Real-time Surface Reconstruction through HRBF Networks

N. A. Borghese[1] S. Ferrari[2] V. Piuri[2]
[1]*Computer Science Dept.* [2]*Information Technologies Dept.*
University of Milan, Italy *University of Milan, Italy*
borghese@dsi.unimi.it *{ferrari,piuri}@dti.unimi.it*

Abstract

A procedure for 3*-D surface reconstruction from range data in strict real-time is presented. The process is based on a connectionist model, the Hierarchical Radial Basis Functions (HRBF) network, which has been proved effective in reconstruction of smooth, space varying surfaces. An efficient data structure and locality assumptions allowed to derive a faster configuration algorithm, without degrading the approximation capabilities. Results show that computing time scales linearly with the number of neurons, and it is comparable with data acquisition time of most of commercial scanners so that reconstruction can actuall be performed in real time.*

1. Introduction

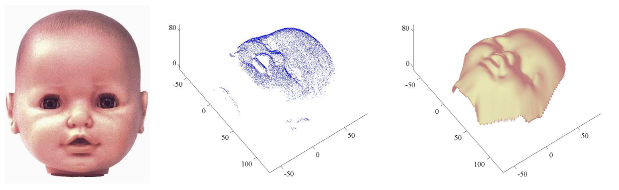
3-D scanners are a powerful and common tool in the digitization of real objects. They usually implement a two-step process: geometry sampling of the surface and meshing. Although most of the systems require several minutes or hours to get a 3-D model, few attempts towards real-time operation are under development [1, 2]. The procedure presented in this paper is aimed in producing 3-D surfaces in strict real-time. We consider only reconstruction of $\mathbb{R}^2 \to \mathbb{R}$ manifolds, also named 2 1/2-D surfaces; however, the process can be generalized to full 3-D manifolds.

The problem can be formalized as follows. Given a data set $S = \{(x_i,\, y_i,\, z_i) \mid 1 \leq i \leq N\}$, find an approximation, $\tilde{s}(\cdot)$, such that

$$||\tilde{s}(x, y) - z||_{(x,y,z)\in S} \leq \epsilon_{\text{th}}, \tag{1}$$

where ϵ_{th} is related to noise, and a suitable measure of the distance between sets is given.

To solve this kind of problems, neural networks, and Radial Basis Functions (RBF) networks [3] among these, have shown to be a suitable tool. Thanks to the quasi-locality



(a) Real object (b) Geometry sampling (c) Surface reconstruction

Figure 1. 3-D acquisition and reconstruction of a real object.

of RBF neurons (usually Gaussians) and their simple network topology, efficient algorithms to configure the network have been developed. Among the RBF models, Hierarchical RBF networks (HRBF) have proven high effectiveness [4, 5] in reconstructing smooth, space varying surfaces. HRBF networks are a self-organizing model composed by a hierarhical set of RBF subnetworks, called *layers*, which provides a multi-scale reconstruction. This is realized by means of a rough approximation $a_1(\cdot)$ and a pool of hierarchical details at decreasing scale: $a_2(\cdot), \ldots, a_M(\cdot)$; the surface approximation at the k-th scale, $\tilde{s}_k(\cdot)$, is given by $a_1(\cdot)$ plus the sum of the first $k-1$ details: $\tilde{s}_k(\cdot) = a_1(\cdot) + \sum_{i=1}^{k} a_i(\cdot)$. The larger is k, the more detailed is the reconstruction.

Configuration of HRBF networks is based on a constructive approach: each layer is created after the other by incorporating details at smaller scales, until (1) is satified. The quasi-locality property of the Gaussian neurons and their regular distribution are exploited in this paper to make the configuration procedure compatible with real-time, without degrading approximation capabilities.

In section 2, the HRBF configuration algorithm is described. Section 3 presents the assumptions, common to pratical situations, which allow to derive a faster configuration procedure. In section 4, a comparison between the original and the fast configuration procedure is reported, while in section 5 the results are discussed. Conclusions follow in

```
A. Initialization: ${r_0(p_j) = z_j}$ (residual = height of sampled points)
   For each layer, $l$
B. Configuration:
      For each neuron, $i$, positioned in $c_i, l$
   B.1. retrieval of the points which belong to the receptive field of the neuron, $S_i, l$
   B.2. if the residual is larger than the measurement error, insert the neuron in $c_{i,l}$
   B.3. estimate of the surface height in the neuron center, $\overline{s}_{i,l}$ and set to that value the weight $w_{i,l}$
C. compute the output of the layer, $\{a_l(p_j)\}$
D. compute the residual, $\{r_l(p_j) = r_{l-1}(p_j) - a_l(p_j)\}$
```

Figure 2. HRBF configuration algorithm.

section 6.

