Proceedings of International Joint Conference on Neural Networks, Orlando, Florida, USA, August 12-17, 2007

Online training of Hierarchical RBF

Francesco Bellocchio, Stefano Ferrari, Vincenzo Piuri and N. Alberto Borghese

Abstract— **An online procedure for configuring the parameters of a Hierarchical Radial Basis Functions (HRBF) network is presented here. The proposed procedure has been implemented and applied to a problem of real-time surface reconstruction. Results show that the algorithm trained online well compares with the batch version.**

I. Introduction

Online learning is a widely diffused learning modality in neural networks [1][2], used in two different domains. The first, is non stationary problems, where the statistical distribution of the input data changes with time [3][4]. The second domain is real-time learning [5]. In this case, online learning can be used to perform a reconstruction of the data mani-fold, while data points are being sampled on the mani-fold itself. This second domain, is less common in the scientific domain; nevertheless it has interesting applications. For instance, the real-time reconstruction of a 3D model the artifact acquired with a 3D digitizer [6] would be of great help to drive the sampling procedure to collect more data points, where the details are missing [7]. Up to now, methods based on splatting [8] have been mainly used to this scope. Splatting displays an elliptical shape centered in the data points and oriented as the local estimated normal of the surface. If the cloud of data points is sufficiently dense, it may provide the perception of a continuous surface, without its mathematical description.

Different approaches have been proposed in the connectionist domain to solve this problem [9], [10].

We propose here to solve it, through an online version of the Hierarchical Radial Basis Functions network (HRBF) model [11]. This can produce in real-time an accurate local multi-scale reconstruction of the surface. This online version can be used in all the domains of low dimensionality, where real-time manifolds approximation is required.

In Section II the batch version of the HRBF training procedure is reported, while the proposed online version is reported in Section III. The algorithm has been implemented and challenged in real-time surface reconstruction problem. Results are reported in Section IV and discussed in Section V.

II. The HRBF Model

Let us assume that the manifold can be described as a $\mathbb{R}^D \rightarrow \mathbb{R}$ function. In this case, the input dataset can be viewed as a height field: $\{(P_i, z_i) \,|\, z_i = S(P_i),\ P_i \in \mathbb{R}^D,\ 1 \leq i \leq N\}$, and the manifold assumes the analytical shape: $z = S(P)$. The output of a HRBF network is obtained by adding the output of a pool of Radial Basis Functions (RBF) networks, organized as a stack of hierarchical layers, each of which is characterized by a decreasing scale:

$$S(P) = \sum_{l=1}^{L} a_l(P; \sigma_l) \tag{1}$$

where σ_l determines the scale of the l-th layer, with $\sigma_l > \sigma_l + 1$. If we suppose that the units are equally spaced on a grid support and a normalized spherical Gaussian function, $G(\cdot;\, \sigma) = \frac{1}{\sqrt{\pi \sigma^2}^D} exp\left(-\frac{||\cdot||^2}{\sigma^2}\right)$, is taken as basis function, the output of each layer can be written as a linear low-pass filter:

$$a_l(P; \sigma_l) = \sum_{k=1}^{M_l} w_{l,k} G(||P - P_{l,k}||; \sigma_l) \tag{2}$$

where M_l is the number of Gaussian units of the l-th layer. The $G(\cdot)$ are equally spaced on a D-dimensional grid, which covers the input domain of the data points: that is the $\{P_{l,k}\}$s are positioned in the grid crossings of the l-th layer. The side of the grid is a function of the scale of that layer: the smaller the scale, the shorter is the side length, the denser are the Gaussians and the finer are the details which can be reconstructed.

The actual shape of the surface in (1) depends on a set of parameters: the *structural parameters*, which are the number, $M = \sum_l M_l$, the scale ensemble, $\{\sigma_l\}$, and the position, $\{P_{l,k}\}$; and the weights associated to each Gaussian: $\{w_{l,k}\}$. Each RBF grid, l, realizes a reconstruction of the surface up to a certain scale, determined by σ_l (low-pass filtered reconstruction). Considerations grounded on the signal processing theory allow, given a certain scale, σ_l, to set the grid side, ΔP_l, as $\sigma_l = 1.465\, \Delta P_l$ and to determine consequently M and the $\{P_{l,k}\}$ [12]. From these observations, the weights $\{w_{l,k}\}$ are set equal to the manifold height in the grid crossings: $w_{l,k} = S(P_{l,k}) \cdot \Delta P_l^D$. As the data set usually does not include the $\{S(P_{l,k})\}$, these values should be estimated. We explicitly observe that, even if the $S(P_{l,k})$ were included, they would be corrupted by noise and an estimate would be the right solution. The data points that lie in an appropriate neighborhood of $P_{l,k}$ can be used to estimate $S(P_{l,k})$ as a weighted average of such subset of data points, $\tilde{S}(P_{l,k})$. This neighborhood, called *receptive field*, $A(P_{l,k})$, can be chosen as a spherical region centered in $P_{l,k}$ with the radius proportional to the grid side, ΔP_l. A possible weighting

