Francesco Bellocchio · N. Alberto Borghese
Stefano Ferrari · Vincenzo Piuri

3D Surface Reconstruction

Multi-Scale Hierarchical Approaches

Springer

3D Surface Reconstruction

Francesco Bellocchio • N. Alberto Borghese
Stefano Ferrari • Vincenzo Piuri

3D Surface Reconstruction

Multi-Scale Hierarchical Approaches

Springer

Francesco Bellocchio
Università degli Studi di Milano
Crema, Italy

N. Alberto Borghese
Università degli Studi di Milano
Milano, Italy

Stefano Ferrari
Università degli Studi di Milano
Crema, Italy

Vincenzo Piuri
Università degli Studi di Milano
Crema, Italy

ISBN 978-1-4614-5631-5 ISBN 978-1-4614-5632-2 (eBook)
DOI 10.1007/978-1-4614-5632-2
Springer New York Heidelberg Dordrecht London

Library of Congress Control Number: 2012948337

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Contents

Chapter 1
Introduction

What is a 3D model? When and why are they used for? How are they created? This chapter presents a brief overview to understand the usefulness of 3D models and how they can be computed from a physical object. The pipeline that creates a 3D model is presented and each step is concisely described. In the rest of the book the main steps of this pipeline are analyzed and described in depth.

1.1 From real objects to digital models

A digital three-dimensional (3D) model is a numerical representation of a real object. Two large families of models can be identified: in volumetric models, the local properties all inside the object are represented, while in surface models the surface and the visual appearance of the object are reproduced. In this book, we will mainly focus on surface models. Such models can be displayed on a computer or smart-phone display, from any desired view point as a 2D image through perspective projection and rendering. It can also be sent to a processing module for further processing, like for instance in the Computerized Aided Machinery domain, where the model is used to automatically produce from the digital model a physical copy of the real object. The latter process is usually referred to as reverse engineering.

Applications based on the three-dimensional model processing are today growing more and more popular due to the increasing availability of three-dimensional graphic devices and the decreasing cost of the computational power. The popularity of 3D digital models arise from the possibility of processing them digitally: their visualization and modification and the display in real-time of the interaction of the model with other digital objects are among the key advantages of digital models with respect to physical copies.

Without claiming to be exhaustive, some typical applications that enjoy the use of 3D modeling are illustrated in the following.

- Archeology, architecture and sculpture —from one side— need to preserve artworks over time, while —from another side— exposure of the artwork to as

F. Bellocchio et al., *3D Surface Reconstruction: Multi-Scale Hierarchical Approaches*,
DOI 10.1007/978-1-4614-5632-2_1, © Springer Science+Business Media, LLC 2013

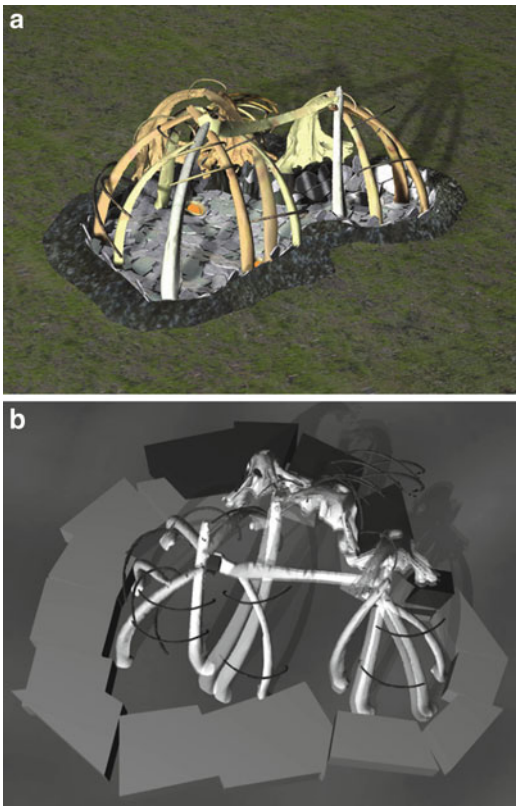

Fig. 1.1 Virtual archeology can be useful in experimental archeology for analyzing the structural properties of an artifact by simulation. In panel (**a**), a 3D model of an early phase of the construction of a whalebone house is shown. In panel (**b**), the model is used for a virtual structural analysis. The statics of the building is evaluated when new elements are added to the structure [151] (reproduced by permission of IEEE)

many people as possible is desirable to support knowledge dissemination and cultural promotion [131][194][151]. Virtual museums allow granting access to artworks to a larger public than real museums without any risk for the exposed objects and, at the same time, they can promote the real museum and attract visitors.

People interested in a single artwork have the possibility of exploring directly its virtual representation that can be accompanied by multi-media information. Furthermore, 3D models can improve both the study of the associated physical artwork and the accuracy of its cataloging and retrieval. Besides, physical simulations can be used to test virtually the properties of the artifacts that cannot be easily constructed. This can be used, for example, for verifying the plausibility of theories on the techniques and the materials that were used for erecting buildings in ancient times: the structural stability of the building can be evaluated in each of the construction phases (Fig. 1.1).

- Fashion design, production, and marketing may take advantage of the use of 3D models of the human body or its parts [68][74]. In the virtual fashion, a customer model can be acquired for accurately evaluating his/her size, as well as any

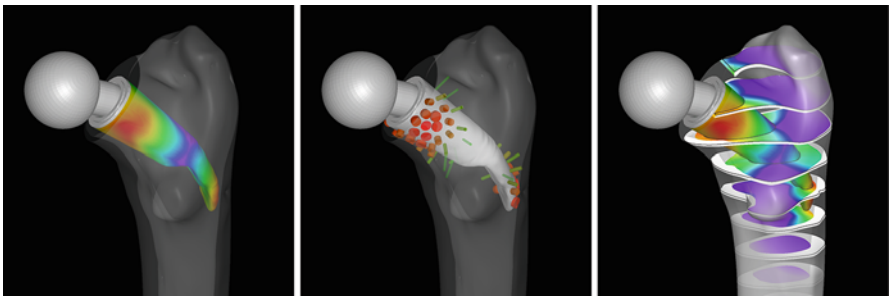

Fig. 1.2 Virtual surgery allows combining the 3D model of a patient with 3D models of tools and implants in order to properly plan surgery. In this figure, the distance between the implant and the surface of the patient's bone is represented in three different visualization modes [79] (reproduced by permission of IEEE)

specific body characteristics of interest. The digital model of the client can be dressed with any cloth item and used to show the fit of the item to the client. This can greatly enhance cloth tailoring by specifically taking into account the body peculiarities for customized solutions. Such systems have started to be deployed recently.

- Features can be identified on a model and their quantitative evaluation can be used to compare objects and classify them. This concept is applied in different contexts. For example, in the manufacturing industry it is applied to quality control [200], while in security it is used for person identification through biometric features [163].
- In medical applications, 3D models of human body parts and organs can offer a virtual view of the body to physicians for observation, diagnosis support, and therapy monitoring [69][247][199][107] (Fig. 1.2). Recently, computerized surgery has been also developed in virtual environments to optimize and tailor the procedure on specific patients with actual surgery performed by a robot guided by the clinician on the basis of 3D models [79][255][228].
- Virtual environments are very important for training to correctly execute critical tasks or dangerous procedures, in which the human error has to be minimized. Virtual surgery is one of these cases: doctors can practice on virtual patients and gain experience and confidence before performing the real surgery without having to resort to cadavers or animals [146]. This has also the advantage for the trainee that he can exercise ad libitum, with neither time constraint nor the constraint of being in a specific place for training. Another instance of virtual training was flight simulation, used both for training and evaluating pilots. Often, in these applications the system is complemented with haptic devices to enrich the virtual reality experience with proprioceptive feed-back [184][233] (Fig. 1.3).
- 3D models are extensively used in architectural design and reverse engineering [250][257]. The 3D models can be used as a powerful tool to show and discuss a project or an idea. They are more and more used also to design mechanical parts

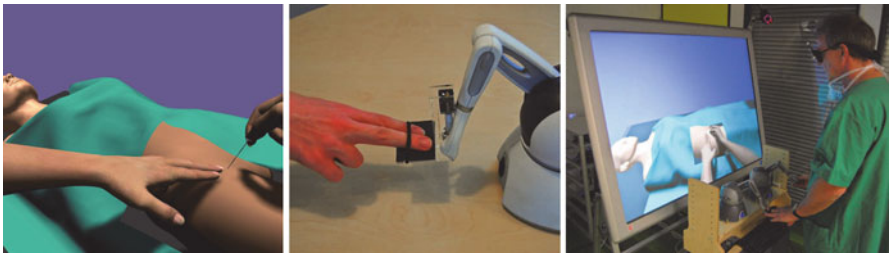

Fig. 1.3 Virtual reality environments can be used for training. In this figure, the training framework is constituted of two haptic devices and a display for providing the user a proprioceptive and a visual feed-back of his interaction with the virtual patient [233] (reproduced by permission of IEEE)

to optimize their interaction with the environment through the visualization of the interaction in real-time. This has been called visual computing. 3D models have a long history as a tool to be used as input to manufacturing in different domains. More recently devices to produce directly a 3D physical object from a 3D CAD model have become available and go under the name of 3D printers. Such devices create a physical object by laying down successive layers of particular material such as plaster, resins or liquid polymers [208].

- The entertainment industry is increasingly using opportunities offered by 3D modeling [115][128]. In the last decade the number of movies with 3D digital characters has grown. On the other hand, the use of the digital 3D model of an actor allows avoiding complex, expensive and time-consuming makeup sessions to create particular physical features, as well as the use of stunt-men in dangerous scenes [153][17] (Fig. 1.5). Similarly, accuracy and sophistication of many modern 3D video games rely on the extensive use of 3D models to create scenes and characters. Besides, many video games, like sport games, are nowadays inspired to real persons. 3D scanning can boost the realism of the game by significantly increasing the fidelity of the avatars avoiding the so called "uncanny valley" [212].

Different applications may have very different requirements. For example, reconstruction in virtual archeology needs a good accuracy and a low invasiveness, but generally scanning time is not an important constraint. On the contrary, in video-conference applications, real-time processing is mandatory, while the modeling quality plays a secondary role. Besides, some a-priori knowledge of the object to be modeled can be useful to achieve a more robust and fast reconstruction procedure. For instance, when the facial characteristics and mimics are of interest, the acquisition and reconstruction techniques can be based on face model warping to achieve better results [39]. Similarly, in industrial quality control the implementation of fast reconstruction with a low cost is important, since the same operations are repeated for many objects of the same type.

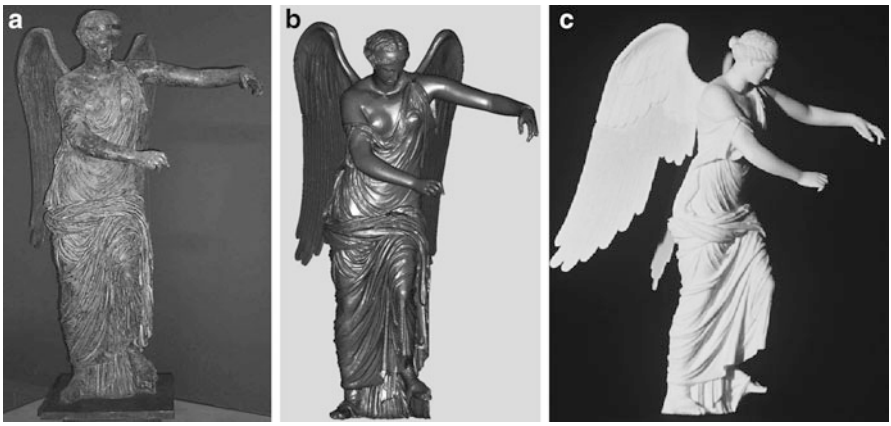

Fig. 1.4 3D printing is a complementary technology to 3D modeling. From a real object (**a**), the 3D model can be obtained (**b**) and reproduced by means of a 3D printer (**c**). The production of physical replicas (which can be realized also at a different scale) can be a cost-effective solution for small number production or when the intrinsic value or fragility of the original object discourage its physical accessibility [208] (reproduced by permission of IEEE)

Fig. 1.5 A digital 3D actor can exhibit a variegated range of expressions. The mimic posture can in fact be synthesized from a neutral face (leftmost) after a suitable parametrization of the expression of the face has been carried out. This information is then used to displace the vertices of the mesh associated to the neutral face obtaining the desired expression [153] (reproduced by permission of IEEE)

Simple objects can be represented as models by means of simple equations: for instance, the equation $x^2 + y^2 + z^2 = r^2$ can be used for representing a sphere centered in the origin, with radius r. The Constructive Solid Geometry (CSG) has been introduced to create more complex objects by combining simple solid objects (e.g., cubes, cones, spheres) by means of union, intersection, and difference operators (Fig. 1.6). For instance, a tube can be seen as the difference between two cylinders with different radius. Unfortunately, this method is not suitable to describe a large class of complex real objects, especially when the surface is very irregular as in Fig. 1.4, and therefore its use is limited to interactive, Computer Aided Design (CAD) modeling.

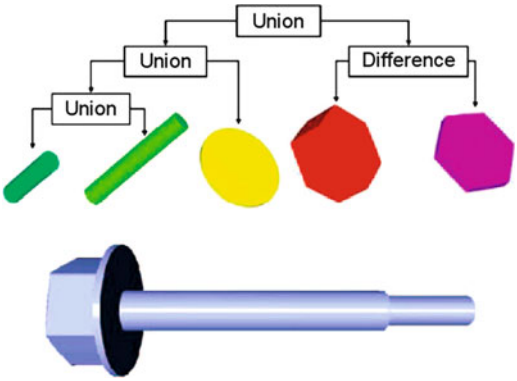

Fig. 1.6 Constructive Solid Geometry (CSG) is a powerful method for describing the shape of an object in terms of boolean operations on primitive solids [101]. It is particularly useful for CAD modeling of mechanical parts (reproduced by permission of IEEE). A large class of complex real objects has not a regular shape and CGS can fail in providing an accurate description of their surface. For this kind of objects, 3D scanning is a viable method for obtaining their 3D model. This is the case, for instance, of the generation of statue models, such as the application described in Fig. 1.4). Among this kind of applications, the Digital Michelangelo project [150][4] is a prominent example, due to the importance of the digitized statues, and the methodology and the tools developed for achieving the results

More complex 3D digital models can be created in two different ways: they can be produced by drawing their surface using advanced functionalities of CAD systems or by measuring directly the surface of the physical object through scanning.

Nowadays, CAD systems operators are able of creating quite complex models by using two main kinds of tools: NURBS and subdivision surfaces. Non-Uniform Rational B-Splines (NURBS, [186]) are a mathematical model that allows generating curves and surfaces with great flexibility and precision. The NURBS are suitable for handling both analytic and free-form shapes. The local aspect of the curve is defined intuitively inserting and displacing a set of points that have a local influence, called control points (Fig. 1.7). NURBS constitute therefore an interactive framework to create models. Instead, subdivision surfaces [60] is a technique to create smooth surfaces as the result of an iterating process that starts from a coarse definition of a polygonal mesh. The smooth surface is computed recursively partitioning the initial polygonal mesh: for each step the polygons of the mesh are subdivided in new polygons by adding vertices that may lie outside the original surface. Depending on the subdivision rule, the new mesh can implement a smoother version of the mesh of the previous step (Fig. 1.8).

Scanning is a process that allows obtaining the 3D model in a fully automatic or semi-automatic way by measuring the geometric features of the object as well as its visual appearance (e.g., color and texture) and, then, by identifying the most appropriate surface digital model that represents the measured data.

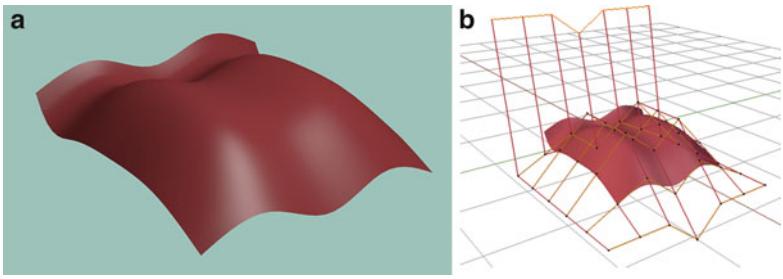

Fig. 1.7 NURBS are a common surface representation paradigm used in CAD tools. The shape of the surface in panel (**a**) is controlled through a polygonal mesh, whose vertices are called control points, shown in panel (**b**); each control point affects the shape of the surface locally proportionally to the value of its weight. The reported images have been obtained using the 3D modeling software suite Blender [2]

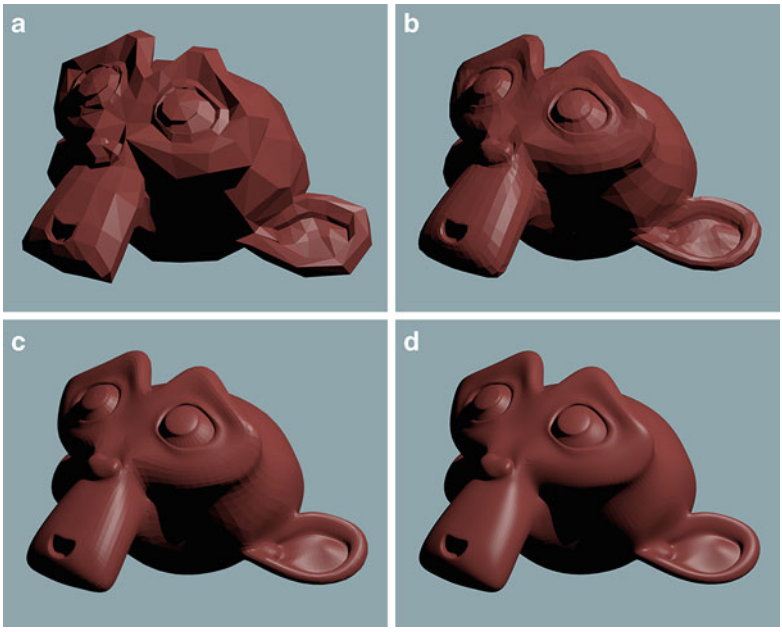

Fig. 1.8 Subdivision surfaces are a framework for making denser a polygonal mesh. When a suitable subdivision rule is used, the resulting mesh becomes smoother and smoother after each iteration. The results of the application of three successive subdivision iterations on the mesh in panel (**a**) are reported in panel (**b**), (**c**), and (**d**), respectively. The reported images have been obtained using the 3D modeling software suite Blender [2]

In the CAD systems the human operator shapes interactively the surface model to represent a given object, while 3D scanners are aimed to produce the digital representation of the surface automatically. Scanners are generally faster than using

a CAD system and achieve a higher (or, at least, quantifiable) level of accuracy of the scanner object. Furthermore, scanning, being essentially a measurement process, does not require any artistic ability to the operator. On the other hand, the creation of a digital model through scanning requires that the model and the measuring system are available on the site. This is the scenario considered inside this book.

1.2 The scanner pipeline

The acquisition process of the surface geometry of a real object and of its visual appearance is called *3D scanning* and the system used to this aim is called *3D scanner*.

3D Scanners can have very different characteristics because their applications are very variegated. They are characterized by different performances and the acquisitions techniques used are based on different physics principles. However, despite of these differences, they all a common pipeline to construct a 3D digital model. In fact 3D model construction is a modular process composed of different sequential steps (Fig. 1.9).

The sensors used in 3D scanner sample the object in a set of points acquiring therefore a finite set of information, such as 3D coordinates of points belonging to the object surface, color fields, and textures. Starting from this information a

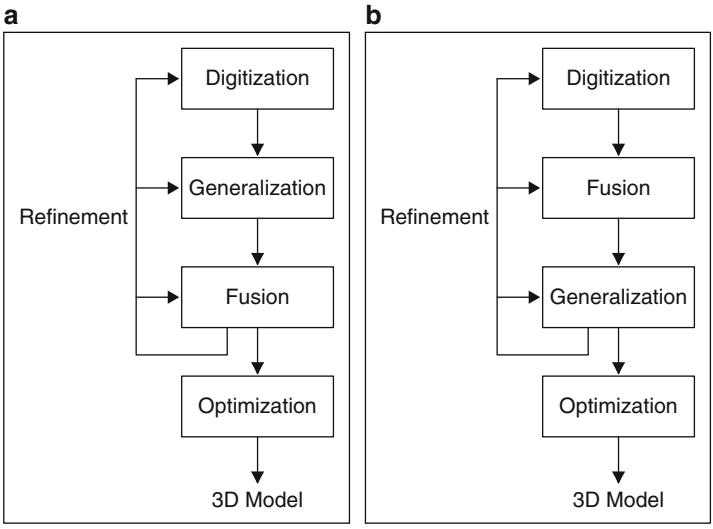

Fig. 1.9 The scheme shows the main steps of the 3D objects digitization process. In panel (**a**) a model of the object is built first for each acquired dataset; the models are fused afterward. In panel (**b**) fusion occurs directly on the raw data points and a single model is obtained from the ensemble of the whole fused data points

3D scanner, through adequate processing, obtains a continuous description of the surface of the object in terms of geometry and visual appearance. This process, in statistical terms, is also known as generalization. Moreover, as the acquisition is realized by means of a sequence of measurements, measured features are affected by measurement error that has to be removed from the reconstructed digital surface. Hence, the reconstruction procedure has to consider strategies for data noise filtering.

In the reconstruction of complex shapes, which cannot be completely surveyed by a single point of view, many sensors have to be used or many acquisitions have to be performed, each producing data on part of the object surface. Information coming from different sensors (or from different acquisitions) has to be consistently fused to come up with a single digital description. This implies two processing steps: registration, where the data are referred to the same reference frame, and merging, where the information from the different sensors are combined.

Depending on the kind of information collected by the sensors, the phases of surface construction and of fusion of partial surfaces can be swapped (Fig. 1.9). In particular, in Fig. 1.9a the fusion is operated between partial surfaces, while in process schematized in Fig. 1.9b the raw data coming from different acquisition sessions is fused together before the surface reconstruction step takes place.

Finally, after the computation of a *good* digital representation of the object, it is possible to transform the model in a format better suited to the final application. For instance, it can be compressed in order to decrease the size of the model.

In Fig. 1.9 a high level scheme of two possible 3D model construction pipelines is reported:

- A set of sensors is used to capture data on the geometry of the surface and of its color and reflectance.
- From these data, a generalization of the acquired data over the whole object surface is computed.
- The information from different sensors or from different acquisition sessions is merged.
- The previous steps can be iterated in order to improve the quality of model (e.g., for obtaining a denser set of data in some regions of the object).
- The 3D digital representation is transformed and optimized using a better paradigm for a specific application.

Each of the above steps can be improved exploiting a priori knowledge about the scene, about the acquisition techniques or about the object to be scanned. Exploiting a priori knowledge allows an improvement of the final 3D model both in terms of quality and computational complexity. Techniques developed for general cases can be improved and customized for particular objects.

In the following, the single steps which constitute the reconstruction of a 3D digital model are now described in depth and the approaches for implementing them are discussed.

1.2.1 Data acquisition

As the physical objects can be very variegated, from a large landscape to a small coin, there are many different systems that can be used for this scope. Basically these systems work measuring the reflection of some kind of radiation produced by the object surface. The output of this step is generally a set of 3D points sampled on the object surface, sometimes enriched with surface color and reflectance information. The ensemble of the 3D points is often referred to as *points cloud*. In the next chapter, a survey on the techniques used for collecting the visual information of an object is presented.

1.2.2 Generalization

The reconstruction of a surface from a three-dimensional points cloud requires the implicit definition of the surface topology.

Let us consider the surface of a human hand to clarify the difference between geometry and topology and the impact of the topology on the reconstruction problem. Points placed on the tip of two fingers of the same hand can be geometrically very close but topologically very far from each other as their topological distance is equal to the sum of fingers length. In fact, the geometric (Euclidean) distance is defined as the shortest line segment that connects the two points while the topological distance is defined as the shortest path connecting the two points, all contained inside the surface. Topological and geometrical distances can be very different. Any procedure based on proximity, like the computation of spatial gradient, may come up with very wrong results if the geometrical distance is considered instead of the topological one. For this reason, the a-priori knowledge on the topology of the object surface can simplify the reconstruction problem. In general case, in which topology is not available a-priori, computation has to be based on an Euclidean geometrical framework. In this case, the sampling density should be high enough so that the sampled surface can be considered locally linear.

Alternatively, the lack of topological information can be handled in two phases. In the first phase, a polygonal model that represents the data at a low resolution is determined. This model can be computed for instance by means of a local planar approximation and it can be used as topological reference [124][123]. In the second phase the surface is constructed defining it as a displacement with respect to the polygonal model computed in the first phase, that acts therefore as a base for the 3D surface.

The problem of surface reconstruction from points clouds has been largely studied in the last years. A detailed taxonomy of the reconstruction techniques can be found in [164], where the methods are classified into two main classes:

- *volumetric techniques*, which are aimed at identifying the volume in space occupied by the object;
- *surface fitting techniques*, which are aimed at identifying the 2D manifold associated to the object surface, inside the 3D space.

Another feature that characterizes the reconstruction techniques is the way in which measurement error is addressed. Simple interpolation of sampled data does not produce good results due to measuring error. However, interpolation can be used as a first approximation that can be improved in next processing phases.

Particular attention is given throughout this book to the effect of the measurement error on the reconstructed surface. This will be described in depth in the Chaps. from 3 to 6.

1.2.3 Integration of multiple scans

When an object is not observable from a single sensor or a very fine detail is needed with respect to the object dimension, a data *integration* step is necessary. In fact, in this case, scanning has to be performed from different points of view, and the information captured from the different views has to be integrated in order to calculate a single representation. This operation can be divided in two steps:

- *registration*, where the different data are represented in the same reference frame;
- *fusion*, where a single model is generated combining the data from different views.

Multi-view scanning can be achieved by acquiring the object in same acquisition session using multiple scanning devices each aimed at part of the object, or by scheduling multiple acquisition sessions, where the additional acquisition sessions will be aimed to acquire data in the objects regions in which the reconstruction quality was too poor.

1.2.3.1 Registration

The *registration* is an operation used in many fields. Whenever the data, coming from different sources of information, have to be related to the same reference system, a registration stage has to be performed. We remark here that registration is required also when the matching of different models of the same object have to be performed. In this latter case, models have to be scaled and aligned parallel to each other to allow proper comparison.

The registration of data from different views corresponds to the inversion of the function that describes the relative motion of the object with respect to the sensor in the two views. This process can be simplified if the motion is known like when the object is placed on a turn table, or the absolute position of the sensors is available.

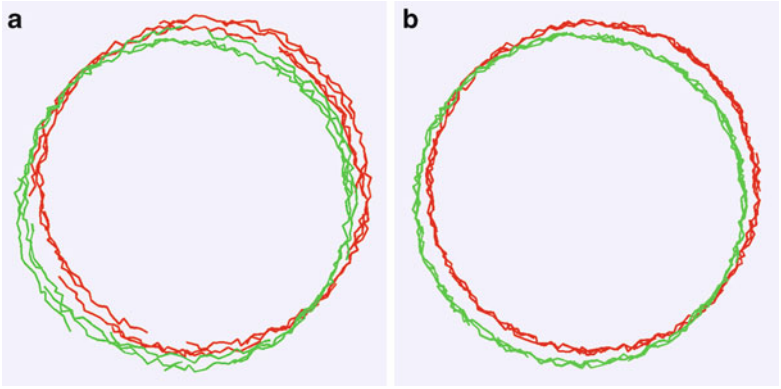

Fig. 1.10 In panel (**a**) the data before the registration. If the registration procedure does not accurately take into account the distribution of the registration error, as proposed in [193], cliques of surfaces can be generated as an effect of the minimization of the local registration error on partially overlapping scans, as shown in panel (**b**). The effect of having two surface cliques is evident

It is also simplified if landmarks are surveyed in the different acquisitions. If there are not landmarks or sensor position information, some features of the object have to be found and matched in the different views to compute the transformation that relates the different acquisitions. As, generally, a common overlapping region for all the views is not available, some views can be accurately registered only with respect some views, but poorly registered with respect to other views (Fig. 1.10). The formation of cliques of views can introduce registration error which results in abrupt spatial or features variations in the fused dataset [193]. Hence, the registration has to solve mainly two kinds of problems:

- a good estimate of the object features;
- a uniform distribution of the registration error.

The algorithm known as *Iterative Closest Point* (ICP) [35] is one of the most used to realize the registration. The algorithm is based on the definition of a distance function among points belonging of the different views. For a set of points, a rototranslation is operated such that the distance among the transformed set of points and the closest corresponding points of a reference surface is minimized. The registration of the different views is obtained iterating this procedure. Since the ICP suffers of local minimum problem, it needs a good initialization and wide overlapping regions among the data that have to be registered for obtaining a satisfactory registration. The effectiveness of ICP has been proved for *every* surface representation modality [35].

In [232], a modified version of ICP is proposed that is based on first fitting a polygonal surface to the point clouds associated to the two views that have to be registered. The vertices are weighted with a value that represents the reliability of its measure. Moreover the registration is guided by the distance of the vertices of one mesh from the faces of the other mesh.

Generally, each acquisition device has an error distribution which depends on the features of the acquired object. For example, the measurement error on the data in the regions close to the object's border is, typically, greater than that in the central regions parallel to the sensor. This information, acquisition and device dependent, can be exploited in the registration procedure to weight the reliability of the points.

The ICP is applied just to the vertices that have a correspondence in the other mesh (i.e., they project over a face of the other mesh) and with a distance smaller than a given threshold. The registration is performed, initially, using a low resolution representation of the surface (that is obtained by means of downsampling the data) in order to increase the speed of convergence of the alignment. The registration of multiple views is computed choosing a reference view and aligning the other views to this one.

In [252] the registration is performed in two steps from the surfaces built for the different views. In the first phase, for each view, the points with high value of gradient are selected. The hypothesis is that the polygonal curves that connect these points have robust features. In the second phase the registration transform (rototranslation) is computed aligning these curves.

1.2.3.2 Fusion

Even after registration, the surfaces belonging to different views may not overlap perfectly, due calibration and reconstruction errors that adds to the measurement error. Moreover, the two surfaces have to be fused so that their description does not become denser in the overlapping region.

The simple unification of the registered data or the partial surfaces would produce a poor representation of the object surface, with artifacts in the overlapping regions. The aim of the fusion procedure is to come up with a single model, improving the representation of the surface in the overlapping region (Fig. 1.11).

The information on the reliability of the sampled data in the two views can be used also inside the fusion process to privilege the data from one of the two views. For instance, in [81], where each view is a *range image*, each pixel of the view is weighted with its distance from the closest border. The range image are registered and a point cloud is obtained from merging them. The position of each point is computed using a weighted average of the pixels of the multiple views. Since the border pixels are not very reliable, their position in the point cloud is estimated using pixels from other (overlapping) range images, resulting in a smooth mapping of those points in the final dataset.

The procedure in [232] fuses the meshes from two views in two steps. In the first step, the two meshes are joined together re-triangulated the vertices that fall inside the overlapping region. In the first step, the parts of the meshes in the overlapping regions are substituted by triangulating the vertices in the overlapping regions.