2. The HRBF configuration algorithm

A HRBF network is composed by a hierarchical set of Gaussian RBF networks, $\{a_l\}$, called *layers*, each of which acts at a different level of detail. The neurons of each layer, l, are distribuited on a regular grid, which has side Δx_l, and share the same scale parameter, σ_l, which determines the level of detail of that layer. The collective behaviour of a grid can be regarded as a low-pass filter [4] and can be formalized as:

$$s(p) = \sum_{l=1}^{L} a_l(p) = \sum_{l=1}^{L} \sum_{i=1}^{M_l} w_{i,l}\, g(||p - c_{i,l}||;\, \sigma_l), \tag{2}$$

where L is the number of layers, and M_l is the number of neurons of the l-th layer. The output of each layer, $a_l(\cdot)$, which is constituted of a RBF network, is a linear combination of the output of the Gaussian neurons of that layer:

$$a_l(p) = \sum_{i=1}^{M_l} w_{i,l}\, g(||p - c_{i,l}||;\, \sigma_l), \quad p \in \mathbb{R}^2 \tag{3}$$

Each coefficient $w_{i,l}$ weights the contribution of the i-th Gaussian belonging to the layer l. It can be shown [4] that its value is equivalent to the surface height in the point $c_{i,l}$. Since such a value is usually not available, it is estimated, $\overline{s}_{i,l}$, in the configuration procedure as a weighted mean of the points, which belong to an appropriate neighborhood of $c_{i,l}$, $S_{i,l}$, called *receptive field*. The HRBF configuration procedure is schematized in fig. 2.

Once all the weights of one layer have been computed, the output, $a_l(\cdot)$, of that layer can be evaluated through (3). The surface height, $\{z_i\}$, is the input to the first layer. The *residual* for the layer l, r_l, is defined on the sampled points, $\{p_i = (x_i,\, y_i)\}$, as the difference between the data to be reconstructed and the output of the first $l-1$ layers:

$$r_l(p_i) = z_i - \sum_{1}^{l-1} a(p_i) \tag{4}$$

and it . The residual is the input for the higher layers.

The residual is used also to decide if a neuron has to be used: the neuron $g_{i,l}$ is inserted only if the residual r_{l-1} in $S_{i,l}$ is larger than the measurement error, ϵ_{th}. The computation of the output a_l in the data points for the residual requires to evaluate the distance $||p_j - c_{i,l}||$ between each input data point $\{p_j \in \mathbb{R}^2\}$ and the position of each neuron, $\{c_{i,l} \in \mathbb{R}^2\}$, of the layer (eq. (2)). Similar considerations apply also to the computation of the coefficients: determination of the points belonging to a receptive field, $S_{i,l}$, requires the evaluation of the distance of all points from the neuron center.

The complexity of the configuration algorithm is therefore $O(M\,N)$, where $M = \sum_{l=1}^{L} M_l$ is the number of the neurons of the network, and $N = |S|$ is the data set cardinality.

In the following, the regular distribution of the neurons of a HRBF network will be exploited to obtain a fast retrieval of the data points, and reduce the complexity.

3. Fast HRBF configuration algorithm

The need to compute the distance between all data points and all Gaussian centers is required by:

- the non-zero response of the Gaussian in the whole domain, $\mathbb{R}^2$ (infinite support);
- the extraction of the points which lie inside the influence region and receptive field (steps B.1 and C in fig. 2).

For practical applications, the support can be made finite: the rapid decrease of the Gaussian towards zero makes its response soon negligible. Hence, a suitable neighborhood, $Q_{i,l}$, of the neuron center can be defined. This neighborhood is called *influence region*. The computation of the output of each neuron is confined in it. In the following, 3σ will be taken as radius of $Q_{i,l}$, where $g(3\sigma)$ is $1.24 \cdot 10^{-4}$ times the value of the value of the Gaussian in its center.

To avoid the need of computing, for each neuron, the distance from all the data points, the key idea is to arrange the data in a data structure that allows fast retrieval of those points, which belong to the neighborhood of a neuron. This is realized by partitioning the input domain such that the points inside the receptive field of every neuron can be extracted without having to examine all data points.