Francesco Bellocchio and N. Alberto Borghese are with the Department of Computer Science, University of Milano, Italy (email: borghese@dsi.unimi.it).

Stefano Ferrari and Vincenzo Piuri are with the Department of Information Technologies, University of Milano, Italy (email: {ferrari, piuri}@dti.unimi.it).

function is:

$$\tilde{S}(P_{l,k}) = \frac{\sum_{P_m \in A(P_{l,k})} S(P_m)\, e^{-\frac{||P_{l,k}-P_m||^2}{\sigma_l^2}}}{\sum_{P_m \in A(P_{l,k})} e^{-\frac{||P_{l,k}-P_m||^2}{\sigma_l^2}}} \quad (3)$$

which is strongly related to the Nadaraya-Watson estimator and maximizes the conditional probability density when the noise is normally distributed, zero mean [13] [14].

Although a single layer with Gaussians of very small scale could reconstruct the finest details, this would produce an unnecessary dense packing of units in all those regions which feature large scale details. Moreover, there might even be not enough points inside $A(P_{l,k})$ to get a reliable estimate of $\tilde{S}(P_{l,k})$ in (3). A better solution is to adaptively allocate the Gaussian units, with an adequate scale in the different regions of the range data domain. This can be achieved by adding and configuring one layer at time, proceeding from the layer featuring the largest scale to the layer featuring the smallest one. For sake of simplicity in the configuration stage, each new layer will feature half the scale of the previous one. However, arbitrary scales could be used for the different layers.

All the layers after the first one will be trained to approximate the residual, that is the difference between the original data and the actual output of the network output by the already configured layers. Hence, the residual, r_l, is computed as:

$$r_l(P_m) = r_{l-1}(P_m) - a_l(P_m) \quad (4)$$

and it is used for estimating the parameters of the l-th layer. $r_0(P_m) = z_m$ is also assumed.

The Gaussians of a new layer are inserted only where a poor approximation is obtained from the previous layers. This is evaluated, for each Gaussian, $P_{l,k}$, through an integral measure of the residuals inside the receptive field of that Gaussian, $A(P_{l,k})$. This measure, which represents the *local residual error*, $R(P_{l,k})$, is computed as the L_1 norm of the local residual as:

$$R(P_{l,k}) = \frac{\sum_{P_m \in A(P_{l,k})} |r_{l-1}(P_m)|}{|A(P_{l,k})|}. \quad (5)$$

As the Gaussian function has an infinite support, the computation of the output of each layer, a_l may be computational very expensive. However, as the Gaussian decreases very fast to zero with the distance from its center, computational time can be saved by allowing each Gaussian to contribute to the computation of the residuals only for those points that belong to an appropriate neighborhood of the Gaussian center. This neighborhood has been called *Influence Region*.

When $R(P_{l,k})$ is over a given threshold, ϵ, the Gaussian is inserted in the corresponding grid crossing of the current layer under construction. As a result, Gaussians at a smaller scales are inserted only in those regions where there are still some missing details, forming a sparse approximation of the data. The introduction of new layers ends when the residual error is under threshold over the entire domain (uniform approximation).

This approach has been compared with classical multiresolution analysis through wavelet basis, and it has proved superior when approximation of noisy data is required [15].

This batch HRBF training procedure exploits the knowledge of the entire input dataset, and adopts local estimates to setup the network parameters. The only preliminary information required is the position and dimension of the data bounding box, which can be computed automatically from the data set. The user may specify the scale of the first layer, otherwise this can be determined by some heuristics on the data bounding box or the data set density.