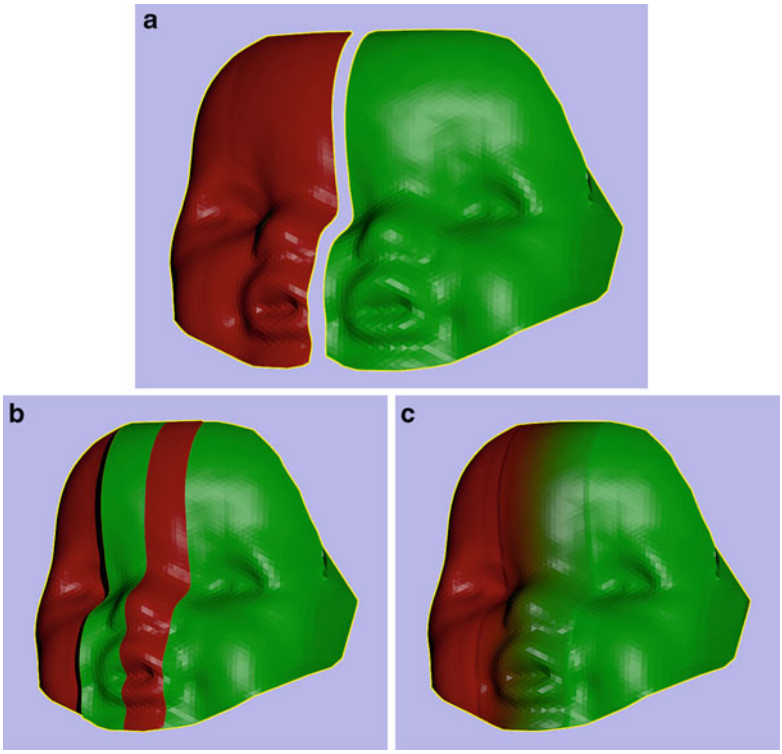

Fig. 1.11 Two partial models (**a**) are registered (**b**) and merged (**c**)

Since this procedure can create small triangles in the overlapping regions, these triangles and all the vertices that do not belong to any other triangle are eliminated and the resulting holes are filled by re-triangulating the vertices on the hole border.

More recently, an interesting approach to fusion is presented in [176]. The integration of the data from multiple views is here performed using the wavelet transform: the original data are decomposed through wavelet transform into low and high frequency components, then the components are separately integrated and finally are composed back.

1.2.4 Optimization

A 3D model can be used in different applications. Depending on the characteristics of the final application, a post-processing of the model can be required. For example, virtual reality applications usually requires models composed of a number of polygons as small as possible, while if the model has to be used in a movie it can be necessary to modify the model for a better inclusion inside a scene or to add handles for digital interaction.

The transformation can be simplified by the use of particular paradigms chosen for the representation of the model. The most common processing procedures applied to a model are:

- conversion
- compression
- watermarking
- animation
- morphing

Conversion of a model is applied to obtain a representation of the information in a format suitable for processing in other applications. A common conversion operation is the triangulation of the model's surface [118] that produces a model constituted of triangular elements that can be directly rendered by graphics cards. An interesting aspect of the conversion is the re-parameterization. In [145] a multiresolution method for the parameterization of a triangular mesh is presented. The original mesh is simplified (in $O(N)$ levels where N is the number of vertices) until a mesh with few polygons is obtained. This low resolution mesh is used as base for the parameterization. The resulting hierarchical description allows a re-computation of a mesh in which the distribution of the triangles can be made more regular or the mesh connectivity has some properties defined a-priori.

The aim of *3D compression* is the description of large and complex models using a limited budget of resources. This is the case, for instance, of models that have to be transmitted through the network or used in the background of Virtual Reality environments or games. The compression can be realized using techniques developed for signal compression: wavelets, entropy coding, and predictive coding have been exploited. Other approaches take explicitly advantage of the properties of 3D models, as: Edgebreaker [202], Subdivision Surfaces [170], and triangle strips [87].

Techniques for 3D compression can be classified into three categories:

- Mesh-based approaches, that realize compression considering the topological relationship among the triangles of the polygonal mesh representing the object's surface (e.g., Edgebreaker);
- Progressive and hierarchical approaches, that transmit a base mesh and a series of refinements (e.g., Compressed Progressive Meshes, subdivision-based approaches and Compressed Normal Meshes);
- Image-based approaches, that encode not an object but a set of its pictures (e.g., QuicktimeVR and IPIX).

In general, compression techniques try to minimize the loss of perceptible information, given a certain amount of resources available (i.e., the maximum number of triangles).

Watermarking [192] is a procedure correlated to compression: it introduces information that is not perceptible by the user of the model, but that can be detected by adequate processing and therefore it constitutes a signature of the digital model. Each model can then be marked with a code and the information can be used to

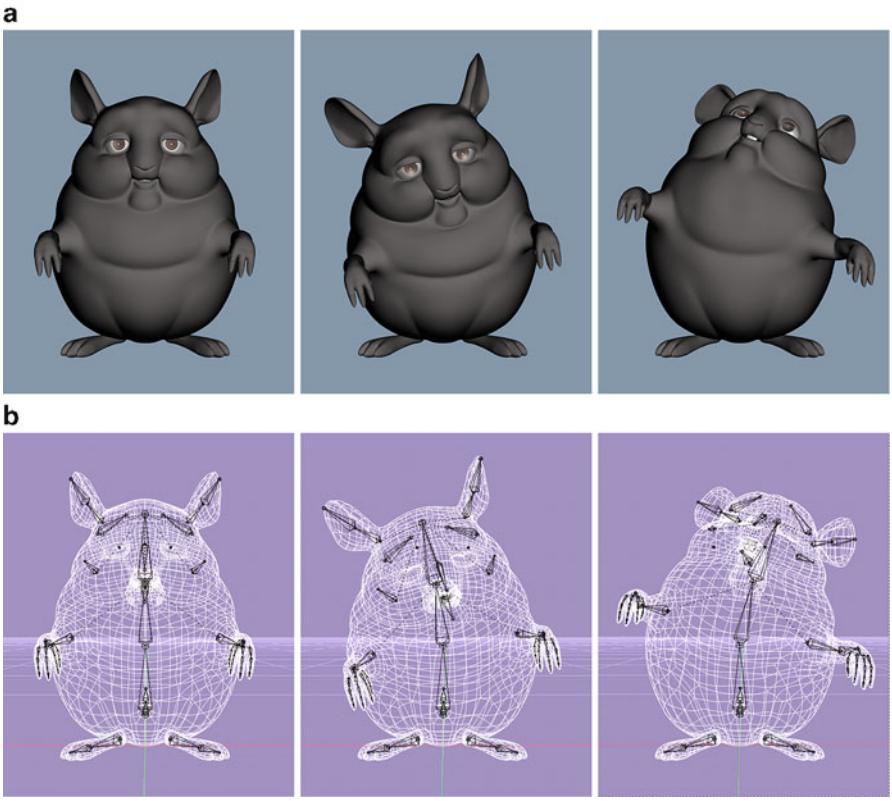

Fig. 1.12 The poses of a chinchilla 3D model shown in panel (**a**) are obtained modifying the mesh through a skeleton, reported in panel (**b**). These images have been obtained using the 3D modeling software suite Blender [2], while the model has been realized in the Peach Project from Blender.Org [1]

recognize it, for example for copyright reasons, without degrade the geometry and the appearance of the model. Only a particular procedure allows the extraction of the marker, allowing the recognition of the model.

The most critical feature of a *watermarking* procedure is its robustness: it has to resist to the common editing operations carried out on digital models, such as scaling, resampling and segmentation.

Animation of a 3D model is obtained associating a motion law (as a function of time) to the surface points (Fig. 1.12b). However, in practice, specifying the motion law for each point of a model is not feasible and a simplified solution is adopted. This is based on the creation of a structure, called skeleton, which is constituted of a set of links connected by joints. The skeleton is associated to the surface through a procedure called rigging; each point of the surface is associated to one (or more than one) joints of the skeleton, possibly with a different weight, and animation is obtained trough rotation of the joints.

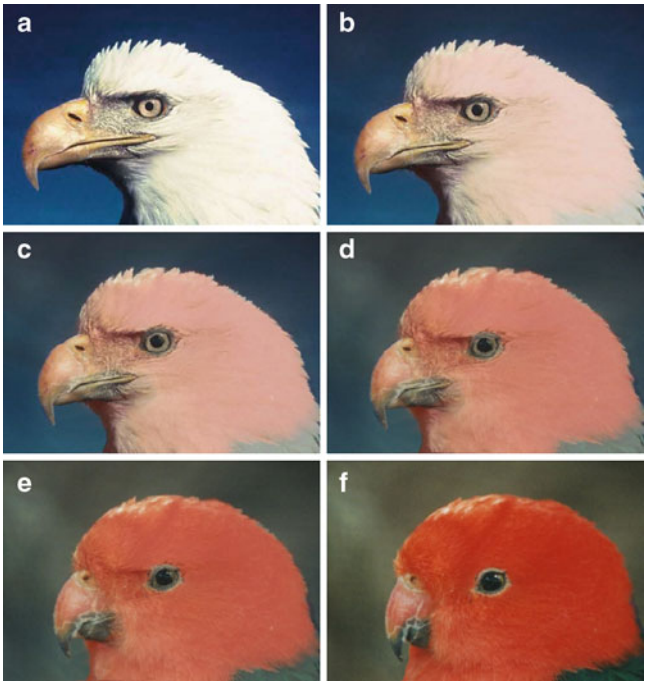

Fig. 1.13 Panels from (**a**) to (**f**) shows some intermediate steps of morphing an eagle model into a parrot one using the software made available at [11]. Note the simultaneous change in geometry and color fields

The skeleton can be substituted by a low resolution surface that can be directly animated as in [145][143]. Such models can be obtained simplifying a high resolution mesh and therefore the correspondences of this mesh with the original one is automatically found.

Morphing is the operation that is used to transform the geometry and visual appearance of an object into that of another object (Fig. 1.13). Morphing is realized by the computation of a smooth time function that relates some features of the two different models, these can be, for instance, the position of corresponding points and their color value [144]. To obtain good results, features characteristic of the two models have to be correctly matched before morphing. For instance, for the face morphing, the nose of the original face should become the nose of the second face. The presence of a hierarchical structure for the representation can simplify morphing. In fact, as the most representative features of the objects should be present at the large scales, the morphing transformation can be performed using mainly the few features present in the layers at low resolution.

In the following we will analyze in depth the process of building a model, starting from its acquisition and set-up procedures.

1.3 Using a multiresolution hierarchical paradigm

In the previous section, the procedure to obtain a 3D model through 3D scanners has been described as a pipeline of different tasks. In each processing stage, both the computational cost and the quality of the output can be affected by the characteristics and the format used for data representation.

Multi-scale hierarchical paradigms are a class of data representation paradigms that enable the description of a surface at different level of detail (LOD). This feature can be exploited to effectively reduce the computational cost of the above described procedure and to increase its robustness [42]. The main properties of these paradigms are multiresolution and spatial locality. Multiresolution allows describing the properties of the surface at different levels of detail. In this sense, it is equivalent to locality in the frequency domain. The spatial locality provides a tools to limit the use of computational resources allocating them to process only the data where the information is effectively located.

Several tasks in the 3D scanning pipeline can take advantage of these characteristics. This is evident for generalization: spatial locality can be exploited to reduce the computational effort, since the computation the 3D model can be carried out operating only on local subsets of the data. On the other hand, multiresolution allows choosing the model such that the residual, that is the difference between such model and the measured data, is as close as possible to the noise distribution or the accuracy degree required by the specifications.

In general, when a hierarchical structure is used to represent a surface, the computational overhead required to compute and store the information is paid back by the savings in using this information (e.g., for accessing the data). Usually, multiresolution and spatial locality enable to decorrelate the information at different level of detail, providing a spatial-frequency characterization of the information. This property can be exploited for locating meaningful regions and for computing features on which registration and morphing are based. Moreover, a multiresolution representation of the partial surfaces allows implementing the integration process on a LOD basis, finding the registration and fusion transformations by means of successive approximations. Since a low resolution model describes only the large scale characteristics of the partial surfaces, a robust estimate of the registration between partial meshes can be carried out. This allows selecting a pool of candidate transformations that, although possibly characterized by a low accuracy, can be refined (or discarded) by using the details of the partial models. The above described searching strategy can be iterated on the different resolutions, incorporating the information of one more level of detail at each iteration. It is worth noting also that introducing the details information only in the refinement steps, besides saving some computational efforts, diminishes the risk of being trapped in local minima. Besides this, also the computational cost of the fusion process can be positively affected by the spatial locality property: this process, in fact, involves only the overlapping

regions of the two partial models, described by a proper subset of the parameters of the models. These are the only data that have to be processed for computing the parameters of the fused model.

Finally, multiresolution representation can be used for incorporating topology information, since a low detailed model usually approximates well the topology of the surface.

Chapter 2
Scanner systems

Generally, the first step in the creation of a 3D model consists in capturing the geometric and color information of the physical object. Objects can be as small as coins or as large as buildings, they can be still or move while scanning, and this has prompted the development of very different technologies and instruments. The aim of this chapter is to present such technologies to explain the techniques on which 3D scanners are based. Comparison in terms of accuracy, speed and applicability is reported, in order to understand advantages and disadvantages of the different approaches. How to use the information captured to compute the 3D model will be discussed in the next chapters.

2.1 3D Scanner systems

A 3D scanner is composed of a set of devices and procedures able to capture the shape and the appearance of an object [30]. Generally, scanners are based on sampling the object surface. Data collected are used in the reconstruction phase to obtain the object digital model.

A 3D Scanner system is, basically, a measuring instrument and therefore it can be evaluated in terms of:

Accuracy The accuracy is an index that describes how close the measurements provided are to their *true value*. Different techniques can be used to determine such index. They can be classified in direct measurement (e.g., using a lattice of known step) and indirect (e.g., average distance of sampled points on a plane from the optimal surface).

Resolution The resolution measures the details density that the system can recover, or in similar way, the minimum distance that two features should have to be discernible.

Speed The speed of an acquisition system is evaluated as the time required for measuring a given feature (for example, the points per second that can be sampled). Obviously, the kind of features measured is equally important.

F. Bellocchio et al., *3D Surface Reconstruction: Multi-Scale Hierarchical Approaches*,
DOI 10.1007/978-1-4614-5632-2_2,

Flexibility The flexibility of a system is the capacity of acquiring a wide class of objects (differing in many aspects like materials and dimensions). It depends on the kind of sensors used and the size of the acquisition field.

Invasiveness The invasiveness is the effect that the acquisition procedure has on the scanned object. The measurement can modify the object in different ways, typically incident light may alter the object surface or surface mechanical contact may damage a fragile object. The latter is a strong constraint for certain classes of objects like archaeological artifacts.

Robustness The robustness of a system describes the sensitivity of the system to the environmental conditions.

Usability The usability of a system describes the technical know-how needed to the user for a correct use of the system.

Cost There are acquisition systems of very different prices. The hardware and the software used by the system determine its cost.

Some of these properties can be easily quantified while others can be evaluated only qualitatively. Moreover, some properties depend on some of the system components. For example, the physical principle used for obtaining the measurement determines the invasiveness of the system. Some features can be incompatible. For instance, in general, a high degree of usability cannot coexist with a high level of flexibility and a high level of accuracy, as both the last two properties typically require a complex system, in which users with a significant degree of expertise are needed.

The physical principle exploited by the system for measuring the object geometrical features is probably the most important characteristic of a 3D scanner, and it can be used to categorize the different systems, as sketched in Fig. 2.1.

The 3D scanning systems use, mainly, two families of acquisition techniques. The first one is based on the interaction between a sensor and the surface of the object; the systems using this approach are called contact 3D scanner. The second one is based on the interaction between a radiation (electromagnetic or sound) and the surface of the object, in this case the systems are called non-contact 3D scanners. In the following these two categories will be illustrated in depth.

2.2 Contact 3D Scanners

In contact 3D scanners the surface of the object is sampled by physically touching it. There are mainly two types of these systems: Coordinate Measuring Machines (CMM) (Fig. 2.2) and Jointed Arm systems.

The first is composed of a tactile probe attached to a vertical arm, which can be moved along the horizontal plane. Motion is produced by translation along three orthogonal axes. The measurement is the direct result of the actuator displacement along each axis and the object has to be placed over a reference plate where the probe can explore it. The movement of the probe can be both automatically and manually operated.

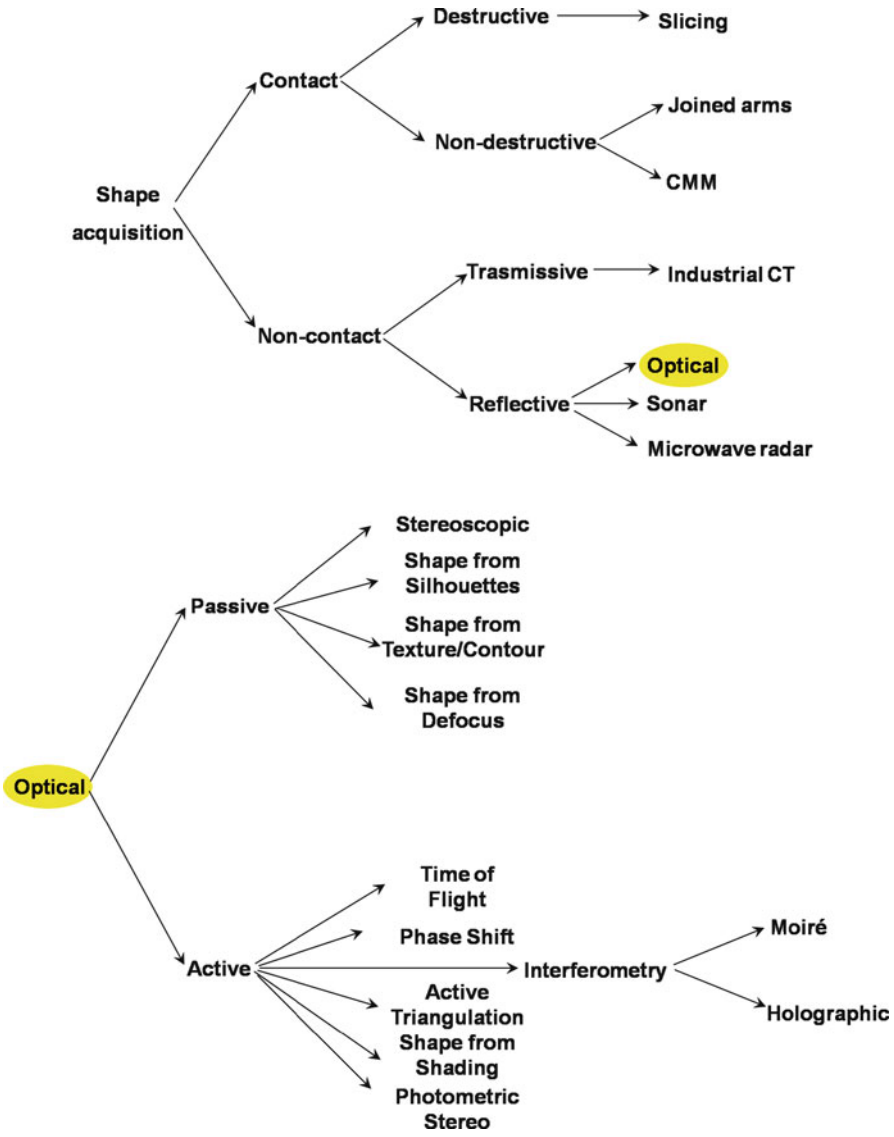

Fig. 2.1 A 3D scanner taxonomy based on the physical principle exploited by the system for measuring the object geometrical features

Generally, these systems enjoy a good accuracy (e.g., the Helmel Checkmaster 112-102 scanner system [3] has an accuracy of 8.6 μm) and they are used mostly in manufacturing. They have two main disadvantages: the working volume is bounded by the structure and sampling is carried out only along the vertical direction.

Fig. 2.2 Coordinate Measuring Machines: the Helmel Checkmaster 112-102 scanner system has a working volume of 300 $\times$ 300 $\times$ 250 mm^3 and an accuracy of 8.6 μm [3] (reproduced by permission of Helmel Engineering Products, Inc., Niagara Falls, NY, USA)

The Jointed Arm is composed of a chain of articulated links with a probe of the end effector. The 3D position of the probe results of the composition of the rototranslation of each of the supporting links. Since, as for the CMM, the position of the probe is computed by mechanical components, these systems are generally sensitive to temperature and humidity variations. For that reason, in order to provide good performance, they require a high quality electronical and mechanical design and manufacturing.

The main difference between CMM and Jointed arm scanners is in their mechanical structure. Since generally the arms have more degrees of freedom, they can be used for a larger class of objects. The arms are typically manually operated, while the CMM can be more easily automated. Both these devices can be very precise (Cam2 Quantum Arm has an accuracy of 0.018 mm [6]), but they are relatively slow compared to the other scanner systems. Furthermore, these methods are invasive and so they are not suitable for fragile objects (e.g. archaeological artifacts). Another disadvantage is the price, as these systems are generally expensive. It is worth noting

that the tactile probe of both these systems could be substituted with another kind of sensor, for instance an ultrasound sensor, to realize non-contact measurement. In this case the system would become a non-contact 3D scanner.

2.3 Non-contact 3D Scanners

In non-contact systems, the sampling of the surface is performed by the interaction between some kind of radiation and the object surface itself. Depending on whether the radiation is supposed to pass through the object or it is reflected by the object surface, these systems can be divided in two sub-categories: transmissive and reflective.

Transmissive Systems - Industrial Computed Tomography

In transmissive systems the object has to be positioned between the emitter (which irradiates the object) and receiver (which collects the radiation attenuated by the object). The main representative of this category is the Industrial Computed Tomography. The radiation is generated by an X-ray tube by means the collision of electrons with particular materials, usually tungsten, molybdenum or copper. The X-ray photons emitted by the collision penetrate the target object, and are captured by a 2D detector as a digital radiographic image. The density and the thickness of the object affect the energy collected by the X-rays detector and a 3D model can be reconstructed from a set of 2D X-ray images of the object taken from different views. The views can be obtained either rotating the object and fixing the source-sensor pair (for example, positioning the object on a turn table) or fixing the object and rotating the source-sensor pair (for example, in medical scanners, where the system revolves around the patient). From this set of 2D radiographs, a 3D model, generally defined as an ensemble of voxels, can be reconstructed [88], [107].

The three-dimensional resolution of the obtained model ranges from a few micrometers to hundreds of micrometers, and depends on X-ray detector pixel size and the set-up. This kind of systems allows the reconstruction of the internal structure of the object and the method is largely unaffected by surface characteristics. It is worth noting that both the density and the thickness of the object affect the energy collected by the X-rays detector. Furthermore tomographic reconstruction is computationally intensive.

Reflective Systems

The reflective systems exploit the radiation reflected by the object surface for estimating the position of the points on the surface. They can be classified upon the type of radiation used. In particular, optical systems use optical radiation

(wavelength between 100 nm and 300 μm), while non-optical systems use other types of waves to make the measurements. Since optical systems form the main category of 3D scanners, they will be considered in depth in the next section.

The class of non-optical systems is constituted of devices based on radar or sonar. Although the type of waves used is very different (radars use electromagnetic microwaves while sonars use sound or ultrasound waves), both are based on measuring the time-of-flight of the emitted radiation. The distance covered by the radiation, which is assumed equal to double the distance of the object from the scanning device, can be computed from the time required for the wave to reach the object and return to the system and from the known speed of the wave. As this principle is used also for a class of optical scanners, it will be better explained in the next section.

Radar systems have a very large depth of field and can measure the surface of objects up to 400 km far away. Moreover they can perform ground penetrating reconstructions [177][59].

A typical application is for air defense and air traffic control. These systems are quite expensive and generally have a low accuracy.

Sonic waves are particularly advantageous underwater, where optical radiation would be distorted or attenuated too much. Nevertheless, these systems are characterized by a low accuracy due to low signal to noise ratio.

2.3.1 Optical 3D Scanners

The ability of reconstructing an object without physically touching it has important advantages: it is applicable to fragile objects and electromagnetic radiation allows high speeds of data acquisition and reconstruction of very large objects like landscapes or buildings. Furthermore, very low cost systems can be realized.

Depending on the source of the radiation, these systems can be divided in two sub-categories: passive and active systems.

2.3.1.1 Passive Optical Scanners

Passive scanners do not emit any kind of radiation by themselves as they measure the radiation reflected by the objects surface. Generally, they are based on Charge-Coupled Devices (CCD) sensors employed by commercial digital cameras. The camera collects images of the scene, eventually from different points of view or with different lens arrangements. Then, the images are analyzed to compute the 3D coordinates of some points in the scene.

Passive scanner can be very cheap as they do not need particular hardware. However, they are generally not able to yield dense and highly accurate data. Moreover, they require a large computational effort to produce a set of 3D points sampled over the surface.

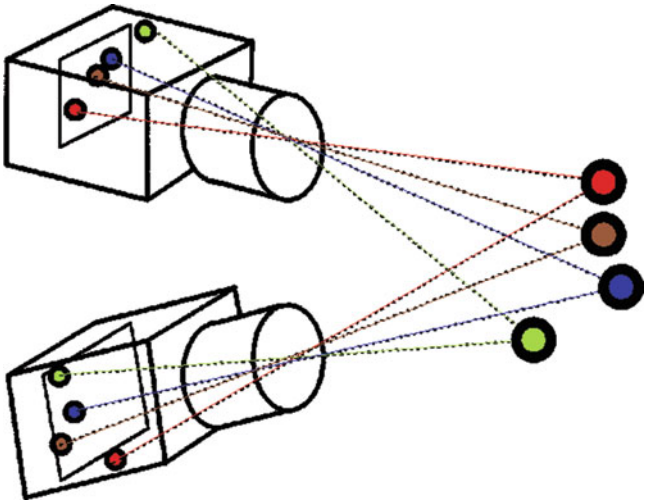

Fig. 2.3 The intersection of the projected rays gives the 3D position of the points

Although they share the same sensor technology, different families of passive optical scanner can be found in the literature [246], which are characterized by the principle used to estimate the surface points: stereoscopic, silhouettes, texture (or contour) and defocusing.

Stereoscopic systems

The stereoscopic systems are based on the analysis of two (or more) images of the same scene, seen from different points of view. The 3D points of the scene are captured by each camera through their 2D projection in the taken images. The first task of the reconstruction algorithm consists in identifying the 2D points in different images that correspond to the same 3D point in the scene (correspondence problem). From the 2D points their associated 3D point can be determined as the intersection of the retro-projected rays (Figs. 2.3 and 2.4). This reconstruction method is known as triangulation. It is worth noting that this method requires complete knowledge of the set-up parameters: the (relative) position and orientation of the cameras, as well as of the cameras internal parameters (focal length, optical center, pixel size and distortion parameters). These parameters are determined during a phase called calibration. Generally, this phase is performed before the scanning session starts, using adequate known objects, like chessboards or simple objects on which the correspondence problem can be easily solved. An estimate of the calibration parameters can be also computed directly from the images of the object (exploiting some additional constraints on the acquired model) [161].

Fig. 2.4 Example of real pair of images where peculiar points are highlighted

The main problem of this kind of systems is the computation of the correspondence between the 2D points on the different images. For this reason, the stereoscopic techniques are generally used for the reconstruction of particular objects in which this problem can be solved easily: points such as the corners of an object, can be identified robustly in an image using standard image processing techniques and therefore these methods work best when structured objects, like, for instance, buildings or furniture have to be reconstructed.

A possible approach for reducing the computational complexity of the correspondence problem consists in capturing many images in which the point of view slightly changes. Since the position of a point on an image will be slightly different from that on the next image, the search for the correspondence for each point can be performed only in a small portion of each image using correlation techniques. Moreover the 3D structure of the scene can be reconstructed from the motion parameters and the internal parameters of the single camera (structure from motion [116]) (Fig. 2.4).

The main advantages of these techniques are the potential low cost of the hardware needed and the non-invasiveness of the method. The generally low density of the data and the sensibility to the calibration phase limit the diffusion of these systems in real applications.

Shape-from-silhouettes systems

The silhouette systems [191][235][66] compute the model as a composition of the contours of the object taken from different points of view. To this aim, the typical scanner of this category is composed of a turn table (where the object is placed on, Fig. 2.5), a uniform background (which simplify the contour extraction procedure), and a single camera. While the object rotates, the camera captures an image from which the contour is extracted. A similar principle has been used to extract clouds of points in [67]. Each contour can be seen as a cone of projected rays that contains the object. The intersection of these cones determines the approximated shape of the object.

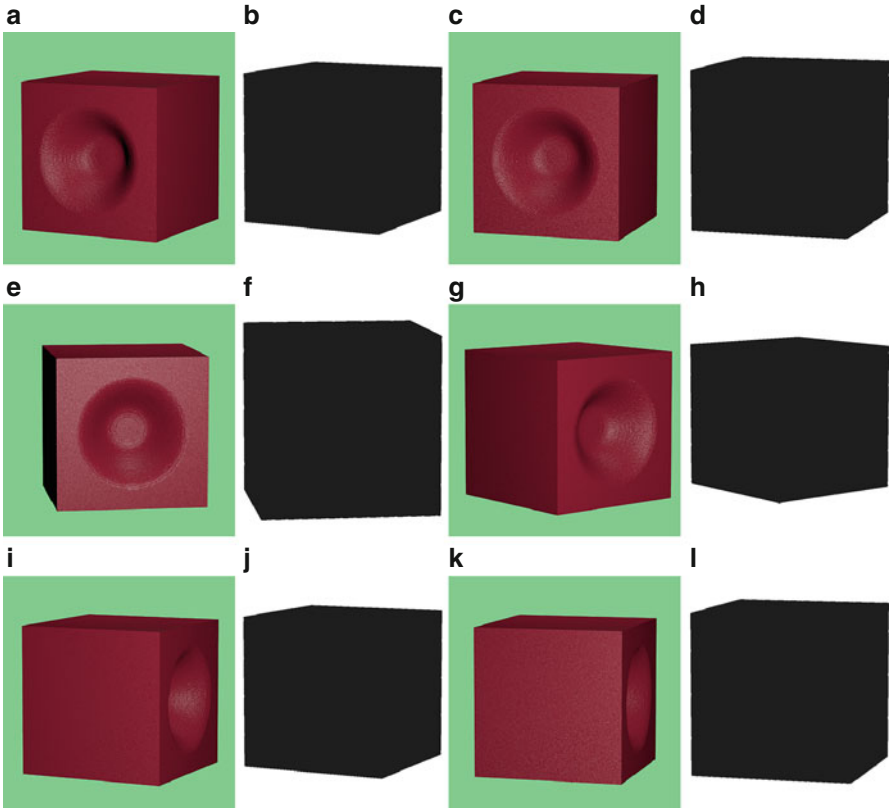

Fig. 2.5 In (**a**), (**c**), (**e**), (**g**), (**i**), and (**k**), the object seen from different view angles. In (**b**), (**d**), (**f**), (**h**), and (**l**), the silhouette of the object for each view. From the silhouettes, only the visual hull of the can be computed and, as shown in the figure, the cavities of the object cannot be detected and reconstructed

Such systems have the benefit to be easily realizable with low cost hardware, but they have the strong limitation that a controlled turntable is required to move the object with respect to the camera. Furthermore they have the strong limitation that only convex objects can be accurately reconstructed. In fact, the cavities of an object are not visible in the projected silhouettes and cannot be reconstructed. This limits the use of these systems in many real applications.

Shape-from-texture and shape-from-contour systems

Techniques that extract information about the object's shape from its texture or contour provide useful clues for 3D digitization and are interesting results of the computer vision theory, but are rarely implemented in real 3D scanners. In fact,

these techniques are not able to compute the 3D position of points on the object surface, but only the surface curvature (up to a scale parameter) or its orientation.

