1. if it is the nearest neighbor to the current observation, or
2. if it is in a neighborhood region proportional to the separation distance $\mathscr{D}_K$.

Using the first criterion, only the nearest neighbor, possibly the most influential to the current observation, is selected and updated. Updating only the nearest is fast and gives good stability. When using the second criterion, on one hand, due to the decay on separation distance $\mathscr{D}_K$, the neighborhood region will reduce; on the other hand, the neuron density will increase during network growth. There will, therefore, always be a sufficient and fairly consistent number of neurons selected, initially from larger areas with sparsely distributed neurons for approximating coarse features, but gradually concentrating on more local regions to refine details. This implies that the local NEKF actually performs a dynamic global-to-local optimization on neuron parameters during network construction.

Using the NEKF, the nth training step updates the selected subset of K' neurons in full dimensionality of weight, width and location $\mathbf{w}_n = \{a_0, \mathbf{w}_k | \mathbf{w}_k = (a_k, \sigma_k, \mathbf{u}_k),\ k = 1, 2, ..., K', K' < K\}$ by

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mathbf{G}_n e_n \ , \tag{8}$$

where $e_n = f(\mathbf{x}_n) - z_n$ is the parameterization error at the nth observation $(\mathbf{x}_n, z_n)$, and $\mathbf{G}_n$ is the Kalman gain calculated by

$$\mathbf{G}_n = \mathbf{P}_{n-1}\mathbf{B}_n \left[\mathbf{R}_n + \mathbf{B}_n^T \mathbf{P}_{n-1}\mathbf{B}_n\right]^{-1} \ , \tag{9}$$

in which $\mathbf{R}_n$ is the variance of the measurement noise, $\mathbf{B}_n = \nabla_{\mathbf{w}_n} f(\mathbf{x}_n)$ is the gradient matrix of the network output $f(\mathbf{x}_n)$ with respect to the network parameter $\mathbf{w}_n$, and $\mathbf{P}_n$ is an error covariance matrix, which is updated by

$$\mathbf{P}_n = \left[\mathbf{I} - \mathbf{G}_n \mathbf{B}_n^T\right] \mathbf{P}_{n-1} + q\mathbf{I} \ , \tag{10}$$

where the scalar q determines the allowed random step in the direction of the gradient vector, and $\mathbf{I}$ is a unit matrix.

The computational cost of NEKF is reduced from $O((4K)^2)$ to $O\left((4K')^2\right)$, where the selected K' neighbors are usually much less numerous than the total K neurons in the network. The minimum cost can consistently be $O\left(4^2\right)$ in the extreme case where only the nearest neighbor is updated ($K' = 1$). Experimental results showed a remarkable gain in performance speed of the NEKF over the GEKF and GD with a comparable accuracy (Section 4.6).

3.5. Effective neuron pruning to enhance the compactness of parameterization

Network size can become large under the growth strategy alone, possibly leading to network overfit. To avoid this, we use a pruning process on those *pseudo* neurons, that make an insignificant contribution to parameterization over a number of consecutive observations. These neurons are very likely to have been added due to noisy points, or become redundant as a result of network optimization. Removing these neurons not only promotes parameterization compactness, but also helps to reduce artifacts caused by measurement errors.

To find such pseudo neurons, at each point observation $(\mathbf{x}_n, z_n)$, we calculate the network output Π_k from each neuron at the input $\mathbf{x}_n$,

$$\Pi_k = a_k \exp\left(-\frac{\|\mathbf{x}_n - \mathbf{u}_k\|^2}{\sigma_k^2}\right) , \tag{11}$$

and then its contribution ratio r_k

$$r_k = \frac{|\Pi_k|}{\sum\limits_{k=1}^{K} |\Pi_k|}. \tag{12}$$

If the ratio r_k is consistently less than the *pruning threshold* $\mathscr{P}$ for ω consecutive observations in sequential learning, that is

$$r_k < \mathscr{P}|_{\omega} , \tag{13}$$

this neuron is detected as a *pseudo* neuron and is removed from the network.

In order to provide an effective validation on the contribution of a neuron, the network pruning requires the ω consecutive observations to be uniformly sampled points from the entire data space. This is another reason for requiring random point orderings in the training set. The performance of neuron pruning is presented in Section 4.3. The adaptive sequential learning algorithm for point-cloud surface parameterization is summarized in Algorithm 1.

4. Experimental results

We implemented the proposed adaptive sequential learning RBF network in C++ for point-cloud surface parameterization. In this section, we present results that demonstrate the performance of the parameterization with regard to mesh reproduction and repair (Section 4.1), multi-LOD (Section 4.2), pruning effectiveness (Section 4.3), adaptivity (Section 4.4), compactness and accuracy (Section 4.5) and finally performance efficiency by using the NEKF (Section 4.6).

4.1. Range surface reproduction and mesh repair from point-cloud parameterization using ASRBF networks

The point-cloud range data used in our experiments of parameterization were obtained from the Ohio SAMPL range scan database [39]. Each range scan was presented by a 200×200 array, consisting of densely distributed surface points and labeled margin areas. These range points have irregular spatial sampling, generally contain measurement noise and many contain data holes.

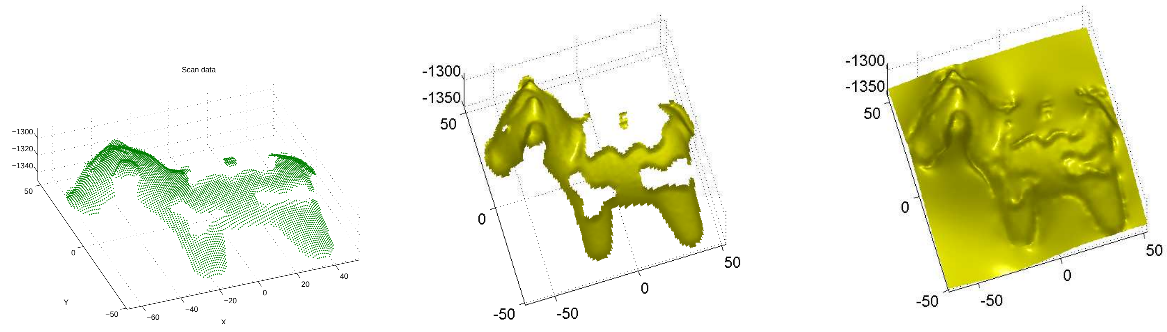
The range scans of different objects vary in physical size and were obtained using different coordinate systems. To enable the parameterization process be carried out with consistently chosen network parameters in a unified network space, while also using same guidelines, each scan was normalized to a unit cube in its x, y and z dimensions. Surface points were then taken with random point orderings to generate a training sequence $\mathscr{T} = \{\mathbf{x}_n, z_n\}\big|_{n=1,2,\ldots,\mathscr{N}}$, including $\mathscr{N}$ 3D point observations $\{\mathbf{x}_n, z_n\}$ in which $\mathbf{x}_n = (x_n, y_n)$ and $x_n, y_n, z_n \in [0,1]$. The $\mathbf{x}_n$ was used as the network input, and z_n its associated output. Details on how the network parameters were defined will be discussed in Section 5.

The parameterization RBF network of a point-cloud range scan was constructed using adaptive sequential learning as described in Section 3. The resulting network can be evaluated anywhere, so that surface vertices can be calculated at any desired resolution and ordering, allowing the production of the parametric surface [1]. For direct comparison between the parametric surface and its original scan, the network was evaluated at the same (x_n, y_n) locations as in the normalized 200×200 range array, with the outputs as their corresponding depth values z_n, and the normalized surface points calculated from the parameterization were then re-scaled to restore the actual aspect ratio and 3D size of the objects.