The learning is based on the following configuration rules:

- given the scale parameter of the first layer and the bounding box, the grids associated to the higher layers can be defined and the Gaussians, in turns, placed at the grid crossings. Hence the number of Gaussians for each layer, M_l, is also defined;
- if the grid scale is divided by two for all the higher layers, their value is completely specified starting from σ_1;
- the weight of each Gaussian is estimated through a local weighted average of the input data for the first layer, and through a local weighted average of the residuals for the next layers (3).

Hence, knowing the entire dataset allows to correctly position the Gaussians and to estimate the weights of the first layer, and of the subsequent ones, with a fast configuration which can be parallelized, but that has to wait that all the data points are available.

III. Online training procedure

When the data set is not entirely known, but grows with time, the schema described in Section II cannot be applied.

In order to illustrate this statement, let us assume that a HRBF has been already configured with a given data set and that a new point, sampled over the manifold, is given. In this case, the local estimate of the manifold height (as in (3)) is not valid anymore and it should be carried out again considering also this new point. This operation has to be carried out for all those Gaussians of the first layer whose Receptive Field contains the new point. This, in turns, modifies the output of the first layer. As a consequence, the residual for the points that belong to the Influence Region of the updated Gaussians, changes, and the weights of those Gaussians of the second layer, whose Receptive Field has a non-empty intersection with the updated region, have to be estimated again. This causes a chain-reaction that, at the end, may involve an important subset of the units of the HRBF network. The need of a new layer can also emerge.

The computational power can be easily not sufficient to sustain the updating of the network parameters, and some approximations have to be accepted to obtain real-time configuration.

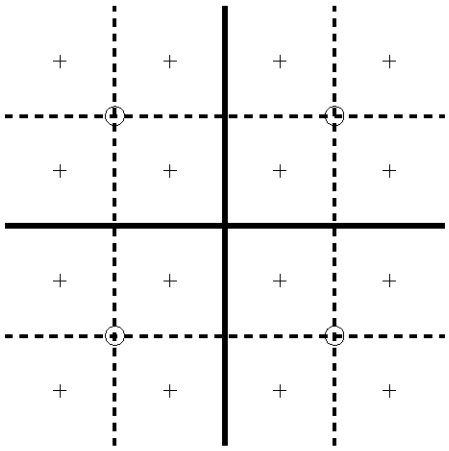


Fig. 1. Partitioning schema of the input space: the close neighborhood of each Gaussian (centered in 'o') is partitioned by the close neighborhood of four Gaussians of the next layer (centered in '+').

The algorithm proposed here is based on operating the network parameters update every Q points (with $Q << N$). In the first phase, Q points are collected and used to update a few quantities associated to those Gaussians whose receptive field includes the Q collected points. In the second phase, the residual error, (5), is computed and new Gaussians are inserted in the network accordingly. The two phases are iterated as far as data points are sampled.

A. Data structures

For each layer, l, input space is partitioned into squares, $\{C_{l,k}\}$, each centered in a different Gaussian center, $P_{l,k}$. As the points, which belong to $\{C_{l,k}\}$ will be closer to $P_{l,k}$ than to any other Gaussian of the l layer, we call $\{C_{l,k}\}$ the *close neighborhood* of the Gaussian l, k. It is clear that, for each layer, l, the set $\{C_{l,k}\}$ constitutes a partition of the input space.

A data structure, containing the data used for training, is associated to each Gaussian, k, of the l-th layer. This structure contains: the 3D coordinates of the Gaussian's center, $P_{l,k}$, its scale parameter, $\sigma_{l,k}$, its weight, $w_{l,k}$, the numerator, $n_{l,k}$ and the denominator, $d_{l,k}$, of (3) and the 3D coordinates of the points which fall inside its *close neighborhood*. As the distance between two adjacent Gaussians of the same layer, ΔP_l, is half of that of the previous layer, ΔP_{l-1}, the close neighborhood of each Gaussian of the l-th layer will be formed by the close neighborhood of four Gaussians of the $l+1$-th layer. This relationship, depicted in Fig. 1, is used to organize in a quad-tree the Gaussians data structure: the data of each Gaussian of the l-th layer, called *father*, points to the data of the four Gaussians of the $l+1$-th layer (its children). Hence, from each Gaussian, (l, k), it is easy to access the data of any other Gaussian whose center belongs to $C_{l,k}$.

