A Hierarchical RBF Online Learning Algorithm for Real-Time 3-D Scanner

Stefano Ferrari, *Member, IEEE*, Francesco Bellocchio, Vincenzo Piuri, *Fellow, IEEE*, and N. Alberto Borghese, *Member, IEEE*

***Abstract*—In this paper, a novel real-time online network model is presented. It is derived from the hierarchical radial basis function (HRBF) model and it grows by automatically adding units at smaller scales, where the surface details are located, while data points are being collected. Real-time operation is achieved by exploiting the quasi-local nature of the Gaussian units: through the definition of a quad-tree structure to support their receptive field local network reconfiguration can be obtained. The model has been applied to 3-D scanning, where an updated real-time display of the manifold to the operator is fundamental to drive the acquisition procedure itself. Quantitative results are reported, which show that the accuracy achieved is comparable to that of two batch approaches: batch HRBF and support vector machines (SVMs). However, these two approaches are not suitable to real-time online learning. Moreover, proof of convergence is also given.**

***Index Terms*—Multiscale manifold approximation, online learning, radial basis function (RBF) networks, real-time parameters estimate, 3-D scanner.**

I. Introduction

ONLINE learning is a widely diffused neural networks learning modality [1]–[3]. It is applied to nonstationary problems, where the statistical distribution of the analyzed data changes over time [4], [5], and to real-time learning [6], where a manifold is constructed and adapted, while data points are being sampled from it.

This second domain, although less common, has interesting applications in all the problems in which the availability of an updated manifold is required to effectively drive the acquisition process itself. For instance, in active 3-D scanning where a laser stripe or spot is projected over an artifact to sample data points over its surface [7], a real-time display of the current reconstructed surface would allow driving the laser toward the areas where the details are still missing in the reconstructed surface [8], [9]. This allows large improvement in the effectiveness of the scanning procedure. In fact, up to now, feedback is provided only by methods based on splatting [10]. These methods display for each data point an elliptical shape centered in the point, whose gray level depends on the estimated normal to the surface. If the cloud of data points is sufficiently dense, it may provide the perception of a continuous surface, but it does not provide any analytical 3-D description of the surface that could be further processed. We introduce here online learning as a powerful tool to obtain the current manifold while data are being collected.

Different approaches have been developed in the connectionist domain to achieve such a goal, most of them have been derived from analogous methods developed for batch learning. Gradient-descent methods are the most studied algorithms [11]–[14]. They were introduced in the 1960s [11] for linear networks and they have then been successively extended to more complex neural networks models, like multilayer perceptrons [12] and radial basis function (RBF) networks [13], [14]. More recently, approaches based on extended Kalman filter (EFK) [15]–[17] have been introduced to speed up learning.

Although, theoretically, the same universal approximation properties have been shown for both batch and online learning [18], gradient-descent-like methods get often stuck in local minima in real applications. For this reason, hybrid approaches to learning have been developed in both batch and online learning domains [3], [13], [19], [20], that are particularly suitable to RBF networks, constituted of linear combinations of quasi-local units, Gaussians in particular.

Platt was the first to propose a growing network model named resources-allocating network (RAN) [21]. In this model, Gaussian units, at decreasing scales, are inserted as training proceeds. For each new point, an additional Gaussian unit is inserted only if both the local reconstruction error, measured in the point, is over threshold and there are no units close enough to that point. If these two conditions are not met, the parameters of the unit closest to the input point are updated through gradient descent. As training proceeds, both the neighborhood size and the width of new units shrink; as a result, the units inserted at the beginning feature a large width, covering most of the input domain, while the units inserted at the end feature a smaller scale, reconstructing the details.

When the unit width shrinks too quickly the model may easily produce a bad reconstruction due to poor coverage of the input domain caused by the lack of units with sufficiently large width. On the contrary, when it shrinks too slowly, many units are inserted, which may require an unacceptable learning time and may produce overfitting.

To partially solve these problems, Fritzke introduced an RBF network model called growing cell structures [22]. Here each unit stores additional information: a neighborhood list to identify the closest units, and an accumulator to store the reconstruction error inside the region covered by the unit. The neighbor-

Manuscript received September 17, 2008; revised August 15, 2009; accepted October 29, 2009. First published December 11, 2009; current version published February 05, 2010.

S. Ferrari, F. Bellocchio, and V. Piuri are with the Department of Information Technology, Università degli Studi di Milano, Crema 26013, Italy (e-mail stefano.ferrari@unimi.it; francesco.bellocchio@unimi.it; vincenzo.piuri@unimi.it).

N. A. Borghese is with the Department of Computer Science, Università degli Studi di Milano, Milano 26013, Italy (e-mail: alberto.borghese@unimi.it).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2009.2036438

hood list is used to adapt the units' width in order to maintain a given overlapping rate between neighbor units, while the accumulator guides the insertion of new units in those regions where the local reconstruction error is higher. For each point presented to the network, the position of the unit closest to the point is updated along with that of the units that lie inside such unit's neighborhood. The width of each unit is also implicitly updated as the average distance from its neighbors while the weights are updated through the delta rule. Moreover, the value of the accumulator associated to the unit closest to the input point is increased. The need to insert a new unit is evaluated periodically, after a fixed number of points has been examined. The new unit's parameters are set as the mean value of the parameters of all its neighbor units.

In order to reduce the total number of units, several pruning techniques, aimed at discarding the units that less contribute to the reconstruction, have been proposed [15], [17], [22]–[24]. They are generally based on an aging rule [3], [22] to select the less used units.

To simplify the learning procedure, grids of Gaussians with the same width, equally spaced, have been introduced [3], [20], [25], [26]. This allows to take full advantage both of the quasi-locality of the Gaussian and of linear filtering theory, to design efficient learning algorithms that can work locally on the data and can adapt the scale locally to the data. Such algorithms, for instance, do not require pruning or maintaining complex data structures.

Grid arrangement can be exploited to achieve real-time approximation and hence is valuable for online applications. The main contribution of this paper consists in a new training procedure applied to the hierarchical radial basis function (HRBF) network model [20], which allows obtaining a multiscale, online, real-time, reconstruction of a manifold [27]. The procedure has been implemented and compared to the batch version of HRBF and to support vector machines (SVMs) [28].

In Section II, the batch version of the HRBF training procedure is summarized. Its online version is introduced in Section III. Results on real data are reported in Section IV and compared with batch HRBF and SVM. They are discussed in Section V and closing remarks are reported in Section VI.

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 data set can be given as: $\{(P_i, z_i) | z_i = S(P_i),\ P_i \in \mathbb{R}^D,\ 1 \leq i \leq N\}$, and the manifold assumes the explicit analytical shape: $z = S(P)$.

The HRBF model is an RBF network where the units of the hidden layer are partitioned in L sets, each of them characterized by a scale parameter σ_l. These sets will be considered sequentially at configuration time, with $\sigma_l > \sigma_{l+1}$, while their output $a_l(\cdot; \sigma_l)$ is added together to provide the overall network output, as follows:

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

Each $a_l(\cdot)$ is indeed a hidden layer of an RBF network, and the HRBF can be considered as a pool of RBF subnetworks operating in parallel, each at a different scale. In the following, we will refer to the lth RBF subnetwork as the lth layer.

If the units are equally spaced on a grid and a normalized Gaussian function $G(\cdot; \sigma) = (\sqrt{\pi\sigma^2})^{-D} \exp(-\|\cdot\|^2/\sigma^2)$ is taken as the basis function, the output of each layer can be written as a linear low-pass filter [25], [29]

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

where M_l is the number of Gaussian units of the lth layer.

The actual output of each RBF network layer $a_l(\cdot)$ depends on the number of the Gaussian units in the lth layer M_l, their position $\{P_{l,k} | P_{l,k} \in \mathbb{R}^D\}$, and their variance $\{\sigma_l | \sigma_l \in \mathbb{R}\}$ that constitute the *structural parameters* of the network. The value of $a_l(\cdot)$ depends also on the $\{w_{l,k} | w_{l,k} \in \mathbb{R}\}$, which are termed *synaptic weights*.

Considering only a single layer, the function $a_l(\cdot)$ realizes a reconstruction of the surface up to a certain scale, determined by σ_l. In this case, signal processing theory allows setting the Gaussian spacing (grid size, ΔP_l) according to σ_l, $\Delta P_l = \sigma_l / 1.465$ [29]: 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. Gridding allows also to automatically set M and the position of the Gaussians, $\{P_{l,k}\}$'s, which is coincident with the grid crossings. With these choices, the weights $\{w_{l,k}\}$ can be computed as $w_{l,k} = S(P_{l,k}) \cdot \Delta P_l^D$ [27].