Easy partitioning cannot be obtained with the circular regions implicitly considered in the definition of $S_{i,l}$ and $Q_{i,l}$; some approximations must therefore be introduced. In particular:

1. The receptive field of the neurons, theoretically circular, will be approximated by the bounding square. From now on, we therefore simply refer to this approximation as the "receptive field".

2. The length of the side of the receptive field is an even multiple of Δx, the grid spacing. More formally, the receptive field side is equal to $2\rho\Delta x$, where $\rho \in \mathbb{Z}$.

3. The Gaussian size (and, hence, the grid side, Δx_l) is halved at every layer: $\sigma_{l+1} = \sigma_l/2$.

The effects of these approximations will be discussed in section 5. In the following sections, the partitioning method and its implementation will be described in detail.

3.1. Partitioning scheme

Let $c_{i,j,l}$ the center of the neuron in the (i,j)-th grid crossing of the l-th layer. It is:

$$\begin{cases} c_{i+1,j,l} - c_{i,j,l} = \Delta x_l \\ c_{i,j+1,l} - c_{i,j,l} = \Delta x_l \end{cases}$$

Without loosing generality, we assume that $c_{0,0,l} \equiv (0,0)$. For each layer, l, a partition of the input space in regular regions $R_{i,j,l}$, called *cells* can be obtained as:

$$R_{i,j,l} = \{(x,y) \mid i\Delta x_l \le x \le (i+1)\Delta x_l, \\ j\Delta x_l \le y \le (j+1)\Delta x_l\}$$

This partition scheme enjoys two properties:

receptive field coverage The receptive field of the i,j,l neuron, $S_{i,j,l}$ is:

$$S_{i,j,l} = \bigcup_{\substack{i-\rho \le h \le i+\rho-1 \\ j-\rho \le k \le j+\rho-1}} R_{h,k,l} \tag{5}$$

where ρ is an appropriate integer value as stated in hypotesis 2 in section 3.

recursion The cells of a lower layer, $\{R_{i,j,l}\}$, are related to those of the immediate higher layer as:

$$R_{i,j,l} = \bigcup_{\substack{2i \le h \le 2i+1 \\ 2j \le k \le 2j+1}} R_{h,k,l+1} \tag{6}$$

The first property defines the receptive field with a single union operation. This property is important for the weight computation. The second property is particularly important for the multi-scale computation. This allows to define the cells of a higher layer efficiently partitioning the cells of the lower layer: four cells of the higher layer constitute a single cell of the immediate lower layer.

Similar considerations apply to the influence regions. In the following only the receptive field will be dealt with, since the operations for the influence region are similar.

3.2. Implementation

The fast HRBF configuration procedure differs from the original one mainly in data pre-processing, which creates a data structure able to represent the partitioning, and is performed just after data collection has been completed before network configuration.

We suppose that the points are stored in an array. Aim of pre-processing is position points, which belong to the same cell, $R_{i,j,l}$, adjacent in the array. This allows to represent each cell by a pointer, $q_{i,j,l}$, and a counter, $n_{i,j,l}$. The pointer, $q_{i,j,l}$, represents the first point in the array belonging to the cell $R_{i,j,l}$, while the counter, $n_{i,j,l}$, counts the number of points which belong to that cell. The partition is represented as a 2D static matrix structure, H, where each cell corresponds to a matrix entry; thus, each matrix entry, $H_{i,j,l} = (q_{i,j,l}, n_{i,j,l})$, stores the pointer and the counter for the corresponding cell.

All the points belonging to a cell, $R_{i,j,l}$, can be easily extracted from $H_{i,j,l}$: these are the $n_{i,j,l}$ points inside the data points array starting from $q_{i,j,l}$. Since the region $S_{i,j,l}$ is obtained as the union of cells (eq. (5)), the points belonging to $S_{i,j,l}$ can be easily collected by means of successive accesses to the cells constituting $S_{i,j,l}$. This can be accomplished without having to compute any distance measure.

The partition for the second layer and the successive ones can be obtained by subdividing the partition of the previous layer (eq. (6)), i.e., by halving each cell along each direction. This suggest to perform the reorganization of the data points array in two steps. In the first step the subarray of the cell points is rearranged such that the points which has the x-coordinate belonging to the first half of the cell come before the other points. In this way, two rectangular subcells are formed. In the second step, a y-coordinates wise rearrangement is applied to each subcells. The rearrangement of the points inside each cell is performed as an in-place partial sorting through a variant of the Quicksort algorithm [6], in which the pivot value is the mean value of the vertices of the cell.

