Online training of Hierarchical RBF

F. Bellocchio[1], S. Ferrari[2], V. Piuri[2], and N.A. Borghese[1]

[1] Department of Computer Science, University of Milano, Italy
borghese@dsi.unimi.it
[2] Department of Information Technologies, University of Milano, Italy
{ferrari,piuri}@dti.unimi.it

Abstract. Efficient multi-scale manifold reconstruction from point clouds can be obtained through the Hierarchical Radial Basis Functions (HRBF) network. An online training procedure for HRBF is here presented and applied to real-time surface reconstruction during a 3D scanning session. Results show that the online version compares well with the batch one.

1 Introduction

Online learning is a widely diffused learning modality in neural networks [1][2]. It is adopted in non stationary problems, where the statistical distribution of the input data changes over time [3], and for real-time learning [4]. In this case, online learning can be used to perform a reconstruction of the data manifold, while data points are being sampled. This second domain, although less common, has interesting applications. For instance, the real-time reconstruction of the surface of an artifact, while it is being 3D scanned [5], would be of great help to drive the sampling procedure where the details are missing [6]. Up to now, methods based on splatting [7] have been mainly used to the scope. These methods display an elliptical shape centered in the data points, whose color intensity changes with the estimated normals of the surface, providing the perception of a continuous surface without giving its analytical description.

We propose here to reconstruct the 3D surface, as the output of Hierarchical Radial Basis Functions (HRBF) network [8], introducing a new on-line training procedure, which can produce in real-time multi-scale reconstruction.

In Section 2 the batch version of the HRBF training procedure is reported, while the proposed online version is reported in Section 3. The algorithm has been implemented and challenged in real-time surface reconstruction problem. Results are reported in Section 4 and discussed in Section 5.

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

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

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, that is the $\{P_{l,k}\}$s are positioned in the grid crossings of the l-th layer. The side of the grid, ΔP_l, is a function of σ_l: the smaller is σ_l, the shorter is ΔP_l, 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 number, $M = \sum_l M_l$, the scale ensemble, $\{\sigma_l\}$, the position, $\{P_{l,k}\}$, and the weights of the Gaussians, $\{w_{l,k}\}$. Each RBF grid, l, realizes a reconstruction of the surface up to a certain scale, determined by σ_l. Signal processing theory allows to set ΔP_l as $\sigma_l = 1.465\,\Delta P_l$ and to determine consequently M and the $\{P_{l,k}\}$ [8]. If only the l-th layer would be used, from the analogy between (1) and linear filtering theory, the weights $\{w_{l,k}\}$ can be as: $w_{l,k} = S(P_{l,k}) \cdot \Delta P_l^D$ [8]. As the data set usually does not include the $\{S(P_{l,k})\}$ (or they could be corrupted by noise), these values should be estimated.

A weighted average of the data points that lie in a neigborhood of $P_{l,k}$ can be used to estimate $S(P_{l,k})$. This neighborhood, called *receptive field*, $A(P_{l,k})$, can be chosen as a spherical region, with the radius proportional to ΔP_l. A possible weighting function, which is related to the Nadaraya-Watson estimator and maximizes the conditional probability density when the noise is normally distributed, zero mean [9], is:

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

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 flat regions and an unreliable estimate of $\tilde{S}(P_{l,k})$ if too few points fall in $A(P_{l,k})$. A better solution is to adaptively allocate the Gaussian units, with an adequate scale in the different regions of the domain by adding and configuring one layer at time, starting from the largest scale one. Each new layer will feature half the scale of the previous one.

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 produced by the already configured layers. Hence, the residual, r_l, is computed as:

$$r_l(P_m) = r_{l-1}(P_m) - a_l(P_m) \tag{3}$$

and it is used for estimating the weights of the l-th layer by substituting $S(P_m)$ in (2). $r_0(P_m) = z_m$ is also assumed.