As the data set usually does not contain the $\{S(P_{l,k})\}$'s, these values can be estimated as a weighted average of the data points that lie inside the neighborhood of the $P_{l,k}$, called *receptive field*, $A_{l,k}$. This can be chosen as the spherical region, centered in $P_{l,k}$, with radius proportional to ΔP_l. A possible weighting scheme, related to Nadaraya–Watson regression [30], is

$$\tilde{S}(P_{l,k}) = \frac{n_{l,k}}{d_{l,k}} = \frac{\sum_{P_m \in A_{l,k}} S(P_m)\, e^{-\|P_{l,k} - P_m\|^2/(\sigma_l/2)^2}}{\sum_{P_m \in A_{l,k}} e^{-\|P_{l,k} - P_m\|^2/(\sigma_l/2)^2}}. \tag{3}$$

This scheme allows also filtering out measurement noise on the data points.

It should be noticed that the single layer of the HRBF model is configured directly using the data points value (2), without the iterations required by gradient-based configuration procedures used both in RBF models (e.g., [13]), or SVM models (e.g., [31]). Such direct configuration is derived from linear filtering theory and it requires that the Gaussian energy is unitary. For this reason, normalized Gaussians are employed in the HRBF model.

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 the $\tilde{S}(P_{l,k})$ where too few points fall inside $A_{l,k}$. A better solution is to adopt a hierarchical scheme by adding and configuring one layer at a time, starting from the largest scale. Although each new layer often features half the scale of the previous one, arbitrary scales could be used for the different layers.

All the layers after the first one are trained to approximate the residual r_l of the previous layer that represents the difference between the original data and the actual output produced

by the network through the already configured layers. Hence, r_l is computed as

$$r_l(P_m) = z_m - \sum_{j=1}^{l} a_j(P_m; \sigma_j) \tag{4}$$

and it is used in place of $S(P_m)$ in (3) for estimating the $\{w_{l,k}\}$ for $l > 1$.

The quality of the local approximation around $P_{l,k}$ is evaluated through the *local residual error* $R(P_{l,k})$ that is defined as

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

The l_1-norm of the local residual inside $A_{l,k}$ has been used to limit the impact of the outliers.

Only if $R(P_{l,k})$ is over a given error threshold ϵ, a Gaussian of a lower scale is inserted in the corresponding grid crossing $P_{l,k}$; otherwise, the Gaussian is not inserted. As a result, at the end of the learning procedure, Gaussian units, at smaller scales, are present in those regions where the most subtle details are located: units were adaptively allocated, each with an adequate scale, in the different regions of the input domain, forming a sparse approximation of the data. The introduction of new layers ends when the residual error goes under threshold over the entire input domain (uniform approximation).

This approach has been compared with classical multiresolution analysis through wavelet basis [20]. While wavelet are most suitable to multiscale analysis, HRBF does produce an output of higher quality when data are affected by noise (approximation problems).

This HRBF training procedure is a batch procedure, which exploits the knowledge of the entire input data set and adopts local estimates to setup the network parameters. This produces a very fast configuration algorithm, which is also suitable to be parallelized; however, all the data points should be available before starting the network configuration.

We summarize here the HRBF configuration steps.

- If the scale is divided by two in each layer, the Gaussians width and position for each layer are completely specified starting from the scale of the first layer σ_1.
- Given the scale parameter of the first layer and the data bounding box, the grids of all the layers are defined along with the maximum number of Gaussians for each layer M_l. We explicitly recall that a Gaussian is inserted only when the local residual error in (5) is over threshold.
- For the first layer, the weight of each Gaussian is estimated through a local weighted average of the input data and through a local weighted average of the residuals for the next layers [cf., (3)].

As the Gaussian function quickly decreases to zero with the distance from its center, processing time can be saved computing the contribution of each Gaussian to the residuals, only for those points that belong to an appropriate neighborhood of the Gaussian center, $P_{l,k}$, called *influence region* $I_{l,k}$. $I_{l,k}$ may be coincident or not with $A_{l,k}$.

III. Online Training Procedure

When the entire data set is not available all together but the points come one after the other, the scheme described in Section II cannot be applied. When a new point P_{new} is added to the data set S_{old}, the estimate in (3) becomes out of date for the first layer and $\tilde{S}(P_{1,k})$ has to be reestimated with the new data set constituted of $S_{\text{old}} \cup P_{\text{new}}$. As a result, $a_1(\cdot)$ changes inside the *influence region* of all the updated units and the residual r_1 changes for all the points inside this area. This requires updating the weights of the second layer for those Gaussians whose *receptive field* intersects with this area. This chain reaction may involve an important subset of the HRBF network's units. Moreover, the new point can prompt the request of a new layer, at a smaller scale.

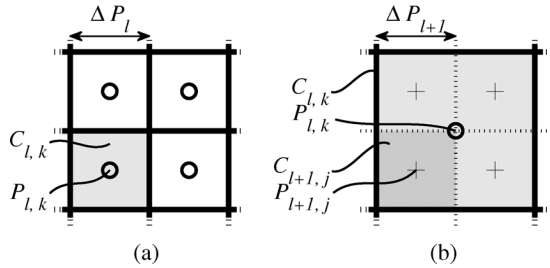


Fig. 1. *Close neighborhood* $C_{l,k}$ of the Gaussian centered in $P_{l,k}$, belonging to the lth layer, is shown in pale gray in panel (a). The close neighborhoods tessellate the input domain, partitioning it in squares which have side equal to that of the lth grid ΔP_l and are offset by half grid side. In the next layer, $C_{l,k}$ is split into four *close neighborhoods*, $C_{l+1,j}$ (quads) according to quad-tree decomposition, as shown in panel (b). Each $C_{l+1,j}$ has a side half the length of $C_{l,k}$, and it is centered in a Gaussian $P_{l+1,j}$ positioned in "+."

One possibility is to reconfigure the entire network computing all the parameters every time a new point is added to the input data set. This solution is computationally expensive and unfeasible for real-time configuration. To avoid this, a few approximations have to be introduced.

The most limiting factor to real-time operation is the shape of the receptive field: as the Gaussians have radial symmetry, their receptive field comes out with a spherical shape that does not allow an efficient partitioning of the data points. To overcome this problem, the receptive field is approximated with a cubic region [27]; this approximation can be accepted as far as the input space has a low dimensionality [32].

Cubic approximation allows organizing the incoming data points and the HRBF parameters into an efficient data structure. For each layer l, the input space is partitioned into nonoverlapping regular boxes $\{C_{l,k}\}$, each centered in a different Gaussian center $P_{l,k}$. As shown in Fig. 1, for a $\mathbb{R}^2 \rightarrow \mathbb{R}$ mapping, the input domain is partitioned into squares $C_{l,k}$, where each $C_{l,k}$ is called the *close neighborhood* of the (l,k)th Gaussian $G_{l,k}$. We explicitly remark that the vertices of each $C_{l,k}$ are shifted of half side with respect to the grid centers.

A particular data structure is associated to each Gaussian $G_{l,k}$. This contains the Gaussian's position $P_{l,k}$, its weight $w_{l,k}$, the numerator $n_{l,k}$, and the denominator $d_{l,k}$ of (3). The structure associated to the Gaussian at the top of the hierarchy (current highest layer h) contains also all the data points that lie inside $C_{h,k}$. To obtain this, when a Gaussian is split during learning, its associated points are sorted locally through qsort algorithm and distributed among the 2^D new Gaussians of the higher layer.