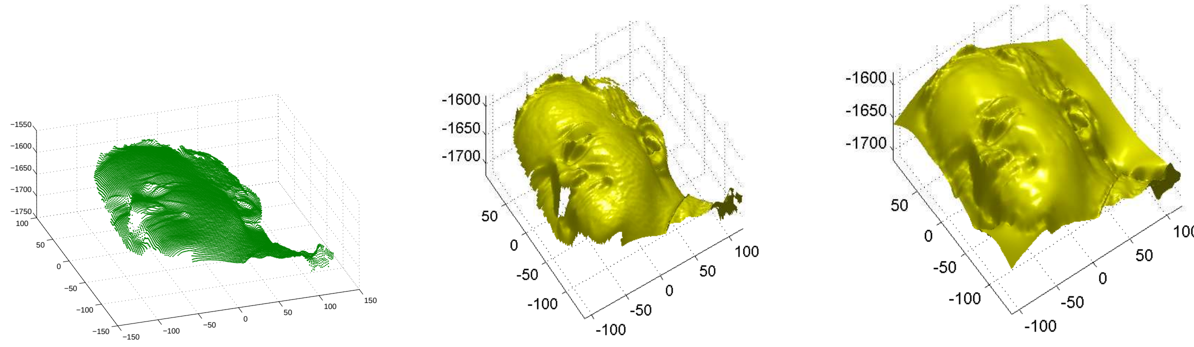
The results of direct meshing from point clouds and corresponding parametric surfaces reproduced from parameterization ASRBF networks are shown in Fig. 3. From these examples, we observe that although the range scans contain bumpiness, jagged protuberances and various irregular data holes, the reproduced parametric surfaces are smooth and complete, preserving high fidelity to the original surfaces. Both low frequency variations and high frequency details are retained, including flat facets and sharp edges/corners. Jagged protuberances (e.g. particularly at sharp edges and corners) and bumpiness (e.g. cheek in Fig. 3 (b)) were smoothly filtered. Benefiting from the extraordinary interpolation and extrapolation capability of the Gaussians, Irregular holes that are large compared to the geometric variation in surfaces (e.g. missing protruding patterns on the “cow” and missing legs of the “Santa”) are convincingly restored. Therefore, our method does not require the availability of multiple instances to train a prior model [40, 41]. Missing data between adjoined faces (e.g. in the “valve” and “Santa”) can also be filled and seamlessly blended into surfaces. The extended smooth margin areas in the parametric meshes, generated by evaluating the entire network space, exhibit the remarkable extrapolation of Gaussians [2]. The accuracy of these parametric surfaces and data compression achieved will be discussed in Section 4.5 with results provided in Table 2.

[1] Alternatively, these vertices can also be used as control points to generate the surface mesh using NURBS or Bezier methods [5]

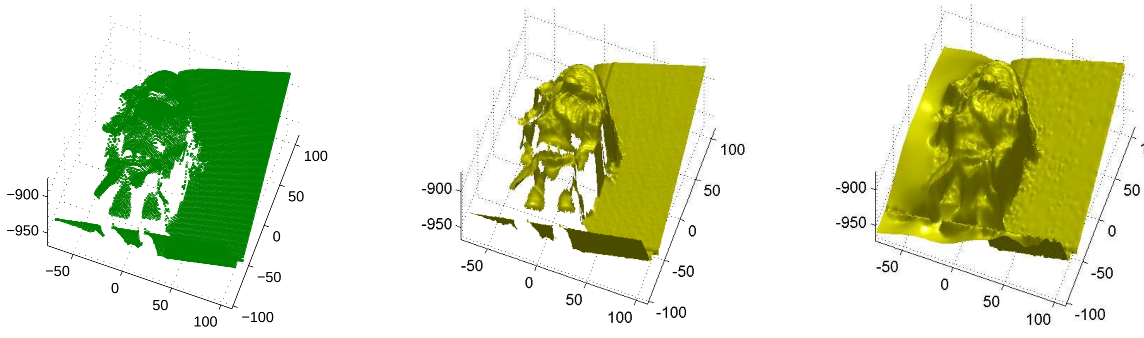
[2] Margin areas in the parametric meshes in Fig. 3 can be removed according to the margin labels in the range scans.



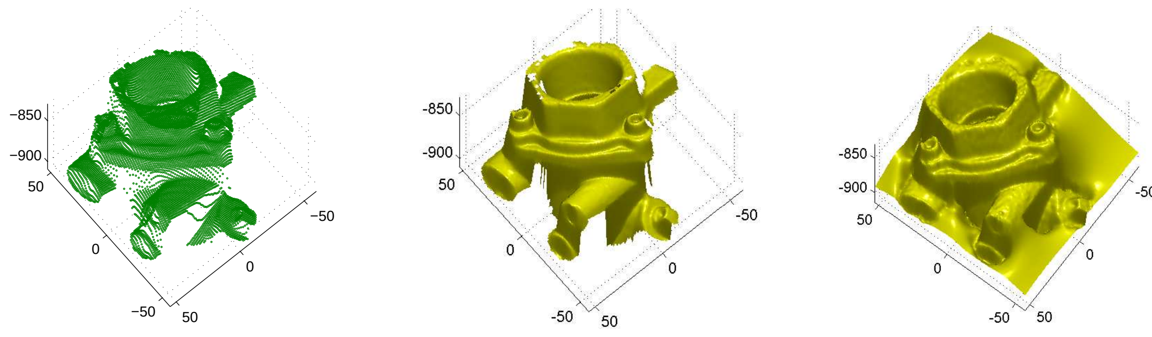
(a) "cow" ($\mathcal{N} = 4,956$) with large, irregularly shaped holes; parametric surface reproduced by $K = 522$ neurons, $\bar{\mathrm{E}} = 0.58\%$.



(b) "face" ($\mathcal{N} = 18,370$) with bumpy facial regions; parametric surface reproduced from 1056 neurons.



(c) "Santa" ($\mathcal{N} = 23,429$) with large portion of missing data at boundaries; parametric surface reproduced from 1901 neurons.



(d) "valve" ($\mathcal{N} = 10,145$) with flat faces and sharp edges; parametric surface reproduced from 1527 neurons.

Figure 3: Direct meshing results (middle column) from clouds of $\mathcal{N}$ points (left), and repaired parametric surfaces generated from ASRBF networks (right).

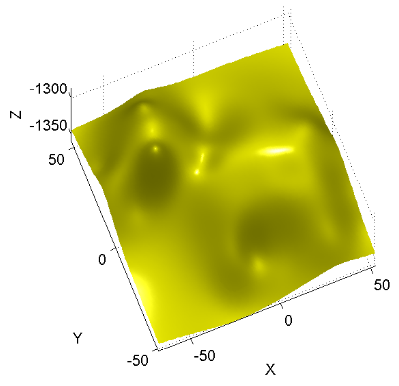
4.2. Multi-level parameterization and multiple LODs

Reproducing a surface with high fidelity to the level of detail in its raw scan is not always desirable. A smooth approximation would be more preferable when point clouds are severely corrupted with noise. Moreover, meshing at different LODs is often required for real-time rendering and multi-resolution purposes.

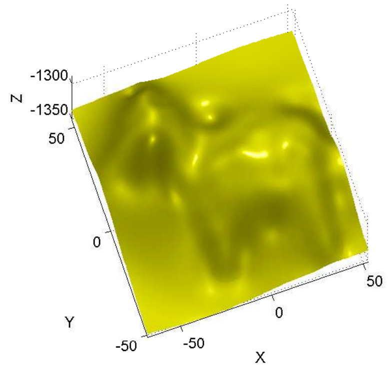
The proposed ASRBF parameterization provides the flexibility of LOD control in two ways. The first is to parameterize the point cloud towards the best resolution provided in the original scan, then compute parametric points and surfaces at degrading sampling levels to get downgraded LODs. This method is efficient when multiple LOD meshes of a surface are required from the parameterization. In the case where only one specific downgraded LOD is required, the second method, which carries out downgraded parameterization at a corresponding detail level, would be beneficial for compactness. To this end, referring to the first novelty criterion (Section 3.3), the neuron separation level $\mathscr{D}_{\min}$ can be used to control the neuron density in the network, and hence the detail level of parameterization. Generally, a larger separation level $\mathscr{D}_{\min}$ will allow a smaller number of neurons that are more sparsely distributed, producing a lower resolution parameterization; conversely, an appropriately smaller value of $\mathscr{D}_{\min}$ will allow more neurons to be added, thus improving parameterization fidelity towards scan details.

Figure 4 demonstrates varying mesh LODs from multi-level parameterization of the “cow”. The “cow” range scan contained 4956 points. We used the data twice in random point orderings to generate training sequences of sufficient length. In Fig. 4(a), the separation value $\mathscr{D}_{\min} = 0.1$ produced a much downgraded parameterization relative to the resolution (200×200) of the original scan. Therefore, only a coarse profile of the “cow” was reproduced from the highly compacted network, with $K = 53$ neurons. When $\mathscr{D}_{\min}$ was reduced to 0.05 (Fig. 4(b)) and 0.03 (Fig. 4(c)), the parameterization generated increasing numbers of neurons ($K = 116$ and 224 respectively), leading to more surface details being presented. When the separation $\mathscr{D}_{\min}$ was set to 0.01 in Fig. 4(d), a highly detailed surface was achieved from the network with $K = 522$ neurons.