The partitioning procedure is illustrated in fig. 3.

The input data set, S, is used to configure the first layer, while the successive ones are configured through the resid-

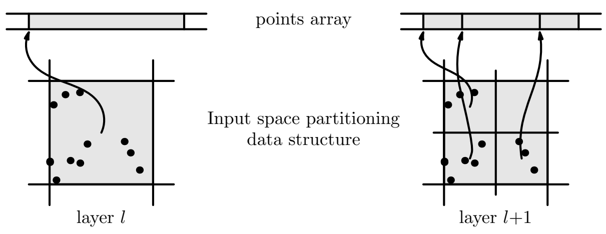


Figure 3. Partitioning schema implementation.

ual. However, the same array can be used to store the data used to configure all the layers. This can be achieved by storing the value of the residual, which is coincident to the data points in the first layer, in the array. This allows to save memory, and to preserve consistency of the partitioning data structure.

4. Results

In order to evaluate the impact of the contraints provided by the assumptions 1-3 in section 3 onto the HRBF configuration task, we applied the procedure described above to the surface reconstruction from noisy data and we compared the results with original HRBF configuration algorithm.

The data set was composed by 16,851 points with an estimated measurement error of 0.5 mm. A four layers HRBF network was used, with an initial value of $\sigma = 16$ mm. For the fast version, the receptive field side was set equal to $4\Delta x$ ($\rho = 2$), while the influence region side was set to $8\Delta x$ long. For the original algorithm, the receptive field radius was $2\Delta x$.

The accuracy perfomance is evaluated through the standard deviation of the reconstruction error, computed as: $\epsilon_{\text{std}} = \text{std}(\{z_j - a_l(p_j)\})$; this reaches asymptotically the measurement error.

The speed perfomance is measured by the time required to compute the HRBF parameters. In the fast configuration algorithm, this comprises the pre-processing and the computation of the weight time.

Table 1. The configuration performace of the original HRBF.

layer	Computing time [s]		grid size	used neurons	ϵ_{std}
	mean	std			
1	1.79	0.187	14×15	175	5.05
2	7.48	1.24	27×29	635	2.45
3	26.4	3.61	53×57	2133	1.23
4	56.0	10.9	105×113	4962	0.765

Table 2. The configuration performance of the fast HRBF.

layer	Preproc. time [ms]		Training time [ms]		Computing time [ms]	
	mean	std	mean	std	mean	std
1	71.0	25.9	480	54.5	551	55.5
2	4.28	14.7	478	41.8	482	41.1
3	4.66	15.3	449	42.0	454	44.1
4	6.64	18.0	290	46.1	297	48.0

layer	grid size	used neurons	ϵ_{std}
1	14×15	177	5.19
2	27×29	635	2.53
3	53×57	2171	1.25
4	105×113	5104	0.779

Computing time of the original algorithm was measured on a base of 20 trials, while 30,000 trials were run for the fast HRBF algorithm. Mean and standard deviation values of these measurements are reported in tabs. 1 and 2.

The experiments were run on a 1 GHz Pentium III PC with a 256 MB memory. The total time spent in configuration for original HRBF is of 91.7 s (with a standard deviation of 15.5 s), while for the fast version is of 1.78 s (with a standard deviation of 0.118 s).

In fig. 4 the absolute value of the difference between the surfaces reconstructed with the original and the fast HRBF configuration algorithm is reported. In regions covered by data points, the difference is almost negligible. In fig. 1c the reconstruction obtained through the fast configuration algorithm is reported. It is qualitatively indistinguishable from the reconstruction obtained through the original algorithm, which is not reported.

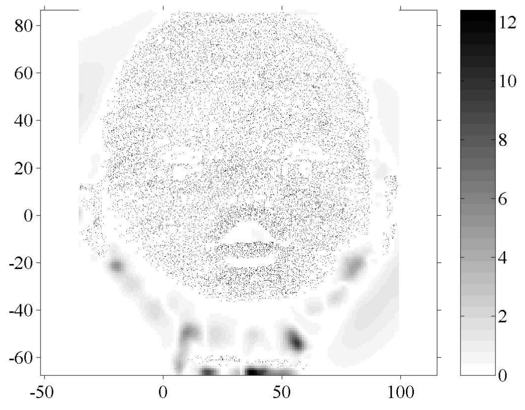


Figure 4. Differences between the two reconstructions.