As $\Delta P_l = \Delta P_{l-1}/2$, the *close neighborhood* of each Gaussian of the lth layer (*father*) is formed as the union of the *close neighborhoods* of the 2^D corresponding Gaussians of the

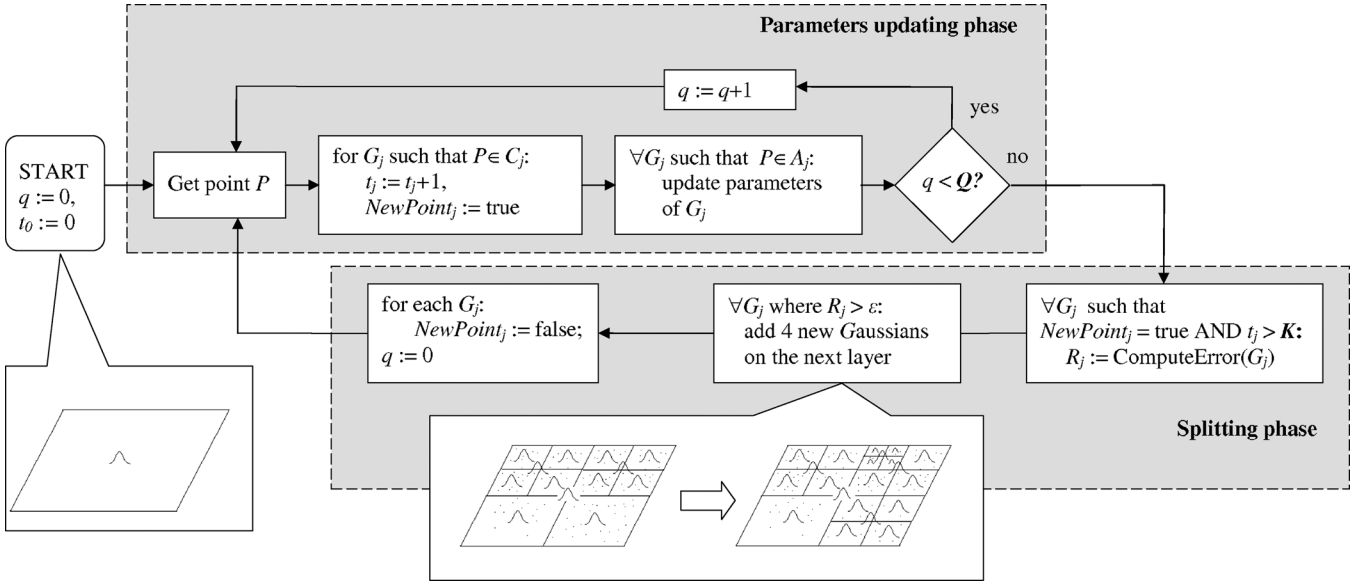


Fig. 2. Schematic representation of the online HRBF configuration algorithm.

$(l+1)$th layer (*children*). This relationship depicted in Fig. 1(b) is taken advantage to organize the data in a quad-tree: the points which lie inside $C_{l,k}$ are efficiently retrieved as those contained inside the close neighborhood of its four children Gaussians.

In the following, we will assume that the side of the *receptive field* $A_{l,k}$ and of the *influence region* $I_{l,k}$ of a Gaussian are set to twice the size of the Gaussian's *close neighborhood* $C_{l,k}$ to allow partial overlapping of adjacent units. However, any relationship such that $A_{l,k}$ and $I_{l,k}$ cover an integer number of *close neighborhoods* produces an efficient computational scheme.

The configuration algorithm is structured as a sequence of steps of weights updating followed by a single step in which residual is evaluated and Gaussians possibly inserted. These two phases, depicted in the schema in Fig. 2, are iterated until new points are added.

The algorithm starts with a single Gaussian positioned approximately in the center of the acquisition volume, with a width sufficiently large to cover the volume. An estimate of the dimension of the acquisition volume is therefore the only *a priori* information needed by the configuration algorithm.

A. First Learning Phase: Parameters Updating

When a new point P_{new} is given, the quantities $n_{l,k}$, $d_{l,k}$, and $w_{l,k}$ (3), associated to the Gaussians such that $P_{\text{new}} \in A_{l,k}$, are updated

$$n_{l,k} := n_{l,k} + r_{l-1}(P_{\text{new}})e^{-\|P_{l,k}-P_{\text{new}}\|^2/(\sigma_l/2)^2} \quad (6a)$$

$$d_{l,k} := d_{l,k} + e^{-\|P_{l,k}-P_{\text{new}}\|^2/(\sigma_l/2)^2} \quad (6b)$$

$$w_{l,k} = \frac{n_{l,k}}{d_{l,k}} \cdot \Delta P_l^D \quad (6c)$$

where $r_{l-1}(P_{\text{new}})$ is computed, likewise as in (4), as the difference between the input data and the sum of the output of the first $l-1$ layers of the actual network computed in P_{new}.

We explicitly notice that the modification of the weight of a Gaussian in the lth layer $G_{l,k}$ modifies the residual of that layer r_l inside the Gaussian's *influence region*. Hence, the terms in (6) should be recomputed for the next layer for all the (already acquired) data points inside the *influence region* of $G_{l,k}$. This would require to recompute the residual of the next layer and so forth up to the last configured layer.

However, this would lead to an excessive computational load, and, in the updating phase, the terms in (6) are modified only by adding the contribution of P_{new} to $n_{l,k}$ and $d_{l,k}$. The rationale is that increasing the number of points, (6c) tends to (3). The residual is then recomputed only in P_{new}, which is sufficient to obtain a good estimate of (3).

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

B. Second Learning Phase: Splitting

After Q points have been collected, the need for new Gaussians is evaluated. To this aim, the reconstructed manifold is examined inside the *close neighborhood* of those Gaussians which satisfy the following three conditions: i) they do not have any *children*, ii) at least a given number K of points has been sampled inside their *close neighborhood*, and iii) their *close neighborhood* includes at least one of the last Q points acquired. These are the Gaussian candidates for splitting. Let us call J their ensemble.

For each Gaussian of J, the local residual $R(P_{l,k})$ (5) is reevaluated for all the points inside its *close neighborhood* using the actual network parameters. If $R(P_{l,k})$ is larger than the given error threshold ϵ, splitting occurs: 2^D new Gaussians at half scale are inserted inside $C_{l,k}$. The points associated to the Gaussian $G_{l,k}$ are distributed among these four new Gaussians depending on which $C_{l,k}$ they belong to [cf., Fig. 1(b)].

We explicitly remark that the estimate of $R(P_{l,k})$ requires the computation of the residual, that is the output of all the previous layers of the network, for all the points inside $C_{l,k}$. To this aim, the output of all the Gaussians (of all the layers) whose receptive field contains $C_{l,k}$ is computed.

As a result, the parameters of an inserted Gaussian $G_{l,k}$, $n_{l,k}$, $d_{l,k}$, and $w_{l,k}$ in (6) are computed using all the points contained

in its *close neighborhood*; for this new Gaussian, no distinction is made between the points sampled in the earlier acquisition stages and the last Q sampled points. The quantities $n_{l,k}$, $d_{l,k}$, and $w_{l,k}$ are set to zero when no data point is present inside $C_{l,k}$ and the Gaussian $G_{l,k}$ will not contribute to the network output.

It should be noticed that, as a consequence of this growing mechanism, the network does not grow layer by layer, as in the batch case, but it grows on a local basis.

C. Proof of Convergence

In [20], the capability of a single-layer HRBF, with a scale σ adequate to approximate a given function $S(\cdot)$, has been proven showing that the residual can be made smaller than any given threshold ϵ. Moreover, it has been shown that the sequence of the residuals obtained with the HRBF schema converges to zero under mild conditions on $S(\cdot)$.

As the online configuration procedure is different from the batch one, the convergence of the residuals obtained with the schema described in Section III has to be proven.

The online schema differs from the batch one for both the computation of the weights [(6) versus (3)] and for the rule of insertion of new Gaussians (in the batch scheme, this occurs layerwise, while in the online schema, it occurs locally during the splitting phase).

We first show that the output of each layer of the online HRBF is asymptotically equivalent to that of the batch HRBF. Let us first consider the case of the first layer.

Let S_p be the input data set constituted of the first p points sampled from S and denote with $\underline{*}$ the operation such that

$$a_1(\cdot) = S_p \underline{*} g(\cdot;\, \sigma_1) \tag{7}$$

is the output of the first HRBF layer, configured using S_p. It can be shown that when p tends to infinite, the function computed in (7) converges to the value computed in (2) for the batch case.