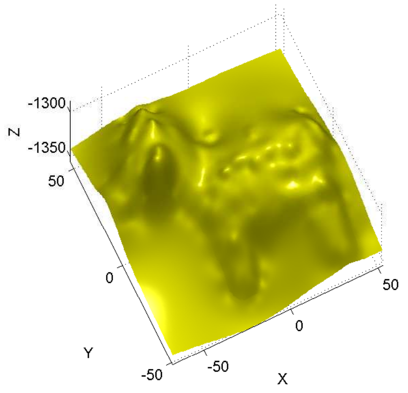
Using the example of the “cow”, Fig. 5 shows how the neuron separation level $\mathscr{D}_{\min}$ affects neuron density and the detail level of parameterization. The sequential training data were generated using four different randomization of the “cow” scan to avoid the possible effects due to the training data length. The normalized parameterization error indicated in Fig. 5 was defined by $\bar{E} = avg\{\|f(\mathbf{x}_n) - z_n\|\}|_{n=1,2,\ldots\mathscr{N}}$. It is the average error between z_n values of all $\mathscr{N}$ points in the scan and their corresponding RBF outputs $f(\mathbf{x}_n)$ at



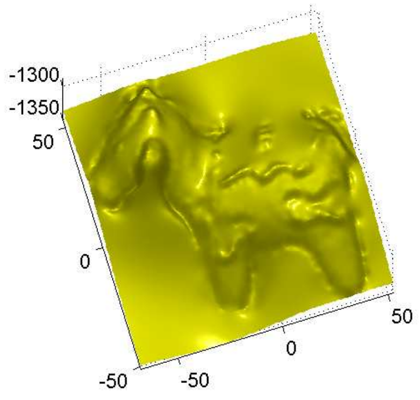
(a) $\mathscr{D}_{\min} = 0.1$, $K = 53$ neurons



(b) $\mathscr{D}_{\min} = 0.05$, $K = 116$ neurons



(c) $\mathscr{D}_{\min} = 0.03$, $K = 224$ neurons



(d) $\mathscr{D}_{\min} = 0.01$, $K = 522$ neurons

Figure 4: Surface LODs of the "cow" generated from multi-level parameterization using ASRBF networks. The detail level of parameterization is controlled by neuron separation $\mathscr{D}_{\min}$. A higher $\mathscr{D}_{\min}$ value downgrades the level of parameterization relative to the resolution in the original scan, and therefore a degraded LOD is obtained from a compact network composed of fewer K neurons.

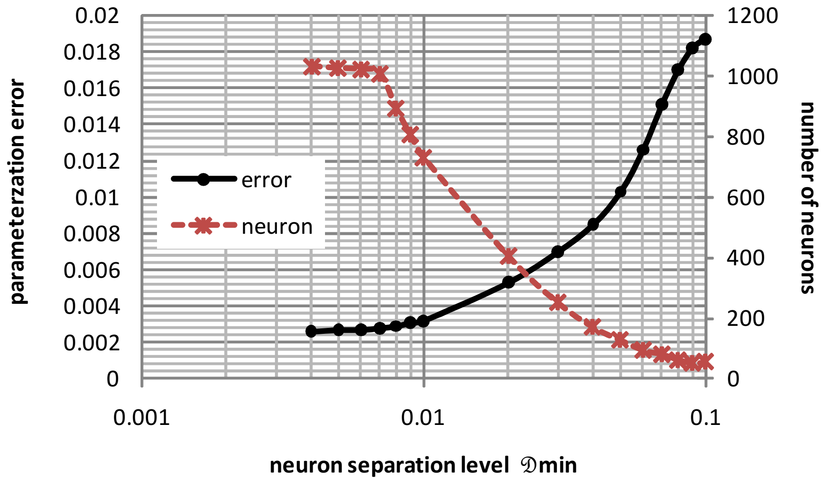


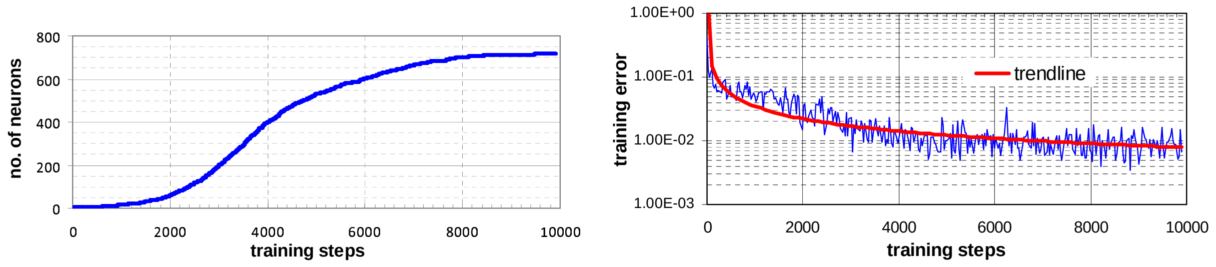
Figure 5: The effect of neuron separation level $\mathscr{D}_{\text{min}}$ on network compactness and parameterization accuracy, as shown by neuron separation level $\mathscr{D}_{\text{min}}$ vs. parameterization error and number of neurons in the "cow" parameterization.

the same location $\mathbf{x}_n = (x_n, y_n)$. We found that when $\mathscr{D}_{\text{min}}$ was 0.01, the parameterization error was around 0.0032, the number of neuron was about 730; when $\mathscr{D}_{\text{min}}$ was greater than 0.01, the parameterization error increased exponentially due to a consistently reducing number of neurons, therefore downgrading parameterization relative to the resolution provided in the original scan; when $\mathscr{D}_{\text{min}}$ was less than 0.01, the parameterization accuracy was incrementally improved until parameterization accuracy achieved 0.0028, and network reached saturation with around 1010 neurons at $\mathscr{D}_{\text{min}} = 0.007$. Obviously, this benefit on accuracy came at the expense of a largely increased number of neurons. Based on our experiments, we considered $\mathscr{D}_{\text{min}} = 0.01$ was appropriate regarding best detail level of parameterization and network compactness.

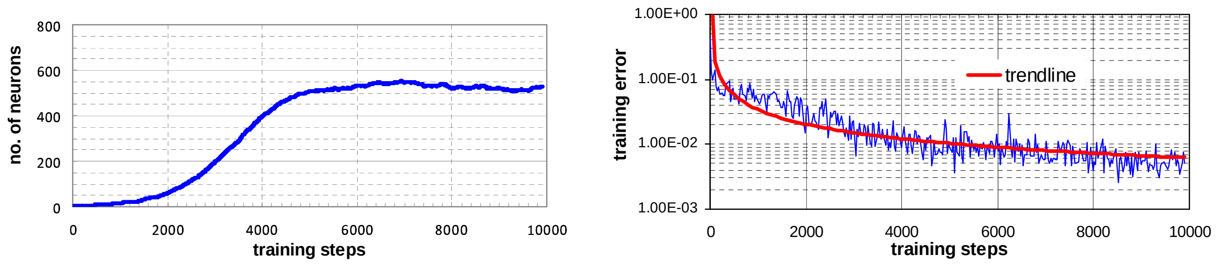
4.3. Parameterization compactness enhanced by neuron pruning

Figure 6 demonstrates the effectiveness of pruning, using the example of parameterization of the "cow". We compared RMS network error (as defined in Eq. 6) and number of neurons involved for cases with pruning (Fig. 6(b)) and without pruning (Fig. 6(a)). To keep the network stable and ensure effective evaluation of neuron contribution, the pruning threshold was set to $\mathscr{P} = 0.001$ for $\omega = 1000$ consecutive observations for the pruning results.

The experimental results shown in Fig. 6 were averaged over 10 trials. Based on these experiments, we observed that, at the start of parameterization, neurons were consistently recruited into the network in both cases (Fig.6 left column), and the RMS error reduced rapidly (Fig.6 right column). When the network error leveled to a steady value after 4000 steps, the with-pruning network in Fig. 6(b) showed effective neuron growth control. Pseudo neurons, exhibiting very limited contributions $r_k < \mathscr{P}$ for 1000 consecutive observations, were detected and removed. By the end of construction, the with-pruning network produced only 522 neurons, while achieving competitive accuracy. However, without pruning, as shown in Fig. 6(a), neurons were added incessantly throughout network construction, resulting in 718 neurons.



(a) without pruning: neurons were consistently added into the network (left), RMS error reduced during parameterization (right).



(b) with pruning: neurons were added and pseudo neurons were removed for effective control of network growth (left), RMS error (right) reduced in a comparable way. A pseudo neuron was detected if its contribution ratio was less than $\mathscr{P} = 0.001$ over $\omega = 1000$ consecutive inputs.