Shape-from-texture is grounded on the hypothesis that the surface of the object is covered by a texture characterized by a pattern that is repeated with regularity. The curvature of the surface can be computed analyzing the distortion of the texture and the field of local surface normals can be estimated from the analysis of local inhomogeneities [20]. Furthermore a diffuse illumination of the scene is required, as the shading can influence the texture analysis.

A similar technique is called shape-from-contour. In this case the surface orientation is computed by the analysis of the distortion of a surface. For example, if the object contour is known to be a circle (e.g., a coin), while the contour of the acquired object is elliptical, the surface orientation that realizes this distortion due to perspective projection, can be estimated.

Shape-from-defocus systems

In the shape-from-defocus systems [149], defocusing obtained changing the camera lens geometry, is used to extract depth information. In these scanners, a conventional camera captures several images of the same scene using different focal lengths. The frequency content of the same region in different images is used for identifying in which image the considered region is on focus. Since from the focal length the distance of the plane of focus from the optical center can be determined, then the distance of the region of focus from the camera can be computed.

Typically, these systems are not able to make very precise reconstruction, as the accuracy depends on the setup, and on the depth of field of the actual lens geometry. Besides, this technique can be applied only on texturized objects. However, these systems can be realized using low cost hardware, and, being optical passive, they are non-invasive.

2.3.1.2 Active Scanners

Active scanners project some kind of electromagnetic pattern, like an infra-red or visible light grid and the image of the pattern over the object surface is captured by an adequate sensor. From the analysis of the captured pattern, the position of feature points over the surface can be obtained. These are the most common scanner systems and are based on three different measuring principle: time of flight (ToF), phase shift and active triangulation.Active scanners based on other principles can be used for specific problems. Among these, for example, interferometry scanners [119] found application for the digitization of very small objects, while illuminant-based techniques [127] have theoretic interest especially for applications where the color of the object has to be captured.

Fig. 2.6 The Leica ScanStation C10 ToF scanner system has an accuracy of 4 mm, for a distance of 1 m to 50 m, and can acquire up to 50 000 points/s [8] (reproduced by permission of Leica Geosystems AG, Switzerland)

Time-of-flight scanners

Time-of-flight (ToF) scanners measure the distance of a point on the surface from the scanner through the measurement of the time employed by the radiation to reach the object and come back to the scanner. Knowing the speed of the radiation and measuring the round-trip time, the distance can be computed. Moreover, from the direction of the radiation emitted, one 3D point of the surface can be identified [110][9]. Hence, changing the direction of the emission, the system will sample a different point over the surface and therefore it can cover the entire field of view.

Depending on the type of waves used, such devices are classified as optical radar (optical waves), radar (electromagnetic waves at low frequency) and sonar (acoustic waves). The systems that use optical waves are the most used. Such systems are sometimes referred to as LIDAR (LIght Detection And Ranging) [241] or LADAR (LAser Detection And Ranging) [16] (Fig. 2.6). They are characterized by a relatively high acquisition speed ($10,000 \div 100,000$ points per second) and their depth of field can reach some kilometers.

Generally, the accuracy of optical ToF scanners is limited because of the high acquisition speed. In fact, a depth accuracy of 1 mm requires a time measurement accuracy in the order of picoseconds. Hence, these systems are generally applied in surveying very large and far objects, such as buildings and geographic features.

The optical properties and the orientation of the surface with respect to the ray direction affect the energy collected by the photo detector and can cause loss of accuracy.

As said above, these systems are often used for geographic reconstruction. Aerial laser scanning is probably the most advanced and efficient technique to survey a wide natural or urban territory. These systems, mounted on an airplane or on a helicopter, work emitting/receiving up to 100,000 laser beams per second. The laser sensor is often coupled with a GPS satellite receiver, which allows recovering with good accuracy the scanner position for each acquired point. Hence, each point can be referred to the same reference system and the acquired points (which form a dense cloud of points) can be related to a cartographic reference frame, for an extremely detailed description of the covered surface [239].

ToF scanners are also often used in environment digitization. A relatively recent application is the digital reconstruction of a crime scene. Using the constructed digital model, police is helped in the scene analysis task.

For this last application, the typical scanner is constituted of a rotating head which permits a wide field of view. For example the model Leica ScanStation C10 has a field of view of 360° horizontal and 320° vertical [8] (2.6).

Another kind of ToF system is the Zcam [129] (produced by 3DVSystems, and lately acquired by Microsoft) which provides in real-time the depth information of the observed scene. The scene is illuminated by the Zcam which emits pulses of infra-red light. Then it senses the reflected light from scene pixel-wise. Depending on the sensed distance, the pixels are arranged in layer. The distance information is output as a gray level image, where the gray value correlates to the relative distance.

Phase shift scanners

Phase shift scanners use a laser beam whose intensity is sinusoidally modulated over time [5]. From the phase difference between the emitted and the reflected signal the round-trip distance can be computed, as the difference is proportional to the traveled distance (Fig. 2.7a). Since the phase can be distinguished only within the same period, the periodicity of the signal creates ambiguity. To resolve this ambiguity, signals at multiple frequencies are used. This method has performances quite similar to the ToF method, but can reach a higher acquisition speed (500,000 points per second).

Active Triangulation Scanners

In active triangulation systems the scene is illuminated by a coherent or incoherent light pattern from one direction and viewed from another. These systems primarily differ in the pattern used (single laser spot, laser line, laser grid, or more complex structured patterns, such as pseudo-random bar code) and for the scanning method (motion of the object or of the scanner). If the source is a low-divergence laser beam,

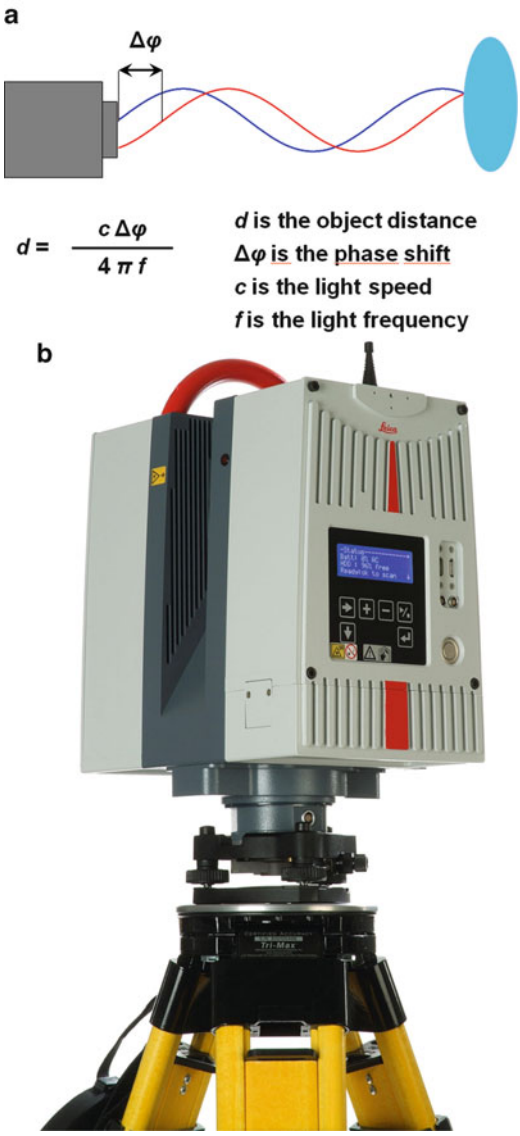

Fig. 2.7 Phase shift working principle is shown in panel (**a**) (published in [30] and reproduced by permission of IGI Global). In panel (**b**), the Leica HDS6200 phase shift scanner system has an accuracy of 5 mm, for a distance 0.4 m to 25 m range, and can acquire up to 1 million points/s [7] (reproduced by permission of Leica Geosystems AG, Switzerland)

the interaction of this radiation with the surface object will produce a spot, which can be detected by a sensor (typically a CCD camera [13]). The orientation and the position of the source and the sensor are typically known. From the spot location on the sensor, the line between the sensed spot and the camera center point can be computed, line l_1 in Fig. 2.8a. As the direction of the laser is known, the line between the laser emitter and the spot can be computed, line l_2 in Fig. 2.8a. The 3D position

Fig. 2.8 A scheme of active triangulation system is shown in panel (**a**). A single spot is acquired in this scheme (published in [30] and reproduced by permission of IGI Global). The Minolta Vivid 9i scanner system [13] based on triangulation is shown in panel (**b**) (reproduced by permission of Konica Minolta, Ramsey, NJ, USA). Such system acquires up to 640 $\times$ 480 points in 2.5 s, with an accuracy of 0.05 mm

of the spot on the object surface can be computed as the intersection between l_1 and l_2 can be computed by triangulation (Fig. 2.8a). If the laser orientation and position are not known, it is possible to calculate the coordinates using two or more cameras using standard stereoscopic methods [48].

Such systems acquire not more than one point per frame. If more complex patterns like lines or grids are used, more points per frame can be captured. As a laser line describes a plane in 3D space, the camera captures the contour resulting from the intersection of this plane and the object surface. Then, for any pixel of the contour, the corresponding 3D point on the object surface can be found by

intersecting the ray passing through the pixel and the 3D plane described by the laser. The use of a grid allows to sample the surface at the grid crossing that can be robustly identified trough standard detectors [229]. However, identifying the grid corners can be more complex than in the single beam case.

More complex structured patterns have been proposed and specific techniques have been proposed to reconstruct the surface from them using a calibrated camera-projector pair [27]. The aim of these techniques is to identify each point in a robust way. For instance in [249] colored stripes are proposed. However, this method presents a severe drawback: both the surface color of the object and the ambient light influence the color of the reflected light. For this reason, to reconstruct a colored (or textured) object, other kinds of coding are preferred.

In [206], a structured-light range-finder scanner composed of a projector and a camera that uses temporal stripe coding is proposed. For every frame, a different pattern of white and black stripes is projected over the object, and the camera captures the stripes pattern distorted by the the surface of the illuminated object. The profiles of the object are identified by identifying the transitions from black to white and white to black over the image and over time. Actually, the entities used for carrying the code are not the stripes, but the stripe boundaries: in this way, a more efficient coding is possible. In fact, a single stripe can carry one bit (the stripe can be black or white), while a boundary can carry two bits (it can be white-black or black-white). This method has shown to run at 60 Hz. Four successive frames are exploited to acquire and reconstruct the model from the actual scanning view as a matrix of 115×77 points. To obtain the whole model the object is rotated continuously while partial models are reconstructed. At the end the partial models are fused together.

Another very efficient scanner system is proposed in [127]. This system projects sequentially three phase-shifted sinusoidal gray-scale fringe patterns that are captured by a camera. The analysis of the intensity of three consecutive frames allows to compute, for each pixel, the phase of the three patterns. This information allows identifying a specific fringe element. The pattern phase maps can hence be converted to a depth map by a phase-to-depth conversion algorithm based on triangulation. The scanner has been shown to measure up to 532×500 points every 0.025 s (about 10^7 points/s), achieving a system accuracy of 0.05 mm. For example, the system has been claimed able to measure human faces, capturing 3D dynamic facial changes. In order to provide a high definition real-time reconstruction, the processing power of a Graphics Processing Unit (GPU) [155][179] is employed taking advantage of the highly parallel structure of the reconstruction algorithm.

A very popular device recently introduced for gaming control, *Microsoft Kinect*, is in fact a real-time 3D scanner. This device is equipped with a RGB camera, an infrared camera, and an infrared projector. The infrared camera captures images of the scene illuminated by the infrared projector with a static structured pattern constituted of pseudo random dots of different diameter (Fig. 2.9a). This simplifies the identification of the spots on the infrared camera. The reconstruction is performed by triangulation similarly to spot based scanners. This system allows the acquisition of 640×480 matrix of points at 30 Hz. The working range of the Kinect is 0.8–3.5 m, while the accuracy decreases with the object distance. Because

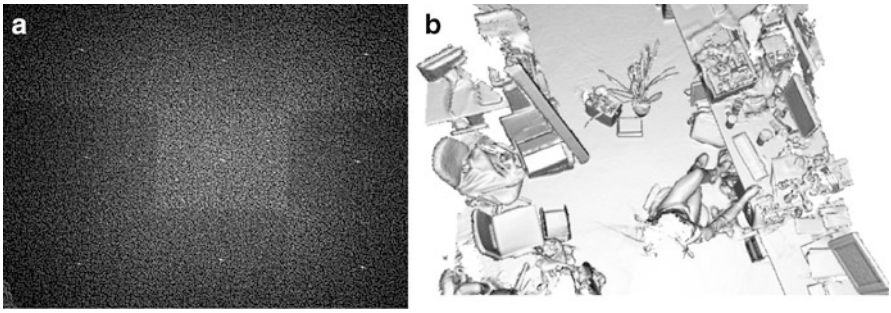

Fig. 2.9 In panel (**a**) the IR pattern projected by Kinect is shown. In panel (**b**) a 3D model obtained using Kinect [174] (reproduced by permission of IEEE)

its low cost (about 100 US dollars), Kinect is used, today, in many research projects as 3D scanner for indoor reconstruction [174] (Fig. 2.9b) and for robust tracking. Kinect has been recently explored as input device of a platform that has the goal to move rehabilitation at home [12][187] (Fig. 2.10).

The active triangulation systems, are in general characterized by a good accuracy and are relatively fast. The strong limitation of these systems is the size of the scanning field. As the coordinates are computed by means of triangulation, the accuracy of these systems depends on the angle formed by the emitted light and the line of sight, or equivalently, by the ratio between the emitter/sensor distance and the distance of the object (d/l Fig. 2.8). In the optimal set-up, the considered angle should be $45°$ [44]. Moreover, the resolution of the CCD, the resolution and the power of the light emitter limit further the active triangulation system to a depth of field of few meters. Hence, these systems cannot be usable for digitization of large objects. Furthermore object's color and ambient illumination may interfere with the measurement.

Shape-From-Shading and Photometric Stereo Scanners

The shape-from-shading problem consists in the estimate of the three-dimensional shape of a surface from the brightness field of an image of that surface. The first formulation of this problem was proposed in the 70's [125]. The work revealed that the problem requires the solution of a nonlinear first-order differential equation called the brightness equation. Today, the shape-from-shading problem is known to be an ill-posed problem, which does not have a single solution [50]. What makes difficult to find a solution for this problem is often illustrated by the concave/convex ambiguity, caused by the fact that the same shading can be obtained both for a surface and its inverted surface, for a different direction of the illuminant. This kind of ambiguity can be widely generalized. In [28] is shown that, given the illuminant direction and the Lambertian reflectance (albedo) of the surface, the same image

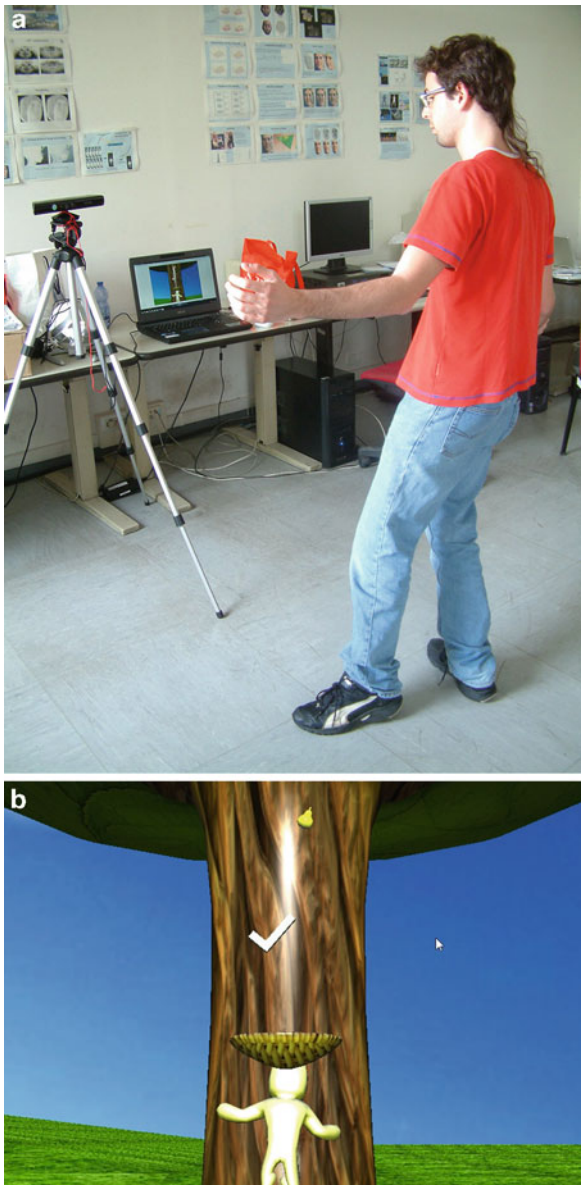

Fig. 2.10 Kinect scanner uses a tracker: the motion of a patient is robustly tracked (**a**) and pasted inside a virtual scenario inside which he is guided to do rehabilitation (**b**). The use of Kinect for this kind of application has been widely studied in [12]

can be obtained by a continuous family of surfaces, which depend linearly on three parameters. In other words, neither shading nor shadowing of an object observed from a single viewpoint can provide the exact 3D structure of the surface.

However the problem can be solved under simplified conditions. The first one is the use of directional light with known direction and intensity. But, again, this simplification is not enough and, in order to solve the problem, knowledge about

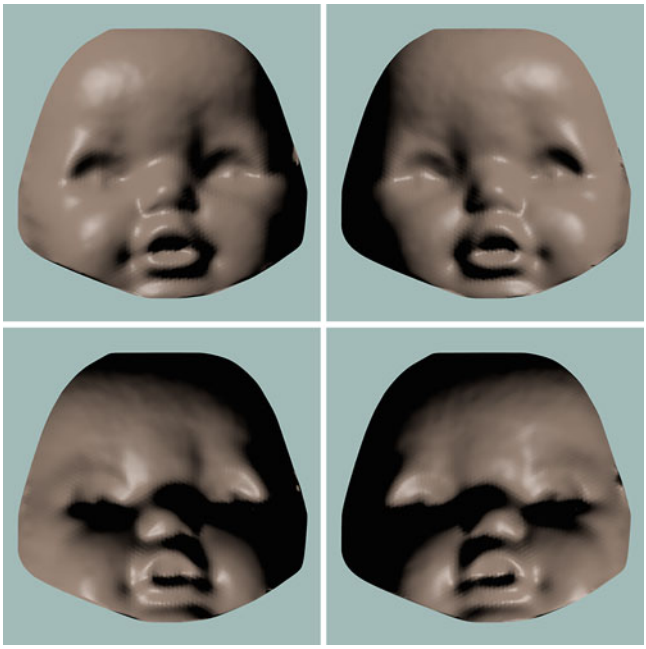

Fig. 2.11 Example of the shape information contained in the shading: different light sources produce different gray level gradients on the image. In each figure, the light comes from a different corner

reflection properties of the object surface is also required. In particular, the surface should be Lambertian, namely the apparent brightness of the surface has to be the same when the observer change the angle of view, and the albedo (i.e., the fraction of light that is reflected) should be known.

As the method implies the use of a known radiation, it can be considered belonging to the active systems class. Under these conditions the angle between the surface normals and the incident light can be computed. However in this way the surface normals are derived as cones around the light direction. Hence, the surface normal in a given point is not unique and it is derived considering also the value of the normals in a neighborhood of the considered point. Moreover, the assumption of a smooth surface is often made.

When a photometric stereo technique is used, the problem is simplified by illuminating the scene from different positions [122] [120]. With this technique, introduced in [248], the estimate of the local surface orientation can be obtained using several images of the same surface taken from the same viewpoint, but under illumination that comes from different directions (Fig. 2.11). The light sources are ideally point sources, whose position is known with respect to the reference system, and they are oriented in different directions. The lights are activated one at a time, one for each captured frame, so that in each image there is a well-defined

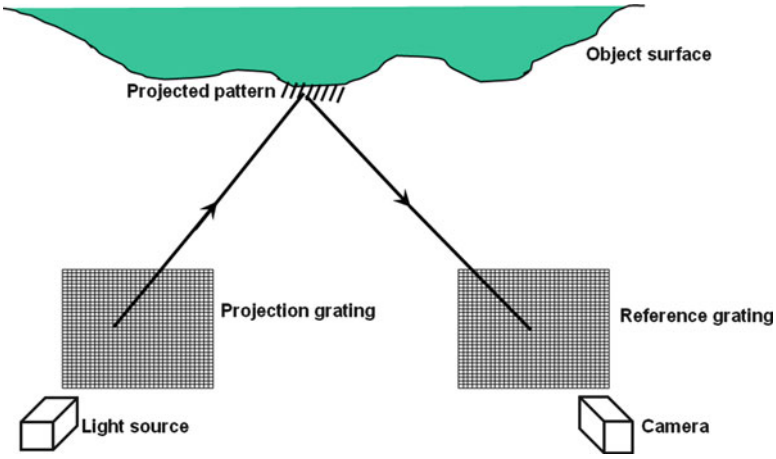

Fig. 2.12 A scheme of Moiré Interferometry system is represented. A regularly spaced grid is projected on the surface of the object and the scene is observed by a camera through a reference grid

light source direction from which to measure the surface orientation. Analyzing the sequence of intensities change of a region, a single value for the surface normal can be derived. In general, for a Lambertian surface, three different light directions are enough to solve uncertainties and compute the normals. This approach is more robust with respect to shape-from-shading, but the use of synchronized light sources implies a more complicated 3D system, which can strongly limit the acquisition volume. On the other hand, the availability of images taken with different lighting conditions allows a more robust estimate of the color of the surface.

Moiré Interferometry systems

Moiré interferometry is a technique that allows to detect and measure deformations in a quasi-planar surface. The method makes use of the interference between some form of specimen grid and a reference grid [130]. The principle of the method is that projecting parallel regularly spaced planes or fringes on the surface of the object, and observing the scene from a different direction, the observed fringes will appear as distorted by the surface shape (Fig. 2.12). The measurement of displacement from the plane can be obtained comparing the observed fringes with the reference fringes. In more detail, the z coordinate can be determined measuring the fringes distance obtained from the superimposition of the grating projected on the grating observed and knowing the projection and observation angles.

This technique allows a high accuracy (on the order of micrometers), but a very small field of view is required. In fact, the grid projected has to be very dense (e.g., with 1000-2000 lines/mm). This characteristic limits the method to microscopic reconstruction.

Holographic Interferometry systems

A hologram is the recording of the interference pattern formed by a reference laser beam and the same beam reflected by the target object. It can be obtained by splitting a laser beam in two parts: one is projected onto the object and the other one goes directly to the camera.Holographic interferometry [209] is a technique for measuring the surface height by comparing the hologram of the same object built under different conditions (e.g., moving or deforming the object, or changing the wavelength of the illuminant). In particular, shape measurement can be achieved by using two slightly different illumination wavelengths for recording two holograms of the object. By properly combining the difference in the phase of the holograms and the angle between the illuminant and the observer, the depth with respect to a reference plane can be estimated. The systems based on this technique are quite expensive, but allow sub-nanometer measurement. Since the field of view is very small, generally, the method is applied for objects of size of few millimeters.

2.3.2 Hybrid Techniques

In the previous pages, an overview of different techniques characterized by complementary strength and weakness has been presented. Many real systems implement more than one technique for exploiting the advantages of each approach. For instance, the CAM2 Laser ScanArm V3 is a Jointed Arm where the probe is an active triangulation laser scanner, combining the precision and the speed of the active system with the mobility of the Jointed Arm.

The systems that combine stereoscopic techniques with structured light are another example. The correspondence problem that arises for the stereoscopic systems can be greatly simplified by the projection of light patterns. Moreover, using multiple cameras the reconstruction can be improved both in accuracy and in speed. Another example of hybrid scanner is the TriDAR sensor [86], which exploits the complementary nature of the triangulation an the LIDAR technologies in a single system that share the components for the operation of the two subsystems.

The combination of multiple techniques generally allows more robust systems and an improvement of the accuracy. The price to be paid is the complexity and the cost of the system.

Table 2.1 Specifications of some 3D scanners

Model	Measuring method	Scan range (depth of field)	Accuracy	Acquisition rate
Leica HDS6200	ToF, phase-based	50 m	5 mm	1 000 000 points/s
FARO Photon	ToF, phase-based	76 m	2 mm (25 m)	120 000 points/s
Leica ScanStation C10	ToF, pulsed range finder	50 m	4 mm	50 000 points/s
I-Site 8800LR	ToF, pulsed range finder	2 000 m	10 mm	8 800 points/s
Helmel Checkmaster 112-102	Physical contact	0.3 mm	0.086 mm	
Metris MCA	Physical contact	2 m	0.028 mm	
Metris K-Robot	Laser triangulation	5 m	0.1 mm	1 000 points/s
Minolta Vivid 9i	Active triangulation	2.5 m	0.05 mm	122 880 points/s

2.4 Evaluation of 3D Scanners

The systems have been classified in a taxonomy that privileges the physical principle exploited to extract the 3D information. However, other properties of the scanning systems can be used as a classification key, like accuracy, resolution, or speed of acquisition. Nevertheless, these properties are more related to an actual implementation of the system than to a class of scanners and hence they are not suited for a robust classification. On the other hand, these properties have to be considered when a scanning system has to be chosen and are often critical for the choice itself. In Table 2.1, the data of some commercial scanners are reported to illustrate the variagated scenario. Depending on the target application, the critical capabilities can differently rank each scanner and guide the choice.

In [57] a method for evaluating the 3D scanners is suggested. It considers some important features like field of view, accuracy, physical weight, scanning time, and it has associated a weight to each feature. By scoring each feature, a single feature can be computed as the sum of the single score for each scanner. Scores given to each feature depend on the application domain which make the scoring arbitrary.

Anyway, the three principal aspects that should be considered in order to choose a 3D scanner are the properties of the objects to be acquired (size and material features), the accuracy required, and the budget, under auxiliary constraints such as the speed of acquisition and the environmental conditions. Some attention should be paid also on the human factors: some devices require a deep knowledge of the principles exploited by the scanner and can be used only by trained people.

An important issue associated to all 3D scanners is the *calibration procedure*. Generally, 3D scanners have different setups and a reliable acquisition of a point cloud is made possible only if the setup parameters are known. The aim of calibration phase is the estimate these parameters. This phase is critical for many types of scanners because the precision of the system is, typically, tight connected to the quality of calibration performed. Moreover the time employed to calibrate can

be very long. The procedure can spend several minutes, even though the scanning sessions can require just few seconds. Calibration procedures typically used for video systems have been employed [256][230][41]. When only subset of calibration parameters are required, simplified calibration procedures can be adopted [45][106]. This is for instance the case of Kinect 3D camera for which simplified procedure have been developed [108][218] that take into account that the relative arrangements of the sensors is known. A particular issue is represented by distortions introduced by the optics. These can be effectively corrected on the field through adequate interpolations [61][116].

However, as available computational power increases, more attention is paid by the scanner designers to making the systems more user friendly. In fact, in the last decade many research reports have been related to the estimate of the calibration parameters without a proper calibration stage. In this track, an interesting approach, mainly oriented to stereoscopic techniques, is the passive 3D reconstruction, which allows estimating the calibration parameters after the acquisition session.

Since devices for the reproduction of 3D content are becoming widely available to the consumer market, compact and easy-to-use devices for capturing 3D contents are likely to be proposed. Hence, it can be envisioned that the miniaturization of components such as CCD sensors or pico-projectors will be exploited for implementing very small, point-and-click optical devices. The light-field photography is another promising technology that can potentially be used for capturing the 3D properties of real objects. A light-field acquisition device is a lensless camera that can capture, for each pixel, not only the intensity and the color of the incoming light rays irradiated from the scene, but also their direction (that is a sampling of what is called the plenoptic function of the scene). These information can be processed through computational photography techniques in order to render the desired image. The principal advantage of such technology is the possibility of choosing a-posteriori the setting of the camera (e.g., focus distance, aperture), but it also enables the extraction of the 3D information of the scene. Although some commercial models are available, this technology is not yet mature for a large consumer market, but in this fast-moving field we can easily envision their broad availability in the next future.

Chapter 3
Reconstruction

Once the geometrical data of a physical object have been sampled on its surface, the next step is the generalization of the sampled data to obtain a continuous description of the object surface, to which visual attributes (like color, textures, and reflectance) can be associated. In this chapter an overview of the techniques used for generalization is presented. These can be subdivided into two broad families: volumetric and surface fitting methods.

3.1 Surface computation from point clouds

The aim of the reconstruction phase is to find the *best* approximating surface for the collection of data gathered in the acquisition phase. There are two aspects that characterize this problem: the a priori knowledge about the object and the kind of information collected in the acquisition phase.

In order to solve this problem, a paradigm for surface representation has to be chosen such that a priori knowledge about the problem can be encoded. For example, if the acquisition is restricted to a certain class of objects, a parametric model can be used to represent the objects of that class. The reconstruction of the surface becomes a search of the parameters that best fit the acquired data.

If the a priori knowledge about the object is limited or the class of objects is large, the reconstruction paradigm should have many parameters in order to accommodate many different shapes and the reconstruction technique should be more complex.

The problem of surface reconstruction can be formulated as an optimization problem. Although a model based approach is more robust [78], it can be applied in few cases. The general case, in which a priori knowledge is strongly limited, is here examined.

F. Bellocchio et al., *3D Surface Reconstruction: Multi-Scale Hierarchical Approaches*,
DOI 10.1007/978-1-4614-5632-2_3,

To transform a cloud of points into a continuous surface, two kinds of problems arise:

- the surface has a continuous description, while the sampling is a form of discretization: the value of the surface in between the sampled points has to be estimated;
- the samples are affected by measurement error: a strategy for error filtering can be necessary.

The goal of surface reconstruction is, then, the computation of an unknown surface approximation using a set of points and, eventually, auxiliary information about the original surface or about the sampling process (e.g., sample density and noise magnitude).

Two large families of solutions have been proposed in the literature, often referred to *spatial subdivision techniques* and *surface fitting techniques*. Some methods produce a coarse surface using spatial subdivision techniques and refine the solution using surface fitting approaches [25]. For a more detailed taxonomy, refer to [165]. In the following the two approaches will be introduced and some implementations will be presented.

3.2 Spatial subdivision techniques

Spatial subdivision techniques are based on searching the volume occupied by the object. The algorithms of this class are composed of the following steps [164]:

1. decomposition of the point space in the cells,
2. selection of the cells crossed by the surface,
3. surface computation from the selected cells.

In [19] the bounding box of the points is regularly partitioned into cubes (Fig. 3.1a–b). Only the cubes that have at least an internal data point are considered (Fig. 3.1b). A first surface approximation is composed of the exterior faces of the cubes that are not shared by any two cubes. These faces are then diagonally divided in order to obtain a triangular mesh (Fig. 3.1c). A smoother version of this surface is obtained moving each vertex of the mesh in the point resulting from a weighted average of its old position and the position of its neighbors (Fig. 3.1d). Then, the reconstruction is improved by adapting the resulting surface to the data points. A critical factor of the volumetric methods based on a regular lattice is the size of the cells.

In [25] tetragonal cells are used. First of all a rough approximation of the surface computed using α-shapes (see Glossary 7.3.2). For each point, a scalar that represents the signed distance between the point and such surface is computed. The algorithm starts with a single tetrahedron that includes all the points, which is then split in four tetrahedrons adding the barycenter to the four vertices; for each of the five points, the signed distance from the approximation surface is computed.