This is evident for this first layer, whose weights are estimated as $\tilde{S}(P_{1,k})$, and $r_0(P) = S(P)$ holds. In this case, the following asymptotic condition can be derived:

$$\lim_{p\to\infty} w^{\mathrm{b}}_{1,k} = \lim_{p\to\infty} \frac{\sum_{m=1}^{p} S(P_m)\, e^{-\|P_{1,k}-P_m\|^2/(\sigma_1/2)^2}}{\sum_{m=1}^{p} e^{-\|P_{1,k}-P_m\|^2/(\sigma_1/2)^2}} \Delta P_1^D = \lim_{p\to\infty} w^{\mathrm{o}}_{1,k} \tag{8}$$

where $w^{\mathrm{b}}_{1,k}$ are the weights computed in the batch algorithm through (3) and $w^{\mathrm{o}}_{1,k}$ are those computed in the online algorithm through (6). It follows that:

$$\lim_{p\to\infty} \Delta r_1(P) = \lim_{p\to\infty} r^{\mathrm{b}}_1(P) - r^{\mathrm{o}}_1(P) = 0 \tag{9}$$

where $r^{\mathrm{b}}_1(P)$ is the residual at the point P computed through (3), and $r^{\mathrm{o}}_1(P)$ is the same residual computed through (6).

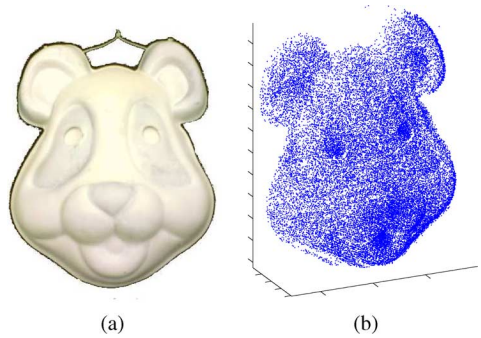
If a second layer is considered, the estimate of its weights can be reframed as

$$n_{2,k} := n_{2,k} + (r^{\mathrm{o}}_1(P_p) + \Delta r_1(P_p)) e^{-\|P_{2,k}-P_p\|^2/(\sigma_2/2)^2} \tag{10a}$$

$$d_{2,k} := d_{2,k} + e^{-\|P_{2,k}-P_p\|^2/(\sigma_2/2)^2}. \tag{10b}$$

As $\lim_{p\to\infty} \Delta r_1(P_p) = 0$ and $d_{2,k}$ always increases with p, the contribution of the initially sampled data points becomes negligible as p increases. As a result, $\lim_{p\to\infty} w^{\mathrm{b}}_{2,k} = \lim_{p\to\infty} w^{\mathrm{o}}_{2,k}$, and also the approximation of the residual of the second layer tends to be equal for the batch and online approaches. The same applies also to the higher layers.

Splitting cannot introduce a poor approximation as the weights of the Gaussians inserted during the splitting phase are robustly initialized with an estimate obtained from at least K points.



(a) (b)

Fig. 3. Typical data set acquired by the autoscan system [8]. The panda mask in (a) is reconstructed starting from 33 000 3-D points automatically sampled on the surface by the autoscan system; these constituted the input to the HRBF network. Notice the higher point density in the mouth and eyes regions.

IV. Results

We have extensively applied the online HRBF model to laser scanning. Digitization was performed through the autoscan system [8], [33], which allows sampling more points inside those regions which contain more details: a higher data density can therefore be achieved in those regions that contain higher spatial frequencies. To this aim, real-time feedback of the reconstructed surface is of paramount importance as shown in [34].

A typical set of sampled data is reported in Fig. 3(b): it is constituted of 33 000 points sampled over the surface of the artifact (a panda mask) in Fig. 3(a). As can be seen in Fig. 4, the reconstruction becomes better with the number of points sampled. Acquisition was stopped when the visual quality of the reconstructed surface was judged sufficient and no significant improvement could be observed when new points were added [compare Fig. 4(e) and (f)].

To assess the effectiveness of the online algorithm, a quantitative analysis of the local and global error has been carried out. Since the true surface is not available, a cross-validation approach has been adopted [35]; from the data set in Fig. 3(b), 32 000 points, randomly extracted, were used to configure the network parameters (training set), and 1000 for testing (test set).

The error, expressed in millimeters, was measured in l_1-norm, ϵ_{mean}, and in l_2-norm, root mean squared error (RMSE) as

$$\epsilon_{\mathrm{mean}} = \frac{1}{N}\sum_{i=1}^{N} |r_T(x_i)| \tag{11a}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} r_T(x_i)^2} \tag{11b}$$

where $r_T(x_i)$ is the reconstruction error on the ith point of the test set, $i = 1, \ldots, N$. To avoid border effects, (11a) and (11b)

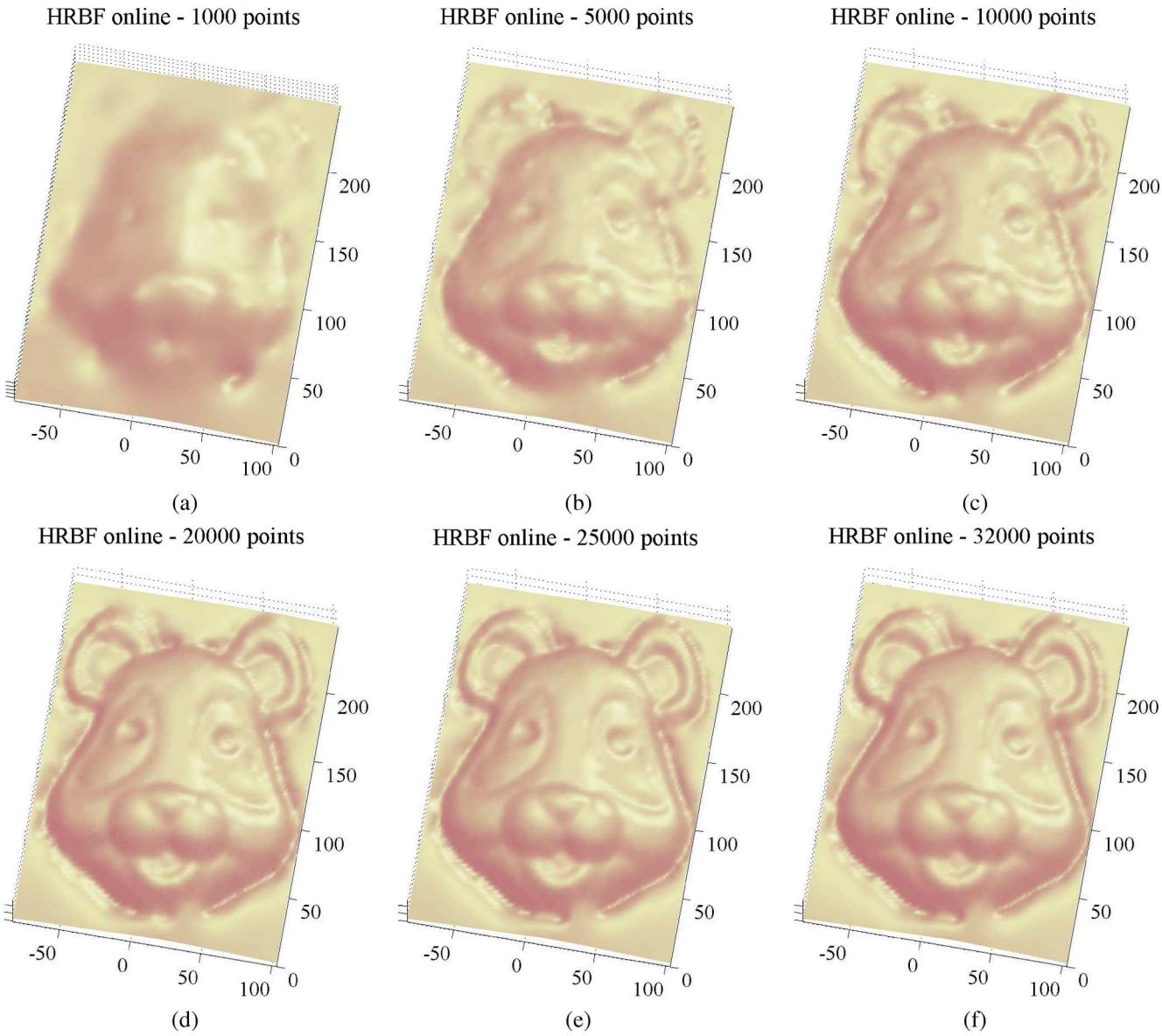


Fig. 4. Panels show the reconstruction with online HRBF after 1000, 5000, 10 000, 20 000, 25 000, and 32 000 points have been sampled.