Figure 6: Parameterization with and without neuron pruning.

4.4. Neuron adaptivity

The adaptivity of parameterization is derived from the adaptive RBF learning: 1) neurons are heuristically located according to the novelty of input; 2) pseudo neurons can be removed by pruning; 3) neuron parameters are adjustable in full dimensionality of location, width and weight.

Figures 7 and 8 illustrate neuron spatial distribution and properties of width and weight in the parameterization network space for the “cow” and “face”. For a more intuitive visualization, neurons are represented by circles displayed in normalized 3D network space at their 2D centers, with their depth z-values evaluated from the network. To allow better insight, neuron width (Fig. 7 and Fig. 8 bottom left) is indicated by the radius of the neuron circle, and the absolute value of neuron weight (Fig. 7 and Fig. 8 bottom right) is presented by the neuron circle diameter. Both display ratios are reduced to 1:3 to lessen visual clustering [3].

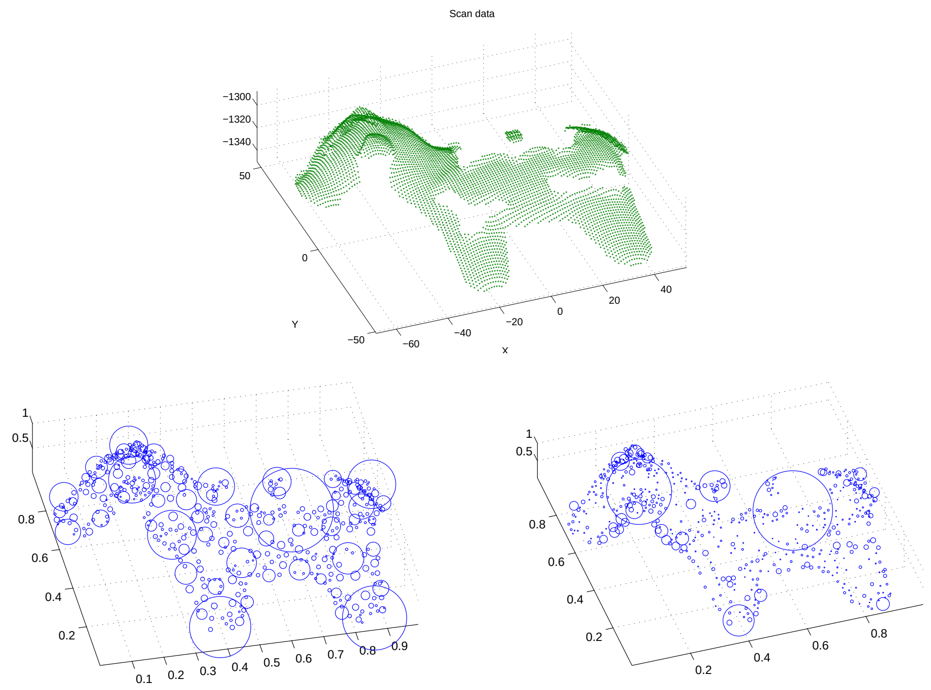


Figure 7: Neuron spatial distribution and properties in the “cow” parameterization. Top: point cloud (4,956 points); bottom left: neuron width of 522 Gaussians in the network; bottom right: neuron weight.

We observe that neuron distribution and density are highly consistent with surface variations. Meanwhile the neuron properties of width and weight reflect spatial features. Neurons with large widths or high weights were located in areas with lower variations. These were usually created at the start of network training to form a smooth base, while smaller and denser neurons were presented in regions with highly variable details. Although there is an inherent tendency by the greedy algorithm to favor capture of lower frequencies before higher ones, smaller neurons consistently retouch the smoothness towards increasing fidelity to local details. Figure 9 shows the average width and weight of neurons at each training step of network construction for the “cow” and the “face”. It visualizes an automatic mechanism of neuron scale decline during adaptive coarse-to-fine RBF fitting.

Table 1 provides statistics on neuron properties for both networks. At the end of training, the “cow” parameterization used 522 neurons, relatively large and weighed, to approximate the smoothly varying

[3]Small neurons in Fig. 7 and Fig. 8 may not be visible due to limited display resolution.

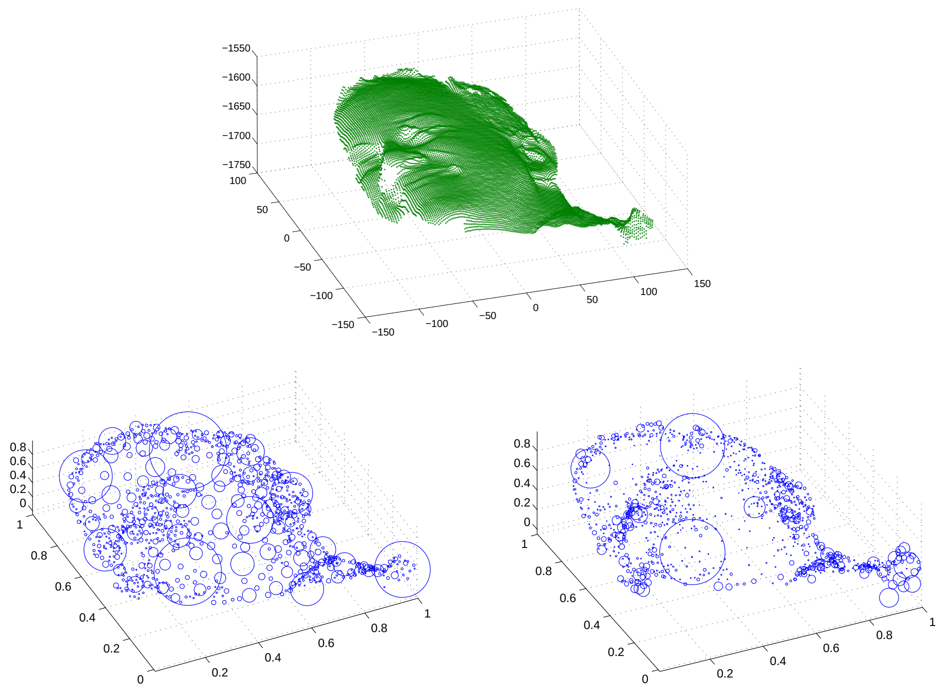
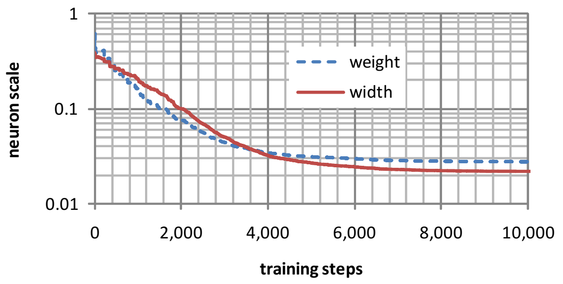
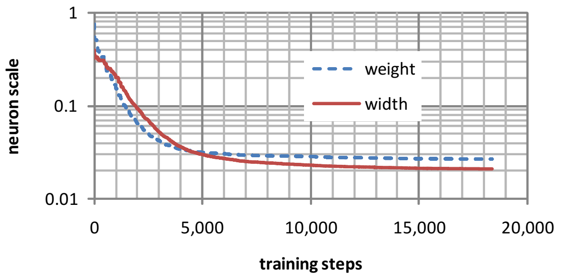


Figure 8: Neuron spatial distribution and properties in the “face” parameterization. Top: point cloud (18,370 points); bottom left: neuron width of 1056 Gaussians in the network; bottom right: neuron weight.



(a) the "cow" parameterization.



(b) the "face" parameterization.

Figure 9: Average neuron width and weight decrease during coarse-to-fine RBF fitting. The results were averaged over 10 trials. To fully investigate the trend of neuron scales, the "cow" data was used twice in random point orderings to obtain training sequences of sufficient length.

body shape; whereas 1,056 neurons were generated by the parameterization for the more complex "face". As indicated by the standard deviations, the neuron weighting of the "cow" varied more than those of the "face". This was probably because the "cow" simultaneously possessed richer features at both low (e.g. smooth variation of the body) and high frequencies (e.g. protruding patterns on the body) than the "face".

parameterization network	neuron width				neuron weight			
	max	min	mean	std	max	min	mean	std
cow (522 neurons)	.4673	.0081	.0307	.0426	.8337	.0033	.0311	.0544
face (1,056 neurons)	.4364	.0038	.0210	.0312	.6896	.0065	.0269	.0477

Table 1: Neuron properties in parameterization RBF networks.