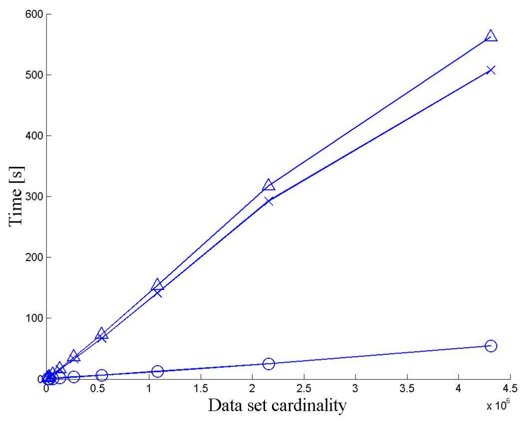


Figure 5. Time spent vs. the data set size: "$\bigcirc$" pre-processing phase, "$\times$" Coefficients computation, "$\triangle$" total computation time.

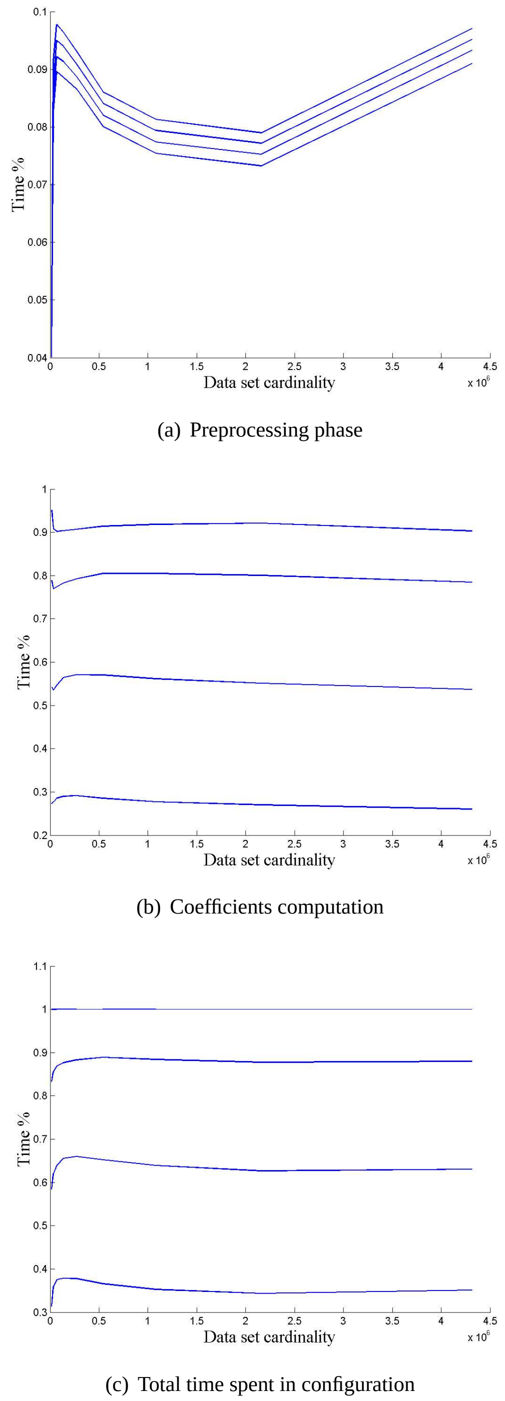
In order to evaluate the impact of the size of the data set on the fast configuration algorithm performances, we made several trials with nine data sets of different size, N_k, $k = 1, \ldots, 9$, where $N_1 = 16,851$, and $N_k = 2^{k-1} N_1$. The number of neurons was the same in all experiments, as those reported in tab. 1. The computing time with respect to the data set size is reported in fig. 5 and the percentage of computing time spent in each configuration procedure phase is given in fig. 6 .

5. Discussion

The key element of the fast HRBF configuration algorithm is data pre-processing. This allows to place the data in the input array so that the points inside each cell can be directly addressed without any sorting procedure. The computing time spent by the fast configuration algorithm is only 2% of the original one, as shown in tab. 2. Moreover, the time spent in surface reconstruction with respect to the number of points (fig. 5) compares well with the time necessary by most of the commercial scanners (e.g., [2]) to collect a data set of the same cardinality.

As shown in Fig. 6, the overhead introduced by the pre-processing is negligible with respect to the total computational time (less than 10%), especially with very large data sets.