B. First phase: parameters adaptation

When a new point, P^*, is acquired, the Gaussian of the first layer, whose receptive field includes the new point, $G^*_{1,k}$, is selected to be updated. For each Gaussian, the numerator, $n_{1,k}$, and the denominator, $d_{1,k}$, are modified according to (3). This procedure is iterated on the next layers, considering for the updating only those Gaussians that are children of $G^*_{1,k}$. Hence, the distance between P^* and the Gaussians' center is computed only for a small subset of the network units. After updating the parameters, the coordinates of the new point are inserted in the data structure of that Gaussian of the higher layer, (l, k), whose close neighborhood contains P^*: $P^* \in C_{l,k}$.

C. Second phase: splitting

After parameters have been reasonably settled, the network will not be able to improve the quality of the reconstruction more as this is limited by the scale of the units inserted in the actual higher layer. Therefore after Q points have been collected, a splitting step is required. The reconstructed manifold is examined in correspondence of the receptive fields of those Gaussians which satisfy three criteria: they have no children; at least a given number of sampled points, K, have been collected inside their close neighbourhood, and their close neighborhood includes at least one of the last Q points processed.

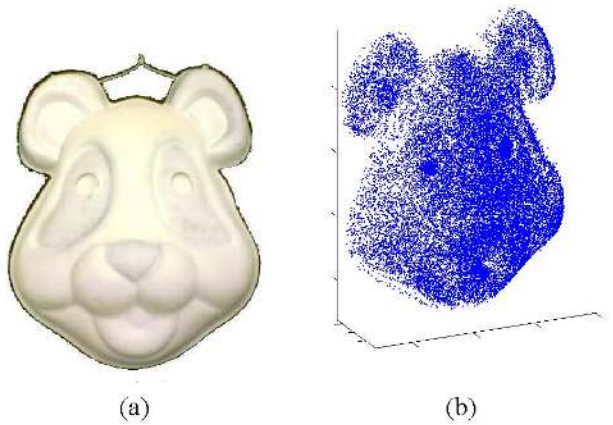
For all the points which fall inside the close neighbourhood of these Gaussians, the network output is computed again for all the layers and, from this, the local residual error associated to these Gaussians (5) is derived and it is compared with the error threshold. We build the residual starting from the Gaussians at the lowest layer, moving towards the higher layers. We explicitly remark that thanks to the quad-tree data structure adopted, the Gaussians considered in each higher layer can be directly accessed as they are children of the corresponding Gaussian of the lower layer.

If the local residual error exceeds the error threshold for the Gaussian (l, k), four Gaussians, featuring a scale parameter $\sigma_{l+1} = \sigma_l/2$, are inserted in the next layer, $l+1$. The points which fall inside $C_{l,k}$ are removed from $C_{l,k}$, sorted and partitioned into the data structure of these four new Gaussians, such that each new Gaussian contains all the data points which belong to its close neighbourhood, $C_{l+1,k}$. For each of these new four Gaussians, $n_{l+1,k}$, and $d_{l+1,k}$ are computed as well.

D. Initialization

The only *a priori* information needed is the bounding box position and side of the input space to be sampled. This information is used to set the parameters of the first layer, which is composed by only one Gaussian. The scale parameter of the first layer, σ_1 will be proportional to the maximum side length of the bounding box, B, namely $sigma_1 = 1.465\,B$, and the center of the Gaussian, $P_{1,1}$, will be positioned at the center of the bounding box: the close neighborhood of the Gaussian of the first layer, $C_{1,1}$, is defined as the square centered in $P_{1,1}$, having side length equal to that of the bounding box.

The parameters of the online training algorithm are the threshold error, ϵ, the number of points of the network



(a) (b)

Fig. 2. The data set used to test the online training procedure (a). The range data has been acquired by the 3D scanner from the mask reported in (b).

TABLE I

PERFORMANCE INDEXES AND PARAMETERS OF EACH LAYER OF THE FINAL HRBF NETWORK

#layer	σ [mm]	#Gauss.	#eff. Gauss.	RMSE [mm]	ϵ_{mean} [mm]	ϵ_{std} [mm]
1	403	1	1	45.5	14.0	43.3
2	201	4	4	28.3	13.0	25.2
3	101	16	12	15.3	8.67	12.7
4	50.4	64	32	7.09	4.30	5.64
5	25.2	256	112	4.09	2.70	3.07
6	12.6	1024	405	2.60	1.94	1.73
7	6.29	4096	1342	1.78	1.37	1.13
8	3.15	16384	2410	1.45	1.20	0.810

adaptation phase, Q, and the minimum number of points for splitting, K. These may depend on the application, as they are related to the noise of the input data and to the computational power of the processing system.