4.5. *Parameterization accuracy and data compression*

Experimental results on the compactness and accuracy of the parameterizations from the examples in Fig. 3 are provided in Table 2, in which $\mathscr{N}$ stands for the number of surface points in a range scan, and K denotes the number of Gaussian neurons generated by the network. Data compression is indicated by: 1) point to neuron compactness ratio $\mathscr{N} : K$, and 2) storage compression ratio $3\mathscr{N} : 4K$, defined as the total storage of $\mathscr{N}$ 3D range points to the total storage of K neurons, each represented by the 4 parameters of width, weight and 2D centers. For a fairer comparison of network performance on different scans, parameterization error $\bar{E}$ was provided in normalized network space, as defined in Section 4.2, indicating a percentage accuracy relative to the data variation range in each scan. The absolute reconstruction errors

point clouds	# of points	# of RBFs	normalized error	data compression		reconstruction error (mm)	
	$\mathcal{N}$	K	$\bar{\mathrm{E}}$	$\mathcal{N}:K$	$3\mathcal{N}:4K$	mean	std
face	18,370	1,056	.0036	17.4:1	13.1:1	.50	.58
Santa	23,429	1,901	.0055	12.3:1	9.2:1	.51	.67
cow	4,956	522	.0058	9.5:2	7.1:1	.36	.46
valve	10,145	1,527	.0069	6.6:1	5.0:1	.61	.89
average	14,225	1,251	.0055	11.5:1	8.6:1	.50	.65

Table 2: Parameterization accuracy and data compression.

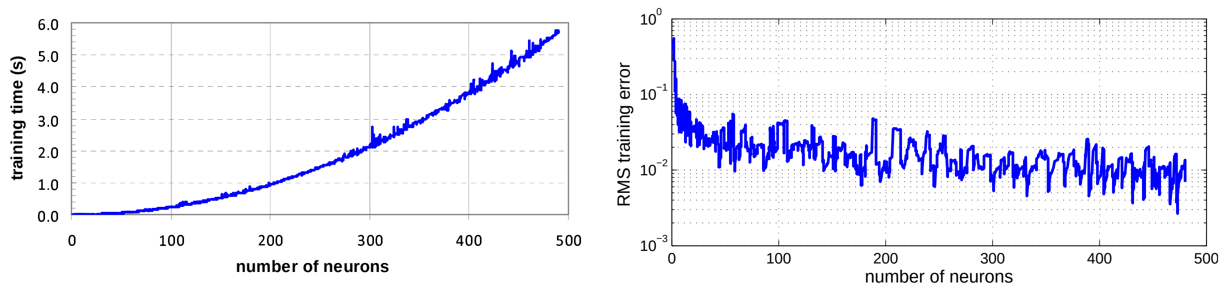
of these parametric surfaces were also given as a means for comparison with other works. It is represented by the mean and standard deviation of the errors between each re-scaled network output (restoring actual aspect ratio and 3D size of the surface) and its corresponding scan point in the real-world measurement of millimeters.

The results show that the ASRBF network provides a compact parameterization of range data with a desired accuracy. The average total number of Gaussians was less than one tenth of the total number of range points, and the average storage compression ratio achieved 8.6 : 1. The normalized parameterization error $\bar{E}$ reached the level of 0.55%. The average accuracy for the parametric surfaces achieved 0.50mm in z, where the average variation of z-values in the four scans was 9.35cm. The “face”, with smooth variations of facial features and curly hair patterns, achieved the highest compression ratio and best accuracy by using Gaussians. However, simultaneously modeling both low frequency (e.g. large background plane in the “Santa”) and high frequency features (in particular the “valve”), was relatively difficult and costly.

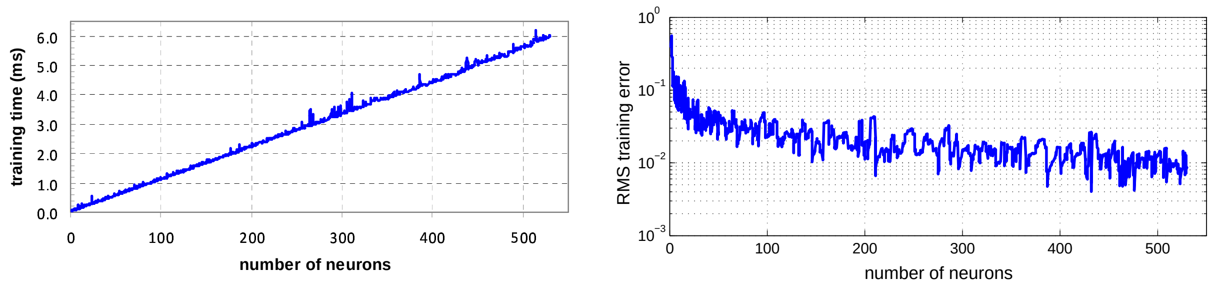
It seems that absolute reconstruction errors did not consistently agree with the normalized $\bar{E}$, due to the data range varying among the different scans. For example, the variation of z-values in the “face”, “Santa” and “valve” were 13.9 cm, 9.3 cm and 8.0 cm respectively, but it was only about 6.2 cm in the “cow”. Therefore, although the normalized error of the “cow” was ranked third, its absolute reconstruction error was comparatively lower than the others.

4.6. Parameterization efficiency

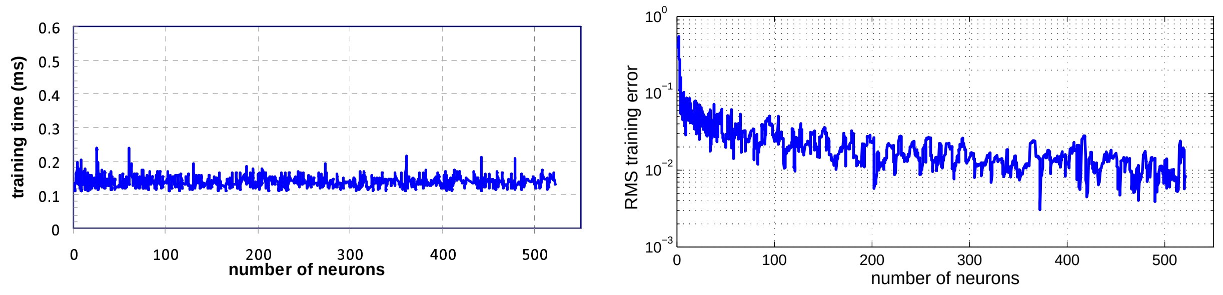
Parameterization time is mainly determined by the amount of time used in updating neuron parameters during network optimization. We therefore compared the speed and accuracy of the neighborhood EKF (NEKF) method with two commonly used methods: gradient descent (GD) [37] and global EKF (GEKF) [38]. The experimental results were obtained using a Pentium 3GHz PC with 1GB RAM. Figure 10 shows



(a) GEKF [38]: parameterization using GEKF produced 490 neurons, showing an exponentially increasing network training cost up to the level of seconds.



(b) GD [37]: parameterization using GD produced 530 neurons, showing a linearly increasing network training cost up to the level of milliseconds.



(c) NEKF: parameterization using NEKF produced 520 neurons, showing a consistent cost at a level of 10^{-4} seconds.

Figure 10: Comparison of how parameterization speed was affected by an increasing number of neurons using GEKF, GD or NEKF. Training time costs are displayed on the left, and their corresponding RMS errors are shown on the right.

results averaged over 10 trials of the “cow”. The NEKF results in Fig. 10(c) demonstrate the extreme case of updating only the nearest neighbor at each learning step.

As shown in Fig. 10 right column, network updating using the three aforementioned methods produced comparable accuracy with similar network RMS errors and error reduction rate, although the GEKF had slightly better accuracy than the GD and NEKF due to a global optimization strategy employed.

However, the network training time (Fig. 10, left column) differed remarkably with increasing number of neurons. The GEKF showed a complexity of $O((4K)^2)$ for K neurons (Fig. 10(a)). The GD appeared to have a linear relationship (Fig. 10(b)). The cost of the NEKF (Fig. 10(c)), however, was nearly constant $O((4)^2)$ when updating only the nearest neighbor. Training time should also remain be consistently low of $O((4K')^2)$, when updating a small set of K' neighbor neurons. Towards the end of network training, the GEKF spent 6.3 seconds to update around 490 neurons, the GD used 6.1 milliseconds for 530 neurons, while the NEKF spent only around 0.12 milliseconds throughout the training process, right through to the case of 522 neurons. Compared to the GEKF, the computational load of NEKF was dramatically reduced, by a factor of 10^4.