Pre-processing for the first layer is the most demanding since the partitions have to be created from scratch. Since the sorting procedure swaps the data points in the array, this phase costs $O(K/2N)$, where K is the number of cells of the partition of the first layer. The construction of the partitions for the subsequent layers is based on a pre-existing data structure of the partition: since each point is analyzed



(a) Preprocessing phase

(b) Coefficients computation

(c) Total time spent in configuration

Figure 6. Time percentage of the total computation time spent vs. the data set size.

only twice, the cost of this partial sorting is $O(N)$.

The partitioning algorithm was implemented by using a static matrix to represent the data of each cell. This implies a memory overhead, which scales as the number of cells and, thus, as the number of HRBF neurons: it is, hence, $O(M)$. No further memory space is required, thanks to the in-place sorting strategy (Section 3.2). A reduction in memory overhead can be achieved using a quad-tree data structure [7]. Quad-tree uses a dynamic hierarchical subdivision of space which allows to save memory as empty cells does not give rise to further subdivision. However, the navigation through the tree structure requires more than one memory access to reach the cell. In practice, this may result in an increase of cache-misses, because the nodes along the path to reach a cell may not be present in the cache at the same time. This slows down the whole procedure.

The time reduction due to fast HRBF algorithm does not affect the accuracy. Errors that might be introduced by the implementation choices (e.g., quantization error [8], the approximations introduced in section 3) do not decrese the reconstruction accuracy, due to the constructive nature of the configuration algorithm. The standard deviation of the reconstruction error for the fast and traditional configuration algorithms are in fact almost identical as shown in tabs. 1 and 2.

Computational complexity for evaluating the weight of a neuron is proportional to the number of points in the neighborhood of that neuron. While passing from one layer to the next higher one, the number of neurons increases four times, the number of points inside each cell decreases of the same factor. Moreover, the number of unused neurons, increases with the number of layers, since the residual tends to be smaller than the measurement error. Thus, the time for computing the weights of each layer decreases as the reconstruction proceedes.

The results reported in this paper are valid in the 2D manifold (D = 2). In fact, we approximated the 2-dimensional sphere of the influence region, with a 2-dimensional cube. This approximation becomes less acceptable with increasing the dimensionality, D, since the volume of a D-dimensional sphere becomes negligible with respect to the volume of its bounding D-dimensional cube for increasing values of D. This configuration algorithm, therefore, cannot be extended to manifolds of higher dimensionality without decreasing the performance both in time and accuracy.

6. Conclusions

HRBF networks have been proved to be effective in reconstructing surfaces from noisy data. However, the infinite support of their neurons limits their use in practical cases. In this paper, the HRBF configuration algorithm has been revised: few assumptions allow to introduce an efficient data structure and to decrease the computational time dramatically. Moreover, since the configuration of each neuron is highly independent from the configuration of the other neurons, the algorithm is suitable for parallel real-time hardware implementation (e.g., on FPGA-based boards).

References

[1] M. Petrov, A. Talapov, T. Robertson, A. Lebedev, A. Zhilyaev, and L. Polonskiy, “Optical 3D digitizers: Bringing life to the virtual world.”, *IEEE Computer Graphics & Applications*, vol. 18, no. 3, pp. 28–37, May–June 1998.

[2] Szymon Rusinkiewicz, Olaf Hall-Holt, and Marc Levoy, “Real-time 3d model acquisition”, in *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. 2002, pp. 438–446, ACM Press.

[3] Tomaso Poggio and Federico Girosi, “Network for approximation and learning”, *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1481–1497, September 1990.

[4] Nunzio Alberto Borghese and Stefano Ferrari, “Hierarchical RBF networks in function approximation”, *Neurocomputing*, vol. 19, pp. 259–283, 1998.

[5] Nunzio Alberto Borghese and Stefano Ferrari, “A portable modular system for automatic acquisition of 3D objects”, *IEEE Transactions on Instrumentation and Measurement*, vol. 49, no. 5, pp. 1128–1136, October 2000.

[6] C. A. R. Hoare, “Algorithms 64: Quicksort”, *Communication of the ACM*, vol. 4, no. 7, pp. 321, 1961.

[7] Hanan Samet, “The quadtree and related data structures”, *ACM Computing Surveys, June 1984*, vol. 16, no. 2, pp. 188–260, 1984.

[8] S. Ferrari, N. A. Borghese, and V. Piuri, “Multiscale models for data processing: an experimental sensitivity analysis”, *IEEE Trans. on Instr. and Meas.*, vol. 50, no. 4, pp. 995–1002, August 2001.