TABLE I
ACCURACY AND PARAMETERS OF EACH LAYER OF THE HRBF NETWORKS

		on-line			pure batch			batch constrained	
#layer	σ	#Gauss. (total)	RMSE	ϵ_{mean}	#Gauss. (total)	RMSE	ϵ_{mean}	RMSE	ϵ_{mean}
1	363.3	1 (1)	47.8	46.2	1 (1)	47.8	46.2	47.8	46.2
2	181.7	4 (5)	30.2	28.0	4 (5)	30.2	28.0	30.2	28.0
3	90.8	16 (21)	13.0	10.7	16 (21)	13.0	10.7	13.0	10.7
4	45.4	46 (67)	6.44	5.10	62 (83)	6.40	5.05	6.39	5.07
5	22.7	160 (227)	3.33	2.68	204 (287)	3.07	2.50	3.03	2.48
6	11.4	573 (800)	2.17	1.66	678 (965)	1.73	1.41	1.72	1.41
7	5.68	2 092 (2 892)	1.16	0.838	2 349 (3 314)	0.849	0.637	0.872	0.657
8	2.84	6 330 (9 222)	0.530	0.391	7 079 (10 393)	0.510	0.373	0.526	0.385

have been computed considering only the points that lie inside an internal region of the input domain; this region has been defined as the region delimited by the convex hull of the data set, reduced by 10%.

Results have been compared with those obtained by the batch configuration algorithm and by SVM through the widely diffused SVM$^{\text{light}}$ package [36] applied on the same data sets.

The experiments have been carried out on a machine equipped with Intel Pentium 4, 2.4-GHz, 512-KB cache, 512-MB RAM.

A. Comparison With the Batch HRBF

Results of the comparison with the batch HRBF approach are reported in Table I. These figures have been obtained with the following parameters: $Q = 100$, $K = 3$, and $L = 8$ layers.

TABLE II
RECONSTRUCTION WITH SEVERAL DATA SETS

		on-line			pure batch			batch constrained	
dataset	#points	#Gauss. (total)	RMSE	ϵ_{mean}	#Gauss. (total)	RMSE	ϵ_{mean}	RMSE	ϵ_{mean}
panda mask	32 000	6 330 (9 222)	0.530	0.391	7 079 (10 393)	0.510	0.373	0.526	0.385
cow mask	33 861	6 360 (9 501)	0.667	0.476	7 680 (11 335)	0.643	0.460	0.660	0.470
doll	15 851	3 366 (6 058)	0.466	0.331	3 312 (6 294)	0.389	0.287	0.447	0.312

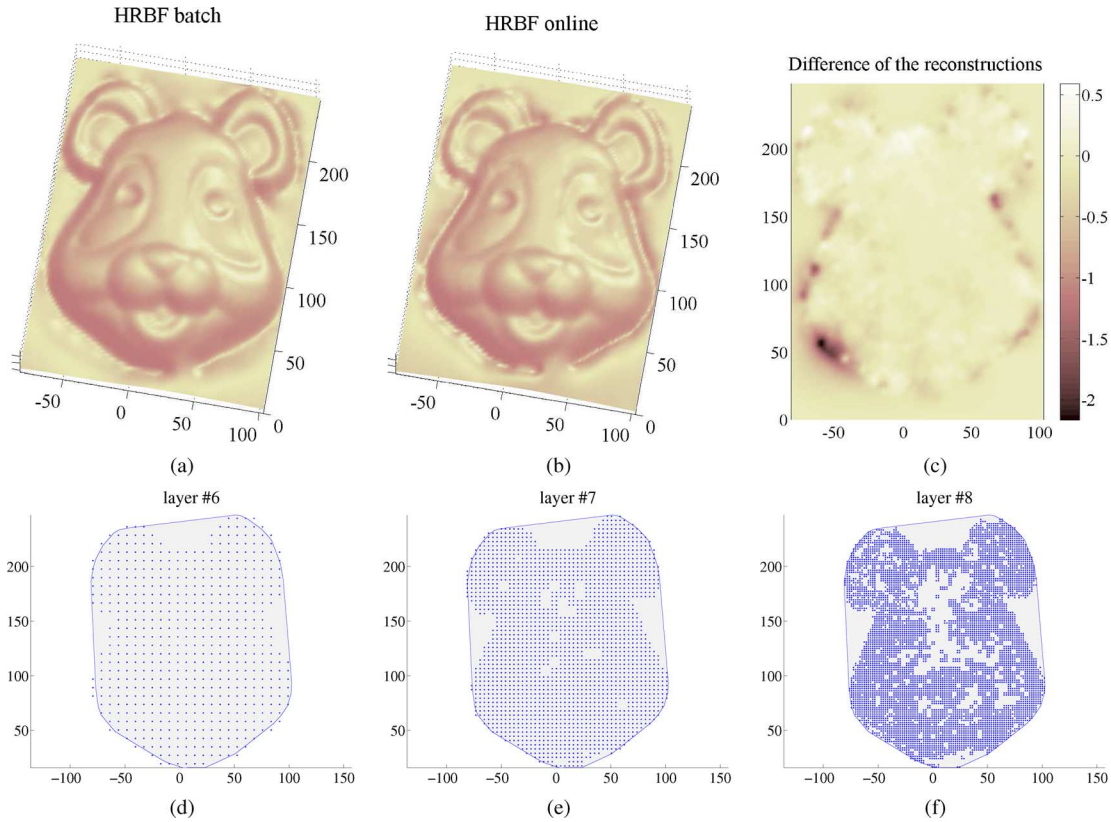


Fig. 5. Reconstruction with (a) HRBF batch and (b) HRBF online. The difference between the two surfaces is shown in panel (c). In panels (d)–(f), the center of the Gaussians allocated by the online algorithm in the last three layers is shown.

The error threshold ϵ was set for all the layers equal to the nominal digitization error that was 0.4 mm. The final reconstruction error of 0.391 mm is very close to ϵ; a total of 9222 Gaussians were allocated over the eight layers, and produce a sparse approximation [cf., Fig. 5(d)–(f)].

The network complexity and the reconstruction error have been compared with those obtained when the network was configured using a batch approach, with the same number of layers of the online version.

Two batch modalities have been considered. In the first one, "pure batch," the configuration procedure summarized at the end of Section II [20] is adopted. In the second approach, "batch constrained," the Gaussians are placed in the same position, and have the same width, of those of the online approach, while the weights are computed through (3), considering all the data points inside the receptive field of each Gaussian, as described in [29].

As shown in Fig. 5, the surface reconstructed by the batch HRBF has a slight better appearance than the online one, especially at the object border as shown by the difference image [Figs. 5(c) and 6]. This was obtained at the expense of a larger number of Gaussians: about 12.7% more than those used in the online approach, being 10 393 versus 9222 (Table I).

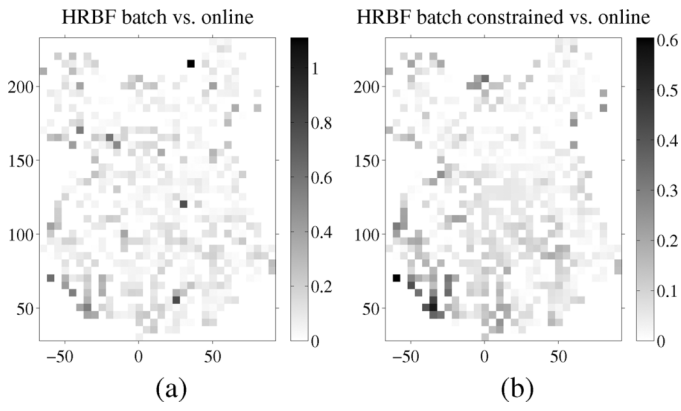


Fig. 6. Difference in the reconstruction error on the points of the test set: (a) online versus pure batch, and (b) online versus batch constrained.

Despite the difference in the computational resources used, the global accuracy of the batch approach is only slightly better

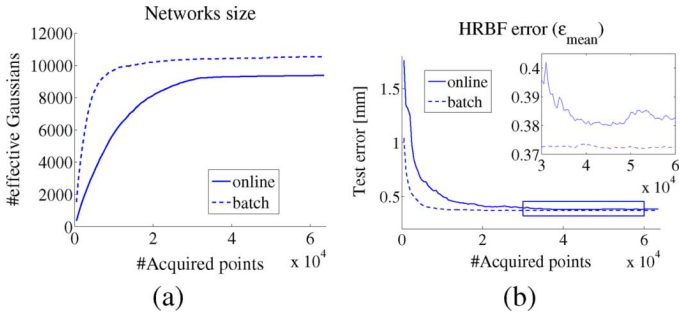


Fig. 7. Number of (a) Gaussian units and (b) mean error as a function of the number of data points used to configure the networks. Data set is grown of 500 data points at a time.