5. Discussions

5.1. Network parameters

The adaptive parameterization RBF network is conceptually simple and straightforward to implement. The network employs a number of parameters. Based on their functions, they can be used flexibly, facilitating the use of the network for different proposes, such as controlling the level of parameterization detail by using different values of $\mathscr{D}_{\min}$, as shown in Fig. 4. After the scan image was normalized to a unit cube in x, y and z dimensions, default values were used and are recommended for the general purpose of surface parameterization and reproduction. In this section, we indicate how the defaults were chosen or automatically calculated from the input data.

Local accuracy threshold E and RMS error E_ω are utilized in the novelty criteria to determine if a new neuron should be added. The value 0.01 was used as default for both the parameters, as this value not only helped to better preserve the fidelity to local inputs, but also to tolerate a certain degree of measurement errors.

The value of separation distance $\mathscr{D}_{\max} = 0.4$ was used to enforce a sparse neuron distribution in normalized space at the beginning of network construction. The neuron separation $\mathscr{D}_{\min}$ helped to control the density level of newly inserted neurons. This value can be obtained by doubling the uni-dimensional sampling density of input data. Thus for a sampling density of $1/200$ in our 200×200 scan array, we set $\mathscr{D}_{\min} = 2 \times 1/200 = 0.01$. The default decay factor $\gamma = 0.99$ provided a moderate decline speed of neuron separation, giving opportunity for coarse-to-fine RBF fitting.

We found that these default values worked effectively on large datasets containing different objects with highly varying and complex spatial features; there was no need to adjust them from one experiment to another. Critically, parameters only need to be set at an approximate level as opposed to a precise value. This relaxation is gained due to the nature of adaptive learning as employed by the ASRBF. For example, the above network parameters are mainly associated with the necessity of inserting new neurons. Using these general settings, neurons added unnecessarily, wrongly inserted due to outliers or having become redundant as a result of network optimization, can still be removed from the network by the pruning process.

In our experiments, only the Gaussian RBF overlap factor ψ was set to slightly different values, varying between $0.7 \sim 0.8$. For example, the "cow", possessing smoother variations, used a ψ value of 0.8, whereas "face", "Santa" and "valve", with relatively sharper details, used a lower ψ value of 0.7. To evaluate the network accuracy achieved from different values of ψ at a sufficient training length, we replicated the data in random orderings to extend the training sequence. The experimental results on the "cow" with ψ values between 0.8 ± 0.1 are shown in Fig. 11, where $\mathscr{N}$ denotes the number of points in the scan. We observe that the different ψ values have a limited effect on network accuracy. This is because the overlap factor ψ is only used to initialize the width of a newly inserted neuron; however each neuron can be later adjusted dynamically during iterative learning and in full dimensionality in terms of location, weight and width. In summary, precise parameter values do not need accurately pre-determined by trial and error, since the network automatically fine-tunes itself by means of adaptive learning.

5.2. Density and size of point-cloud data sets

The parameterization RBF network is capable of handling non-uniformly sampled data with varying densities and sizes. In a normalized space, a large number of scan points in the "face" (18,370 points) and "Santa" (23,429 points) presented higher densities; whereas the size and density of the "cow" (4956 points) and "valve" (10,145 points) were relatively lower. Unlike other RBF methods, which could encounter over-fitting problems when dealing with dense point sets, the parameterization RBF network essentially favors large and dense data sets, because its neurons are generated only according to *novel inputs* rather than from all points. Sufficient length of point cloud data is desired to produce high detail level of parameterization.

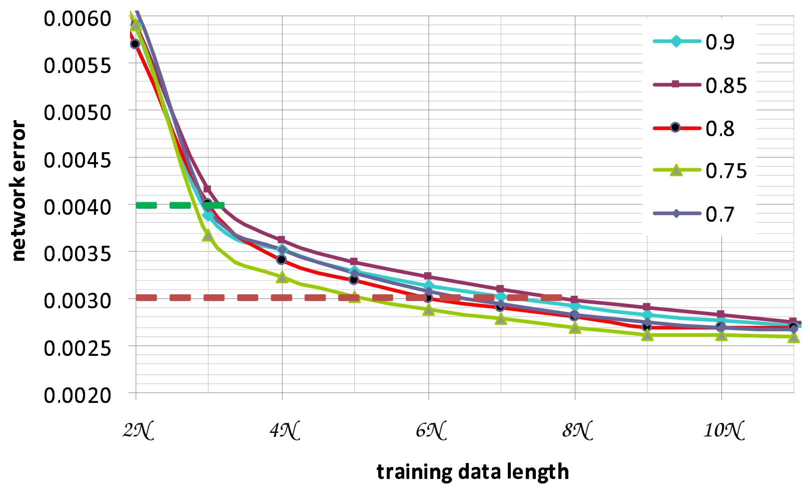


Figure 11: Limited effect of different ψ values on network accuracy. $\mathscr{N}$ denotes the number of points in the scan.

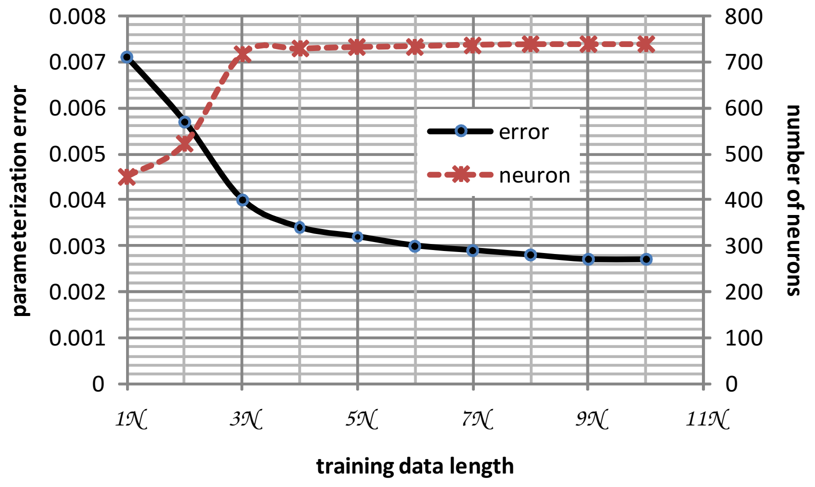


Figure 12: Replication of original data improves parameterization accuracy: replication of scan data vs. parameterization error and number of neurons. $\mathscr{N}$ denotes the number of points in the scan.

When a surface was somewhat under-sampled relative to its geometric complexity (e.g. the "cow"), we replicated the data in random orderings to extend the training sequence. Although this method did not create any new inputs, it did increase the chance of a point being a novel instance during a longer process of network optimization, thereby promoting the accuracy of parameterization. In the example of the "cow", Figure 12 demonstrates how the number of data point replications of the original scan affected the accuracy and number of neurons in the network. When using a single replication of scan data (from $\mathscr{N}$ to $2\mathscr{N}$), a moderate parameterization error $\bar{E} = 0.0058$ was achieved with 522 neurons; when using the data three times ($3\mathscr{N}$), the error was significantly reduced to $\bar{E} = 0.004$ with increased number of neurons ($K = 718$). After replicating the data to four sets ($4\mathscr{N}$), the network achieved saturation, so that the extended training sequence length by another $\mathscr{N}$ caused neuron number to increase only a couple each time. After $8\mathscr{N}$, the neuron number remained constant at 738. However, the network accuracy was still improved incrementally as the result of network optimization. This indicates that a certain number of training replications can help deal with under-sampled data sets and always improve network accuracy in sequential learning.

5.3. Performance comparisons

point cloud	GEKF [38]				GD [37]				NEKF			
(#points)	T_{para} (hrs)	T_{rep} (s)	#RBFs	$\bar{\text{E}}$ (10^{-3})	T_{para} (s)	T_{rep} (s)	#RBFs	$\bar{\text{E}}$ (10^{-3})	T_{para} (s)	T_{rep} (s)	#RBFs	$\bar{\text{E}}$ (10^{-3})
cow (4,956)	0.75	.74	490	5.5	4.1	.8	530	6.2	1.4	.8	522	5.8
face (18,370)	23.6	5.8	965	3.2	61.4	6.7	1097	4.1	14.5	6.6	1056	3.6
valve (10,145)	42.5	4.7	1452	7.1	35.8	5.4	1690	9.1	8.7	5.3	1527	7.5
Santa (23,429)	98.2	13.6	1786	4.9	153.4	17.5	2159	7.0	33.1	15.6	1901	5.5
average (14,225)	41.3 hrs	6.2 s	1,073	5.1	63.7 s	7.6 s	1,369	6.6	14.4 s	7.1 s	1,252	5.6

Table 3: Parameterization performance when using GEKF, GD and NEKF.