IV. RESULTS

We applied the neural model described in section III to real-time surface reconstruction from data output by automatic digitizers.

In particular, the data set reported in Fig. 2b, are considered here. It is constituted of a total 32,000 3D points acquired by sampling one point after the other over the surface of the object represented in Fig. 2a using the 3D scanner described in [7]. The configuration parameters were set as follows: $Q = 100$, $K = 9$, $\epsilon = 0.8$. ϵ was set equal to the digitizer accuracy.

Figure 3 reports the surface reconstructed at different steps of the acquisition process: Fig. 3a features an early reconstruction (after 1,000 points have been sampled), Fig. 3b-c present the surface in two intermediate steps (after 5,000 and 10,000 points, respectively), and in Fig. 3d the final surface is reported.

We have first investigated the error produced by the network and the number of units employed, and compared it with the batch HRBF version. A total of eight layers are created, with a scale of the final layer of 3.17 mm

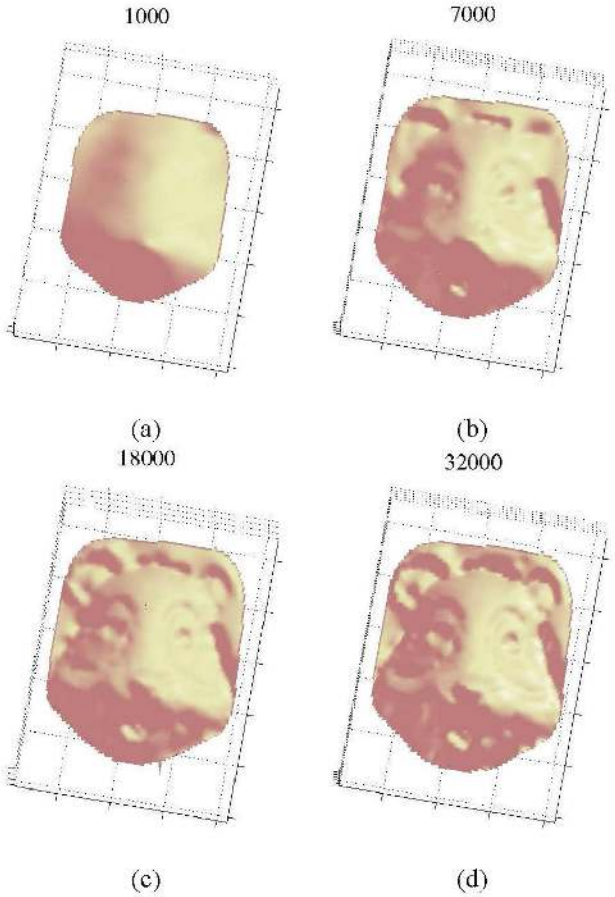


Fig. 3. Surface reconstruction as the acquisition proceeds. Subfigures (a), (b), (c), and (d) show the reconstruction after 1000, 7000, 18000, and 32000 points, respectively.

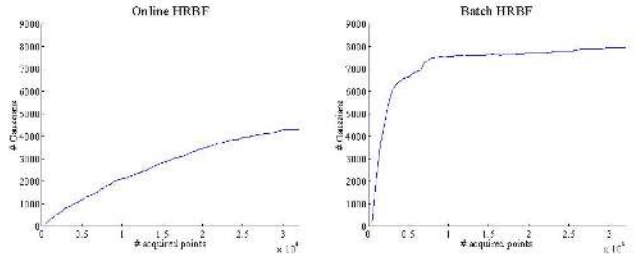


Fig. 4. The number of allocated Gaussians with respect to the number of acquired points for the final online HRBF (a), and the batch HRBF (b).

TABLE II

PERFORMANCE INDEXES AND PARAMETERS OF EACH LAYER OF THE BATCH HRBF NETWORK

#layer	σ [mm]	#Gauss.	#eff. Gauss.	RMSE [mm]	ϵ_{mean} [mm]	ϵ_{std} [mm]
1	403	4	4	25.3	21.6	13.9
2	201	9	13	14.5	12.1	13.3
3	101	25	34	11.7	9.56	11.7
4	50.4	81	86	9.16	7.47	9.14
5	25.2	289	257	5.24	4.02	5.24
6	12.6	1089	847	2.68	1.97	2.68
7	6.29	4225	2922	1.52	1.05	1.52
8	3.15	16641	8400	0.919	0.636	0.919