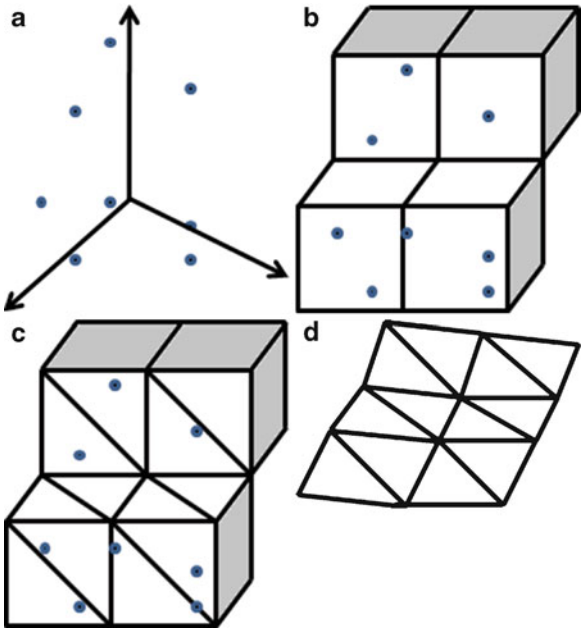

Fig. 3.1 Example of surface reconstruction by means volumetric method. In panel (**a**) the point cloud, in panel (**b**) the partition into cubes of the point cloud space, in panel (**c**) the first triangular mesh, and in panel (**c**) the a smoother version of the preview mesh

The tetrahedrons whose vertices have the same sign of their scalar (all positive or all negative) are deleted. For each of the remaining tetrahedrons an approximation of the surface that passes through it (a Bernstein-Bézier polynomial surface) is computed. If the approximation error of the surface with respect to the points inside the tetrahedron is over a given threshold, the procedure is iterated.

In [182] the set of points is partitioned by means of the *k-means* clustering algorithm. Each cluster is represented by a planar polygonal approximation and the model obtained is used as a first approximation of the surface. In order to avoid the problem of putting in the same cluster points from different faces of a thin object, the points' normals (estimated by the local properties of the dataset) are also used in clustering.

In [168] a mesh deforming procedure is used. The initial model is a non-self-intersecting polyhedron that is either embedded in the object or surrounds the object in the volume data representation. Then the model is deformed with respect to a set of constraints. For each vertex, a function that describe the cost of local deformations is associated; the function considers the noise of the data, the reconstruction features and the simplicity of the mesh.

In [19] a polygonal model is transformed in a physical model and is used in a data-driven adaptation process. The vertices are seen as points with mass and the edges are seen as springs. Moreover, each vertex is connected to the closer data point by a spring. The evolution of the model can be expressed as a system of differential linear equations and the solution can be found iteratively as the equilibrium state. After the computation of the equilibrium state, the physical model

is transformed back into a geometrical model. The solution represents a trade-off between smoothness of the surface and data fitting. The trade off is determined by the chosen physical parameters.

The physical modeling approach is, also, used in [227]. In this case the aim is to obtain a method for reconstruction and recognition. The paradigm, called Deformable Superquadrics, consists of a class of models based on parameterized superquadric ellipsoids that are deformed to fit points clouds. The ellipsoids are used to fit the global shape, while the local features are represented by means of splines defined over the superquadrics support. The main advantage of the use of a physical model is due to the intuitiveness with which the degrees of freedom can be related to the data features.

Iterative adaptation of the surface parameters has been widely studied in the connectionist domain where the problem is seen as a particular instance of learning from examples. The *Self-Organizing Map* (SOM) model [140], also known as *Kohonen features map*, consists of a set of units, $\{u_j\}$, organized in a reticular structure, and characterized by a value, $w_j \in \mathbb{R}^D$, that represents the position of the unit in the data space. The SOM is configured with the aim of approximating the distribution of a given set, $P \subset \mathbb{R}^D$. For each adaptation step, an element of the dataset p_i is extracted and the closer unit in data space, u_k, called *winning unit*, and the units connected to it are adapted with the following rule:

$$w_j(i+1) = w_j(i) + \eta \cdot h_j(k) \cdot (p - w_j(i))$$

The parameter η is the learning rate, and controls the speed of adaptation, while the value assumed by $h_j(k)$ is inversely proportional to distance of the units j and k, evaluated on the reticular structure (topological distance). When applied to three-dimensional data, the reticular structure will represent the polygonal surface and it will be a 2D manifold embedded in the 3D space that smoothly approximates the input data points.

The SOM presents two kinds of problems: convergence and dead units. The quality is greatly affected by the initial position of the units: a good starting configuration allows saving computational time, hence, a preprocessing phase devoted to obtain an initial model close to the data can be useful in this sense. Moreover, some units, called *dead units*, may form. These are units that do not move significantly during the adaptation phase because they are pushed too far from any of the data. These units may significantly degrade the quality of the obtained surface. To solve the problem of dead units, in [23] a two steps algorithm is proposed. In the first step (direct adaptation), the input is presented to the SOM and the respective units are updated, as in the standard one, while in the second step (inverted adaptation) the units are randomly chosen and they are updated using the closest input data. Hence, potential dead units can be brought near to the data and can effectively participate to the fitting. A similar technique is used in [26], where the attraction of data points outside the SOM mesh is increased. Hence, especially during the first steps of learning, the mesh is expanded toward the external points (Fig. 3.2a–c). When the SOM lattice is exploited to represent the surface

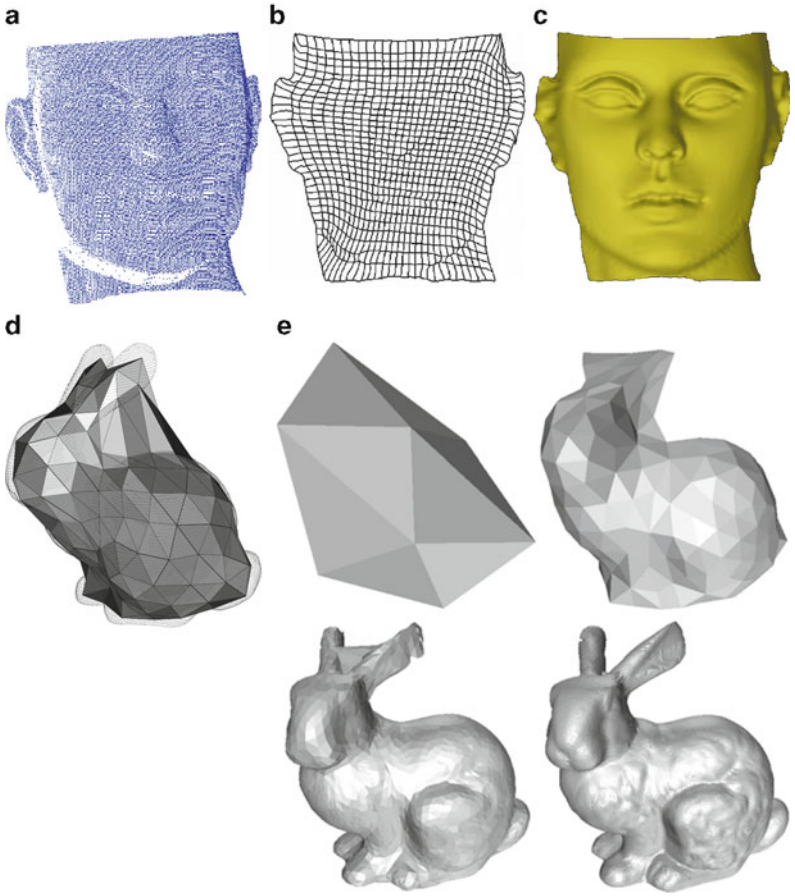

Fig. 3.2 3D surface reconstruction through SOM. The data set in panel (**a**) is reconstructed by the SOM shown in panel (**b**) as a lattice and in panel (**c**) as a shaded surface [26] (reproduced by permission of IEEE). The SOM training procedure has been modified to improve the reconstruction accuracy at the border of the dataset, by increasing the attraction of the data points that lie outside the mesh. Spurious triangles are a common drawback of SOM based surface reconstruction meshes. In panel (**d**), this defect is evident in the region of the bunny's ears. In [162], an iterative procedure based on SOM is proposed. A low resolution mesh is adapted using the SOM training algorithm. At the end of the training, unstable vertices are removed and large triangles are subdivided. If the reconstruction accuracy of the resulting refined mesh does not satisfies the specifications, the procedure can be iterated. This allows to obtain a sequence of meshes at increasing resolution, as those in panel (**e**), as shown in [162] (reproduced by permission of IEEE)

that reconstruct the data points, some precautions have to be taken to avoid self-intersection and spurious triangles. For instance, in [162], the surface reconstruction has been structured in three phases. The mesh obtained by the training of the SOM is processed to find a better fitting to the data by applying mesh operators such as edge swap and vertex removal to the unstable vertices of the mesh (i.e., those vertices

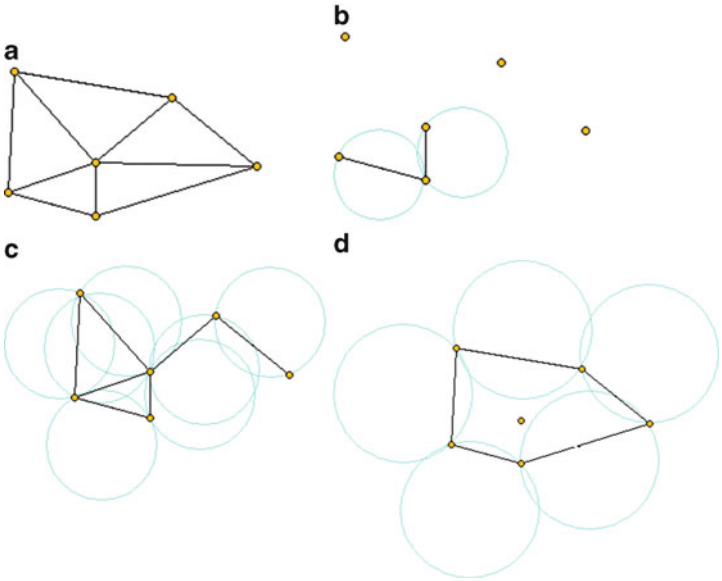

Fig. 3.3 Example of α-shape computation in 2D space. In (**a**) the Delaunay triangulation for a set of six points is shown, in (**b**), (**c**), and (**d**) the results of the edges elimination based on three different values of α

that are relatively far from the data points). Finally the mesh is refined by adding new vertices: large triangle are subdivided and vertices with high connectivity are splitted. The process can then be iterated obtaining a sequence of increasing resolution meshes (Fig. 3.2d–e).

There are some methods based on the deletion of elements. This approach is called *sculpturing* or *carving*. In [85] a reconstruction algorithm based on α-shape (see Glossary 7.3.2) has been proposed. Delaunay triangulation (see Glossary) is first computed for the given data set and then every tetrahedron which cannot be circumscribed by a sphere of radius α is deleted: the remaining triangles compose the α-shape of the dataset. It is worth noting that for $\alpha = 0$ the α-shape is the dataset itself, while for $\alpha = \infty$ the α-shape is the Delaunay triangulation of the dataset. In Fig. 3.3a–d the working principle for the 2D case is represented. In Fig. 3.4, instead, the surfaces reconstructed from a cloud of points through 3D α-shape for several values of α are reported [226].

The following criterion is adopted for the computation of the faces that constitute the surface: the two spheres of radius α that pass through the vertices of each triangle are computed. The triangle belongs to the surface if at least one of the two spheres does not contain any other point of the dataset. This method is very simple and has only α as global parameter. If the sampling density is not uniform, the use of a single α value can cause a poor reconstruction. In fact, if the value of α is too large, the reconstruction can be too smooth, while for a too small value of α the reconstruction can have undesired holes. In [226], a suitable value for α is locally

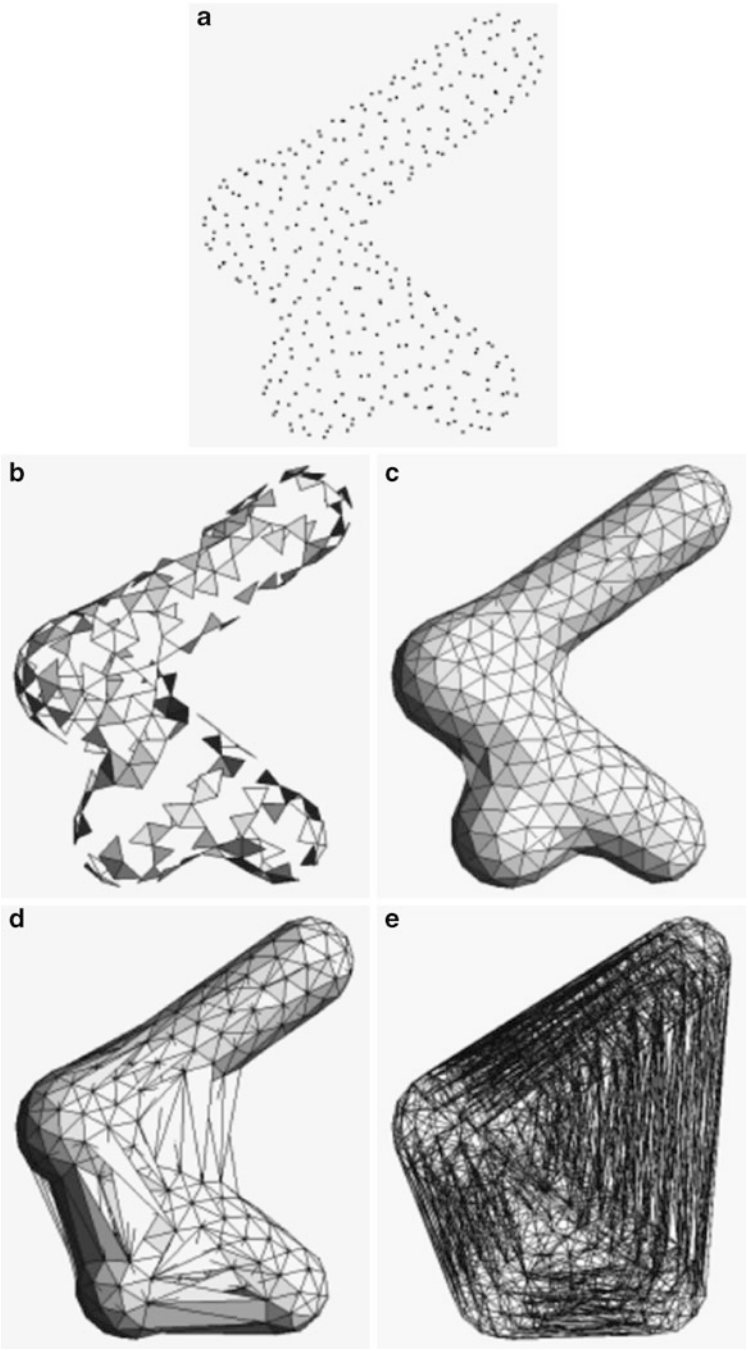

Fig. 3.4 Example of α-shape computation in 3D space. In (**a**) the points cloud, in (**b**)–(**e**) the α-shape surface reconstruction using $\alpha = \{0.19, 0.25, 0.75, \infty\}$ [226] (reproduced by permission of IEEE)

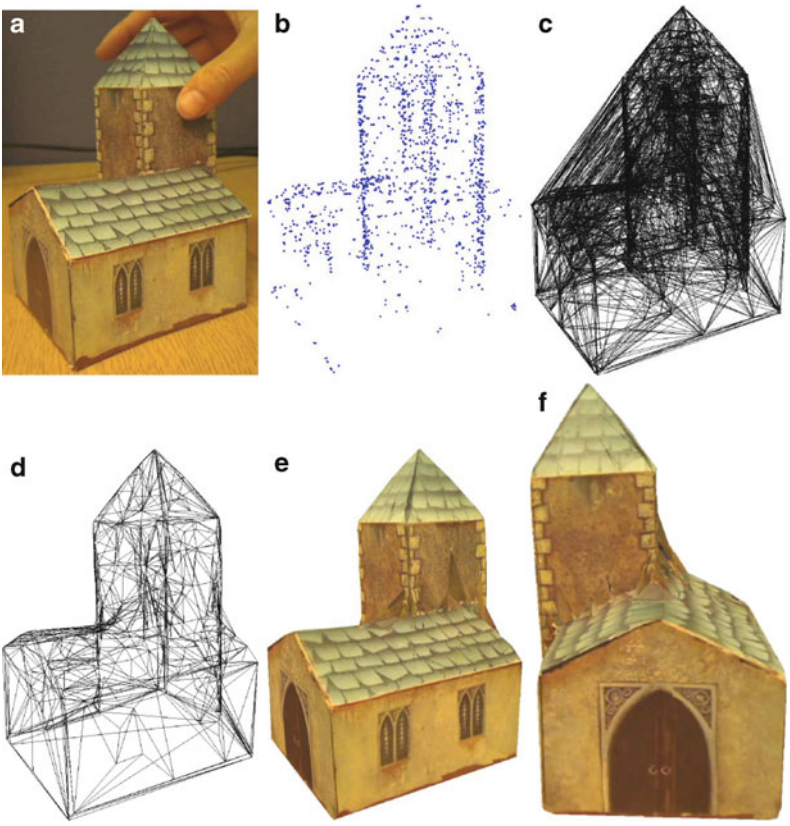

Fig. 3.5 Carving associated to 3D Delaunay tetrahedralisation [181]. The real object in panel (**a**) is observed from different point of views and the images are used to obtain the point cloud in panel (**b**). Their 3D Delaunay tetrahedralisation is reported in panel (**c**). It is then carved by eliminating all the triangles that occlude the background elements present in the acquired images, obtaining the surface mesh in panel (**d**). Texture can then be applied obtaining the 3D textured model shown in panels (**e**) and (**f**) (reproduced by permission of IEEE)

adapted to the data, considering the normal of the potential triangle vertices; if they are almost collinear, the α value is locally shrinked for eliminating the triangle from the mesh.

In [181], a carving technique based on background occlusion is proposed (Fig. 3.5). It should be noted that it requires auxiliary information that can be gathered during the scanning session. A method that, from the set of the tetrahedrons obtained from the Delaunay triangulation, computes the elimination of those tetrahedrons considered external to the object is proposed in [40]. The external tetrahedrons are selected using a heuristic (the tetrahedrons that have the following elements toward the outside of the object: two faces, five edges and four points or a single face, three edges and three points) (Fig. 3.6). The aim of the procedure is obtaining a

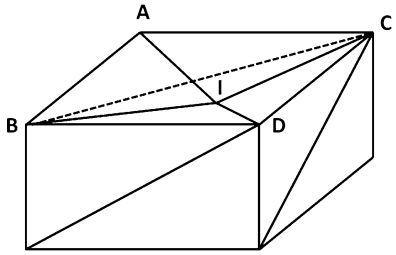

Fig. 3.6 Example of a tetrahedrons elimination procedure [40]. The tetrahedrons $ABCI$ and $BCDI$ are eliminated because they have the following elements toward the outside of the object: single face, three edges and three points. In this way, the cavity determined by I can be made visible

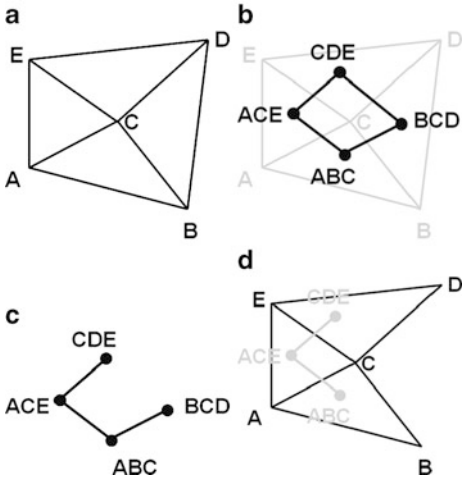

Fig. 3.7 Example of spanning tree computation in 2D space. In (**a**) the triangulation of a set of five points is shown and in (**b**) the graph associated to it. The nodes represent the triangles and the edges represent the edges in common between two triangles. In (**c**) the minimum spanning tree is depicted. In (**d**) the triangulation resulted from pruning the minimum spanning tree

polyhedron of *genus* 0 (see Glossary 7.3.2) whose vertices are all the data points, in other words, all the data points are on the surface. This is obtained by iteratively eliminating one tetrahedron at a time. For each tetrahedron which has at least one face on the surface, a value, called decision value, is computed and is used as a key to sort the tetrahedrons. After evaluating all the tetrahedrons, the ones with the largest values will be eliminated first. The threshold is defined as the maximum distance between the faces of the tetrahedron and the points that lie on the sphere circumscribing the tetrahedron itself. Hence, tetrahedrons that are large and flat will be eliminated first. These tetrahedrons, generally, hide surface details. This algorithm does not allow the reconstruction of surfaces with non-zero genus.

An alternative approach [210] is based on the duality between Delaunay triangulation and Voronoi diagram (see Glossary 7.3.2). The minimum *spanning tree* of the Voronoi diagram (where each node corresponds to the center of the tetrahedron of the Delaunay triangulation of the data points and the length of each edge corresponds to the distance of the connected centers) is computed and some heuristics are applied in order to prune the tree (Fig. 3.7). The elimination of a node

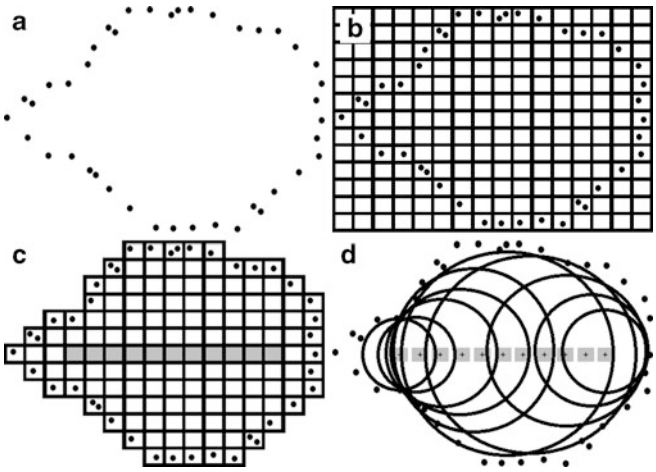

Fig. 3.8 Example of medial axis approach to model surfaces in 2D space. In (**a**) the point cloud. In (**b**) the input points partitioned into a regular lattice. In (**c**) external squares are eliminated and the squares with the maximal distance from the axis are selected and the medial axis of the object is identified. In (**d**) the circles with radius equal to the minimum distance to the points are represented

in the spanning tree corresponds to elimination of the tetrahedron associated in the triangulation (Fig. 3.7d).

In [34], the principle of *sculpturing* of an initial triangulation obtained from the α-shape algorithm is used. The elimination criterion is different from the previous methods, as it is based on a local measure of smoothness. The α-shape is used to find the general structure of the surface and the elimination phase is used for the details.

In [38], a different method based on the definition of medial axis has been proposed. It is called *medial axis transform*, and it is defined as the set of centers of the maximal spheres, which are the spheres internal to the surface that are not included in other internal spheres (Fig. 3.8d). This set defines the medial axis of the object represented by the data set. The approximation of the medial axis is computed from the *bounding box* of the input points partitioned by a regular lattice (Fig. 3.8b). The voxels that contain the points are considered as the border of the volume included by the surface. Starting from the external voxels, those that do not contain points are removed (Fig. 3.8c). For each voxel a value representing the distance between the voxel and the surface is computed. A distance equal to zero is assigned to the voxels that contain points. Then the distance is iteratively propagated to the adjacent voxels that do not contain points, increasing its value. For the voxels with maximal distance it is assigned a sphere with a radius equal to the value of the distance (Fig. 3.8d). To each sphere, a field function is associated. These constitute the primitives for the definition of a field scalar value in each point of the space, which, in turn, provides the implicit surface. The global field is defined as the sum of a subset of these primitives, which are selected by using an iterative optimization procedure.

The methods based on the Delaunay triangulation use the hypothesis that the points are not affected by error. However, as generally these points are the result of a measurement, they do have noise. For this reason, a noise filtering phase can be needed for the vertices of the reconstructed surface.

3.3 Surface fitting methods

The alternative to spatial subdivision techniques is represented by surface fitting. These techniques are divided in two classes, often referred as *surface reconstruction* or *function reconstruction* [124]. Surface reconstruction can be further divided in other two classes: *implicit reconstruction* and *parametric reconstruction*. In implicit reconstruction, the goal is to find a smooth function $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ such that the input points $\{k_1, ..., k_n\} \subset \mathbb{R}^3$ are close to the zero set of $f(\cdot)$, $Z(f) = \{x \in \mathbb{R}^3 | f(x) = 0\}$. $Z(f)$ represents the estimate of the surface. In a second stage, a contouring algorithm can be used to obtain a simple surface that approximates $Z(f)$. In parametric reconstruction, instead, the surface is represented as a parametric function on 2-dimensional parameter domain embedded in $\mathbb{R}^3$. For example a popular parametric formulation is the Non-Uniform Rational B-Spline (NURBS) [186] which is defined as:

$$f(u, v) = \sum_i \sum_j B^h_{i,j} N_{i,k}(u) M_{i,l}(v) \tag{3.1}$$

where $N_{i,k}(u)$ and $M_{i,l}(v)$ are the B-spline basis functions of order k and l, and $B^h_{i,j}$ are the coordinates of the control points. This approach has the limit that is in general very difficult to make a surface defined on suitable control points from unorganized collection of scanned point data. For these cases, in practice, the parametric reconstruction is used just for a rough initial model description that has to be improved using other techniques. From the other side, this approach is particularly suited for interactive modeling because moving the control points is easy to control the locally deformation of the model surface.

The objective of the *function reconstruction* approach can be stated as follow: given a set of pairs $\{(x_1, z_1), ..., (x_n, z_n)\}$, where $x_i \in \mathbb{R}^2$ and $z_i \in \mathbb{R}$, the goal is to determine a function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ such that $f(x_i) \approx z_i$. More formally, $f(x_i) = z_i + \epsilon_i$, where ϵ_i is a random variable with a given distribution.

The implicit reconstruction approach allows complete 3D reconstruction, but the surface is described in terms of distance function. The input domain represents the 3D coordinates of points, the codomain is a measure of the distance between the points, and the surface is implicitly computed by the function itself. This means that the points belonging to the surface have to be estimated by searching the zero set elements. This is typically more expensive with respect to the case of function reconstruction, in which the function itself directly represents the surface.

Besides, having the analytic representation of the surface, a resampling of the surface at the required resolution can be easily obtained. With function reconstruction approach the description of a complete 3D model cannot be realized using a single explicit function, as a single explicit function is able to describe just 2.5D surface (i.e., similar to a bas-relief representation). A further step for registering and merging the different 2.5D reconstructions is then needed.

The function reconstruction approach is the core of this book and it will be treated in depth in the next chapters.

An example of implicit reconstruction approach has been presented in [124]. In this work the *signed distance function* concept is used for the approximation of the implicit surface. This function computes for each point in the 3D space the signed distance from the point to the surface along the surface normal of the point itself (the estimation of the surface normals in the point is required for this computation). The signed distance function of the surface assumes a positive value for external points and negative value for internal points. The isosurface (i.e., the region of the space where the signed distance function is zero) is then the target surface. The 3D space occupied by the cloud of points is regularly partitioned into a voxel grid. For each voxel vertex, the value of the signed distance function is estimated considering the distance between the vertex and the plane that approximates the points in the neighborhood of the vertex. Then, the voxel which have vertices of opposite sign are selected, since they are the voxels through which the surface passes. For improving the surface resolution the *marching cubes* algorithm [154] is applied to the selected cubes. Marching cubes is one of the most popular algorithm for the computation of the isosurface from signed distance functions. Each selected cube has some vertices with signed function over zero and some others under zero, the isosurface of value equal to zero will pass through the cube. Considering the value of the signed function in the vertices the triangular patches that divide the cube in regions internal and external to the isosurface are computed (Fig. 3.9). The representation of the surface is generated connecting the obtained triangular patches.

A similar approach is proposed in [203], but with a different technique for the computation of the signed distance function. In this case the value of the signed distance function in the vertices of the voxel is computed as the weighted average of the signed distance between every point in the eight neighboring voxels and the vertex considered. As they consider the case in which the points are collected using a laser scanner, the measurement error is largely effected by the angle between the point normal and the line of sight of the 3D scanner. In particular the measurement error will be large when the surface normal is close to being orthogonal to the line of sight of the 3D scanner. For this reason they use the cosine of that angle to weight the contribute of each point. Since the distance function determines implicitly the surface and obviously it is not a priori known, the technique for estimating the distance function is a critical aspect for the methods of this kind.

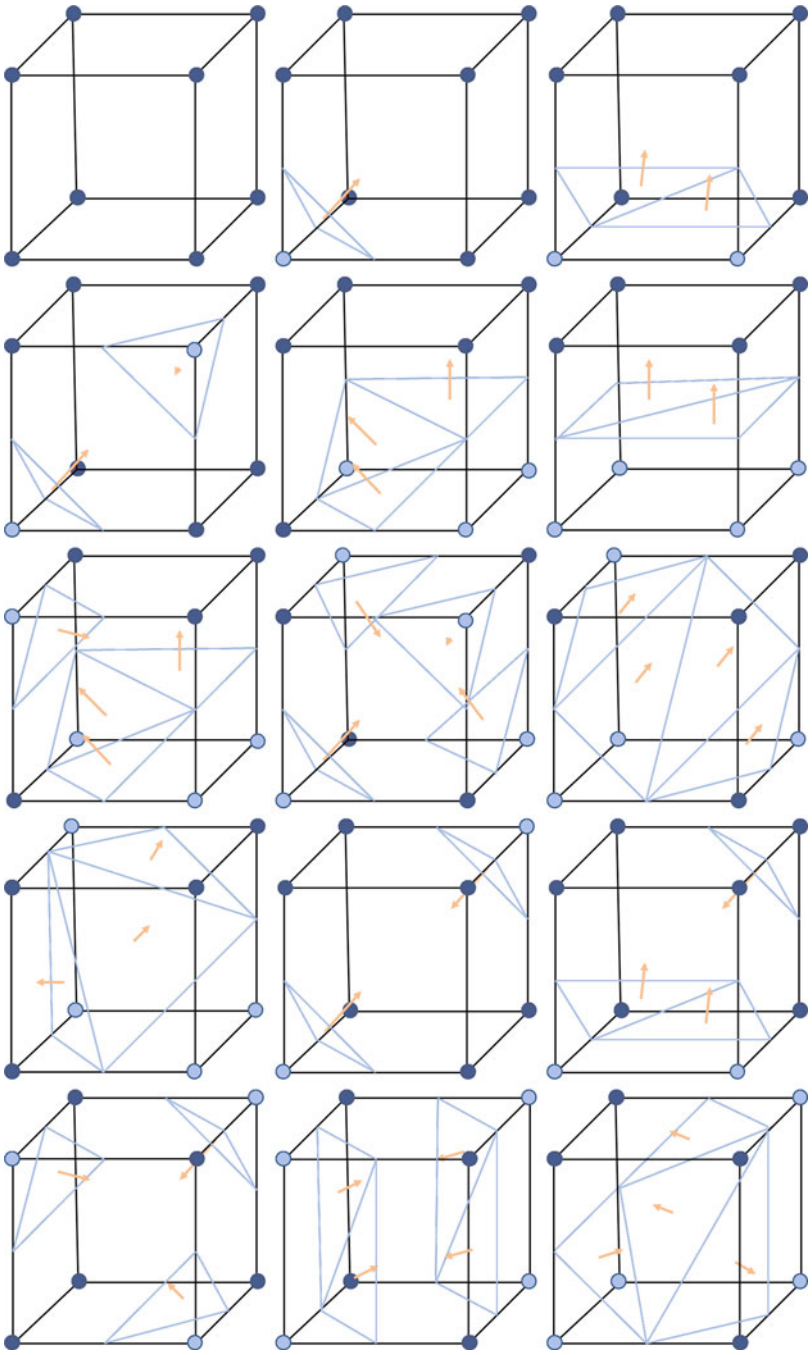

Fig. 3.9 The 15 possible intersections of a cube and a the surface are shown. The color of each vertex represents if the signed distance function is positive (dark) or negative (light). Depending on the signed distance function value in the vertices, different triangular patches are computed

3.4 Multiresolution approach

When very large data sets are considered, the data density is not needed to be uniformly dense as details may concentrate in few regions. Both the volumetric and the surface fitting methods can be realized considering the measured information at different scales. Many phenomena of real world have an informative content that is appreciable at different space scales. Some features of an object can be associated to the global shape, while other features can be associated to the details, localized in space. The large scale features are generally related to the object type, while the small scale features are related to the details and can be used to differentiate objects of the same type. The large scale features have typically a low frequency content (hence a small variability), while the small scale features have a high frequency content. If an organization of the information based on the scale would be available, few resources would be generally sufficient to describe the behavior at a large scale. Conversely, the configuration of the resources for the small scale features can be performed, considering only the data belonging to small regions.