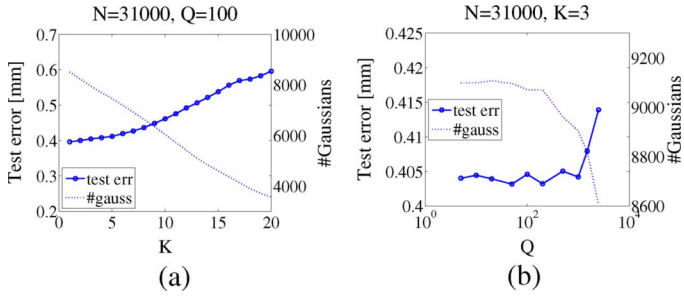


Fig. 8. Test error and number of Gaussian units (a) with respect to K with Q fixed to 100, and (b) with respect to Q with K fixed to 3.

TABLE III
COMPARISON BETWEEN SVM AND HRBF

paradigm	ϵ_{mean} train [mm]	ϵ_{mean} test [mm]	conf. time [s]	#units
SVM ($C = 1$)	0.683	0.645	$3 \cdot 10^3$	18 643
SVM ($C = 10$)	0.303	0.288	$2.11 \cdot 10^4$	12 737
SVM ($C = 10$)	0.296	0.288	$2.68 \cdot 10^5$	13 272
HRBF batch	0.416	0.405	3.54	11 143
HRBF on-line	0.393	0.388	105	9 277

than the online one, being of 0.373 mm versus 0.391 mm (4.82%).

We remark here that acquisition was stopped when the visual appearance of the model (reconstructed with the online approach) was considered sufficient.

We have therefore investigated if there was room for further accuracy improvement by acquiring more data points. To this aim, we have plotted in Fig. 7 the rate of Gaussian allocation and of error decrease as a function of the number of data points. As is clearly shown, the batch version grows and converges faster than the online version: it achieves ϵ_{mean} of 0.391 mm using only 8500 data points. The figure shows also that the error of the online model can be slightly lowered adding new points down to 0.381 mm, closer to the batch approach. To achieve such an error, only 99 more Gaussians are required.

B. Comparison With Support Vector Machine

SVMs [31], [37] are a supervised learning paradigm for classification and regression. In the basic formulation of SVM for regression, the approximation is formulated as a linear combination of kernel functions. The configuration phase selects a subset of the training points [support vectors (SVs)], which are used as centers of the kernels. The estimate of the weights and, hence, the selection of the SVs, is obtained as the solution of a quadratic programming optimization problem.

Although different kernels can be used, Gaussian kernel is used here for sake of comparison; a value of $\sigma = 2.84$ mm was chosen that is equivalent to the smallest scale parameter used by the HRBF network. The error parameter ϵ was set to 0.4 mm, the same value used for the HRBF models. Results are reported in Table III. The regularization parameter C, which bounds the weights magnitude and is usually set through cross validation, was set here to several values (1, 10, and 100) for evaluating its impact on the SVM performances. Results are reported in Table III.

V. DISCUSSION

Results are consistent for different artifacts (cf., Fig. 10); real-time visualization of the actual surface has been of great value in directing the laser spot for more time in the most critical regions, collecting more points there. The best results are obtained when the points are sampled mostly uniformly in the first few layers. This allows a more robust estimate of the weights of the Gaussians of these layers, as the Gaussians of the first layers cover all a large portion of the input space. In all the experiments, data acquisition was stopped when the visual appearance of the reconstructed model was considered satisfactory. Alternatively, data acquisition could be stopped when splitting no more occurs. The error was measured as the mean value of the test error averaged over ten randomizations on the same data set.

The online configuration algorithm contains two parameters: K and Q (Section III); their influence has been experimentally assessed by training the model with different combinations of K and Q. This gives an insight on the general behavior of the online algorithm as a function of these parameters.

Fig. 8(a) shows that the number of Gaussians decreases while the test error increases with K. This is due to the fact that, increasing K, more points are collected inside the close neighborhood of a Gaussian, before splitting it. Therefore, in this case, given the same total number of points, although the single weights can be estimated better, a smaller number of split operations would occur. This suggests to use a very low value of K; in fact, $K = 1$ produces the smallest test error of 0.378 mm. This is paid with a larger number of Gaussians that reaches a total of 9968 for $K = 1$. A tradeoff has been adopted here choosing $K = 3$, which produces a total number of Gaussians of 9222, about 92.5% of those obtained setting $K = 1$.

The behavior of the test error as a function of Q is shown in Fig. 8(b). For small values of Q, the behavior is almost the same: the error starts increasing after $Q = 1000$, although the increase is of small amplitude (about 0.01 mm from $Q = 1000$ to $Q = 2500$). The number of Gaussian units instead decreases monotonically with Q, with a marked decrease above $Q = 300$. This can be explained by the fact that, when a new Gaussian is inserted, its weight is initialized using all the already acquired points that belong to its close neighborhood (Section III-B). Afterward, its weight is updated considering only the points inserted inside its *receptive field*, as in (6). Therefore, increasing Q, the weight associated to each new Gaussian can be computed more reliably as more points are available for its estimate. However, when Q assumes too large values with respect to the

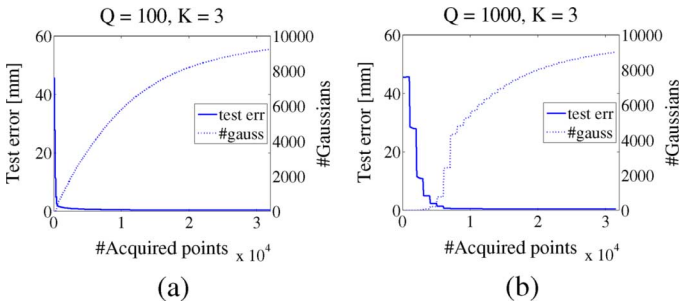


Fig. 9. Test error and number of Gaussian units reported with respect to N for small and large values of Q: (a) Q was fixed to 100, and (b) Q was fixed to 1000.

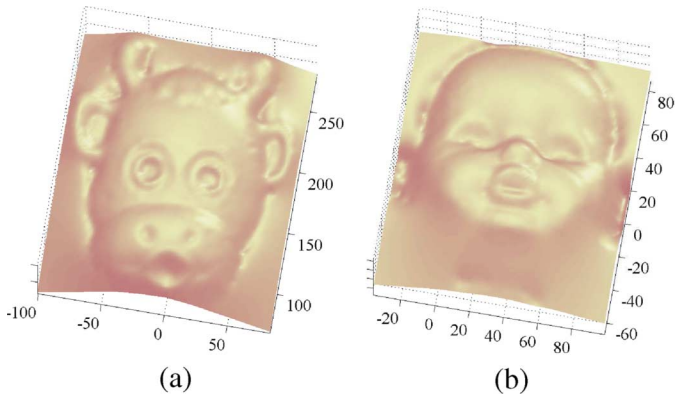


Fig. 10. HRBF online reconstruction of the data set (a) "cow" (33 861 points, 9501 Gaussians), and (b) "doll" (15 851 points, 6058 Gaussians).

number of available data points, not enough splits occur, and the reconstruction becomes poorer. This situation is depicted in Fig. 8(b) where for a relatively large value of Q, the test error tends to increase as an effect of the decrease in the number of the allocated Gaussians. From the curve in Fig. 8, one could derive that the optimal value of Q would be ≈ 1000, as it allows a low reconstruction error with a reduced number of Gaussians. However, the price to be paid for such saving in computational units is the loss of interactivity. In fact, increasing Q, the split operation occurs less frequently in time. As this operation produces the largest decrease in the reconstruction error, a long time has to elapse before the user can see a large change in the appearance of the reconstructed model, and the improvement in the model quality is not smooth in time. Moreover, the reconstruction error decreases less quickly; this is well captured in Fig. 9. Hence, the use of large values of Q is not interesting for such applications where interactive online learning is required and the value of $Q = 100$ has been adopted here as it is approximately twice the data sampling rate, that is, of 60 points/s.