Table 3 shows the performance results of parameterization using ASRBF networks. We presented results on parameterization time (T_{para}), the time (T_{rep}) used to reproduce all points at original inputs, number of Gaussian RBF (#RBF) and normalized parameterization error ($\bar{\text{E}}$). We compared our results using NEKF to the results using GEKF [38] and GD [37]. The experiments were carried out on PC with 3GHz Pentium processor and 1GB of RAM. For comparison, consistent network parameters were used on each range scan for GEKF, GD and NEKF, so that parameterization with each of the three methods was carried out to the same desired accuracy level while similar numbers of neurons were produced.

From the results shown in Table 3, we observed that parameterization using GEKF appeared to achieve a slightly better accuracy, but this came at expense of increasing training time by several orders of magnitude. This was due to the high complexity $O((4K)^2)$ required by the GEKF to update all K neurons at each input. This computational cost overhead become unmanageable for large networks and training sets. Parameterization time using GD increases linearly with the number of neurons and training points. Comparing GEKF to GD, average training time used by the GEKF was 41.3 hours, whereas the GD required only 63.7 seconds. The NEKF achieved the fastest parameterization, at an average of 14.4 seconds. Only a handful of neighbor neurons were updated at each training input. Its average training time was further reduced to one fourth of that used by GD. Furthermore, it also improved on GD accuracy by 25%, using 8.5% fewer neurons.

The time used for reproducing a point from the parameterization network is directly proportional to the number of RBF neurons. In our experiments, the average time used to compute one RBF output was around 3.3×10^{-7} seconds. Since each of the three methods generated a similar number of neurons, the reconstruction times T_{rep} are therefore on a similar level for each scan. By comparison, NEKF parameterization used less time ($T_{rep} = 7.1$ seconds) on average to produce 14,225 parametric points, while retaining a relatively high accuracy of 5.6×10^{-3}, while using fewer neurons (1,252 neurons).

In addition, the proposed adaptive dynamic RBF network structure outperforms networks with fixed structures. We compare our approach with the most relevant and state-of-art multi-layer hierarchical RBF (HRBF) network structures [34, 35]. Each layer in the HRBF network contains a regular grid of Gaussians at decreasing scales. A neuron is located according to the extent of local mapping error, thus the grids do not need to be fully filled. Although neurons still have to be placed on the partition grids, neuron sizes vary at multi-scales by means of halving the width at every higher layer. Once a neuron is inserted, it cannot be moved or discarded. The network construction time increases linearly to the size of scan data and the number of neurons involved.

The HRBF network has been applied to point-cloud surface modeling. In [35], a 4-layer HRBF network was used to approximate a human face scan containing 12,641 points. Experimental results showed that the reconstruction error achieved 0.77 mm, with 5,570 Gaussians being generated by the network. The compression ratio $\mathscr{N} : K$ was thus 2.3:1. In [34], results based on a toy face scan (16,851 points) showed a similar reconstruction accuracy of 0.779 mm and compression ratio $\mathscr{N} : K = 2 : 1$ involving the use of 8,087 Gaussians in a 4-layer HRBF network.

In comparison, thanks to the adaptive learning and dynamic network structure, as shown in Table 2, the

ASRBF achieved higher compression ratios at comparable accuracy. This is particulary evidenced by the "face" data set ($\mathscr{N} = 18,370$ points). Only $K = 1,056$ Gaussians were generated for the "face" using the ASRBF parameterization network. The compression ratio $\mathscr{N} : K = 17.4 : 1$ achieved by the ASRBF was significantly higher than that by the HRBF networks, while ASRBF also achieved a comparable level of reconstruction accuracy of 0.50mm.

5.4. Comparison with Self-Organizing Maps

The ability of SOM to learn topological maps from input data distributions has been explored for surface reconstruction from vertices in a mesh or a point cloud [30, 31]. This section compares results on the accuracy of meshes reproduced from SOM [29] and the proposed ASRBF. Firstly, range scan data was used to train the ASRBF network and SOM network. Surface vertices calculated from the obtained ASRBF network and from the SOM were meshed using the same function Ball Pivoting in Meshlab [42]. The accuracies of the ASRBF mesh and the SOM mesh with respect to original mesh generated from the raw scan were computed using Metro Tool [43].

Accuracy measures presented by the Hausdorff distance, the mean and RMS deviations from each original mesh to its reconstructed mesh, are shown in Table 4. The Hausdorff distance indicates the maximum difference between two meshes; however, the mean and RMS values are more descriptive due to being less susceptible to the influence of outliers. The accuracy measures in Table 4 were averaged over 10 trials, presented as a percentage of the bounding box diagonal length in the original mesh. Since range data input was over a 200×200 square, for fair comparison, we chose the most square-like SOM grid with the number of SOM nodes and the the number of neurons in the ASRBF as close as possible. The size of the SOM grid used for each point cloud is given in the first column of SOM table entries.

The results show that the ASRBF networks produce better accuracy than standard SOM when using similar numbers of neurons. As described in previous sections, the advantage of ASRBF based mesh reproduction is primarily driven by the benefits from: 1) the extraordinary interpolation and extrapolation capability of Gaussian kernels to provide better surface fidelity; 2) network structure flexibility and neuron adaptivity to the unorganized underlying data. The SOM requires a grid structure composed of point nodes. The 3D locations of SOM point nodes are used directly as vertices on the mesh. Although the SOM net can be broken where necessary to adapt to the input data, the pre-defined number of nodes and fixed connection linkages ultimately constrain the flexibility and accuracy of the topological map generated by SOM.

point clouds	ASRBF				SOM [29]			
	# of neurons	Hausdorff dist.	mean	RMS	# of nodes	Hausdorff dist.	mean	RMS
face	1,056	1.5%	.09%	.14%	1,056 (33×32)	2.6%	.16%	.31%
Santa	1,901	2.6%	.09%	.15%	1,892 (44×43)	4.7%	.17%	.38%
cow	522	1.5%	.12%	.19%	529 (23×23)	2.2%	.19%	.33%
valve	1,527	2.9%	.13%	.19%	1,521 (39×39)	3.6%	.25%	.49%
average	1,251	2.1%	.10%	.17%	1,250	3.3%	.19%	.38%

Table 4: Comparison on mesh accuracy reproduced from the ASRBF and SOM.

6. Conclusions

We presented a neural network based method to solve the problem of point-cloud surface parameterization. The network employs a dynamic structure through adaptive learning. It allows Gaussian neurons to be fitted according to the novelty of inputs, while also being fully adjustable on their locations, widths and weights. Compared to approaches using RBFs at fixed locations and at pre-defined scales, our approach achieved a high compression ratio and a comparable level of accuracy. The developed NEKF method dramatically reduces the training cost to a manageable time, enabling parameterization of extensive point clouds, involving the use of large scale of networks. Experimental results show that complete surfaces can be accurately reproduced from unified low-storage parameterization networks, and multiple LODs can be easily obtained. Our adaptive learning strategy contributes to the general problem of effective RBF fitting, and thus could be of value to other RBF based methods. The possibility of effectively absorbing nonuniform mapping residuals in areas with learning difficulties may be addressed by extending the adaptivity to a multi-layer network. This is left for future work.

References

[1] A. Sheffer, E. Praun, K. Rose, Mesh parameterization methods and their applications, Foundations and Trends in Computer Graphics and Vision 2 (2) (2006) 105–171.

[2] X. Sun, E. Hancock, Quasi-isometric parameterization for texture mapping, Pattern Recognition 41 (5) (2008) 1732–1743.

[3] M. Floater, K. Hormann, Surface Parameterization: a Tutorial and Survey. Advances in Multiresolution for Geometric Modelling, Springer Berlin Heidelberg, 2005.

[4] M. Duckham, L. Kulik, M. Worboys, A. Galton, Efficient generation of simple polygons for characterizing the shape of a set of points in the plane, Pattern Recognition 41 (10) (2008) 3224–3236.

[5] M. Floater, K. Hormann, Parameterization of triangulations and unorganized points, in: Tutorials on Multiresolution in Geometric Modelling, 2002, pp. 287–315.

[6] E. Zhang, K. Mischaikow, G. Turk, Feature-based surface parameterization and texture mapping, ACM Transactions on Graphics 24 (1) (2005) 1–27.

[7] C. Gotsman, X. Gu, A. Sheffer, Fundamentals of spherical parameterization for 3D meshes, in: ACM SIGGRAPH, 2003, pp. 358–363.

[8] E. Praun, H. Hoppe, Spherical parametrization and remeshing, ACM Transactions on Graphics 22 (3) (2003) 340 – 349.