The multiresolution and hierarchical paradigms are based on these concepts. They are structured to perform a description at different scales. There are mainly two ways to obtain a multiresolution description:

- *coarse to fine* (Fig. 3.10a), where the description at low resolution is extracted first, and then the resolution is increased till the details are reconstructed;
- *fine to coarse* (Fig. 3.10b), where the description at the maximum resolution is performed first, and then the description at smaller resolution is obtained.

A coarse to fine multiresolution paradigm is presented in [254], where a cloud of points is recursively approximated using superquadrics. In the initial phase, the superquadric that best approximates the data is estimated (Fig. 3.11b). For

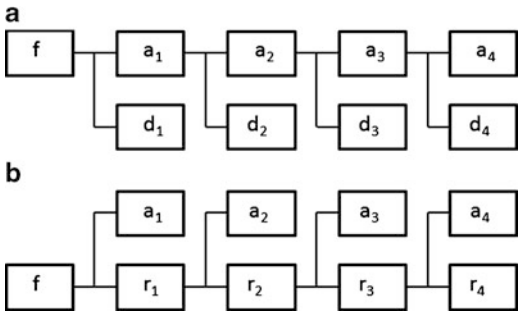

Fig. 3.10 The two approaches to obtain a multiresolution representation of the target surface f. In the fine to coarse approach, (**a**), the scale of approximation increases with the layers. The function a_i $(0 \leq i \leq 4)$ represents the approximation of a_{i-1}, where $a_0 = f$, obtained by layer i, while d_i represents the approximation error of the layer i: $d_i = a_{i-1} - a_i$. In the coarse to fine approach, (**b**), the scale of approximation decreases with the layers. a_i represents the approximation of r_{i-1}, where $r_0 = f$, obtained at layer i. r_i represent the approximation error of the layer i: $r_i = r_{i-1} - a_i$

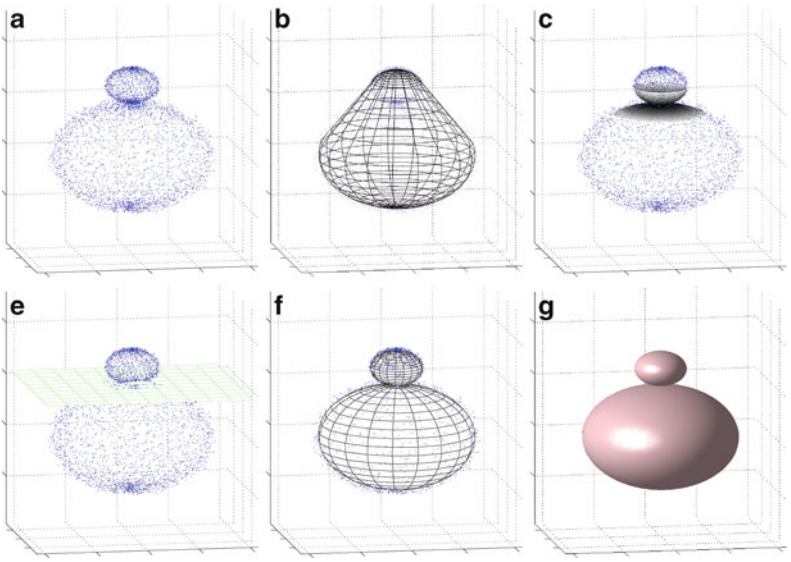

Fig. 3.11 In (**a**) a 3D point cloud is shown. In (**b**) the first approximation computed using a superquadric. In (**c**) the colored region represents where the residual of the points is large. In (**d**) the dividing plane for the region with large residual is shown. In (**e**) the second approximation on the two subset is depicted and in (**f**) reported as a shaded surface

each point, the residual of the fitting (i.e., the distance between the point and its approximation) is computed (Fig. 3.11c) and a plane is fitted to data, characterized by a large error (Fig. 3.11d). The plane is then used for splitting the cloud of points into two disjoint parts. For each subset, the superquadric that represents the data in the optimal way is computed (Fig. 3.11e). If the procedure is iterated, for example computing the dividing plane for the subsets with a higher error, a set of models, with increased complexity that can be associated to different resolutions, is determined.

The hierarchical paradigms are an interesting family of the multiresolution paradigms. They have a structure characterized by levels (or layers), where each level is characterized by reconstruction scale. The reconstruction at a certain scale is obtained using all the levels at scales higher or equal to the scale chosen. Therefore, the representation presents a hierarchical structure.

In the next chapters two kinds of *coarse to fine* hierarchical paradigms will be presented: in particular the first one is called Hierarchical Radial Basis Functions (HRBF) networks [46][47], treated in depth in Chap. 5, while the second one, called Hierarchical Support Vector Regression (HSVR) [90], will be treated in depth in Chap. 6.

The HRBF model is composed of a pool of subnetworks, called layers, and its output is computed as the sum of the output of its layers. The output of each layer is computed as a linear combination of Gaussians. For each layer, the Gaussians are

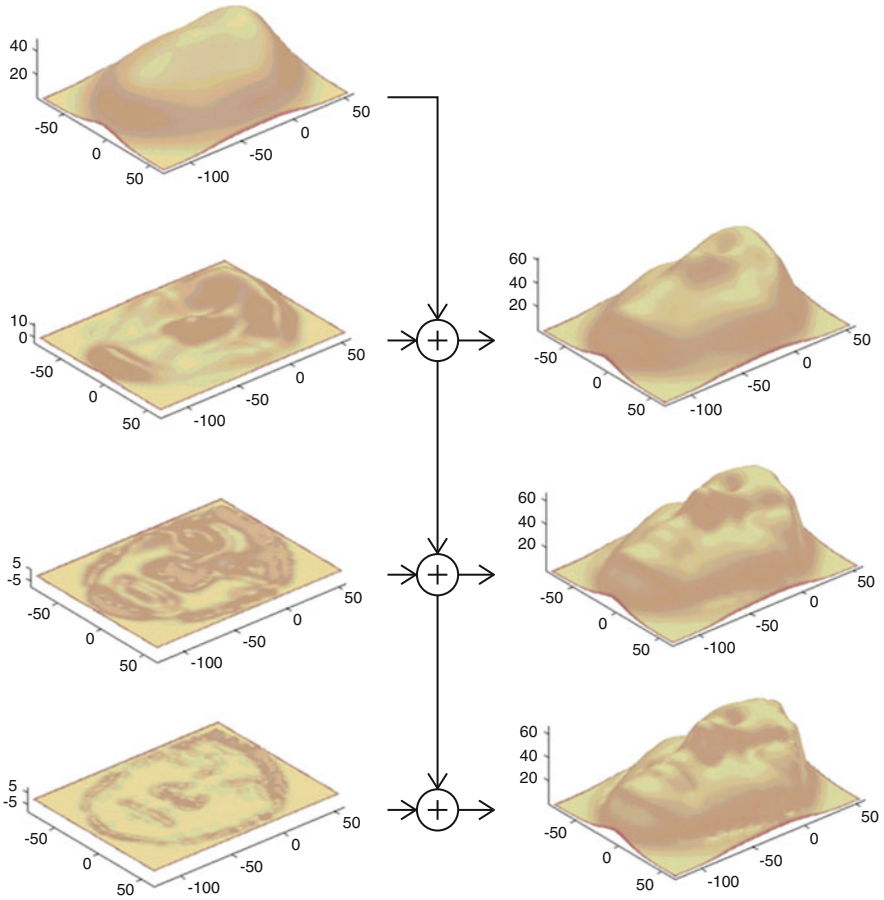

Fig. 3.12 The output of four layers of a HRBF model are depicted in the left. The sum of the output of the first layers gives the multi-scale approximation, in the right

regularly spaced on the input domain and they have the same standard deviation, which acts as the scale parameter of the layer.

The configuration of the coefficients of the linear combination is based on the analysis of the points that lie in the region of the Gaussian. A first layer of Gaussians at large scale performs an approximation of the global shape of the surface. The detail is computed as the difference of the dataset and the approximation computed by the first layer. The scale parameter is then decreased and a new layer is inserted to approximate the detail. The process can be iterated till the maximal possible resolution is reached (Fig. 3.12).

The Hierarchical Support Vector Regression model works in a very similar way. The idea again is based on a model organized in layers, where each layer performs the reconstruction at a certain scale. The main differences with respect to the HRBF

model are that the basis functions are placed at the input points position (in general, they are not regularly spaced), and the weights are found as the solution of an optimization problem.

In a similar way, the *Multilevel B-splines*, proposed in [147], performs the reconstruction using the superimposition of different levels of B-splines. The coefficients of the first level are computed by means of a least squares based optimization, considering only the points that lie in the influence region of the base. The computation starts from a reticular structure of control points. The detail is computed as difference between the original data and the approximation performed by the first layer. A new layer of B-spline, with a denser reticular structure of control points is then used for the reconstruction of the detail. A similar approach, called Hierarchical Spline, [99] works only with regularly spaced data. For this case, a fine to coarse algorithm has been proposed [100], where the multiresolution representation is derived from a initial B-spline representation at the maximal resolution.

The *Multiresolution Analysis* based on the wavelet transform [157][63] works with regularly spaced data. An MRA is characterized by a pair of functions, called scaling function and wavelet, that span complementary spaces. The basis for these spaces are formed respectively by shifted copies of the scaling function and the wavelet. The scaling function is usually smooth and features the low frequency components, while the wavelet varies rapidly and features the high frequency components.

A function can be decomposed as the sum of an approximation and a detail function, which belong respectively to the space formed by the scaling function (approximation space) and the wavelet (detail spaces). Function decomposition can be iterated, which allows to obtain the function representation with a fine to coarse approach: the process starts from the data and it is iterated on the coefficients of the approximation function to obtain the coefficients for the coarser scales. At the end, the multiresolution reconstruction can be performed by adding the coarsest approximation and the details up to a given scale. Hence a function can be represented as a linear combination of shifted copies of the scaling function and the wavelet. The coefficients of the approximation and of the detail are determined by the convolution between the data samples and a suitable pair of digital filters (low-pass filter for the approximation and high-pass filter for the detail).

In [183], a MRA based approach for not regularly spaced data is presented. The data considered are a three-dimensional cloud of points. The approach is based on the hypothesis that the surface has the topology and the genus of a sphere. The initial polygonal mesh is computed by adapting a sphere to the set of 3D points. To this mesh, the wavelet transformation defined on a spherical domain [211] is applied.

In [98] another *fine to coarse* technique based on wavelets for not regularly spaced data is presented (but limited to one-dimensional data). The scaling function and wavelet coefficients are obtained by means of the minimization of the reconstruction error, through the resolution of a linear system.

Chapter 4
Surface fitting as a regression problem

A brief overview of the methods for surface reconstruction has been presented in the previous chapters. In this chapter the attention will be focused on a particular class of methods that see surface reconstruction as a multivariate approximation problem. Some of the most popular techniques of this kind will be presented and pros and cons will be discussed. An evolution of two of these techniques, namely Radial Basis Function Neural Networks and Support Vector Machines, will be the topic of the next two chapters.

4.1 The regression problem

The prediction of the value of a variable from its observations is known in statistical literature as *regression*. This problem has been studied in several disciplines in the area of computer science and applied mathematics. In particular, the prediction of real value variables is an important topic for machine learning. When the value to be predicted is categorical or discrete, the problem is called classification. Although in machine learning most of the research work has been concentrated on classification, in the last decade the focus has moved toward regression, as a large number of real-life problems can be modeled as regression problems.

The regression problem is identified using different names [234], for example: functional prediction [219], real value prediction [251], function approximation [213] and continuous class learning [196]. In this chapter, the name *regression* is used, because it is the historical one. More formally, a regression problem can be defined as follows:

Given a set of samples $\{(x_i, y_i) \,|\, i = 1, \ldots, n\}$*, where* $x_i \in X \subset \mathbb{R}^D$ *and* $y_i \in Y \subset \mathbb{R}$*, the goal is to estimate an unknown function* f*, called regression function, such that* $y = f(x) + e$*, where* e *is a (zero mean) random variable which models the noise on the dataset.*

The elements of the vector x will be indicated as input variables (or attributes) and y will be indicated as output (or target) variables. The regression function

F. Bellocchio et al., *3D Surface Reconstruction: Multi-Scale Hierarchical Approaches*,
DOI 10.1007/978-1-4614-5632-2_4,

describes the dependency between x and y. The goal of a regression problem is to estimate this dependency from a set of samples and eventually from a priori knowledge about the regression function f. The solution of a regression problem will be indicated with $\hat{f}$. As $\hat{f}$ can be evaluated also for values of x that are not in the training set, the regression function represents a generalization of the training set. The generalization capability of a solution can be evaluated challenging $\hat{f}$ over a set of samples (usually called test set) not used in the computation of $\hat{f}$.

It should be noticed that the regression problem is more general than the problems in which the training samples are assumed to be noise free (for example, interpolation problems). Typically, the techniques developed for noise-affected data can provide good solutions also in the cases in which noise is absent. Viceversa the techniques developed for the noise free case are not able to filter the noise and the solution tends to reproduce also the noise on the training samples. In this case, typically, the solution does not generalize well. When $\hat{f}$ is characterized by a low error on the training samples and a high error on the other data, overfitting did occur. The regression techniques can be considered robust because they are aimed at noise filtering and at avoiding overfitting.

The importance of the regression techniques is highlighted in all the applications in which a domain expert in not available. Thanks to these techniques, the problem can be addressed just using the data. Furthermore, it can be considered a technique to extract knowledge automatically. In fact, as the solution allows predicting the output corresponding to input values not used in estimating $\hat{f}$, it can be used to get new knowledge about the considered domain.

The regression techniques can be divided in two classes: parametric and non-parametric. In the following this classification is presented and some examples of the methods belonging these two classes are presented.

4.2 Parametric versus non-parametric approaches

One of the most used approaches in regression is to limit the search for the solution to a subset of functions. In particular, the limitation can be realized searching the solution among a family of parametric functions. Although a choice of a small family of parametric functions can have the collateral effect of excluding the function that actually generates the dataset (which would be the "true" solution of the regression problem), this approach offers the advantage of a simpler solution (e.g., it is less computationally expensive than other methods). Simple linear regression in statistical analysis is an example of a parametric approach. In this approach, the variable y is supposed to change at a constant rate with respect to the variable x. Hence, the solution of the regression problem will be a hyperplane that satisfies the following linear system:

$$y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \cdots + \beta_D x_{i,D} + \varepsilon_i, \quad i = 1, \ldots, n \tag{4.1}$$

The first subscript, i, denotes the observations or instances, while the second denotes the number of input variables. Hence a system of n linear equation can be used for computing $D + 1$ parameters, β_j, $j = 0, \cdots, D$, of the hyperplane. The term ε_i represents the error of the solution on the sample i, namely $\varepsilon_i = y_i - \hat{y}_i$, where $\hat{y}_i = \hat{f}(x_i)$, the value of the solution (the hyperplane) for x_i. In general, the criterion used is based on the minimization of an error (or loss) function on the training points. An error measure often used in the applications is the the sum of squares difference between the predicted and the actual value of the examples (least squares criterion).

In parametric models, the mathematical structure of $\hat{f}$ is given and it is expected to be suitable to the data. This allows largely reducing the number of the parameters β_i at the price of a more complex shape of $\hat{f}$. This approach is reasonable only when adequate a-priori information on the data is available. When, the a priori knowledge is poor or not available, a non-parametric approach is generally preferable. The non-parametric approach is based on the use of a generic structure of $\hat{f}$ able to model a large set of functions, requiring only very weak assumptions on the distribution underlying the data. An example of non-parametric approach is the locally weighted regression that will be presented in Sect. (4.4).

The choice of the possible set of functions among which the solution has to be searched is known as model selection problem. It is a critical task due to the *bias-variance dilemma* [111]. It can be shown that the expectation value of the error committed by a model, M, for the solution of a regression problem can be expressed as the sum of three components: $\text{Var}(M) + \text{Bias}(M) + \sigma_\varepsilon$.

The first term, $\text{Var}(M)$, describes the robustness of a model with respect to the training set. If different training sets, sampled by the same distribution, determine very different solutions, the model M is characterized by a large variance. The second term, $\text{Bias}(M)$, describes how the best approximation achievable by the model, M, is a good estimate of the regression function f. The third term, σ_ε describes the measurement error on the data and it represents the limit of the accuracy of the solution.

For example, if the regression function is a quadratic polynomial and the regression problem is solved using a linear model (Fig. 4.1a), the variance can be small (the solution can be robust with respect to different training sets), but the bias is large (the best, in some sense, solution for the model is not a good estimate of the regression function). On the contrary if the model is non linear (Fig. 4.1-b), the variance can be large (as the solution tends to overfit the data and it does change for different training sets), but the bias is small (as the model is able to reproduce a quadratic polynomial, then a good estimate of the regression function f is possible). In general a good model realizes a trade-off between bias and variance.

The solution can be computed in both parametric and non-parametric approaches mainly using two classes of methods: local methods, that locally search the solution locally averaging the data, and global methods, in which the solution is found solving a global optimization problem.

In the following, some popular parametric and non-parametric regression techniques are presented.

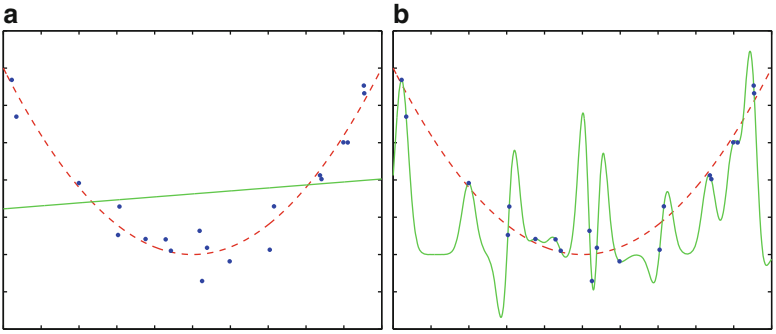

Fig. 4.1 In (**a**) and (**b**) the dashed line represents the target function that has been sampled in the reported points. In (**a**) the solid line is a solution computed by a linear model. In (**b**) the solid line is a solution computed by a complex model. The reconstruction is characterized by a small bias and large variance

4.3 Instance-based regression

Instance-based approaches form a family of non-parametric methods that instead of configuring a model for explicit generalization use a strategy for the prediction based only on the direct comparison of new instances with those contained in the training set. In other words, the parameters of the model are exactly the training set instances, and the processing of training data is deferred to the time when a prediction has to be performed. For this characteristic, such methods are called also *lazy learning* algorithms. This class of algorithms is derived from the *nearest neighbor classifier* [70], which is the simpler member of this family. These methods were initially devoted to the classification [75][15] and they have been extended later to the regression [136] case.

In the instance-based techniques for regression, the examples are composed of a set of input variables, whose value is either categorical or numerical, and the output variables are continuous. The label value predicted for an instance is computed as a function of the training set elements. For example, the *nearest neighbor method* considers the most similar training samples (called neighbors) of the new instance, and it computes the variable value of the new instance as a function (typically the average) of the output variables of its neighbors subset. The similarity can be computed considering the Euclidean distance between instances.

There are many variants of instance-based algorithms in literature. The simplest one is called *proximity algorithm*. The similarity is computed as the complement of the Euclidean distance, $S(x_i, x_j) = 1 - ||x_i - x_j||_1$, where all the input variables have been normalized in the continuous range $[0,\ 1]$. The value of the output variable for a sample x is computed as the weighted sum of the values of the output variables of similar training set elements:

$$\hat{f}(x) = \frac{\sum_{x_j \in T_i} S(x, x_j)\, y_j}{\sum_{x_j \in T_i} S(x, x_j)} \tag{4.2}$$

where T_i is the subset of the training set composed of the training set elements most similar to x.

This method performs good approximation only for a sufficiently large training set and when f can be locally linearized. The method is based on the assumption that all the input variables have the same importance with respect to the output variable. If this assumption is not verified, the method has to be modified applying a weight for each input variable. Since all the training examples have to be stored, for large dataset the amount of memory required can be huge. In order to reduce the quantity of memory needed, averaging techniques have been proposed [15].

The complexity of this class of methods as well as its performance depends on the number of nearest neighbors considered. The most effective number of nearest neighbors for a specific problem can be found using different heuristic techniques such as cross-validation (see Glossary 7.3.2) [139].

4.4 Locally weighted regression

Locally weighted methods [22] belong also to the family of *lazy learning* algorithms. Like the instance-based methods, the prediction is operated using a subset of the instances in the training set. Hence, the training set instances, which are represented as points in D-dimensional Euclidean space, strongly influence the prediction on a local basis. The main difference between the instance-based and the locally weighted methods is that, while the formers compute the prediction by averaging the training set elements closer to the considered position, the locally weighted methods perform the prediction using a locally estimated model. The local models are generally linear or non-linear parametric functions [22]. These methods are also based on weighting the data to give more importance to relevant examples and less importance to less relevant examples. The same effect can also be obtained replicating the important instances. The relevance is computed, analogously to the similarity of instance-based methods, measuring the distance between a new instance and each training point. The functions used to weight the training points contribution are called *kernels*; one of the most used is the Gaussian function. For example the prediction for the instance x can be obtained using the Nadaraya-Watson estimator [55] as:

$$\hat{f}(x) = \frac{\sum_i y_i\, K_\sigma(x_i, x)}{\sum_i K_\sigma(x_i, x)} = \frac{\sum_i y_i\, e^{\frac{-||x_i - x||^2}{\sigma^2}}}{\sum_i e^{\frac{-||x_i - x||^2}{\sigma^2}}} \tag{4.3}$$

where the sums are limited to an appropriate neighborhood of x, and σ is a parameter which controls the width of the influence region of the kernel.

The actual form of the solution, $\hat{f}$, has to be chosen in order to minimize a given training error function, L, called loss function:

$$L = \sum_{i=1}^{n} E(\hat{f}(x_i), y_i) \tag{4.4}$$

where E is a general function that measures (or weights) the difference between $\hat{f}(x_i)$ and y_i, i.e., the error made in using $\hat{f}(x_i)$ in place of y_i. Generally, these functions are of two kinds: squared error, $E(\hat{f}(x_i), y_i) = (\hat{f}(x_i) - y_i)^2$, and absolute error, $E(\hat{f}(x_i), y_i) = |\hat{f}(x_i) - y_i|$.

It can be shown that weighting the loss function (such that nearby instances are more considered than those that are distant), is equivalent to directly applying a weighting function on the data. For instance, considering a constant local model as the solution, $\hat{y}$, in order to require that it fits the nearby points well, the loss function can be chosen as:

$$L(x) = \sum_{i=1}^{n} (\hat{y} - y_i)^2 K(x_i, x) \tag{4.5}$$

It can be shown that the best estimate $\hat{y}(x)$ that will minimize the loss function $L(x)$ is $\hat{y} = \hat{f}(x)$, where $\hat{f}(x)$ is computed as in (4.3).

Different weighting functions can be applied to this approach [22] and this choice can produce very different performances. Locally weighted methods have shown higher flexibility and interesting properties, such as smoothness, than the instance-based methods. However, the choice of an appropriate similarity function is critical for the performance of the model. This can be a problem when it is difficult to formalize when two points can be considered close. As mentioned before, the computational complexity of the training phase is reduced to the minimum (it is realized just storing the dataset), but the prediction phase can be computationally expensive. In order to limit the cost of the prediction for large data sets, suitable data structures can be used for storing the training set. For instance in [22] a *k-D* tree data structure is used for speeding up this phase. A *k-d* tree is a binary data structure that recursively splits a D-dimensional space into smaller subregions in order to reduce the search time of the relevance points for an instance.

4.5 Rule induction regression

Rule induction regression methods have been developed for the regression problems which provide interpretable solutions. These methods have the aim of extracting rules from a given training set. A common format for interpretable solutions is the Disjunctive Normal Form (DNF) model [244]. Rule induction models have been initially introduced for classification problem [244] and then extended for regression [245].

These approaches have characteristics similar to *decision trees* [49], as both models induce a recursive subdivision of the input space, but the rule induction algorithms, generally, provide models with better explanatory capabilities since the rules found are not mutually exclusive. Furthermore, these rules are more compact and have a greater predictive power than decision trees [245], while decision trees show better performances when many high order dependencies among the input variables exist [102]. Decision trees are extremely suited to find the relevant input variables in high dimensional cases when only a small subset of them is meaningful; one of their disadvantage is that partitioning of the input space can cause discontinuities in prediction at the regions boundaries.

The solution provided by a regression tree is generally composed by a pool of *if-then* rules, such as, for example:

$$\text{if } x \in R_k \text{ then } \hat{f}(x) = a_k = \text{median}\{y_i^k\} \tag{4.6}$$

where R_k is a region of the input space, a_k is a constant, and $\{y_i^k\}$ is the set of training points that lie inside R_k. In this example, the output variable value for the sample x is computed as the median of the output variables of the training points that lie inside R_k. This is a common choice, as the median is the minimizer of the mean absolute distance.

The general procedure for obtaining the regression tree is described in the following. The tree is initialized by associating the whole training set to the root of the tree. Then a recursive splitting procedure is applied to each node, until the cardinality of the dataset associated to each node is below a given threshold. For this purpose, for each node, the single best split (e.g., the one that minimizes the mean absolute distance of the training examples of the partitioned subset) is applied, generating two children for each node. As the goal is to find the tree that best generalizes new cases, a second stage of pruning generally follows to eliminate the nodes that cause overfitting.

In the regression trees domain, a single partition of R_k represents a rule, and the set of all disjoint partitions is called rule-set. In rule induction approach, instead, the regions for rules need not be disjointed: a single sample can satisfy several rules. Hence, a strategy to choose a single rule to be applied from those that are satisfied is needed. In [244], the rules are ordered according to a given criterion (e.g., the creation order). Such ordered rule-set is called *decision list*. Then, the first rule in the list is selected.

The rule-based regression model can be built, as the regression tree, adding a new element at a time (i.e., the one that minimizes the distance). Each rule is extended until the number of training examples which are covered by it, drops below a given threshold. The covered cases are then removed and rule induction process can continue on the remaining cases. This procedure is very similar to the classification case [167][65].

In [245] a rule-induction regression algorithm based on the classification algorithm *Swap-1* [244] is presented. Swap-1 starts with a randomly chosen set of input variables to create a candidate rule and swaps all the conjunctive components with

all the possible components. This swap includes the deletion of some components from the candidate rule. The search finishes when there are no swaps that improve the candidate rule anymore, where the evaluation of each candidate rule is performed on their ability to predict the value of the output variables on the examples that satisfy the rule condition. Each time a new rule is added to the model, all the examples covered by that rule are removed. This procedure is iterated until the training set becomes empty. A pruning step is performed after the creation of the rule-set: if the deletion of a rule does not decrease the accuracy of the model the rule is deleted.

Since Swap-1 is designed for categorical input variables, its regression version implies a preprocessing to map the numeric input variables into categorical ones. This is realized using a variant of the *K-means* algorithm [134]:

- the output value of the examples y_i is sorted;
- an approximately equal number of contiguous values y_i is assigned to each class;
- an example is moved to an adjacent class if the distance between the example and its class mean is reduced.

The rule induction regression algorithm is realized computing Swap-1 on the mapped data and then pruning and optimizing the rule-set. After the pruning, the optimization is performed by searching the best replacement for each rule such that the prediction error is reduced [245].

The predicted value can be computed as the median or mean value of the class elements, but a parametric or a non-parametric model can also be applied to each class.

4.6 Projection pursuit

Local averaging produces a poor accuracy when the input space has a large numbers of attributes (*curse of dimensionality*). In fact, data density decreases exponentially with the increase in the number of dimensions [103][114] [21]. This means that in higher dimensional spaces, the number of training points should be huge for computing meaningful local average, and this is rarely true in real applications.

The Projection Pursuit Regression approach [138] is more suited to these cases. The idea is to compute a global regression by using an iterative procedure that performs successive refinements. The procedure computes at each step a smooth approximation of the difference between the data and what has been already modeled in the previous steps. The model is represented by the sum of the smooth approximations, S_m, determined at each iteration:

$$\hat{f}(x) = \sum_{m=1}^{M} S_m(\beta_m^T \cdot x) \tag{4.7}$$

where β_m is the parameter vector (projection), x is an instance, S_m is a smooth function, and M is the total number of sub-models computed (number of iterations).

The critical part of the algorithm is the a priori choice of the smooth functions and the determination of the β_m. The β_m vector can be found according to a fitting criterion that requires the minimization of a given loss function. In particular β_m has to maximize the following:

$$\beta_m = \max_{\beta} R(\beta) = \max_{\beta} 1 - \frac{\sum_{i=1}^{n}(r_{i,m} - S_m(\beta^T \cdot x_i))^2}{\sum_{i=1}^{n} r_i^2} \tag{4.8}$$

where $r_{i,m}$ represents the error on the i-th training point at the m-th iteration, $r_{i,0} = y_i$. The iteration steps are performed until $R(\beta)$ is greater than a given threshold. The smooth function S can be chosen in many different ways [138][201][198]. In [201] the performances of the algorithm are compared for three different smooth functions: smoothing splines, super-smoothers, and polynomials. Also smooth functions based on local averaging of the residuals [103] can be used.

4.7 Multivariate Adaptive Regression Splines

As mentioned in Sect. 4.5, tree-based recursive partitioning regression suffers from lack of continuity at the partitions boundaries, which affects the accuracy. Furthermore tree methods are not able to provide good approximation for some classes of target functions (e.g., linear or additive functions). The Multivariate Adaptive Regression Splines (MARS) model addresses these two problems [102]. The solution computed has the following form:

$$\hat{f}(x) = c_0 + \sum_{m=1}^{M} c_m B_m(x) \tag{4.9}$$

where c_i is a constant, B_m is a basis function, and M is the total number of basis functions. Hence, the output of the model is computed as a linear combination of basis functions $B_m(x)$. Each basis function $B_m(x)$ takes one of the following three forms:

1. a constant 1
2. a *hinge function*, namely a function of the form $\max(0, x - \tau) = [x - \tau]_+$ or $\max(0, \tau - x) = [\tau - x]_+$, where τ is a constant, called knot, whose value is automatically determined by MARS.
3. a product of two or more hinge functions.