The quality of the approximation may be evaluated, for each Gaussian, $P_{l,k}$, through an integral measure of the residuals inside $A(P_{l,k})$. This measure, which represents the *local residual error*, $R(P_{l,k})$, is computed as the L_1 norm of the local residual as:

$$R(P_{l,k}) = \frac{1}{|A(P_{l,k})|} \sum_{P_m \in A(P_{l,k})} |r_{l-1}(P_m)|. \tag{4}$$

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

As the Gaussian function decreases very fast to zero with the distance from its center, computational time can be saved by allowing each Gaussian to contribute to the residuals only for those points that belong to an appropriate neighborhood of the Gaussian center, $P_{l,k}$, called *influence region*, $I(P_{l,k})$.

This batch HRBF training procedure exploits the knowledge of the entire input dataset, and adopts local estimates to setup the network parameters with a fast configuration which can be parallelized, but that has to wait that all the data points are available.

3 Online Training Procedure

When the data set is not entirely known, but grows one point at a time, the schema described in Section 2 cannot be applied. In fact, let us assume that a HRBF has been already configured with a given data set, S_{old} and that a new point, P_{new} is sampled over the manifold.

In this case, the estimate in (2) becomes out of date for all the units (1, k) such that $P_{\text{new}} \in A(P_{1,k})$, and has to be estimated with the new data set, $S_{\text{old}} \cup P_{\text{new}}$. This modifies a_l inside the influence region of the updated units. As a consequence, the residual for the points that belong to this region changes, making out of date the weights for all those Gaussians of the second layer whose *receptive field* has a non-empty intersection with this region. This causes a chain-reaction that, at the end, may involve an important subset of the units of the HRBF network. Moreover, the need for a new layer can also occur.

If the computational power cannot be sufficient to sustain the updating of the network weights for every new input data, some approximations have to be accepted to obtain real-time configuration.

The algorithm proposed here is based on updating the network parameters every Q points (with $Q << N$). In the first phase, a few internal variables, associated to those Gaussians whose *receptive field* includes the collected points, are computed and the network weights are updated. In the second phase, the residual error, (4), is computed, and new Gaussians are inserted where it is over threshold. The two phases are iterated as far as data points are sampled.

3.1 Data Structures

For every layer, l, input space is partitioned into squares, $\{C_{l,k}\}$, each centered in a different Gaussian center, $P_{l,k}$. The points, which belong to $C_{l,k}$ will be closer to $P_{l,k}$ than to any other Gaussian of the l-th layer. We therefore call $C_{l,k}$ the *close neighborhood* of the (l, k)-th Gaussian.

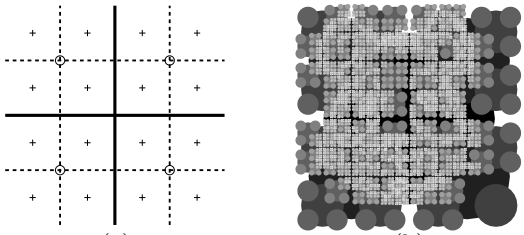

Fig. 1. (a) The *close neighborhood* of each Gaussian ('o') is partitioned into four *close neighborhoods* of Gaussians of the next layer ('+'). (b) During the training, the partitioning is iterated adaptively. For sake of clarity, the *close neighborhood* of the Gaussians is depicted as a circle, having a diameter slightly smaller than the *close neighborhood* side. The circles changes their color from black to white depending on the layer (lower to higher) they belong to.

At the (l, k)-th Gaussian is associated a data structure containing: $P_{l,k}$, $\sigma_{l,k}$, the points inside $C_{l,k}$, its weight, $w_{l,k}$, the numerator, $n_{l,k}$, and the denominator, $d_{l,k}$, of (2). As $\Delta P_l = \frac{1}{2}\Delta P_{l-1}$, the *close neigborhood* of each Gaussian (*father*) of the l-th layer will be formed by the *close neigborhood* of the four corresponding Gaussians (*children*) of the $(l+1)$-th layer. This relationship, depicted in Fig. 1a, is used to organize in a quadtree the Gaussians data: from each *father* Gaussian, it is easy to access to every *children* of the higher layers.