[9] J. Schreiner, A. Asirvatham, E. Praun, H. Hoppe, Inter-surface mapping, ACM Transactions on Graphics 23 (3) (2004) 870–877.

[10] S. Dong, P. Bremer, M. Garland, V. Pascucci, J. Hart, Spectral surface quadrangulation, in: ACM SIGGRAPH, 2006, pp. 1057 – 1066.

[11] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, A. Wu, An efficient k-means clustering algorithm: Analysis and implementation, IEEE Trans. Pattern Anal. Mach. Intell. 24 (7) (2002) 881–892.

[12] Y. Miao, J. Feng, C. Xiao, Q. Peng, A. Forrest, Differential-based segmentation and parameterization for point-sampled surfaces, J. of Computer Science and Technology 22 (5) (2007) 749–760.

[13] R. Morales, Y. Wang, Z. Zhang, Unstructured point cloud surface denoising and decimation using distance RBF K-nearest neighbor kernel, in: PCM Advances in Multimedia Information Processing, 2011, pp. 214–225.

[14] M. Zwicker, C. Gotsman, Meshing point clouds using sphereical parameteriztion, in: Proc. Eurographics Symposium on Point-Based Graphics, 2004, pp. 173–180.

[15] F. Fernández, C. Hervásand, P. Gutiérrez, A dynamic over-sampling procedure based on sensitivity for multi-class problems, Pattern Recognition 44 (8) (2011) 1821–1833.

[16] C. Silva, S. Ranganath, L. Silva, Cloud basis function neural network: A modified RBF network architecture for holistic facial expression recognition, Pattern Recognition 41 (4) (2008) 1241–1253.

[17] A. Alexandridisa, H. Sarimveisb, K. Ninosb, A radial basis function network training algorithm using a non-symmetric partition of the input space - application to a model predictive control configuration, Advances in Engineering Software 42 (10) (2011) 830–837.

[18] Q. Meng, M. Lee, Automated cross-modal mapping in robotic eye/hand systems using plastic radial basis function networks, Connection Science 19 (1) (2007) 25–52.

[19] B. Choi, J. Lee, Comparison of generalization ability on solving differential equations using backpropagation and reformulated radial basis function networks, Neurocomputing 73 (2009) 115–118.

[20] Y. Lu, N. Sundararajan, P. Saratchandran, Performance evaluation of a sequential minimal radial basis function (RBF) neural network learning algorithm, IEEE Trans. Neural Networks 9 (2) (1998) 308–318.

[21] J. Carr, R. Beatson, J. Cherrie, T. Mitchell, W. Fright, B. McCallum, T. Evans, Reconstruction and representation of 3D objects with radial basis functions, in: ACM SIGGRAPH, 2001, pp. 67–76.

[22] H. Liu, X. Wang, W. Qiang, A fast method for implicit surface reconstruction based on radial basis functions network from 3D scattered points, Int J Neural System 17 (6) (2007) 459–65.

[23] Y. Lin, C. Chen, M. Song, Z. Liu, Dual-RBF based surface reconstruction, Visual Computer 25 (2009) 599–607.

[24] N. Pears, T. Heseltine, M. Romero, From 3d point clouds to pose-normalised depth maps, Int. Journal of Computer Vision

89 (2010) 152–176.

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

[26] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, H. Seidel, Multi-level partition of unity implicits, ACM Trans. Graph. 22 (3) (2003) 463–470. doi:http://doi.acm.org/10.1145/882262.882293.

[27] Y. Ohtake, A. Belyaev, H. Seidel, Sparse surface reconstruction with adaptive partition of unity and radial basis functions, Graphical Models 68 (1) (2006) 15 – 24.

[28] D. Chen, B. Morse, B. Lowekamp, T. Yoo, Hierarchically partitioned implicit surfaces for interpolating large point set models, Geometric Modeling and Processing 4077 (2006) 553–562.

[29] T. Kohonen, The self-organizing map, Neurocomputing 21 (1998) 1–6.

[30] A. Junior, A. Neto, J. de Melo, Surface reconstruction using neural networks and adaptive geometry meshes, in: Proc. IEEE Int. Joint Conf. on Neural Networks, 2004.

[31] L. Varady, M. Hoffmann, E. Kovacs, Improved free-form modelling of scattered data by dynamic neural networks, J. for Geometry and Graphics 3 (2) (1999) 177–181.

[32] J. Barhak, A. Fischer, Parameterization and reconstruction from 3D scattered points based on neural network and PDE, IEEE Trans. Visualisation and Computer Graphics 7 (1) (2001) 1–16.

[33] G. Knopf, A. Sangole, Interpolating scattered data using 2D self-organizing feature maps, Journal Graphical Models 66 (1) (2004) 50–69.

[34] A. Borghese, S. Ferrari, V. Piuri, Real-time surface reconstruction through HRBF networks, IEEE international workshop on haptic virtual environments and their applications (2002) 19 – 24.

[35] S. Ferrari, M. Maggioni, N. A. Borghese, Multiscale approximation with hierarchical radial basis functions networks, IEEE Trans. on Neural Networks 15 (1) (2004) 178–188.

[36] J. Platt, A resource-allocating network for function interpolation, Neural Comput. 3 (2) (1991) 213–225.

[37] N. Karayiannis, Reformulated radial basis neural networks trained by gradient descent, IEEE Trans. on Neural Networks 10 (3) (1999) 657–671.

[38] D. Simon, Training radial basis neural networks with the extended Kalman filter, Neurocomputing 48 (2002) 455–475.

[39] Range image database at Ohio SAMPL, http://sampl.ece.ohio-state.edu/data/3DDB/RID/minolta.

[40] W. Smith, E. Hancock, Facial shape-from-shading and recognition using principal geodesic analysis and robust statistics, Int. J. of Computer Vision 76 (1) (2008) 71–91.

[41] V. Blanz, T. Vetter, A morphable model for the synthesis of 3D faces, in: Proc. SIGGRAPH, 1999, pp. 187–194.

[42] Meshlab, http://meshlab.sourceforge.net.

[43] P. Cignoni, C. Rocchini, R. Scopigno, Metro: Measuring error on simplified surfaces, Computer Graphics Forum 17 (2) (1998) 167–174.

Author Biography

QINGGANG MENG received B.Sc. and M.Sc. degrees in Electronic Engineering from Tianjin University, China and Ph.D. degree in Computer Science from Aberystwyth University, UK. He is a Senior Lecturer in the Department of Computer Science, Loughborough University, UK. His research interests include biologically and psychologically inspired learning algorithms and developmental robotics, service robotics, robot learning and adaptation, multi-UAV cooperation, driver's distraction detection, human motion analysis and activity recognition, activity pattern detection, pattern recognition, artificial intelligence and computer vision. He is a member of the IEEE and a fellow of the Higher Education Academy of United Kingdom.

BAIHUA LI received B.Sc. and M.Sc. degrees in Electronic Engineering from Tianjin University, China and Ph.D. degree in Computer Science from Aberystwyth University, UK. She is a Senior Lecturer in the School of Computing, Mathematics & Digital Technology, Manchester Metropolitan University, UK. Her current research interests include computer vision, pattern recognition, advanced 3D computer graphics, human motion analysis and behavior understanding from multi-modality imaging and sensory data. About 40 fully refereed research papers have been published in leading national/international journals and conferences, including IEEE Trans. SMC, PR and IVC. She takes a role as reviewer and Program Committee member for a number of high quality journals and conferences. She is a member of the BMVA.

HORST HOLSTEIN received the degree of B.S. in Mathematics from the University of Southampton, UK, in 1963, and obtained a Ph.D. in the field of rheology from Aberystwyth, UK, in 1981. He is a Lecturer in the Department of Computer Science, University of Wales, Aberystwyth, UK. His research interests include computer graphics, motion tracking, computational bioengineering and geophysical gravi-magnetic modelling.

YONGHUAI LIU received his first PhD degree from Northwestern Polytechnical University, People's Republic of China, in 1998, and his second PhD degree in computer science from The University of Hull, UK, in 2001.Currently he is a senior Lecturer at Aberystwyth University. He is a guest editor for the special issue of Computer Vision and Image Understanding Journal on the registration and fusion of range images published in 2002. He has served as a program committee member and a referee for more than 30 international conferences and journals. He has published more than 100 papers in international conference proceedings and journals. His primary research interests lie in computer graphics, image registration, motion estimation, pattern recognition, image processing, machine learning, 3D vision and artificial intelligence. He is a member of the IEEE and a fellow of the Higher Education Academy of United Kingdom.