Therefore, although the value of K and Q may be subjected to optimization with respect to network accuracy or size, the resulting values may not be adequate for real-time applications. In particular, as Q produces a very similar test error over a wide range of values, it has been set here according to the data sampling rate to guarantee interactivity; this value is lower than the one that would produce the smallest network. This value could be even lowered, but at the price of a large overhead, as the split phase is the most computationally expensive. Therefore, a value of Q related to a data rate seems the most reasonable. In any case, we remark that using more Gaussians than the minimum required is not very sensitive to overfitting, because of the mechanism of local weights computation, provided that enough points have been sampled inside the receptive field of the Gaussians, that is guaranteed by a reasonable value of K. This is shown in Fig. 9, where the test error does not increase with the number of Gaussian units.

As K is concerned, its value is related to the amount of noise on the sampled points: the larger the noise, the greater should be K to reduce the estimate error on the weights. However, possible bias in the weights estimate can be recovered in the higher layers thanks to the incremental optimization construction. For this reason, a very low value of K can be chosen: a value of $K < 5$ worked well in all our experiments and produced a good reconstruction with a reasonably low number of Gaussians.

The parameter L decides the level of detail in the reconstructed surface as it sets the smallest value of σ that is related to the local spatial frequency. It could be set in advance when this information is available to avoid the introduction of spurious high-frequency components; otherwise, L is incremented until the error in (11) goes under threshold or a maximum number of Gaussians is inserted. It should be remarked that in this latter case, if Q were too small, L could increase more than necessary.

The mechanism used in the weight update phase that does not require the reconfiguration of the whole network produces a slight bias in the weights. This can be appreciated in Table I, where the accuracy obtained when the weights are estimated considering all the data points (batch constrained) is compared with that obtained with the online approach described here. In fact, the manifold height in $P_{l,k}$ (3) is estimated as the ratio between $n_{l,k}$ and $d_{l,k}$, obtained as the runtime 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 (where all the Gaussian weights are computed using the height of all sampled points inside their *receptive field*). In the higher layer, where the residual in the already acquired points is not updated, the estimate of the weights may contain a bias. This, in turns, may produce a reconstruction error. However, in the splitting phase, the bias in the weights and the reconstruction error are corrected in the *close neighborhood* of the splitted units, as the residual is recomputed there, using all the data points inside the $C_{l,k}$. Moreover, due to the nonorthogonality of the Gaussian basis function, the HRBF model is able to compensate the reconstruction error in one layer, with the approximation achieved by the next layer [38].

The maximum number of layers does not determine only the maximum spatial frequency that can be reconstructed, but it has also a subtle effect. In fact, the online configuration approach, differently from the batch one, can introduce Gaussians at the kth level also when $(k-1)$th level has not been completed: a Gaussian can be split before the neighbor Gaussians of the same layer. This is the case when higher frequency details are concentrated inside the *receptive field* of that Gaussian, as the parameters of each Gaussian are updated independently from those of the others. Therefore, one branch of the network can extend before another branch.

TABLE IV
NUMBER OF GAUSSIANS OF THE LAST FOUR LAYERS OF NETWORKS CONFIGURED WITH DIFFERENT MAXIMUM NUMBERS OF LAYERS L

L \ l	6	7	8	9
7	565	1948	—	—
8	565	1848	4185	—
9	565	1840	3777	2683

When the maximum number of layers of the network L is too low for the frequency content of the given data set, the error inside the close neighborhood of some Gaussians of the last layers will contain also the highest frequency details. As a consequence, the weights in these regions may contain a bias that produces a local poor reconstruction. This error affects also the close neighborhood of the adjacent units through the influence region and the receptive field of the corresponding Gaussians. This, in turns, may induce splitting of these adjacent units and produces networks of different structures when a different maximum number of layers is prescribed (cf. Table IV).

Differently from other growing networks models [3], [17], [22], pruning is not adopted here, as all the units participate in the reconstruction because of the configuration mechanism; pruning can be considered useful when dealing with time-variant systems or when some of the data are not pertinent, but the problem addressed here does not belong to this class, and the additional complexity for managing pruning does not seem justified here.

Online HRBF shares with other growing networks models the criterion for inserting new units: the insertion is decided on the basis of a local error measure, which is fundamental to achieve real-time operation. The other element which allows real-time operation is the grid support, which guides Gaussian positioning. This is shared also by [3]. However, in [3], all the weights are recomputed after the insertion of new Gaussians, while here only a subset of the weights is recomputed thanks to the hierarchical *close neighborhood* structure. This produces a large saving, especially for large networks. It should be noticed that this growing strategy implicitly implements an active learning procedure [39], as the data points that participate in the configuration of the new Gaussians are only those that carry an overthreshold error. Grid support has been also adopted by [25] and [26]; however, in their approach, global optimization is used that makes the configuration procedure computationally heavy.

Global optimization is also adopted by SVM; this has an impact on the computational time required for the configuration that is $3 \cdot 10^3$ s for $C = 1$, which produces a high error of 0.645 mm and $21.1 \cdot 10^3$ s for $C = 10$, which reduces the error to 0.288 mm. For $C = 100$, the test error does not improve, while the processing time becomes unbearable.

Besides the computational time, SVM reconstruction is heavily affected from the value of the C parameter (cf. Table III). As this value cannot be estimated *a priori*, the usability of the SVM for large data sets is limited only to the cases in which C can be given *a priori*. Moreover, an adequate value of σ should also be given, while in HRBF models, σ is decreased layer by layer from an arbitrarily large value.

As other kernel methods, SVMs can be declined in an online training version (e.g., [40]); in this case, starting from the optimal solution for a given data set, the solution is updated when a new point is added to the data set. For very large data sets, the incremental optimization can be more efficient than searching the global optimal solution over all the data sets. However, at least for the data sets used for our experiments, the use of an incremental version did not produce improvement both in computational time and accuracy with respect to the batch SVM approach [41], resulting in unsuitable for such real-time interactive applications.

VI. CONCLUSION

An online training procedure for real-time, online, manifolds estimation is here presented along with its application to 3-D scanning. The procedure allows a more effective scanning procedure, providing as a feedback to the operator the surface obtained up to that time. The presented model, derived from the batch HRBF model, produces results comparable to it, and can be effectively used in all the domains of low dimensionality. A proof of the equivalence between the two models in the asymptotic case has been provided.

REFERENCES

[1] D. Saad, *On-Line Learning in Neural Networks*. Cambridge, U.K.: Cambridge Univ. Press, 1998.
[2] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Oxford Univ. Press, 1995.
[3] A. Alexandridis, H. Sarimveis, and G. Bafas, "A new algorithm for online structure and parameter adaptation of RBF networks," *Neural Netw.*, vol. 16, no. 7, pp. 1003–1017, 2003.
[4] A. Rubaai, R. Kotaruand, and M. Kankam, "Online training of parallel neural network estimators for control of induction motors," *IEEE Trans. Ind. Appl.*, vol. 37, no. 5, pp. 1512–1521, Sep./Oct. 2001.
[5] R. Kiran, S. Jetti, and G. Venayagamoorthy, "Online training of a generalized neuron with particle swarm optimization," in *Proc. Int. Joint Conf. Neural Netw.*, 2006, pp. 5088–5095.
[6] J. Fan, N. Dimitrova, and V. Philomin, "Online face recognition system for videos based on modified probabilistic neural networks," in *Proc. Int. Conf. Image Process.*, 2004, vol. 3, pp. 2019–2022.
[7] Z Corporation, Burlington, MA, 2009 [Online]. Available: http://www.zcorp.com/Products/3D-Scanners/ZScannerandtrade-700/spage.aspx
[8] N. A. Borghese, G. Ferrigno, G. Baroni, S. Ferrari, R. Savaré, and A. Pedotti, "AUTOSCAN: A flexible and portable 3D scanner," *IEEE Comput. Graphics Appl.*, vol. 18, no. 3, pp. 38–41, May/Jun. 1998.
[9] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy, "Real-time 3D model acquisition," in *Proc. 29th Conf. Comput. Graph Int. Tech.*, 2002, pp. 438–446.
[10] S. Rusinkiewicz and M. Levoy, "QSplat: A multiresolution point rendering system for large meshes," in *Proc. SIGGRAPH*, 2000, pp. 343–352.
[11] B. Widrow and M. Hoff, "Adaptive switching circuits," in *IRE WESCON Conv. Record*, 1960, pp. 96–104.
[12] W. G. U. Muller and A. Gunzinger, "Fast neural net simulation with a DSP processor array," *IEEE Trans. Neural Netw.*, vol. 6, no. 1, pp. 203–213, Jan. 1995.
[13] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proc. IEEE*, vol. 78, no. 9, pp. 1481–1497, Sep. 1990.
[14] J. Freeman and D. Saad, "Dynamics of online learning in radial basis function networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 56, no. 1, pp. 907–918, 1997.
[15] L. Yingwei, N. Sundararajan, and P. Saratchandran, "A sequential learning scheme for function approximation using minimal radial basis function neural networks," *Neural Comput.*, vol. 9, no. 2, pp. 461–478, 1997.
[16] C. Takenga, K. Anne, K. Kyamakya, and J. Chedjou, "Comparison of gradient descent method, Kalman filtering and decoupled Kalman in training neural networks used for fingerprint-based positioning," in *Proc. IEEE 60th Veh. Technol. Conf.*, 2004, vol. 6, pp. 4146–4150.
[17] Q. Meng and M. Lee, "Error-driven active learning in growing radial basis function networks for early robot learning," *Neurocomputing*, vol. 71, no. 7–9, pp. 1449–1461, 2008.