3.2 First Phase: Parameters Updating

When a new point, P_{new}, is acquired, the parameters $n_{1,k}$, $d_{1,k}$, and, hence, $w_{1,k}$ of the Gaussians of the first layer such that $P_{\text{new}} \in A(P_{1,k})$ are updated:

$$n_{1,k} := n_{1,k} + S(P_{\text{new}}) \cdot e^{-||P_{1,k} - P_{\text{new}}||^2/\sigma_1^2} \tag{5}$$

$$d_{1,k} := d_{1,k} + e^{-||P_{1,k} - P_{\text{new}}||^2/\sigma_1^2} \tag{6}$$

This procedure is iterated on the higher layers, considering for updating only the *children* of the updated Gaussians. According to (3), in the higher layers, $r_l(P_{\text{new}})$ is used instead of $S(P_{\text{new}})$. The computation of the r_l's is limited to P_{new}: the contribution of the old points to $n_{l,k}$ and $d_{l,k}$ is not modified.

Overall, the processing of P_{new} produces a modification of the network output in those regions covered by Gaussians whose *receptive field* contains P_{new}. However, only a small subset of the network units is involved, for each P_{new}.

After updating the weights, P_{new} is inserted into the data structure associated to the Gaussian such that $P_{\text{new}} \in C_{l,k}$.

3.3 Second phase: Splitting

When Q points are collected, the reconstructed manifold is examined in correspondence of the *receptive fields* of those Gaussians which satisfies three conditions: they have no *children*, at least a given number of sampled points, K, was sampled inside its *close neighborhood*, and whose *close neighborhood* includes at least one of the last Q points processed. For each of these Gaussians, the residual error is computed inside their *receptive field* and compared with the error threshold, ϵ, as in (4). As for the updating phase, for each layer, the Gaussians that contribute to the output computation are selected among the *children* of the units considered in the previous layer. If the reconstruction is not satisfactory for the Gaussian (l, k), four Gaussians, are inserted in the next layer, $l+1$, and the points collected by the Gaussian (l, k) are distributed between the new Gaussians. The parameters of the new Gaussians ($n_{l,k}$, $d_{l,k}$, and $w_{l,k}$) are then set using the points contained in their *close neighborhood*.

Not all the four *children* Gaussians always have data points inside their *close neighborhood*. The weight of the Gaussians with an empty *close neighborhood* is set to zero and they do not contribute to the output of the network.

3.4 Initialization

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

The parameters of the online training algorithm, ϵ, Q, and K, may depend from the application, as they should be set proportional to the noise of the input data and to the computational power of the processing system.

4 Results

We applied the neural model described in Section 3 to 3D scanning, where the artifact's surface in Fig. 2a is reconstructed in real-time. In particular the Autoscan system [5] is used here, which allows to sample for a longer time in those regions where more details are concentrated. The acquired 3D input dataset is reported in Fig. 2b: it is composed of 16,000 3D points. Acquisition was stopped when the visual quality of the reconstructed surface (Fig. 2c-f) was judged sufficient and no significant improvement in the model quality was observed adding new points (Fig.2e and Fig.2f). Alternatively the procedure could be stopped when no splitting occurs. In this example, $\epsilon = 0.8$, $Q = 100$, and $K = 9$.