Hence the MARS set of basis functions is:

$$B := \{[x^j - \tau_i^j]_+, [\tau_i^j - x^j]_+ |\ \tau_i^j = x_i^j, j \in \{1, 2, \cdots, D\}, i \in \{1, 2, \cdots, n\}\} \tag{4.10}$$

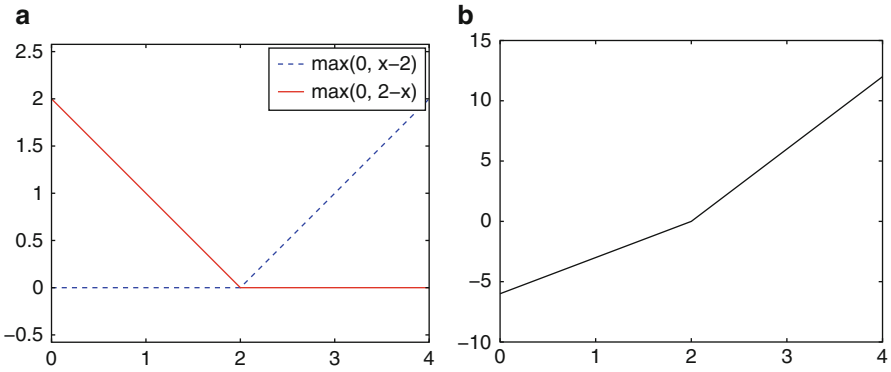

Fig. 4.2 In (**a**) two examples of hinge functions. In (**b**), a piecewise linear function defined as a linear combination of the functions in (**a**) as $f(x) = 6[x-2]_+ - 3[2-x]_+$

where D is the input space dimension, n is the number of training points, x_i^j denotes the value of j-th input variable for the i-th sample. If all the training points are distinct, there are $2\,D\,n$ basis functions.

Hinge functions are a key part of MARS models. In Fig. 4.2a are reported two examples of hinge functions. A pair of hinge functions such as that in Fig. 4.2a is called *reflected pair*. As the hinge function is zero in part of its support, it can be used to partition the data into disjoint regions, each of which can be processed independently. In particular the use of hinge functions allows building piecewise linear and non-linear functions (Fig. 4.2b). MARS operates by adding a pair of functions that are symmetrical with respect to the knot point which corresponds to each training point, in 1D spaces. In higher dimensional spaces, a function is added for each component of the training point.

The MARS algorithm presented in [102] is composed by two procedures:

Forward stepwise The basis functions are identified starting from the constant basis function, the only initially present. At each step, the split that minimizes a given approximation criterion among all the possible splits of each basis function is chosen. This procedure stops when the maximum value, chosen by the user, $M_{\max}$, is reached. Since the model found generally overfits the data, a *backward* pruning procedure is then applied.

Backward stepwise At each step, the basis function whose elimination causes the smallest increase in the residual error is deleted. This process produces a sequence of models, $\{\hat{f}_\alpha\}$, which have an increasing residual error, where α expresses the complexity of the estimate. In order to choose the best estimated model, cross-validation can be applied.

In [225] a variant of the MARS algorithm in which the backward procedure is not used is proposed. This is substituted by a penalized residual sum of squares introduced in the forward procedure, and the problem is reformulated as a *Tikhonov regularization problem*.

4.8 Artificial Neural Networks

Artificial Neural Networks (ANN) is a family of models belonging to the class of parametric approaches [37][83][121][190], which have been inspired by the computational model of the human brain neural network. Among the models of this family, the most common is the feedforward multilayer perceptron (MLP). This model is composed of a sequence of layers of computational units, called neurons, which perform a simple processing of the inputs that they receive. Each neuron is characterized by a function, called activation function, which is used for the computation of its output. Among others, the linear and logistic (reported in the following) functions are commonly used for this aim:

$$B(z) = \frac{1}{1 + \exp(-z)} \tag{4.11}$$

In Fig. 4.3, a scheme of the MLP model is depicted. The MLP processes the data layer-wise: each layer receives the input from the output of the previous layer and provides the input to the next layer. Usually, only the neurons of consecutive layers are connected. Each connection is characterized by a scalar parameter called weight, which is a multiplicative coefficient that is applied to the value transmitted by the connection itself. The input of each unit is the weighted sum of all the contributions carried by the incoming connections; hence, it is a linear combination of the output of the units of the previous layer. The first and the last layers are generally composed of linear neurons, while the intermediate layers, called hidden layers, are composed of non-linear (commonly logistic) neurons. The scheme in Fig. 4.3 represents a MLP with d input units, a single hidden layer, and a single output unit.

The MLP models can be used to solve efficiently a large class of regression and classification problems. It has been proved that MLP with at least one hidden layer is a universal approximator [126]. However the addition of extra hidden layers may

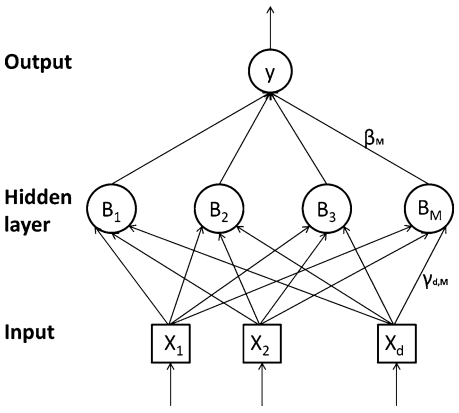

Fig. 4.3 A scheme of single hidden layer feedforward neural network model

allow to perform a more efficient approximation with fewer units [37]. The output of the network of Fig. 4.3 is computed as:

$$\hat{f}(x) = \sum_{m=1}^{M} \beta_m B(\gamma_m^T \cdot x) \tag{4.12}$$

where M is the number of hidden units, β_m is the weight (a scalar) of the connection between the output unit and m-th hidden unit, γ_m is the weights vector of the connections between the input units and the m-th hidden unit.

In general the neural networks can have more complex schemes than that shown in Fig. 4.3. For example, in recurrent neural networks the output is fed-back to the input or to the intermediate layers. In fully connected networks, the input of the units in the lower layers is sent to all the units of the higher layers and not only to those of the next one. The number of hidden units of a network determines the trade-off between bias and variance of the model (see Sect. 4.2). The more hidden units are used in a network, the better the training data are fit. However if a too large number of units is used, the model can show overfitting and a poor predictive capability.

The parameters of a neural network are found, generally, minimizing a loss function (as in Sect. 4.4) that measures the distance between the output measured in the input data set and the output computed by the network for the same input values. The distance function usually adopted is sum of squares (Euclidean distance).

Since MLP output is not linear (and the corresponding loss function is not quadratic) with respect to the model parameters, the optimal value of the parameters cannot be found directly (e.g., as solution of a linear system). Hence iterative optimization is usually performed, based on the computation of the gradient of the loss function. The gradient can be computed in several ways; the most popular algorithm is *backpropagation* algorithm [205] because of its computational efficiency. It exploits the property of the derivatives of a composite function for decomposing the gradient in simpler expressions. Starting from the output layer, the gradient of the loss function can be decomposed in its components relative to each unit of the previous layer and the recursive application of the above mentioned property allows to propagate the computation of the gradient with respect to the parameters of the network toward the input layer.

The gradient of the loss function is then used in a minimization algorithm. There are several iterative methods to perform the minimization of the loss function. The simple gradient descent is one of them. The updating of the parameters is computed as a fraction, η, of the gradient of the loss function. The choice of the proper value of η, called learning rate, is generally, difficult. A too small value of η, can produce many steps to reach a minimum, while a too large value can produce an oscillating behavior around the optimal value, and the minimum is never reached. To speed up the convergence, the η parameter can be varied during the learning phase and many heuristics have been proposed to this aim.

However, gradient descend is, generally, an inefficient method as all first order methods are, and second order approaches have been explored. The second order methods are based on the fact that in a minimum of a function, its gradient is zero. As a consequence, the Taylor expansion of the function computed with respect to the minimum will not have first order terms. Hence, for points close to the minimum, the function can be effectively approximated by using only the second order derivatives and the value of the function in the minimum. In particular, the Taylor expansion of a function, $f(\beta)$, of a single variable β, close to a minimum β^* results:

$$f(\beta) = f(\beta^*) + \frac{1}{2}(\beta - \beta^*)^2 \left(\frac{d^2 f}{d\beta^2}\right)_{\beta=\beta^*} + \mathrm{O}(\beta^3) \tag{4.13}$$

The derivative of the previous equation with respect to β gives an approximation of the gradient close to a minimum β^*:

$$\frac{d f(\beta)}{d\beta} = (\beta - \beta^*) \left(\frac{d^2 f}{d\beta^2}\right)_{\beta=\beta^*} \tag{4.14}$$

It results that the step that separate the considered point β from the minimum, $(\beta - \beta^*)$, can be computed directly as the ratio between the first derivative of the function with respect to β and its second order derivative in the minimum. This result can be extended to the multi-variate case for which the second derivative becomes the Hessian matrix, $H(\cdot)$, evaluated in the minimum β^*. As the Hessian matrix (in the minimum) is unknown, it has to be approximated. For this aim, several iterative techniques based on the use of different approximations have been developed. Two popular methods are the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [52] and the Levenberg-Marquardt (LM) algorithm [148][158]. The BFGS requires that the starting parameter vector is close to the minimum to be effective: other techniques, like those based on the simple gradient descent, can be to reach a good starting point. The LM algorithm does not suffer from this problem, but it requires more memory than BFGS as it has to store several matrices. Hence, for large networks, it becomes inefficient.

The determination of the parameters through iterative optimization procedures is called *learning* in neural networks terminology. Other equivalent terms often used in this domain are *configuration*, that is used when also the number of units is considered in the optimization procedure (i.e., it includes the model selection) and *training*, that puts the accent on the iterative use of the data to compute the parameters by changing their value to the optimum.

In the next chapters two kinds of neural models, namely Radial Basis Function Neural Networks and Support Vector Machines, and their hierarchical version, will be treated in depth. Both these models can be seen as particular MLPs with a single layer that make use of alternative training technique.

4.9 Wavelet regression

An interesting approach to perform function approximation is *wavelet regression* [76] [77] [80]. This approach derives from the unification and development of many ideas from the field of filtering theory, physics and applied mathematics. The *wavelet* transform is a representation of a function $f(x) \in L^2(\mathbb{R})$ using a linear combination of basis function. The basis functions, called *wavelets*, are generated by scaling and translating the same function called *mother wavelet*, $\psi(x) \in L^2(\mathbb{R})$. Hence, the actual shape of each wavelet component $\psi_{a,b}(x)$ depends on two real parameters, a and b that represent the scale and the position of the wavelet, respectively.

The computation of the solution is based on the Multi-Resolution Analysis (MRA). MRA defines for a function $f(x)$ a series of approximating function $\{P_j[f(x)]\}_{j \in Z}$ that converges to the function itself. The function $P_j[f(x)]$ is a linear combination of functions obtained by translating a single function, ϕ, called *scaling function*: $P_j[f(x)] = \sum_k \lambda_{j,k}\phi_{j,k}(x)$. The shape of the scaling function depends on a scale parameter j that determines the approximation resolution while the parameter k controls its position in the input domain. It has been proved [157] that there exists a sequence $\{h_k\}$ such that:

$$\phi(x) = 2\sum_k h_k \phi(2x - k) \tag{4.15}$$

This relation is called *refinement equation*.

The difference between an approximation and the next one is called *detail*, and can be expressed as: $P_{j+1}[f(x)] - P_j[f(x)] = Q_j[f(x)]$. The function $f(x)$ can be expressed as:

$$f(x) = \sum_j Q_j[f(x)] = \sum_{j,k} \gamma_{j,k}\psi_{j,k}(x) \tag{4.16}$$

Then the function $f(x)$ can be obtained also as linear combination of the detail functions which can be obtained as a linear combination of copies (scaled and translated) of the wavelet. As for the scaling function, it has been proved [157] that there exists a sequence $\{g_k\}$ such that:

$$\psi(x) = 2\sum_k g_k \psi(2x - k) \tag{4.17}$$

If an orthogonal basis is chosen, the coefficients $\{\lambda_{j,k}\}$ can be obtained by the orthogonal projections of the function $f(x)$ on the corresponding scaling function:

$$\lambda_{j,k} = \langle f, \phi_{j,k} \rangle \Rightarrow P_j[f(x)] = \sum_k \langle f, \phi_{j,k} \rangle \phi_{j,k}(x) \tag{4.18}$$

In a similar way, the coefficient $\gamma_{j,k}$ can be obtained as the projection of $f(x)$ on the wavelet:

$$\gamma_{j,k} = \langle f, \psi_{j,k} \rangle \Rightarrow Q_j[f(x)] = \sum_k \langle f, \psi_{j,k} \rangle \psi_{j,k}(x) \tag{4.19}$$

The orthogonality constraint limits the construction of wavelets. For example, the Haar wavelet is the only wavelet that is symmetric and orthogonal with real values defined on a compact domain. The definition on a compact domain allows an accurate implementation of the wavelet transformation. The sequences $\{h_k\}$ and $\{g_k\}$ (4.15, 4.17) can be seen as the coefficients of a pair of digital filters (more precisely, Finite Impulse Filters). These filters are characterized by a finite number of not zero coefficients. Therefore, the coefficients of the transformation can be computed by means of the convolution of the function f with these filters.

Relaxing the orthogonality constraint by defining an MRA which makes use of biorthogonal basis allows obtaining two pairs of filters. The first one can be used for the transform (i.e., for computing the coefficients), while the second one can be used for the inverse transform (i.e., for computing the function from the coefficients). The generalization to biorthogonal wavelet allows more flexibility in the choice of the basis functions. In this case, there are two additional functions $\tilde{\phi}(x)$ and $\tilde{\psi}(x)$, that represent the *dual scaling function* and *dual wavelet*. These functions generate a dual MRA.

In the case of a pair of biorthogonal MRA, one can be used for computing the coefficients and the other for the approximation of the function. The projection operators P_j and Q_j become:

$$P_j[f(x)] = \sum_k \langle f, \tilde{\phi}_{j,k} \rangle \phi_{j,k}(x) \tag{4.20}$$

$$Q_j[f(x)] = \sum_k \langle f, \tilde{\psi}_{j,k} \rangle \psi_{j,k}(x) \tag{4.21}$$

The discrete wavelet transformation assumes the form:

$$f = \sum_{j,k} \langle f, \tilde{\psi}_{j,k} \rangle \psi_{j,k} \tag{4.22}$$

The properties of orthogonality and biorthogonality allow the definition of a fast algorithm for the computation of the wavelet transform. This algorithm exploits the relation between the basis functions and the filters associated to them. The projection operations on the approximation and detail spaces are formulated as filtering operations using low-pass FIR and high-pass FIR filters.

The original formulation of the wavelet is based on the constraint of regularly spaced input data. In the regression problems this constraint is generally not satisfied. A method called *Lifting Scheme* [220] [221] [222] defines an approach

to use wavelet for the case of not regularly spaced data. In the Lifting Scheme the computation of the wavelet is divided in a chain of weighted averaging and sub/over sampling operations. If the data are not regularly spaced, the basis functions are not scaled and translated copies of the same function, but they have to be adapted to the training data. The increased flexibility of the method is paid with a higher complexity of the analysis-synthesis procedure. When the data are sparse, a connectivity structure which describes the topological relationship between the data is required.

Chapter 5
Hierarchical Radial Basis Functions Networks

In this chapter a particular kind of neural model, namely the Hierarchical Radial Basis Function Network, is presented as an effective hierarchical network organization. In a similar way, in the next chapter another kind of multi-scale model, namely Support Vector Machines, will be presented.

5.1 Radial Basis Functions Networks - RBF

In this chapter, a particular kind of neural network, which belongs to the perceptron family [121], is considered. The models of this family are characterized, as the other neural networks models, by the activation function of their neural units and the connection scheme. In particular, the units with radial symmetry allow obtaining a good approximation with a limited number of units. Each unit is characterized by a *width* that defines its influence region: only the input inside this region produces a significant activation of the related unit. This locality property contributes to improve the efficiency of the learning process, as only a subset of the training data can be considered for the configuration of each unit. Moreover, locality allows avoiding the interference of examples in distant regions that can produce a collinear error on the output of a layer that compensates the error generated by closer points, with the result that the learning procedure may get trapped in a local minimum. The networks based on such units are called *Radial Basis Functions* (RBF) Networks [190][169][188].

Using Gaussians as basis functions, a RBF network can be described as a function $\hat{f}(x) : \mathbb{R}^D \rightarrow \mathbb{R}$, as follows:

$$\hat{f}(x) = \sum_{k=1}^{M} w_k G(x; \mu_k, \sigma_k) = \sum_{k=1}^{M} w_k \frac{e^{-\frac{||x-\mu_k||^2}{\sigma_k^2}}}{\sqrt{\pi^D}\sigma_k^D} \tag{5.1}$$

F. Bellocchio et al., *3D Surface Reconstruction: Multi-Scale Hierarchical Approaches*,
DOI 10.1007/978-1-4614-5632-2_5, © Springer Science+Business Media, LLC 2013

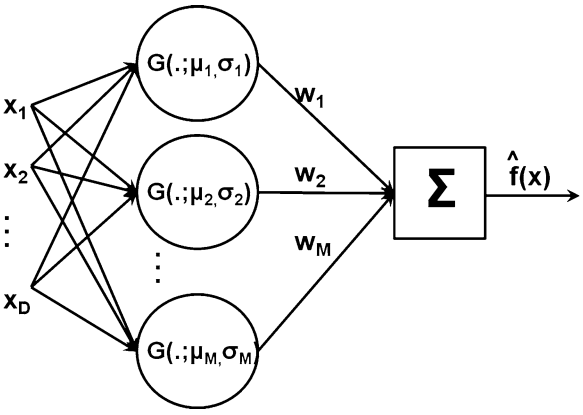

Fig. 5.1 The RBF network structure. Each Gaussian (represented here with a circle) receives the input $\boldsymbol{x}$; the output of the network is computed as the sum of the contribute of each Gaussian, k, weighted by its coefficient w_k

where M is the number of Gaussian units used, $w_k \in \mathbb{R}$ is the multiplicative coefficient (or weight) of each Gaussian, $\sigma_k \in \mathbb{R}$ is the width of the k-th Gaussian, $\mu_k \in \mathbb{R}^D$ is its position in the input domain and D is the input space dimension. σ_k determines the size of the influence region of the k-th Gaussian: in fact, each Gaussian assumes a value significantly higher than zero in the spherical region centered in μ_k and radius proportional to σ_k. We explicitly note that the Gaussians in (5.1) have unitary norm. In Fig. 5.1 a schematic representation of a RBF network is reported.

In this scenario, the learning problem can be formulated as follows. Given a set of points $\{(x_i, y_i)|x_i \in \mathbb{R}^D, y_i \in \mathbb{R}, 1 \leq i \leq n\}$, the goal is to find a set of parameters $\{M, \mu_k, \sigma_k, w_k\}$ such that a given error function $E(\hat{f}(x), y)$ is minimized.

The determination of the optimal parameters is a non-trivial problem as it implies to solve a non linear optimization problem that easily has many local minima.

Moreover, the problem is ill-posed, because the number of surfaces that pass through the given points is infinite. In order to choose the best approximation among the possible solutions, some constraints have to be introduced, as well as a criterion to evaluate the suitability of each solution. The smoothness of the solution is a natural constraint: small displacements of a point have to correspond to small variations of the surface.

These constraints can be exploited inside the regularization framework where the learning problem is formulated introducing a smoothness term in the cost function that has to be minimized [189][112][253]. It has been proved that a linear combination of Gaussians centered in the data points is able to minimize such cost function. Despite this clear formulation, in general, an efficient algorithmic solution does not exist [243][172]. Simple techniques based on stochastic gradient have been proposed [112], but they suffer of local minima, which may prevent reaching the optimal solution.

To find the global minimum, specialized and computationally intensive algorithms have been proposed and they are based on exploring exhaustively the parameters space [33][62][113][137][173][36]. Although, in principle, these tech-

niques are able to find the global minimum, they are based on the exhaustive exploration of parameters space and then they are therefore unfeasible in real scenarios because they are computationally expensive.

A different strategy, called *hybrid learning* [169][51], derives from the observation that the parameters in (5.1) can be divided in two classes depending on their role: structural parameters and synaptic weights. M, $\{\mu_k\}$ and $\{\sigma_k\}$ belong to the first class, while the weights, $\{w_k\}$, belong to the second class. The different nature of these parameters suggests to use different algorithms to determine their values. With hybrid techniques the position of the units $\{\mu_k\}$ can be determined by using clustering algorithms (e.g., [159][94] and the number of units, M, can be chosen a priori. The parameters $\{\sigma_k\}$, define the behavior of the function in regions between the samples and they can be determined using heuristics [169][188][178][216]. When the structural parameters are set, the equation (5.1) becomes a linear system where the unknowns are the weights, $\{w_k\}$. Although the weights can be computed solving the system, for networks with high number of units this solution is not feasible because of numerical instability or memory allocating problems and different strategies have been explored. These are based on estimating the weights of the units starting from the data local to each unit.

The growing structures [188][104][105] are an improvement over hybrid schemes. The number of units is not given a priori, but the units are added to the network one after the other until a given criterion is satisfied. These schemes are iterative and a good estimate of the network parameters, generally, requires a time consuming learning phase. A similar approach is followed by boosting regression [197].

Another approach, pioneered in digital filtering and in *computer vision*, is based on the use of regularly spaced Gaussians placed on a lattice covering the input space [185][207]. The disadvantage of this approach is the rigidity of the structure used. The distance among the units is the same in all the input space: the resulting function can suffer of *overfitting* in some regions and is not able to reproduce the finest details in other regions. Furthermore, the presence of overfitting would imply a waste of resources due to the use of too many units. This problem can be solved in several ways.

In [64] a method based on building a model adding each time an orthogonal basis is proposed. Such model allows both improving the generalization ability of the previous model and reducing the number of the units. The weights of the orthogonal basis are found by an iterative minimization of a cost function. The technique is based on a modified Gram-Schmidt orthogonalization procedure and can be computational intensive for large networks. Besides, numerical problems may arise.

A different approach is represented by a model called *Hierarchical Radial Basis Functions* (HRBF) [46][97][89][29]. It is based on the idea of inserting units only where they are needed without resorting to any iterative procedure to determine the network parameters. HRBF is one of the two hierarchical models considered in the present book for surface reconstruction and the remaining sections of this chapter are dedicated to it. It enables the non-iterative computation of the

network parameters using a limited number of hyperparameters (Sect. 5.2), which can be intuitively related to the computational resources budget. In Sect. 5.3 an incremental, on-line configuration procedure will be presented.

5.2 Hierarchical Radial Basis Functions Networks - HRBF

The *Hierarchical Radial Basis Functions* (HRBF) network [46][97][89][29] combines the characteristics of growing structures and considerations grounded in signal processing, allowing a fast and robust estimate of both the structural parameters and the weights of the network. The HRBF configuration algorithm can be derived starting from a simple RBF network with a particular structure and expliciting its similarity with a Gaussian digital filter which allow to devise a non-iterative procedure for computing the parameters of the RBF network directly from the data. This will be done in Sect. 5.2.1, while in Sect. 5.2.2 the above mentioned procedure will be exploited for formulating a hierarchical approach that allow to adapt the computational resources to the frequency content of the data. The resulting algorithm will be summarized and extended to the surface reconstruction problem in Sect. 5.2.3, while in Sect. 5.2.4 the approximation properties of the HRBF networks will be analyzed.

5.2.1 Gaussian Regular RBF networks and Gaussian Filters

For sake of simplicity, the reconstruction of functions $\mathbb{R} \rightarrow \mathbb{R}$ is considered for the description of the HRBF model. Then the concepts will be extended to the case of a surface in 3D space: $\mathbb{R}^2 \rightarrow \mathbb{R}$. In the $\mathbb{R} \rightarrow \mathbb{R}$ case, the output of the RBF model in (5.1) assumes the form:

$$\hat{f}(x) = \sum_{k=1}^{M} w_k G(x; \mu_k, \sigma_k) = \sum_{k=1}^{M} w_k \frac{e^{-\frac{||x-\mu_k||^2}{\sigma_k^2}}}{\sqrt{\pi}\sigma_k} \tag{5.2}$$

In the following, a RBF network composed of regularly spaced Gaussian units which have the same standard deviation, σ, will be considered and referred as regular RBF network. Since the size of the region of the input domain in which each Gaussian affects the approximation is proportional to σ, this parameter describes also the scale at which the RBF operates (σ is also referred to as scale parameter). In this case, (5.2) can be reframed as the convolution of the weights sequence $\{w_k\}$ with a Gaussian function $G(x; \sigma)$, more precisely with a regular sampling of $G(\cdot)$. Filtering theory concepts can be applied to find an efficient algorithm for computing the value of the structural parameters and the weights of a regular RBF network. Some concepts useful to characterize a Gaussian filter will be introduced

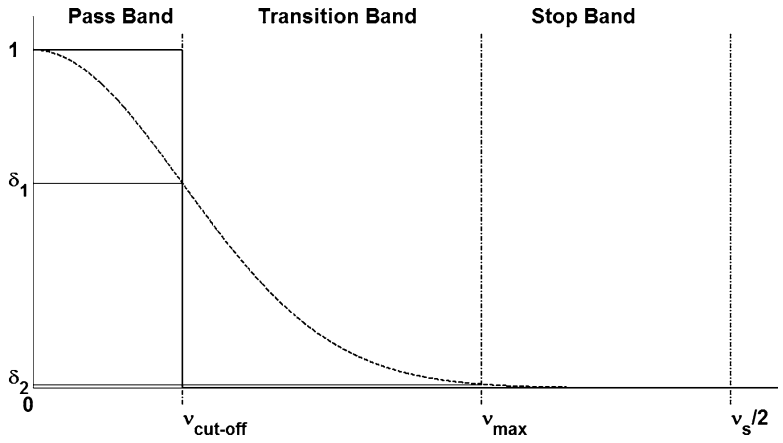

Fig. 5.2 Gaussian filter and its characterizing parameters

in Sect. 5.2.1.1. Then, in Sects. 5.2.1.2 and 5.2.1.3, the regular RBF network model will be generalized to models that simplify the theoretical derivation of the HRBF learning algorithm.

5.2.1.1 Gaussian Filters

The ideal low-pass filter keeps all the components at frequency lower than $\nu_{\text{cut-off}}$ unaltered and deletes all the components at higher frequencies. This filter is not realizable in practice. In the frequency domain, a real low pass filter, $\tilde{F}(\nu)$, is characterized by two frequencies, $\nu_{\text{cut-off}}$ and $\nu_{\max}$, that identify three bands: *Pass Band*, *Stop Band* and *Transition Band* (Fig. 5.2). Two suitable thresholds δ_1 and δ_2, $\nu_{\text{cut-off}}$ and $\nu_{\max}$ are identified by the following conditions:

$$\delta_1 \leq |\tilde{F}(\nu)| \leq 1 \quad \text{for} \quad 0 \leq \nu \leq \nu_{\text{cut-off}} \tag{5.3}$$

$$0 \leq |\tilde{F}(\nu)| \leq \delta_2 \quad \text{for} \quad \nu_{\max} \leq \nu \leq \infty \tag{5.4}$$

In the Pass band, the frequency components are kept almost unchanged, in the second one, the Stop band, they are almost deleted and in the last one, the Transition band, they are attenuated progressively. The Gaussian filter (Fig. 5.2) is formulated as:

$$G(x, \sigma) = \frac{1}{\sqrt{\pi}\sigma} e^{-\frac{x^2}{\sigma^2}} \tag{5.5}$$

and its Fourier transform is:

$$\tilde{G}(\nu, \sigma) = e^{-\pi^2 \sigma^2 \nu^2} \tag{5.6}$$

The following relationships exist between σ and the cut-off and maximum frequency in equations (5.3) and (5.3):

$$e^{-\pi^2\sigma^2\nu_{\text{cut-off}}^2} = \delta_1 \Rightarrow \nu_{\text{cut-off}} = \frac{\sqrt{-\log \delta_1}}{\pi\sigma} \tag{5.7}$$

$$e^{-\pi^2\sigma^2\nu_{\text{max}}^2} = \delta_2 \Rightarrow \nu_{\text{max}} = \frac{\sqrt{-\log \delta_2}}{\pi\sigma} \tag{5.8}$$

5.2.1.2 Regular continuous RBF networks

Equation (5.2) describes a neural network that, in practical cases, is constituted of a finite number of units. Let us consider this as a particular case of the general case in which the network is constituted of an infinite number of contiguous equally spaced units, which will be referred in the following as the continuous RBF network. This continuous RBF network is described by a support region $\mathbb{U} \subset \mathbb{R}$ (the input range where the Gaussians are defined), a weight function $w(x) : \mathbb{U} \to \mathbb{R}$ (that represents the density function of the weights in (5.2)) and a function $\sigma(x) : \mathbb{U} \to \mathbb{R}^+$. If the distance between the Gaussians tends to zero, equation (5.2) can be generalized using a regular continuous RBF network in which $\mathbb{U} \equiv \mathbb{R}$ and $\sigma(x) = \sigma$:

$$f(x) = \int_{\mathbb{R}} w(c)\, G(x - c|\sigma)\, \mathrm{d}c \; = w(x) * G(x;\sigma) \tag{5.9}$$

Due to the convolution theorem [133], (5.9) implies:

$$\tilde{f}(\nu) = \tilde{w}(\nu)\tilde{G}(\nu;\sigma) \tag{5.10}$$

where the function $f(x)$ is obtained from the function $w(x)$. In the regression, however, we have the inverse problem: we have to find a rule for obtaining a function $w(x)$ that provide a good estimate of $f(x)$. If $w(x)$ were replaced by the function $f(x)$ itself, the function $\hat{f}(x)$ would be obtained as:

$$\hat{f}(x) = \int_{\mathbb{R}} f(c)\, G(x - c|\sigma)\, \mathrm{d}c \; = f(x) * G(x;\sigma) \tag{5.11}$$

and in the frequency domain:

$$\tilde{\hat{f}}(\nu) = \tilde{f}(\nu)\tilde{G}(\nu;\sigma) \tag{5.12}$$

From (5.12) it is clear that $\hat{f}(x)$ will be a smooth version of $f(x)$. In fact all the $f(x)$'s frequency components over $\nu_{\text{cut-off}}$ will be attenuated progressively by the convolution with the Gaussian filter.