[18] N. Murata, , D. Saad, Ed., "A statistical study of on-line learning," in *On-Line Learning in Neural Networks*. Cambridge, U.K.: Cambridge Univ. Press, 1998, pp. 63–92.
[19] J. Moody and C. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Comput.*, vol. 1, pp. 281–294, 1989.
[20] S. Ferrari, M. Maggioni, and N. A. Borghese, "Multi-scale approximation with hierarchical radial basis functions networks," *IEEE Trans. Neural Netw.*, vol. 15, no. 1, pp. 178–188, Jan. 2004.
[21] J. Platt, "A resource-allocating network for function interpolation," *Neural Comput.*, vol. 3, no. 2, pp. 213–225, 1991.
[22] B. Fritzke, "Growing cell structures—A self-organizing network for unsupervised and supervised learning," *Neural Netw.*, vol. 7, no. 9, pp. 1441–1460, 1994.
[23] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Neural Information Processing System*. Cambridge, MA: MIT Press, 1993, vol. 5, pp. 164–172.
[24] M. Orr, J. Hallam, K. Takezawa, A. Murray, S. Ninomiya, M. Oide, and T. Leonard, "Combining regression trees and radial basis function networks," *Int. J. Neural Syst.*, vol. 10, no. 6, pp. 453–465, 2000.
[25] R. M. Sanner and J.-J. E. Slotine, "Gaussian networks for direct adaptive control," *IEEE Trans. Neural Netw.*, vol. 3, no. 6, pp. 837–863, Nov. 1992.
[26] J. I. M. Martinez, "Best approximation of Gaussian neural networks with nodes uniformly spaced," *IEEE Trans. Neural Netw.*, vol. 19, no. 2, pp. 284–298, Feb. 2008.
[27] S. Ferrari, I. Frosio, V. Piuri, and N. A. Borghese, "Automatic multiscale meshing through HRBF networks," *IEEE Trans. Instrum. Meas.*, vol. 54, no. 4, pp. 1463–1470, Aug. 2005.
[28] P.-H. Chen, C.-J. Lin, and B. Schölkopf, "A tutorial on ν-support vector machines," [Online]. Available: http://www.kernel-machines.org
[29] N. A. Borghese and S. Ferrari, "Hierarchical RBF networks and local parameter estimate," *Neurocomputing*, vol. 19, no. 1–3, pp. 259–283, 1998.
[30] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural Comput.*, vol. 7, no. 2, pp. 219–269, 1995.
[31] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, "Support Vector Regression Machines," in *Neural Information Processing Systems*, M. Mozer, M. I. Jordan, and T. Petsche, Eds. Cambridge, MA: MIT Press, 1996, pp. 155–161.
[32] S. Ferrari, G. Ferrigno, V. Piuri, and N. A. Borghese, "Reducing and filtering point clouds with enhanced vector quantization," *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 161–177, Jan. 2007.
[33] N. A. Borghese and S. Ferrari, "A portable modular system for automatic acquisition of 3-D objects," *IEEE Trans. Instrum. Meas.*, vol. 49, no. 5, pp. 1128–1136, Oct. 2000.
[34] F. Bellocchio and S. Ferrari, Università degli Studi di Milano, Milano, Italy, 2009 [Online]. Available: http://www.dti.ummi.it/ferrari/hrbf_online/hrbf_online.wmv
[35] P. Craven and G. Wahba, "Smoothing noisy data with spline functions. Estimating the correct degree of smoothing by the method of generalized cross-validation," *Numer. Math.*, vol. 31, no. 4, pp. 377–403, 1978/79.
[36] T. Joachims, "Making large-scale SVM learning practical," in *Advances in Kernel Methods—Support Vector Learning*. Cambridge, MA: MIT Press, 1999.
[37] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge, U.K.: Cambridge Univ. Press, Mar. 2000.
[38] S. Ferrari, N. A. Borghese, and V. Piuri, "Multiscale models for data processing: An experimental sensitivity analysis," *IEEE Trans. Instrum. Meas.*, vol. 50, no. 4, pp. 995–1002, Aug. 2001.
[39] M. Hasenjager and H. Ritter, "New learning paradigms in soft computing," in *Active Learning in Neural Networks*. Heidelberg, Germany: Physica-Verlag, 2002, pp. 137–169.
[40] J. Ma, J. Theiler, and S. Perkins, "Accurate on-line support vector regression," *Neural Comput.*, vol. 15, no. 11, pp. 2683–2703, 2003.
[41] G. Montana and F. Parrella, "Learning to trade with incremental support vector regression experts," in *Proc. 3rd Int. Workshop Hybrid Artif. Intell. Syst.*, 2008, pp. 591–598.

Stefano Ferrari (M'09) received the M.Sc. degree in computer science from the Università degli Studi di Milano, Milano, Italy, in 1995 and the Ph.D. in computer and automation engineering from the Politecnico di Milano, Milano, Italy, in 2001.

Since 2002, he has been an Assistant Professor at the Dipartimento di Tecnologie dell'Informazione, Università degli Studi di Milano. His research interests are related mainly to neural networks and soft-computing paradigms and their application to the computer graphics, signal processing, and measurement systems.

Francesco Bellocchio received the computer science degree from the Università degli Studi di Milano, Milano, Italy, in 2007. He is currently working towards the Ph.D. degree at the Department of Information Technology, Universita' degli Studi di Milano, Crema, Italy.

His research interests are related mainly to neural networks and soft-computing paradigms and their application to the real-time 3-D surface reconstruction and signal and image processing.

Vincenzo Piuri (S'84–M'86–SM'96–F'01) received the Ph.D. in computer engineering from Politecnico di Milano, Milano, Italy, in 1989.

Since October 2000, he has been Full Professor of Computer Engineering at the Università degli Studi di Milano, Crema, Italy. His research interests include biometrics, signal and image processing for industrial applications, theory and industrial applications of neural networks, and intelligent measurement systems. His original results have been published in more than 250 papers in book chapters, international journals, and proceedings of international conferences.

Prof. Piuri is a Distinguished Scientist of the Association for Computing Machinery (ACM). He was an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS and the IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT. He was President of the IEEE Computational Intelligence Society and Vice President for Publications of the IEEE Instrumentation and Measurement Society. He is Vice President for Publications of the IEEE Systems Council and Vice President for Education of the IEEE Biometrics Council.

N. Alberto Borghese (M'97) graduated in electrical engineering with full marks and honors from Politecnico di Milano, Milano, Italy, in 1985.

Since 2001, he has been an Associate Professor at the Department of Computer Science, Università degli Studi di Milano, Milano, Italy, where he teaches the courses on intelligent systems and robotics, and is the Director of the Laboratory of Applied Intelligent Systems. He was visiting scholar at the Center for Neural Engineering of the University of South California (USC), Los Angeles, in 1991, at the Department of Electrical Engineering, California Institute of Technology (Caltech), Pasadena, in 1992, and at the Department of Motion Capture of Electronic Arts, Canada, in 2000. He coauthored more than 40 peer-reviewed journal papers and holds nine international patents. His research is focused on the development and application of methods and algorithms based on computational geometry and statistical data analysis. His main areas of application are as follows: machine learning, medical imaging, 3-D scanning, motion capture and artificial vision, analysis and modeling human motion control, and cognitive functions. His tight connections with industry provide both interesting applications and funding for the research.