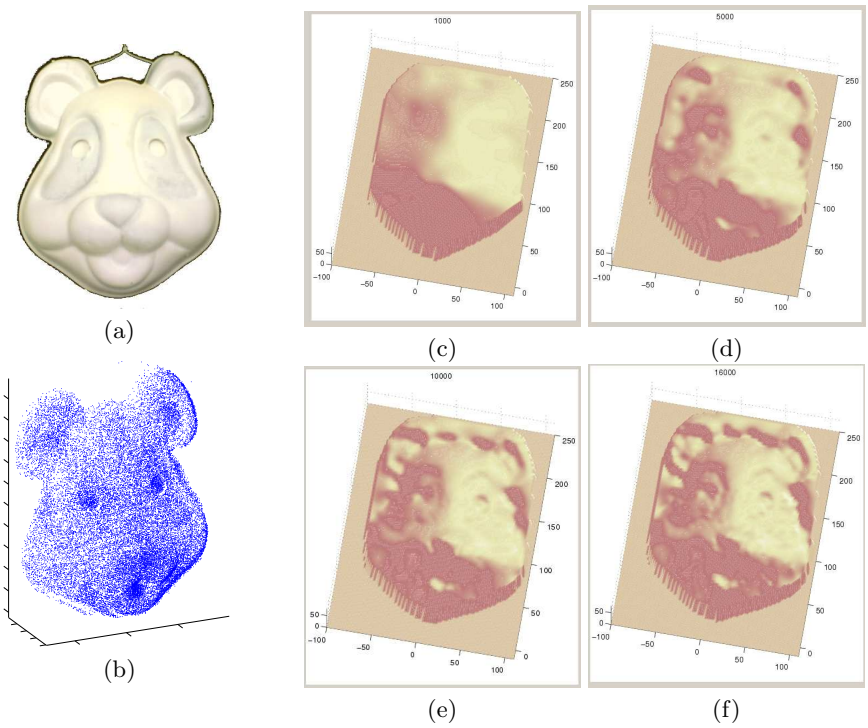

Fig. 2. A typical data set used to test the online training procedure. From the mask reported in (a), 16,000 points, which constitute the input data set (b), are sampled by the Autoscan system [5]. Notice the higher point density in the mouth and eyes regions. Surface reconstruction as the acquisition proceeds. Panels (c), (d), (e), and (f) show the reconstruction after 1000, 5000, 10000, and 16000 points have been sampled.

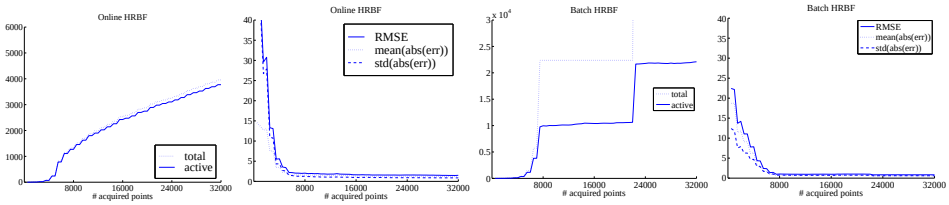

Fig. 3. Performance indexes of the batch HRBF model with respect to the number of processed points. The total number of Gaussians created and the number of those that contribute to the reconstruction (i.e., those with an associated non-zero weight) for online, (a), and batch, (c), HRBF. The error figures: RMSE, mean and standard deviation of the absolute error for the online, (b), and batch, (d), HRBF.

As can be seen in Fig. 3a the number of units increases with the number of data points, improving the quality of the reconstructed model. However, comparing the number of allocated Gaussians (Table 1), with the maximum number of Gaussians for each layer, the reconstruction is obtained with a very sparse Gaussians set (cf. Fig. 2b) as only $3,774$ units over $87,381$ are used. Only a small portion of the Gaussians created are not used in the reconstruction. The reconstruction error decreases rapidly at the beginning, and continues to decrease down to 1.22 mm, which is close to the measurement error with the adopted setup, in accordance with the good visual quality of the reconstructed model.

Table 1. Performance Indexes and Parameters of Each Layer of the final HRBF Network

		online						batch		
l	σ_l	max #Gauss.	#Gauss.	#eff. Gauss.	RMSE [mm]	ϵ_{mean} [mm]	ϵ_{std} [mm]	RMSE [mm]	ϵ_{mean} [mm]	ϵ_{std} [mm]
1	330	1	1	1	46.0	15.1	43.4	30.1	26.3	14.9
2	165	4	4	4	30.7	14.0	27.3	16.0	13.5	14.0
3	82.4	16	16	16	13.4	7.83	10.8	11.4	9.37	11.4
4	41.2	64	64	50	5.87	3.71	4.54	7.79	6.15	7.79
5	20.6	256	204	181	3.33	2.21	2.49	4.36	3.28	4.36
6	10.3	1024	672	632	2.23	1.63	1.53	2.51	1.82	2.51
7	5.15	4096	1744	1674	1.59	1.26	0.970	1.47	1.02	1.47
8	2.58	16384	1240	1206	1.49	1.22	0.868	0.980	0.672	0.977
9	1.29	65536	12	10	1.49	1.22	0.868	0.830	0.581	0.825