(Table I). In order to compare the online and the batch approaches, the evaluation has been carried out creating a new batch HRBF from scratch every 500 points. As it can be seen, only a subset of the maximum number of Gaussians is considered: for instance only 1,342 over 4,096, and 2,410 over 16,384, and 12 over 65,53 are considered for the last two layers. This produce a sparse approximation. Comparing these figures with those in batch HRBF trained using the while dataset (Table II), it is evident that the online HRBF version adopts a more conservative Gaussian allocating strategy. This consideration is evident in Figs. 4a and b, where the number of Gaussian units allocated in the network with respect to the number of the acquired data points is shown for the online and batch versions. As it can be seen, in the online version the number of units increases linearly in the first stages to saturate around 25,000 points. In the batch version, instead the number of units tends to allocate more units and to saturate around 7,500 points.

The accuracy has been evaluated through the RMS error, the mean of the absolute error, ϵ_{mean}, and its standard deviation, ϵ_{std}. The figures from the online version (Table I) and the batch version (Table II) are quite dissimilar, but the batch HRBF uses nearly the double of the online HRBF. For sake of comparison, we pruned the batch HRBFs in order to obtaining a network with the same number of units of the online HRBF. The pruning operation has been carried out by removing the less significant units, i.e., the ones with the smallest coefficients. The error evolution as a function of the number of sampled points is shown in Figs. 5a, b, and c.

V. Discussion

The key elements in updating the network parameters are $n_{l,k}$ and $d_{l,k}$. These are initialized in the splitting phase, when a new Gaussian and its data structure is created, and they are updated every time a new sampled point falls inside the Gaussian's, (l, k), receptive field, $A(P_{l,k})$.

During the parameters adaptation phase, it should be remarked that, for sake of speed, the residual of a point is computed only when it is sampled. For this reason the residual on the previously acquired point can be biased and therefore it can bias the local residual error of the associated Gaussian. Therefore the associated weight can be biased along with the local surface height estimate.

However, due to the non-orthogonality of their basis functions, the HRBF network is able to recover from a poor estimate of the surface in one layer, with the approximation produced by the next layer.

Figure 5a shows that the reconstruction error decreases when the network grows in the first stages of learning to saturate when the model details have been captured, towards the end of the acquisition session. This can be used also as a stopping criterion for the acquisition process itself.

From Fig.s 5a and b it can be noticed that the reconstruction error achieved by the online learning algorithm is greater than the one achieved by the batch HRBF. This is due to the different Gaussian positioning and the different units allocation policy, as the online HRBF can create