In general, a good approximation of $f(x)$ can be obtained if it does not contain significant frequency components over the cut-off frequency ν_M of $w(x)$. Therefore, the parameter σ in (5.7) has to be set such that the following constraint is satisfied:

$$\nu_{\text{cut-off}} > \nu_M \tag{5.13}$$

5.2.1.3 Regular discrete RBF networks

In real cases there is a limited number, N, of samples. Let us consider the case in which a function, f, is reconstructed from a regular sampling of the function itself, $\{(x_i,\, f_i) \,|\, f_i = f(x_i)\}$, with sampling step Δx, using a linear combination of Gaussians centered in the samples. In the points $\{x_i\}$ the function can be reconstructed as:

$$\hat{f}(x_i) = \sum_{k=1}^{N} f_k\, G(x_i; x_k, \sigma)\Delta x \;=\; \frac{\Delta x}{\sqrt{\pi}\,\sigma} \sum_{k=1}^{N} f_k e^{-\frac{(x_i - x_k)^2}{\sigma^2}} \tag{5.14}$$

Equation (5.14) can be extended on the whole real line using the following interpolation:

$$\hat{f}(x) = \sum_{k=1}^{N} f_k\, G(x; x_k, \sigma)\Delta x \;=\; \frac{\Delta x}{\sqrt{\pi}\,\sigma} \sum_{k=1}^{N} f_k e^{-\frac{(x - x_k)^2}{\sigma^2}} \tag{5.15}$$

In (5.15), also the Gaussian filter is sampled, with sampling frequency $\nu_s = \frac{1}{\Delta x}$. This fact, for the sampling theorem, introduces another constraint:

$$\nu_{\text{max}} < \frac{\nu_s}{2} \tag{5.16}$$

Relations (5.7, 5.8) can be modified as follows:

$$e^{-\pi^2\sigma^2\nu^2_{\text{cut-off}}} \;=\; \delta_1 \tag{5.17}$$

$$e^{-\pi^2\sigma^2\nu^2_{\text{max}}} \;=\; \frac{\delta_2}{2} \tag{5.18}$$

δ_2 has been halved because, when $\nu_{\text{max}} = \nu_s/2$, the Gaussian receives the same contribution from both the main lobe and from the closest replica (aliasing effect). The contribution of the other Gaussians replicas could also be considered, but they are too far and their contribution is too small for significantly influence the reconstruction.

It can be proved that with a sampling frequency of ν_s, the reconstructed function does not have (significant) components over $\nu^\star_M$, where:

$$\nu^\star_M = \sqrt{\frac{\log\,\delta_1}{\log\,\delta_2/2}}\,\frac{\nu_s}{2} \tag{5.19}$$

Equation (5.19) can be compared with Shannon constraint. The latter asserts that the maximum frequency that can be reconstructed by a sampled signal is equal to half the sampling frequency. Since $\delta_1 > \delta_2$, the maximum reconstructed frequency by the model in (5.15) will be smaller than that indicated by the Shannon theorem. In particular, δ_1 is set to $\sqrt{2}/2$, according with common practice (attenuation of 3 dB), and δ_2 is set to 0.01, the following is obtained from (5.19):

$$\nu_s \;=\; 7.829\,\nu_M^{\star} \tag{5.20}$$

Hence, the sampling frequency should be almost about eight times the maximal frequency that has to be reconstructed. Decreasing the value of δ_1 or increasing the value of δ_2 (i.e., when δ_1 approaches δ_2), the ratio (5.19) tends to two, but the quality of the reconstruction decreases. In particular, if δ_2 increases there will be an increase of aliasing. If δ_1 decreases there will be greater attenuation of higher frequency components.

Using the above relationships and setting the maximum frequency to $\nu_M^{\star}$, a regular discrete RBF network can be configured, setting the number, M, the position, $\{\mu_k\}$, and the width, σ, of the Gaussians (compatible with the sampling frequency ν_s). The weights $\{w_k\}$ will be proportional to the input data themselves.

For example, given $\{(x_i, y_i)|x \in \mathbb{R}, y \in \mathbb{R}, i \in \{1, ..., n\}\}$ a set of regularly sampled points, $\Delta x = x_{i+1} - x_i \;\forall i$, a regular discrete RBF network can reconstruct frequency components up to $\frac{1}{8\Delta x}$ (due to the equation (5.20)). If this is compatible with the maximal frequency component of the function, then the function can be reconstructed by placing a Gaussian unit in every input point: $\mu_k = x_k,\; M = n$. We can derive from the constraints on the frequencies that the value of σ cannot be less than $1.465\Delta x$. The weights correspond to the value assumed by the sampled points times the Δx: $w_k = y_k \Delta x$.

It is worth noting that, given a maximum frequency below $\frac{1}{8\Delta x}$, a suitable subsampling of the data set can be applied to match the associated sampling distance. In this case, the number of Gaussians, their centers are selected according to the reduced data set.

5.2.2 The hierarchical approach

The previous technique is not efficient when the function $f(\cdot)$ presents different frequency content in different regions of the input domain. The presence of high frequency details in just some regions of the input domain would require to use a high number of units also in the regions where the frequency content is low. Instead, these regions would be more efficiently reconstructed with a lower number of Gaussians with a larger σ.

The reconstruction can be realized, in a more efficient way, using more than one network, each one characterized by a different value of σ. The first network has the goal of realizing an approximation of the function at a very large scale, $a_1(x)$. The value of the cut-off frequency of this network is chosen relatively small, say ν_1.

For the estimate of the Gaussian weights a suitable input points subsampling can be realized, and the approximation $a_1(x_i)$ of $f(x_i)$ can be computed for all the examples in the data set.

$a_1(x)$ can be seen as a first rough approximation of the data. The data points will not generally be on $a_1(x)$ and a residual can be defined for each point as:

$$r_1(x_i) \; = \; \{y_i - a_1(x_i)\} \tag{5.21}$$

A second network, characterized by a higher cut-off frequency ν_2, $\nu_2 > \nu_1$, is created aimed to approximating the residuals r_1, thus realizing a second approximating function, $a_2(x)$. The actual approximation of the data is realized adding $a_1(x)$ and $a_2(x)$.

A new value of the residuals, $r_2(x)$ can be computed for the data points as:

$$r_2(x_i) \; = \; \{r_1(x_i) - a_2(x_i)\} \tag{5.22}$$

A third network, with a cut-off frequency ν_3, $\nu_3 > \nu_2$ can now be inserted and the procedure can be iterated until the residual is under a chosen threshold.

The described procedure performs a multi-resolution representation of the sampled function [42] where the representation at the l-th resolution is given by the sum of the output of the first l networks. In real applications, the data are usually not regularly spaced and affected by measurement error. In this scenario the configuration procedure described in the previous section cannot be used anymore. In the following section, this problem will be characterized and addressed for enabling the HRBF configuration algorithm to deal with real data.

5.2.2.1 Generalization to non regular spaced data affected by noise

The procedure described in Sect. 5.2.1.3 supposes regularly spaced noise-free data sampled in the same position of the Gaussians center. If this hypothesis is not verified, a technique for the estimate of the value assumed by the function, $f(\cdot)$, in the position of the Gaussians center is needed and it has to be estimated. We will call this estimate, $\check{f}(\cdot)$. $\check{f}_k = \check{f}(\mu_k)$ is estimated locally: a subset A_k of the dataset is selected according to the distance of the data points from μ_k. Then, the points in A_k are used to provide the value of $\check{f}_k$. The region to which the points in A_k belong to, is called *receptive field* of the k-th Gaussian.

The criteria used to determine A_k are mainly two. The first one is based on the number of elements used in the estimation: for each Gaussian the n input points closer to its center, μ_k are selected. The second one is based on the distance: for each Gaussian every point closer to μ_k than a given threshold, ρ, is selected. Both the methods have advantages and disadvantages.

In [46], the estimate of $\check{f}_k = \check{f}(\mu_k)$ is based on a weighted average [22] of the points that belong to the set A_k, where the weight of each sample decreases with the distance of the sample from the center of the Gaussian. The implicit assumption is that the points closer to the Gaussian center are able to provide a more reliable estimate of the function in that point. In particular, the weight function used is again a Gaussian centered in μ_k, and $\check{f}_k$ can be computed as:

$$\check{f}(\mu_k) = \frac{\sum_{x_r \in A_k} y_r e^{-\frac{(x_r - \mu_k)^2}{\sigma_w^2}}}{\sum_{x_r \in A_k} e^{-\frac{(x_r - \mu_k)^2}{\sigma_w^2}}} \tag{5.23}$$

where the receptive field, A_k, is:

$$A_k = \{x_r \mid ||x_r - \mu_k|| < \rho\} \tag{5.24}$$

The parameter σ_w in the weighting function is set equal to the value of σ of the RBF layer, although $\sigma_w < \sigma$ would be more conservative as it avoids to possibly neglect the high frequency components of the function. ρ, that determines the amplitude of A_k, is set equal to Δx which is the spacing between two consecutive Gaussians ($\mu_x - \mu_{x-1} = \Delta x, \forall x$).

However, for a given layer, the computation of $\check{f}_k$ is not required over the whole input domain, as previous layers could have produced a function that is accurate enough in some input regions. In this case, the weights are better set to zero in these regions and no new Gaussians have to be inserted there.

To this aim, before computing $\check{f}_k$ we evaluate if the local residual value is below threshold. Such threshold can be for instance associated to the measurement noise, ε. Hence, the k-th Gaussian is inserted in the l-th layer (i.e., w_k is set to $\Delta x\, f_k$) if and only if the average value of the residual in the neighborhood of μ_k, R_k, is over ε, namely if the following is satisfied for the k-th Gaussian:

$$R_k = \frac{\sum_{x_r \in A_k} ||y_r||}{|A_k|} > \varepsilon \tag{5.25}$$

for the first layer, and:

$$R_k = \frac{\sum_{x_r \in A_k} ||r_r||}{|A_k|} > \varepsilon \tag{5.26}$$

for all the other layers. Otherwise, w_k is set to zero.

If A_k contains too few samples to provide a reliable estimate of $\check{f}_k$, an alternative criterion should applied for estimating w_k. For instance, the value of ρ can be increased until A_k is enough populated of input points (e.g., if $|A_k| < \alpha$, for a suitable α).

5.2.3 Batch HRBF configuration

The elements illustrated in the previous sections allow designing a recursive algorithm for setting up the parameters of a HRBF network to approximate a function from a finite set of noisy points. The HRBF configuration algorithm can be summarized as follows. Let us consider an input data set: $\{(x_i,\, y_i)|i = 1, \ldots, n\}$.

1. A set of L cut-off frequencies is chosen, following some criteria, such that: $\{\nu_l|\ l = 1, ..., L\}$ such that $\nu_l < \nu_{l+1}$. Each frequency is associated to the Gaussians of one layer.
2. From the set of frequencies, the value of σ_l and Δx_l are computed for each layer:

$$\sigma_l = \frac{\sqrt{\log 2/2}}{\pi \nu_{\text{cut-off}}} \tag{5.27}$$

$$\Delta x_l = \frac{\sigma_l}{1.465} \tag{5.28}$$

3. The first residual is set as: $r_0 = \{r_0(x_k) = y_k\}$.
4. For each layer, $l = 1, \ldots, L$:
   5. The Gaussians center is defined, $\{\mu_{l,j}|\mu_{l,j} - \mu_{l,j-1} = \Delta x_l\}$, such that they cover the input domain covered by the dataset. The number, M_l of the Gaussians of the l-th layer will be: $M_l = \left\lceil \frac{\max(x_i) - \min(x_i)}{\Delta x_l} \right\rceil$.
   6. For each Gaussian, j, the subset $A_{l,j}$ of input points is selected, as the points inside the receptive field, is a spherical neighborhood of the (l, j)-th Gaussian, centered in $\mu_{l,j}$, $A_{l,j}$ of the Gaussian j, that is a circular region centered in $\mu_{l,j}$ with radius proportional to Δx_l.
   7. If (5.26) is satisfied, and if enough points fall into the receptive field (i.e., $|A_{l,j}| > \alpha$), the coefficient $w_{l,j}$ is computed as:

$$w_{l,j} = \Delta x_l \frac{n_{l,j}}{d_{l,j}} = \Delta x_l \frac{\sum_{r:x_r \in A_{l,j}} y_r \; e^{-\frac{(x_r - \mu_{l,j})^2}{\sigma_w^2}}}{\sum_{r:x_r \in A_{l,j}} e^{-\frac{(x_r - \mu_{l,j})^2}{\sigma_w^2}}} \tag{5.29}$$

   8. The approximation function $a_l = \{a_l(x_i) = \sum_j w_{l,j} G(x_i; \mu_{l,j}, \sigma_l)\}$ is computed.
   9. The residual is computed as: $r_l = \{r_{l-1}(x_i) - a_l(x_i)\}$.

□

The output function is the sum of all the approximation functions computed:

$$\tilde{f}(x) = \sum_{l=1}^{L} \sum_{j} w_{l,j} G(x; \mu_{l,j}, \sigma_l) \tag{5.30}$$

L reconstructions with different resolution are then available:

$$\tilde{a}_l(x) = \sum_{h=1}^{l} a_h(x) = \sum_{h=1}^{l} \sum_{j} w_{h,j} G(x; \mu_{h,j}, \sigma_h) \tag{5.31}$$

Despite there are not theoretical constraints, the set $\{\nu_1, ..., \nu_l\}$ is computed, generally, such that each frequency is doubled with respect to the previous one. In this way a wide range of frequencies can be covered by using a small number of layers.

Since σ and ν are closely related, in many applications providing directly the value of σ is preferable. Furthermore, the number of layers has not to be set a priori but a termination criterion can be used: for example, the procedure can be iterated until the residual goes under a given threshold on the entire domain.

From equation 5.31, we see that the output of all the Gaussians in all the layers should be computed for each point in the input space, that would require a huge computational effort. However, due to its exponential decay, the Gaussian assumes values significantly larger than zero only in the region close to its center and some windowing to zero can be introduced. In fact the Gaussian value at the distance of 3σ from its center is $1.24 \cdot 10^{-4}$ times its value in the center.

This allows to limit the computation of the contribution of each Gaussian to the output of the layer only in a suitable neighborhood of its center. We call this region *influence region* of the Gaussian k, I_k. This region is a spherical region with radius τ proportional to Δx:

$$G_{l,k} = \begin{cases} \frac{1}{\sqrt{\pi}\sigma_{l,k}} \exp\left(-\frac{||x-\mu_{l,k}||)^2}{\sigma^2}\right), & ||x - \mu_{l,k}|| < \tau_{l,k} \\ 0, \quad \text{elsewhere} \end{cases} \tag{5.32}$$

Summarizing, the hyperparameters of the HRBF network, that is the parameters associated to the configuration procedure are the following:

- width of the weighting function, σ_w, (5.23);
- radius of the receptive field, ρ, (5.24);
- radius of the influence region, τ, (5.32);
- minimum number of samples, α, for a reliable estimate of the weights;
- set of the Gaussian centers for each layer, $\{\mu_{l,k}\}$;
- Gaussian width for each layer, $\{\sigma_l\}$;
- number of layers, L;
- error threshold ε, (5.25).

The parameters $\{\mu_{l,k}\}$, $\{\sigma_l\}$, and L define the structure of the HRBF network (Fig. 5.3).

The scale parameters values, $\{\sigma_l\}$, can be set using some a priori knowledge on the frequency content of the function as in (5.27). However, when this information is missing, σ_0 can be set equal to the extension of the input domain to which the data belong to. If no information on the scale of the details is available, the width

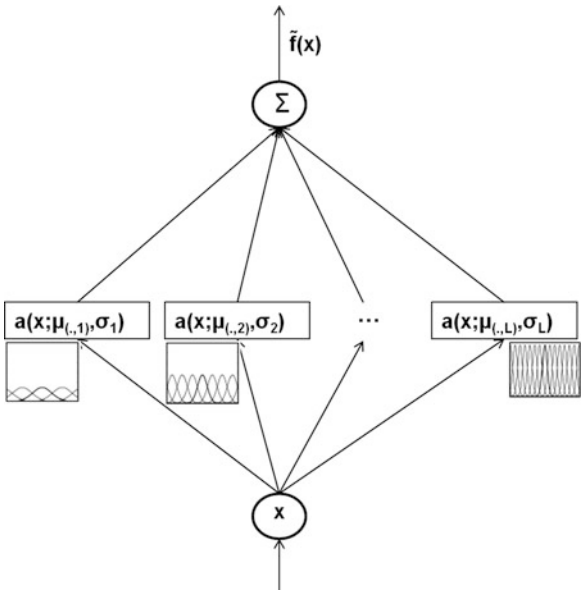

Fig. 5.3 The surface is reconstructed as the sum of L contributions in a HRBF model. Each layer (represented here with a rectangle) receives the input $\boldsymbol{x}$; the output of the model is computed as the sum of the output of each layer

of the Gaussians can be halved at each layer. The position of the units of each layer can be easily determined starting from the center of the domain moving laterally by Δx (5.28). The number of layers, L, could be determined a priori considering the computational resources available (e.g., the maximum number of units), or can be determined run-time considering the error achieved by the last configured layer. Otherwise it can be determined by a trial and error strategy or by cross-validation.

The hyperparameters σ_w, ρ, τ and α have been chosen through empirical considerations and they have been adequate in all the experiments that we have made. The error threshold ε depends on the root mean squared error on the data and the degree of the approximation accuracy of the HRBF network.

The hyperparameter ρ controls the width of the receptive field, and, hence, it is set proportional to the width of the units of the layer, σ_l, or to their spacing, Δx_l. Hence, reasonable choices can be: $\rho = \sigma_l$ or $\rho = 2\Delta x_l$. A too small value of ρ can make the estimate of $\check{f}_k$ unreliable due to lack of samples. On the other hand, large values of ρ can make the estimate computationally expensive. Moreover, the use of samples far from the Gaussian center is questionable.

The radius of the influence region affects both the accuracy in the approximation of the Gaussian output and the computational cost of the training procedure. We have considered, somehow arbitrarily the contribution of a Gaussian in points distant more than 3 σ from its center equal to zero. This can be a reasonable value because, as described before, the Gaussian value at the distance of 3σ from its center is generally negligible. The minimum number of samples required for a reliable estimate depends both on the variability of the function and on the amount of noise on the data. However, since only a rough approximation is required, few samples are sufficient for the scope (3 to 5 can be a reasonable value for α).

The error threshold, ε, determines if the Gaussians of the new layer will be added to the network and then it determines the accuracy of the approximation of the whole HRBF network. It should be set from the accuracy of the measurement instruments used for obtaining the data set. In the case of the 3D scanning, it can be estimated by using several techniques (e.g., average distance of sampled points on a plane with respect to the optimal plane), or from the calibration procedure. If a priori information is not available, a trial and error strategy could be used to estimate the error threshold value.

The configuration procedure of a HRBF network requires operations, local to the data, to compute the parameters. This produces a very fast configuration algorithm that is also suitable to be parallelized.

5.2.3.1 Extension to the two-dimensional case

Due to the properties of the Gaussian function, the extension to the multivariate case is straightforward. In particular, we will consider here the two-dimensional case since it is of interest for the applicative problem of surface reconstruction although generalization to spaces of higher dimensions is trivial. In the $\mathbb{R}^2 \rightarrow \mathbb{R}$ case, the Gaussian filter results:

$$G(x, x_k; \sigma) = \frac{1}{\pi\sigma^2} e^{-\frac{||x-\mu_k||^2}{\sigma^2}} \tag{5.33}$$

where $x \in \mathbb{R}^2$. As $x = (x_{(1)}, x_{(2)})$, the (5.33) can be written as:

$$\begin{aligned} G((x_{(1)}, x_{(2)}); \sigma) &= \frac{1}{\pi\sigma^2} e^{-\frac{x_{(1)}^2 + x_{(2)}^2}{\sigma^2}} \\ &= \frac{1}{\pi\sigma^2} e^{-\frac{x_{(1)}^2}{\sigma^2}} \frac{1}{\pi\sigma^2} e^{-\frac{x_{(2)}^2}{\sigma^2}} = G(x_{(1)}; \sigma) G(x_{(2)}; \sigma) \end{aligned} \tag{5.34}$$

and its Fourier transform can be written as:

$$\hat{G}(\nu; \sigma) = e^{-\pi^2\sigma^2\nu_1^2} e^{-\pi^2\sigma^2\nu_2^2} = e^{-\pi^2\sigma^2||\nu||^2} \tag{5.35}$$

as the Fourier transform of a separable function is equal to the product of the one-dimensional transform of its components. It follows that all the considerations and relationships for the one-dimensional case are valid also for the two-dimensional one.

We stress that, the relationship between the continuous and discrete case, as shown in (5.14), leads to:

$$\hat{f}(x; \sigma) = \sum_{k=1}^{N} f_k \, G(x_{(1)}; \mu_k; \sigma) \, \Delta x_{(1)} \; G(x_{(2)}; \mu_k; \sigma) \, \Delta x_{(2)} \tag{5.36}$$

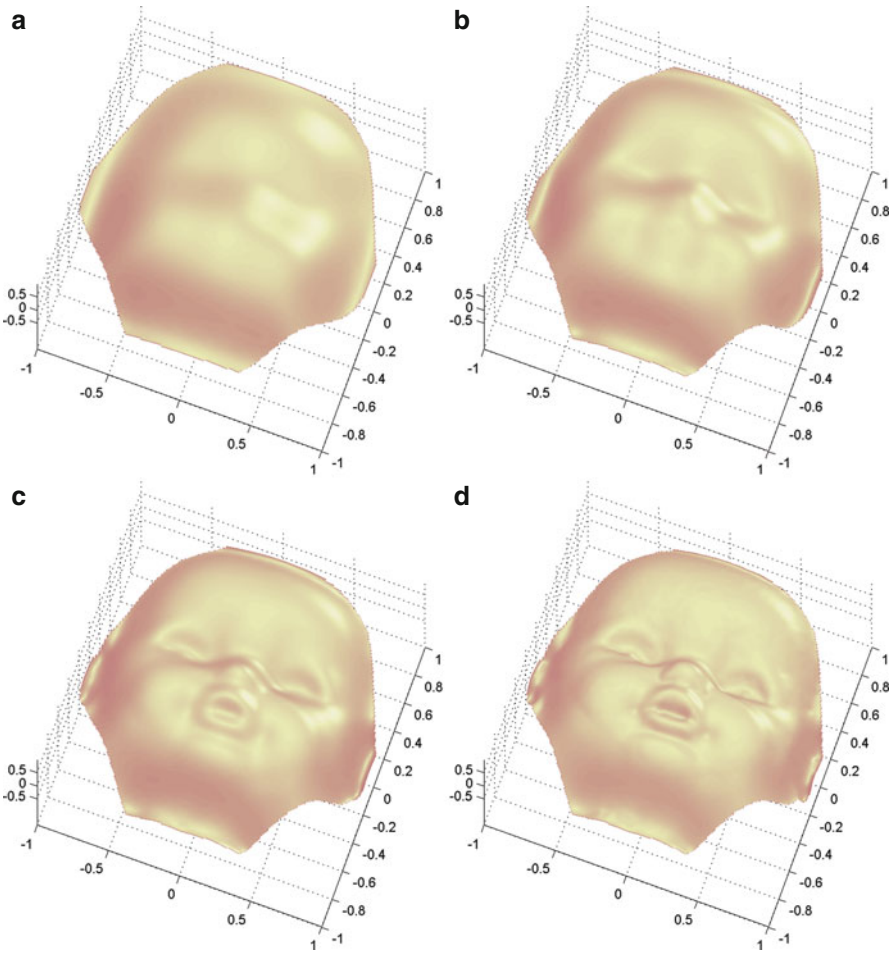

Fig. 5.4 Reconstruction of a doll with the HRBF model using (**a**) 5 layers, (**b**) 6 layers, (**c**) 7 layers, and (**d**) 8 layers. The hierarchical structure of the model allows to choose the number of layers that match the required visual quality

It follows that in (5.29) of the HRBF configuration algorithm (Sect. 5.2.3) the proportionality constant between the estimated value of the function and the coefficient of the Gaussian, becomes Δx_l^2.

In Fig. 5.4 an example of reconstruction realized using the HRBF model is reported. The dataset [96] has been acquired by the systems described in [48]. The four surfaces represent the output of the HRBF model considering the first 5 layers (a), 6 layers (b), 7 layers (c), and 8 layers (d). As can be seen, the reconstruction improves when new layers at smaller scales are added, reproducing finer and finer details. As a result a multi-scale reconstruction is obtained. In Fig. 5.5 the support of the Gaussians, i.e. the regular lattices where the Gaussians are placed,

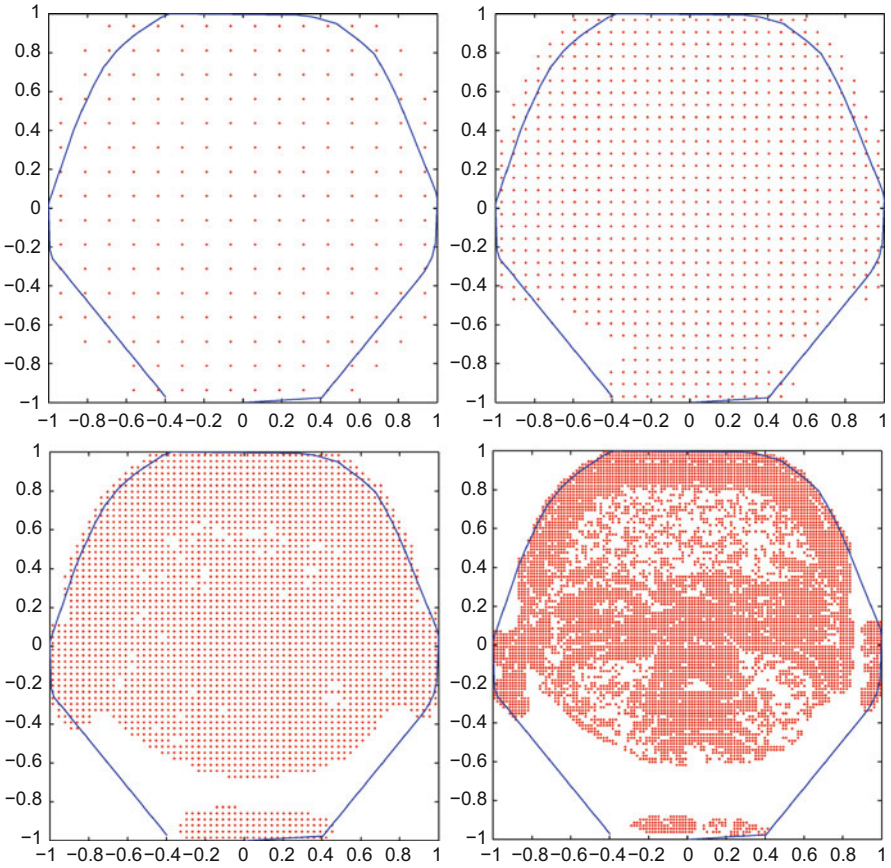

Fig. 5.5 The grids of Gaussians for the 5th (**a**), 6th (**b**), 7th (**c**), and 8th (**d**) layers of the HRBF model used for the reconstruction of the doll in Fig. 5.4. Note the sparseness of the Gaussians

are depicted. The centers of the Gaussians effectively used (i.e., those that have not a zero weight) are reported as a cross mark. It can be noticed that in the last layers the Gaussians are inserted only in some regions on the input space, as in the others the approximation is already below the threshold ε after the first layers.

5.2.4 Approximation properties of the Batch HRBF network model

In [97], the approximation properties of the HRBF model have been investigated formally. It is shown that a RBF network with units equally spaced on a lattice can be regarded as a Riesz basis and therefore it can approximate a large class of

functions, namely the compactly supported in $\mathcal{C}^\infty$ functions that have equilimited derivatives (i.e., there exists a supremum of the norms of the derivatives of function), provided that a suitable scale factor σ_j is chosen.

These considerations have been extended from a RBF to a HRBF network. In HRBF networks, the function to be approximated changes at each layer, as this is the residual left by the previous layers and it becomes smaller and smaller with the number of layers, decreasing the value of σ_l the (5.30) will be able to reconstruct $f(x)$.

In multi-dimensional input spaces, the part of the input domain that can be used as support of the HRBF network can be shrinked considering for instance the convex hull of the data points or any other a-priori information on the domain extension.

In [93][92] HRBF networks are applied to regularly sampled data in order to enable a proper comparison with Wavelet MRA. In particular, accuracy with respect to parameters representation and precision (fixed and floating point) have been investigated. Results show that HRBF does not lose accuracy since the residual computed in the HRBF configuration procedure for each layer incorporates any error introduced in the output of the previous layer (e.g., errors in the representation of the units, of the coefficients, or limited precision in the computation). This HRBF configuration process allows error compensation in the higher layers. This is not the case in MRA, where the error in the coefficients propagates through the cascade algorithm.

5.3 Real-time, incremental surface construction through HRBF networks

For the HRBF networks, the scheme described in the previous section cannot be efficiently applied for real-time reconstruction. In fact, if an HRBF network has been already configured using a given data set, when a new sample, $(x_{\text{new}}, y_{\text{new}})$, becomes available, the output of (5.23) becomes out of date for the first layer. Then $\check{f}(\mu_k)$ has to be estimated again for all the Gaussians by using the new data set constituted of the old data set plus the new sample. 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 if the average residual becomes over threshold in some regions of the input domain.

A brute force solution is to reconfigure the entire network re-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, an on-line version of the previous scheme has been developed [91][32] and it is presented in the following.

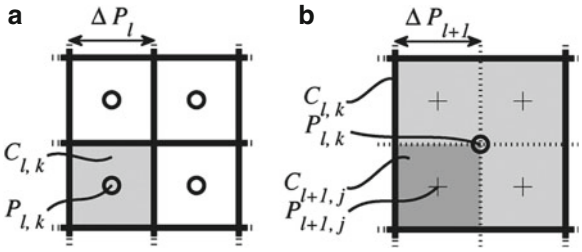

Fig. 5.6 The close neighborhood $C_{l,k}$ of the Gaussian centered in $\mu_{l,k}$, belonging to the l-th layer, is shown in pale gray in (**a**). The ensemble of the close neighborhoods tessellates the input domain, partitioning it in squares which have side equal to that of the l-th grid Δx_l and are 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 a quad-tree decomposition scheme, as shown in (**b**). Each $C_{l+1,j}$ has the side half the length of $C_{l,k}$, and it is centered in a Gaussian $\mu_{l+1,j}$ positioned in "+". Originally published in [91] (reproduced by permission of IEEE)

On-line learning algorithms work updating the model parameters analyzing one sample at a time. Several reasons may induce to prefer this modality with respect to the batch learning. In real cases, the training set can be too large for applying a batch methods efficiently. This is the case, for instance, of linear models when matrix inversion, required to determine the parameters, cannot be performed due to excessive memory needed. Besides, the peculiarity of the application may require on-line learning. For instance, when the input data come one at a time or when partial approximations are required during the acquisition of the training set. The latter is particularly valuable in 3D scanning, to better direct the acquisition process in most critical areas.

The most limiting element to real-time operation is the shape of the receptive field: as the Gaussians have radial symmetry, their receptive field comes out with a $D-$dimensional circular shape that does not allow an efficient partitioning of the data points. To overcome this problem, the receptive field can be approximated with a $D-$dimensional square region [96][43]. However, this approximation can be accepted as far as the input space has a low dimensionality [94][204].

This idea 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 non-overlapping regular $D-$dimensional cubes $C_{l,k}$, each centered in a different Gaussian center $\mu_{l,k}$. As shown in Fig. 5.6, for a $\mathbb{R}^2 \rightarrow \mathbb{R}$ mapping, the input domain is partitioned into $2D$ squares $C_{l,k}$, where each $C_{l,k}$ is called the *close neighborhood* of the (l,k)-th Gaussian $G_{l,k}$. The vertices of each $C_{l,k}$ are shifted of $\frac{1}{2}\Delta x_l$ with respect to the grid centers.