Table 1 reports the number of units of the final networks and the residuals of each layer, and compares these data with those obtained from the batch version of a HRBF network applied to the same dataset of Fig. 2. To the scope, every $Q = 1000$ data points, a new batch HRBF network is created from scratch. We constrained the maximum number of layers and the scale parameters of the batch HRBF network to be equal to the corresponding parameters of the online version. In the batch HRBF the value of K is set to 3: due to the complete knowledge of the input dataset, few points are considered sufficient to estimate the surface height. The final error of the two HRBF models was quite comparable, 1.22 ± 0.868 mm versus 0.581 ± 0.83 mm. However, as expected, the batch version converged faster to a smaller error than the on-line version, as shown in Fig 3. Moreover, the batch network grows faster.

5 Discussion

The manifold height in $P_{l,k}$ (2) is estimated as the ratio between $n_{l,k}$ and $d_{l,k}$, which is obtained as a run-time sum of the values derived from each sampled point. However, this value is equal to that in the batch model only for the first layer as the residual is subject to changes as the points are sampled, and the

contribution of the older points in the residual estimation may become unreliable as the training proceeds. However, due to the non-orthogonality of their basis functions, the HRBF model is able to compensate the error in one layer, with the approximation achieved by the next layer [10].

The choice of a suitable value for K may be critical as it can lead to an early splitting: if K is too small, the weight estimate of the new Gaussians may be unreliable because the contribution of the first points will not be changed. This effect could be mitigated by using an aging strategy for the estimated n and d. On the other hand, the more the points are processed, the more the weights of the lower layers become reliable. Hence, a late splitting involves a more accurated reconstruction. However, using a too large value of K, the Gaussians are inserted slowly, and the reconstruction details cannot became apparent.

Depending on the rate the example are available and by the computational power of the system, idle time may exploited for re-processing old points. This activity does not increase the input data points density and do not contribute to splitting, but should compensate the bad estimate operated in the early stages.

6 Conclusion

An online training procedure for the HRBF model is here presented, illustrated by an application to real-time surface reconstruction problem. The accuracy of the online procedure results comparable with that of the batch versiont. This online version can be used in all the domains of low dimensionality, where real-time manifolds approximation is required.

References

1. Saad, D., ed.: On-Line Learning in Neural Networks. Cambr. Univ. Press (1998)
2. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford Univ. Press (1995)
3. Rubaai, A., Kotaruand, R., Kankam, M.: Online training of parallel neural network estimators for control of induction motors. IEEE Trans. on Ind. Appl. **37**(5) (2001) 1512–1521
4. Fan, J., Dimitrova, N., Philomin, V.: Online face recognition system for videos based on modified probabilistic neural networks. In: Proc. of the 2004 Int. Conf. on Image Processing, 2004. ICIP '04. Volume 3. (2004) 2019–2022
5. Borghese, N.A., Ferrigno, G., Baroni, G., Ferrari, S., Savaré, R., Pedotti, A.: Autoscan: a flexible and portable 3D scanner. IEEE C. G. & A. **18**(3) (1998) 38–41
6. Rusinkiewicz, S., Hall-Holt, O., Levoy, M.: Real-time 3d model acquisition. In: Proc. of the 29th conf. on Comp. graph. and int. tech., ACM Press (2002) 438–446
7. Rusinkiewicz, S., Levoy, M.: Qsplat: A multiresolution point rendering system for large meshes. Proc. of SIGGRAPH 2000 (2000) 343–352 ISBN 1-58113-208-5.
8. Ferrari, S., Frosio, I., Piuri, V., Borghese, N.A.: Automatic multiscale meshing through HRBF networks. IEEE Trans. on I. & M. **54**(4) (2005) 1463–1470
9. Girosi, F., Jones, M., Poggio, T.: Regularization theory and neural networks architectures. Neural Computation **7**(2) (1995) 219–269
10. Ferrari, S., Borghese, N.A., Piuri, V.: Multiscale models for data processing: an experimental sensitivity analysis. IEEE Trans. on I. & M. **50**(4) (2001) 995–1002