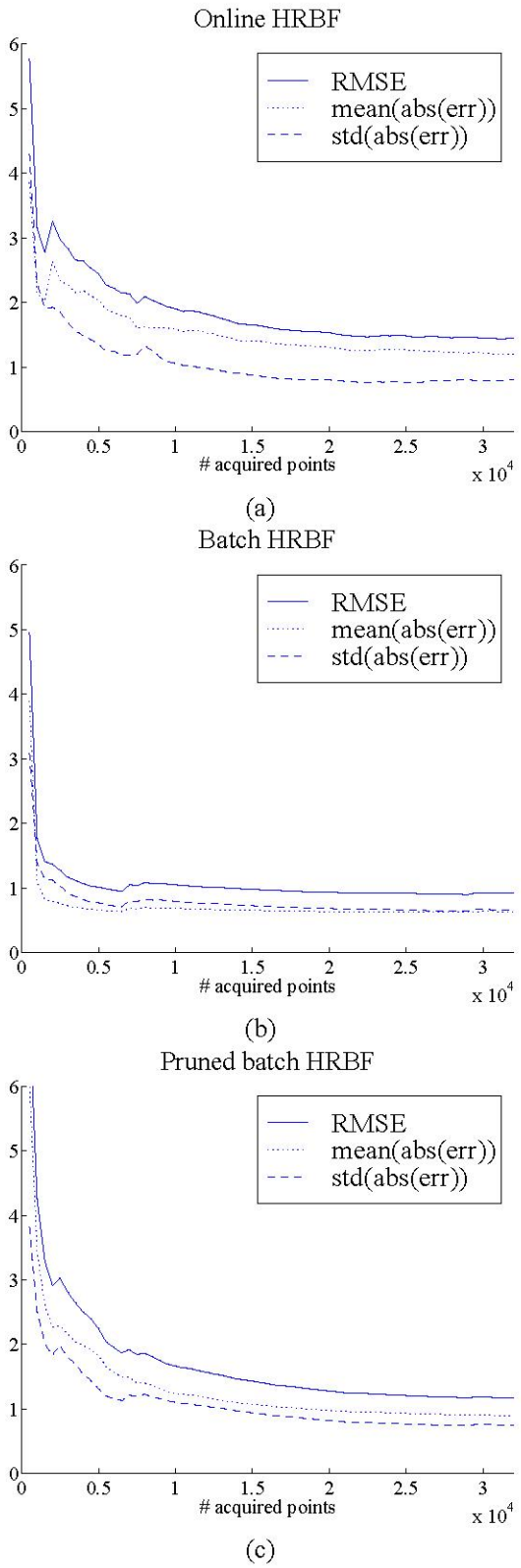


Fig. 5. Performance indexes with respect to the number of processed points. The error figures (RMSE, mean and standard deviation of the absolute error) are reported for the final online HRBF (a), the batch HRBF (b), and the pruned batch HRBF.

new Gaussians only in the region explored during the last parameter adaptation phase (III-C). However, the comparison with the pruned batch HRBF (Figs 5a and c) makes evident that the units allocated by the online HRBF are the most significant ones.

VI. Conclusion

An online training procedure for the HRBF model is presented here and applied to a real-time surface reconstruction problem. The online HRBF model is being extensively used to reconstruct artefacts' surfaces. Different parameters set are under investigation to define optimal behavior. From the preliminary data set obtained, the performance of the online procedure results comparable with the performance achieved by the batch version.

References

[1] D. Saad, Ed., *On-Line Learning in Neural Networks*, Cambridge University Press, 1998.

[2] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[3] A. Rubaai, R. Kotaruand, and M.D. Kankam, "Online training of parallel neural network estimators for control of induction motors," *IEEE Trans. on Industry Applications*, vol. 37, no. 5, pp. 1512–1521, sep-oct 2001.

[4] Jung-Wook Park, G. K. Venayagamoorthy, and R.G. Harley, "Mlp/rbf neural-networks-based online global model identification of synchronous generator," *IEEE Transactions on Industrial Electronics*, vol. 52, no. 6, pp. 1685–1695, Dec. 2005.

[5] Jun Fan, N. Dimitrova, and V. Philomin, "Online face recognition system for videos based on modified probabilistic neural networks," in *Proceedings of the 2004 International Conference on Image Processing, 2004. ICIP '04*, Oct. 2004, vol. 3, pp. 2019–2022.

[6] S. Rusinkiewicz, O. Hall-Holt, and M. 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.

[7] N. A. Borghese and S. Ferrari, "A portable modular system for automatic acquisition of 3-D objects," *IEEE Trans. on Instr. and Meas.*, vol. 49, no. 5, pp. 1128–1136, October 2000.

[8] Szymon Rusinkiewicz and Marc Levoy, "Qsplat: A multiresolution point rendering system for large meshes," *Proceedings of SIGGRAPH 2000*, pp. 343–352, July 2000, ISBN 1-58113-208-5.

[9] B. Fritzke, "Growing cell structures — a self-organizing network for unsupervised and supervised learning," *Neural Networks*, vol. 7, no. 9, pp. 1441–1460, 1994.

[10] "Wen-Chang Chen", ""neural-network-based photometric stereo for 3d surface reconstruction"," in *Proc. of IJCNN 2006*, July 2006, pp. 404–410.

[11] S. Ferrari, I. Frosio, V. Piuri, and N. A. Borghese, "Automatic multiscale meshing through HRBF networks," *IEEE Trans. on Instr. and Meas.*, vol. 54, no. 4, pp. 1463–1470, Aug. 2005.

[12] N. A. Borghese and S. Ferrari, "Hierarchical RBF networks and local parameter estimate," *Neurocomputing*, vol. 19, no. 1–3, pp. 259–283, 1998.

[13] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural Computation*, vol. 7, no. 2, pp. 219–269, 1995.

[14] C. M. Bishop, *Pattern Recognition and Machine Learning*, Information Science and Statistics. Springer, 2006.

[15] S. Ferrari, M. Maggioni, and N. A. Borghese, "Multi-scale approximation with hierarchical radial basis functions networks," *IEEE Trans. on Neural Networks*, vol. 15, no. 1, pp. 178–188, Jan. 2004.