The configuration algorithm is structured in a set of updating steps, each of them required any time a new input sample is given to the network. These steps are followed by a single step in which the average residual is evaluated and Gaussians

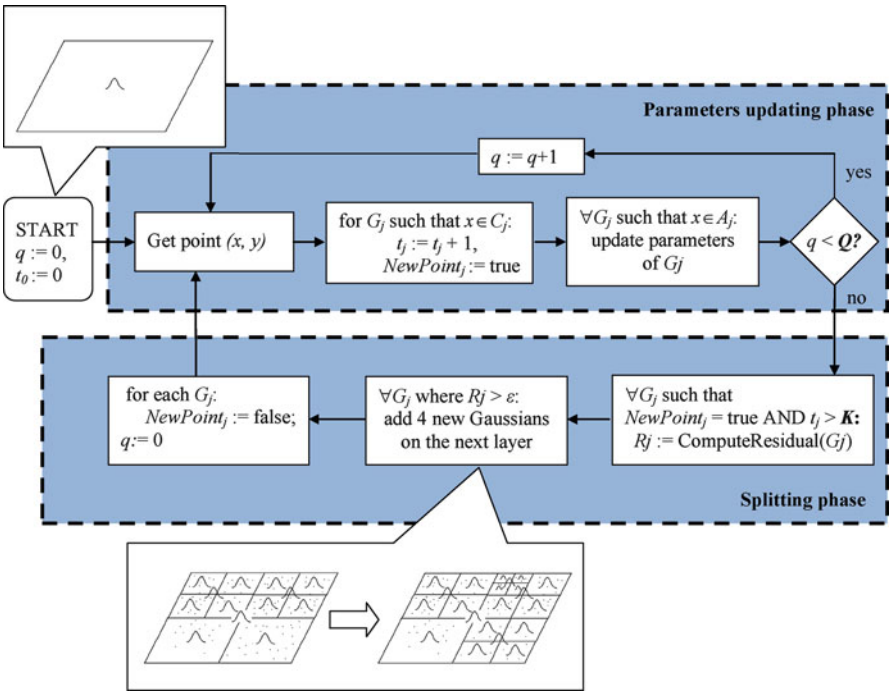

Fig. 5.7 Schematic representation of the on-line HRBF configuration algorithm. Originally published in [91] (reproduced by permission of IEEE)

can be inserted. This latter operation is called also split as 2^D new Gaussians can be inserted at the next layer, inside the area covered by the Gaussian of the current layer. These two phases, depicted in the scheme in Fig. 5.7, 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 covered by the data. An estimate of the dimension of the volume is therefore the only a priori information needed by the configuration algorithm.

A particular data structure is associated to each Gaussian $G_{l,k}$. This contains the Gaussian's position $\mu_{l,k}$, its weight $w_{l,k}$, the numerator $n_{l,k}$, and the denominator $d_{l,k}$ of (5.29). The structure associated to the Gaussian at the top of the hierarchy (current highest layer h) contains also all the samples that lie inside $C_{h,k}$. To obtain this, when a Gaussian is split during learning, its associated samples are sorted locally by mean of the *qsort* algorithm and distributed among the 2^D new Gaussians of the higher layer.

As $\Delta x_l = \Delta x_{l-1}/2$, the *close neighborhood* of each Gaussian of the l-th layer (*father*) is formed as the union of the *close neighborhoods* of the 2^D corresponding Gaussians of the $(l+1)$-th layer (*children*). This relationship, depicted in Fig. 5.6b,

is exploited 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, it is assumed 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.

5.3.1 First Learning Phase: Updating the Parameters

When a new point x_{new} is given, the quantities $n_{l,k}$, $d_{l,k}$, and $w_{l,k}$ (5.29), associated to the Gaussians such that $x_{\text{new}} \in A_{l,k}$, are updated

$$n_{l,k} := n_{l,k} + r_{l-1}(x_{\text{new}}) e^{-||\mu_{l,k} - x_{\text{new}}||^2/(\sigma_l/2)^2} \tag{5.37}$$

$$d_{l,k} := d_{l,k} + e^{-||\mu_{l,k} - x_{\text{new}}||^2/(\sigma_l/2)^2} \tag{5.38}$$

$$w_{l,k} = \frac{n_{l,k}}{d_{l,k}} \cdot \Delta x_l^D \tag{5.39}$$

where $r_{l-1}(x_{\text{new}})$ is computed, like in (5.21), as the difference between the input data and the sum of the output of the first $l-1$ layers of the present network computed in x_{new}.

It is explicitly noticed that the modification of the weight of a Gaussian in the l-th layer $G_{l,k}$ modifies the residual of that layer r_l inside the Gaussian's influence region. Hence, the terms in (5.37)–(5.39) should be recomputed for all the layers starting from the first layer upwards.

However, this would lead to an excessive computational load, and, in the updating phase, the terms in (5.37)–(5.39) are modified only for the last configured layer, l. The rationale is that increasing the number of points, (5.39) tends to (5.29). After computing the new numerator, denominator and weight for the Gaussian l, k, the residual is computed in x_{new}.

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

5.3.2 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 (that is there are no Gaussians in a higher layer that share the same support), 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 Gaussians candidated for splitting. Let us call J their ensemble.

For each Gaussian of J, the local residual $R_{l,k}$ is re-evaluated for all the points inside its close neighborhood using the current network parameters. If $R_{l,k}$ 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. 5.6b].

We remark that the estimate of $R_{l,k}$ 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.

The parameters of a newly inserted Gaussian $G_{l+1,k}$, $n_{l+1,k}$, $d_{l+1,k}$ and $w_{l+1,k}$ in (5.37, 5.38, 5.39) 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 $n_{l+1,k}$, $d_{l+1,k}$ and $w_{l+1,k}$ are set to zero when no data point is present inside $C_{l+1,k}$ and the Gaussian $G_{l+1,k}$ will not contribute to the network output.

It is worth noting 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.

5.3.3 On-line HRBF convergence

In [97] it was shown that the sequence of the residuals obtained with the HRBF scheme converges to zero under mild conditions on $f(\cdot)$. As the on-line configuration procedure is different from the batch one, the convergence of the residuals obtained with the on-line scheme has to be proved.

The on-line and the batch scheme differ for both the computation of the weights [(5.37)–(5.39) versus (5.29)] and for the rule of insertion of new Gaussians (in the batch scheme, this occurs layer-wise, while in the on-line scheme, it occurs locally during the splitting phase).

It is first shown that the output of each layer of the on-line HRBF is asymptotically equivalent to that of the batch HRBF. Let us first consider the case of the first layer.

Let f_p be the input data set constituted of the first p points sampled from f and denote with $\star$ the convolution operation:

$$a_1(\cdot) = f_p \star G(\cdot; \sigma_1) \tag{5.40}$$

that produces the output of the first HRBF layer, configured using f_p. It can be shown that when p tends to infinity, the function computed in (5.40) converges to the value computed in (5.30) for the batch case.

This is evident for this first layer, whose weights are estimated as $\check{f}(\mu_{l,j})$, and $r_0(x_i) = y_i$ holds. In this case, the following asymptotic condition can be derived:

$$\lim_{p\to\infty} w^{\mathrm{b}}_{1,k} = \frac{\sum_{m=1}^{p} y_m \, e^{-||\mu_{1,\,k}-x_m||^2/(\sigma_1/2)^2}}{\sum_{m=1}^{p} e^{-||\mu_{1,\,k}-x_m||^2/(\sigma_1/2)^2}} \Delta x_1^D = \lim_{p\to\infty} w^{\mathrm{o}}_{1,k} \tag{5.41}$$

where $w^{\mathrm{b}}_{1,k}$ are the weights computed in the batch algorithm by (5.29) and $w^{\mathrm{o}}_{1,k}$ are those computed in the on-line algorithm by (5.37)–(5.39).

It follows that:

$$\lim_{p\to\infty} \Delta r_1(x_i) = \lim_{p\to\infty} r^{\mathrm{b}}_1(x_i) - r^{\mathrm{o}}_1(x_i) = 0 \tag{5.42}$$

where $r^{\mathrm{b}}_1(x_i)$ is the residual at the point x_i computed by (5.29), and $r^{\mathrm{o}}_1(x_i)$ is the same residual computed by (5.37, 5.38, 5.39).

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

$$n_{2,\,k} := n_{2,\,k} + (r^{\mathrm{o}}_1(x_p) + \Delta r_1(x_p)) \cdot e^{-||\mu_{2,\,k}-x_p||^2/(\sigma_2/2)^2} \tag{5.43}$$

$$d_{2,\,k} := d_{2,\,k} + e^{-||\mu_{2,\,k}-x_p||^2/(\sigma_2/2)^2} \tag{5.44}$$

Since $\lim_{p\to\infty} \Delta r_1(x_p) = 0$ 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 on-line 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.

5.3.4 *Experimental results*

The on-line HRBF model has been extensively applied to 3D scanning. Points were acquired by the Autoscan system, that is based on acquiring the 3D position of a laser spot through a set of cameras that can be set-up according to the acquisition needs: a very flexible set-up is therefore obtained. Autoscan [47][48] 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 a real-time feedback of the reconstructed surface

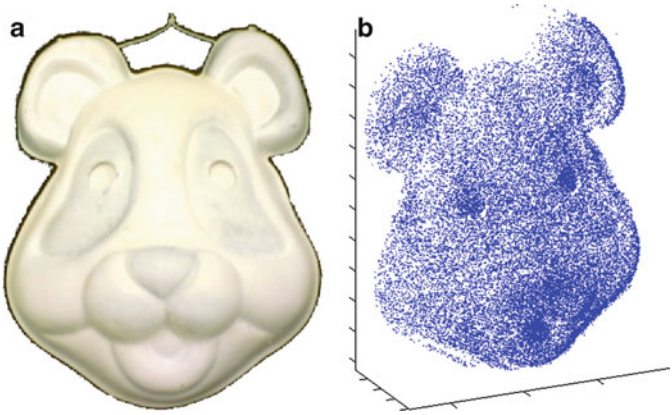

Fig. 5.8 A typical data set acquired by the Autoscan system [48]: the panda mask in (**a**) has been sampled into 33 000 3D points to obtain the cloud of points shown in panel (**b**). These points constitute the input to the HRBF network [91] (reproduced by permission of IEEE). Note the higher point density in the mouth and eyes regions

during scanning is of paramount importance to achieve optimal data collection as shown in [14].

A typical set of sampled data is reported in Fig. 5.8b. It is constituted of 33 000 points sampled over the surface of the artifact (a panda mask) shown in Fig. 5.8a. As can be seen in Fig. 5.9 the reconstruction becomes better and better with the number of points sampled. Acquisition was stopped when the visual quality of the reconstructed surface was considered sufficient by the operator and no significant improvement could be observed when new points were added (compare Fig. 5.9e and 5.9f).

To assess the effectiveness of the on-line 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 [71]: from the dataset in Fig. 5.8b, 32 000 points, randomly extracted, were used to configure the network parameters (training set), and 1 000 for testing (test set).

The error, expressed in millimeters, was measured in both the l_1-norm, mean absolute error, $\varepsilon_{\text{mean}}$, and in l_2-norm and in the l_2-norm, root mean squared error, RMSE, as:

$$\varepsilon_{\text{mean}} = \frac{1}{N}\sum_{i=1}^{N} |r_T(x_i)| \tag{5.45a}$$

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} r_T(x_i)^2} \tag{5.45b}$$

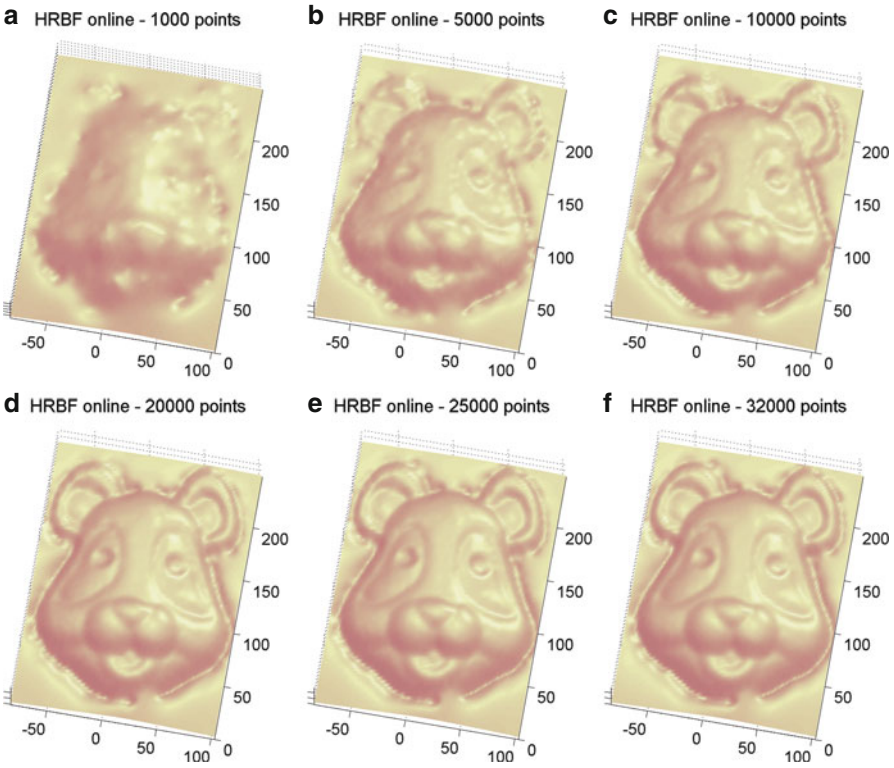

Fig. 5.9 Panels show the reconstruction with on-line HRBF after 1 000, 5 000, 10 000, 20 000, 25 000, and 32 000 points have been sampled [91] (reproduced by permission of IEEE)

where $r_T(x_i)$ is the reconstruction error on the i-th point of the test set, $i = 1, \ldots, n$. The error was measured as the mean value of the test error averaged over ten randomizations on the same dataset.

To avoid border effects, (5.45) 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%.

5.3.5 *Comparison with the batch HRBF model*

Results of the comparison with the batch HRBF model are reported in Table 5.1. These figures have been obtained with the following parameters: $Q = 100$, $K = 3$, $L = 8$ layers. The error threshold, ε, was set for all the layers equal to the nominal digitization error, that was 0.4 mm and the final reconstruction error of 0.391 mm, is very close to this value. A total of 9 222 Gaussians were allocated over the eight layers and produce a sparse approximation (cf. Figs. 5.10d–f).

Table 5.1 Accuracy and parameters of each layer of the HRBF networks

		on-line			pure batch			batch constrained	
#layer	σ	#Gauss. (total)	RMSE	$\varepsilon_{\text{mean}}$	#Gauss. (total)	RMSE	$\varepsilon_{\text{mean}}$	RMSE	$\varepsilon_{\text{mean}}$
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

Published in [91] (reproduced by permission of IEEE).

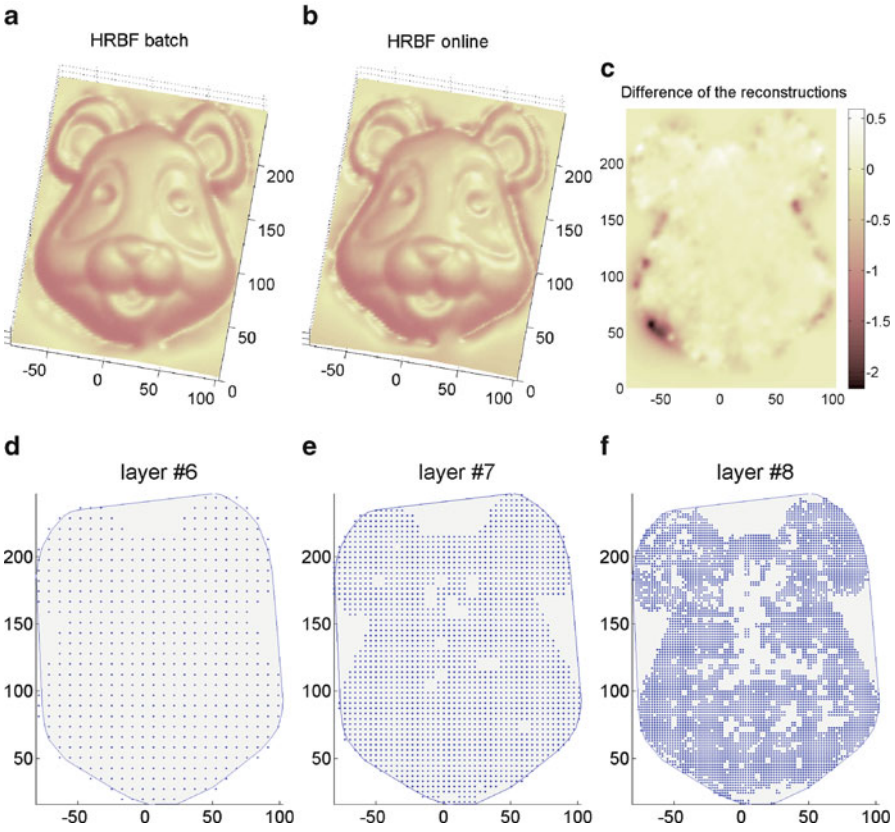

Fig. 5.10 Reconstruction with (**a**) HRBF batch and (**b**) HRBF on-line. The difference between the two surfaces is shown in panel (**c**). In panels (**d**)–(**f**) the center of the Gaussians allocated by the on-line algorithm in the last three layers is shown. Originally published in [91] (reproduced by permission of IEEE)

The network complexity and the reconstruction error have been compared with those obtained when the network was configured using a batch approach [95], with the same number of layers as the on-line version.

Two batch modalities have been considered. In the first one, *pure batch*, the configuration procedure described in Sect. 5.2.3 [97] is adopted. In the second approach, *batch constrained*, the Gaussians are placed in the same position, and have the same width as those of the on-line approach, while the weights are computed by (5.29), considering all the data points inside the receptive field of each Gaussian, as described in [46].

As shown in Fig. 5.10, the surface reconstructed by the batch HRBF has a slight better appearance than the on-line one, especially at the object border as shown by the difference image (Figs. 5.10c and 5.11). This was obtained at the expense of a larger number of Gaussians: about 12.7% more than those used in the on-

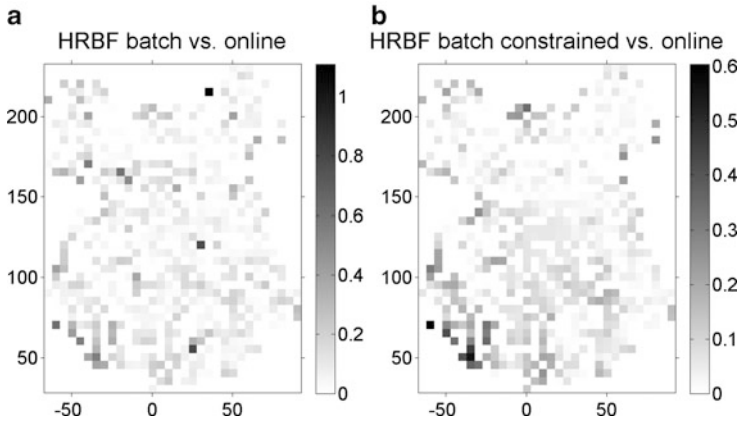

Fig. 5.11 Difference in the reconstruction error on the points of the test set: on-line vs. pure batch (**a**), on-line vs. batch constrained (**b**). Originally published in [91] (reproduced by permission of IEEE)

line approach, being 10 393 versus 9 222 (Table 5.1). Despite the difference in the number of Gaussians, the global accuracy of the batch approach is only slightly better than the on-line one, being of 0.373 mm versus 0.391 mm (4.82%).

As the acquisition was stopped when the visual appearance of the model (reconstructed with the on-line approach) was considered adequate by the operator, we have investigated if there was room for further accuracy improvement by acquiring more data points. To this aim, the rate of Gaussians allocation and of error decrease as a function of the number of data points is plotted in Fig. 5.12. The figure shows also that, adding new points, the error of the on-line model can be slightly lowered down to 0.381 mm, closer to the batch approach. To achieve such an error only 99 more Gaussians are required. However, as it is clearly shown, the batch version grows and converges faster than the on-line version: it achieves $\varepsilon_{\text{mean}}$ of 0.391 mm using only 8 500 data points. This is due to the fact that in the batch approach all the parameters are computed together for each layer and they are therefore optimized.

Results are consistent for different artifacts (cf. Table 5.2 and Fig. 5.13). 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. 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, and these Gaussians cover each 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 by the operator. Alternatively, data acquisition could be stopped when splitting inside the HRBF model does not occur anymore.

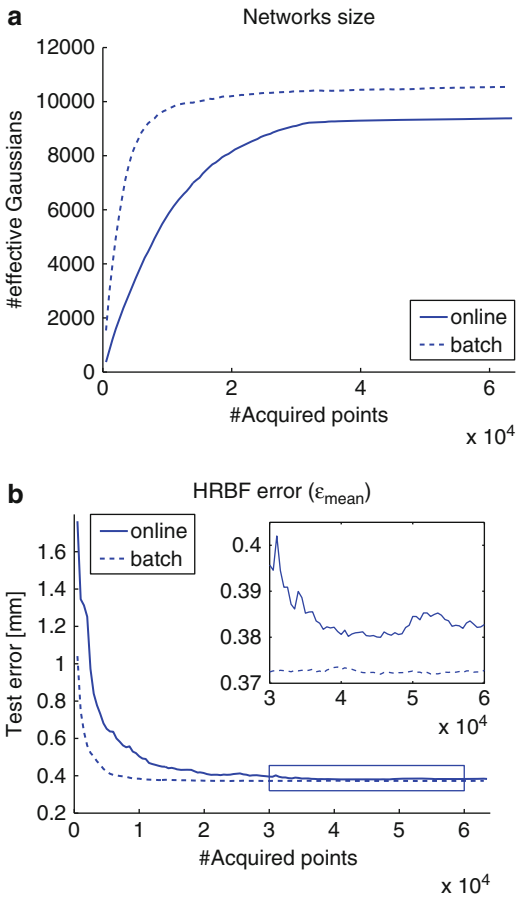

Fig. 5.12 Number of Gaussian units (**a**) and mean error (**b**) as a function of the number of data points used to configure the networks. Data set is grown of 500 data points at a time. Originally published in [91] (reproduced by permission of IEEE)

The on-line configuration algorithm contains two hyperparameters: K and Q (Sect. 5.3.2). Their influence on the general behavior of the on-line algorithm has been experimentally assessed by training the model with different combinations of K and Q.

Figure 5.14a 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 using 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 $9\,968$ for $K = 1$. A trade-off has been adopted here choosing $K = 3$, which produces a total number of Gaussians of $9\,222$, about 92.5% of those obtained setting $K = 1$.

Table 5.2 Reconstruction with several datasets

		on-line			pure batch			batch constrained	
dataset	#points	#Gauss. (total)	RMSE	$\varepsilon_{\text{mean}}$	#Gauss. (total)	RMSE	$\varepsilon_{\text{mean}}$	RMSE	$\varepsilon_{\text{mean}}$
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

Published in [91] (reproduced by permission of IEEE).

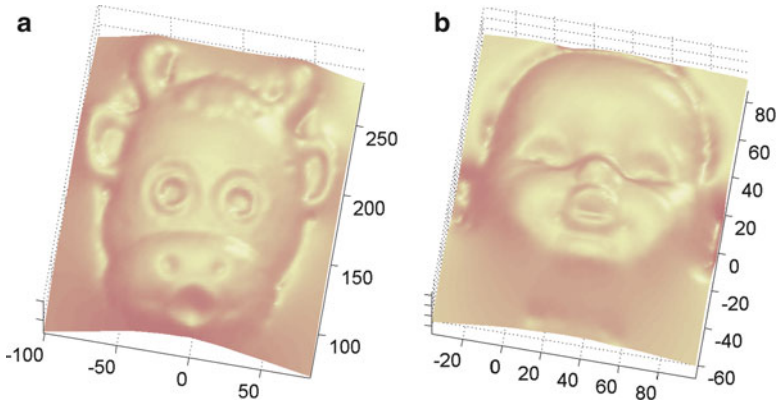

Fig. 5.13 HRBF on-line reconstruction of the dataset (**a**) *cow* (33861 points, 9501 Gaussians), and (**b**) *doll* (15851 points, 6058 Gaussians) [91] (reproduced by permission of IEEE)

The behavior of the test error as a function of Q is shown in Fig. 5.14b. For small values of Q, the behavior does not changes significantly with Q: the error starts increasing after $Q = 1\,000$, although the increase is of small amplitude (about 0.01 mm from $Q = 1\,000$ to $Q = 2\,500$). 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 (Sect. 5.3.2). Afterward, its weight is updated considering only the points inserted inside its *receptive field*, as in (5.37, 5.38, 5.39). 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 number of available data points, not enough splits are allowed to occur (these are at most N/Q), and the reconstruction becomes poorer. This situation is depicted in Fig. 5.14b where for a relatively large value of Q, the test error tends to increase with Q, as an effect of the decrease in the number of the allocated Gaussians.

From the curve in Fig. 5.14, it can be derived that the optimal value of Q would be about $1\,000$, 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 some loss in the degree of interactivity. In fact, when Q increases, splitting occurs less frequently and, as it produces the largest decrease in the reconstruction error, a longer time has to elapse before the user can see a large change in the appearance of the reconstructed model while the user would see, ideally, the reconstructed model improve swiftly in time. Moreover, the reconstruction error (computed as in 5.45a) decreases less quickly as shown in Fig. 5.15b. Therefore, although the value of K and Q may be subjected to optimization with respect to network accuracy or size, the resulting value 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

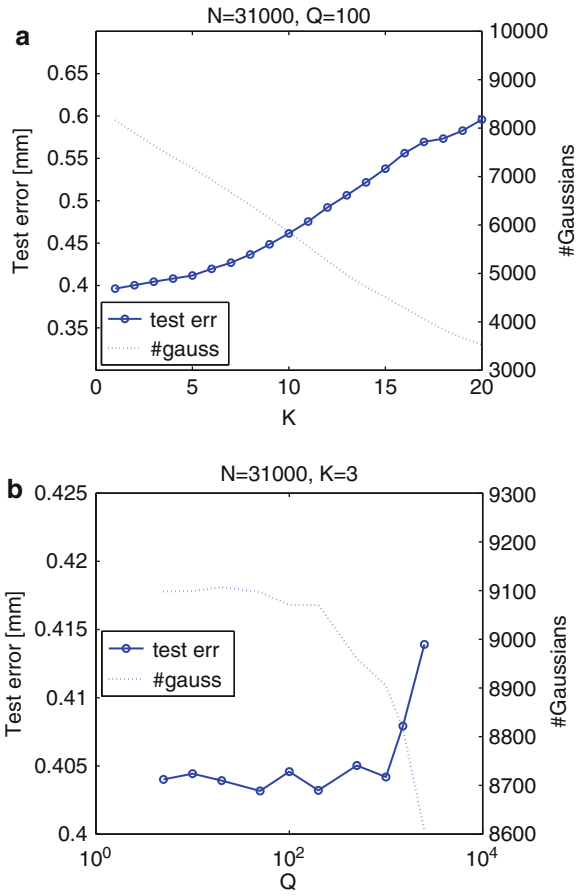

Fig. 5.14 Test error and number of Gaussian units with respect to K with Q fixed to 100 (**a**) is in a larger font than the title in the panel, and with respect to Q with K fixed to 3 (**b**). Originally published in [91] (reproduced by permission of IEEE)

as small as possible, according to the data sampling rate, to maximize interactivity. We remark that the value set, $Q = 100$, is lower than the one that would produce the smallest network.

This value could be even be lowered at the price of a large overhead, as the split phase is the most computationally expensive. Therefore a value of Q related to data rate seems the most reasonable for applications that require interaction rates. Although this might introduce more Gaussians, than the minimum required, the network will not be sensitive to overfitting, because of the mechanism of local computation of the weights. This is shown in Fig. 5.15 where the test error does not increase with the number of Gaussian units.

As far 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 estimation error on the weights. However, possible bias in the weights estimated can be recovered in the higher layers thanks to the incremental optimization construction.

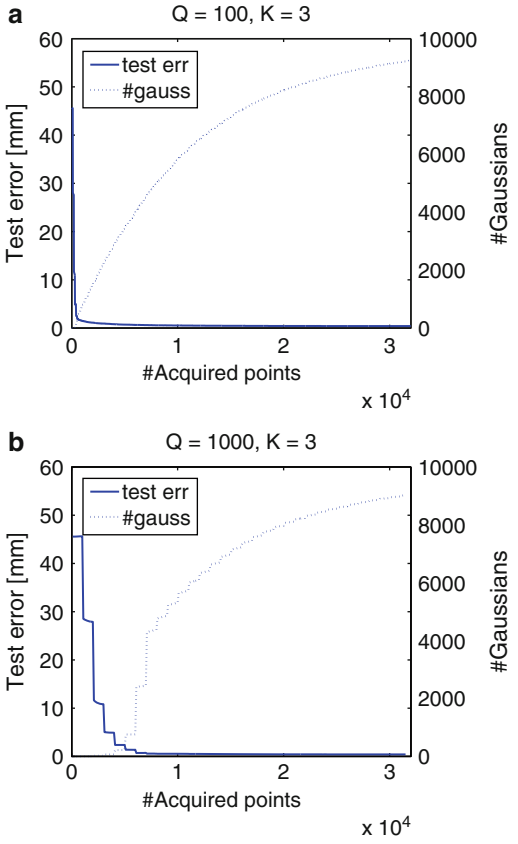

Fig. 5.15 Test error and number of Gaussian units reported with respect to N for small and large values of Q: in (**a**), Q was fixed to 100, and, in (**b**), Q was fixed to 1 000 [91] (reproduced by permission of IEEE)

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, 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 can be incremented until the error in (5.45) 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, may introduce a slight bias in the weights. This can be appreciated in Table 5.1, where the accuracy obtained when the weights are estimated considering all the data points (batch constrained) is compared with that obtained with the on-line approach described here.

In fact, the value output by the current network in $\mu_{l,k}$ (5.29) is computed as the ratio between $n_{l,k}$ and $d_{l,k}$, obtained as the run-time sum of the values derived from each sampled point for the higher layer. However, this value is equal to that output by the batch model only for the first layer, in which the weight of all the Gaussians is computed using all the sampled points inside the Gaussians *receptive field*). In the higher layers, where the residual in the already acquired points is not updated, the estimate of the weights may introduce a bias that, in turns, may produce a bias in reconstruction.

However, as this bias increases the value of the residual, in the splitting phase it is taken into account by the weights of the new Gaussians inserted in the higher layer. In fact, the residual is recomputed there for the last layer, using all the data points inside the $C_{l,k}$. Moreover, due to the small angle between the spaces spanned by two consecutive approximations, the HRBF model is able to compensate the reconstruction error in one layer, with the approximation produced by the next layer [97].

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 on-line configuration approach, differently from the batch one, can introduce Gaussians at the k-th level also when $k-1$-th level has not been completed, as the parameters of each Gaussian are updated independently from those of the others. Therefore one branch of the network can grow before another branch; 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.

When the maximum number of layers of the network, L, is too low for the frequency content of the given dataset, the error inside the close neighborhood of some Gaussians of the last layers will contain also the highest frequency details. As a consequence, the reconstruction in these regions can be poor. This error affects also the close neighborhood of the adjacent units by 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 5.3).

Differently from other growing network models [104][18][166], pruning is not adopted here, as all the units participate in the reconstruction because of the configuration mechanism. Pruning could be considered when dealing with time-variant systems or when some of the data are not pertinent or are outliers far from the true data distribution. The problem addressed here does not belong to this class and the additional complexity for managing pruning is not justified.

On-line HRBF shares with other growing networks models the criterion for inserting new units: the insertion is decided on the basis of an error measure, computed locally, that is fundamental to achieve real-time operation. The other element which allows real-time operation is the adoption of a grid support for the units, which guides Gaussians positioning. This is shared also by [18]. However, in [18] all the weights are recomputed after the insertion of new Gaussians, while

Table 5.3 The 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

Published in [91] (reproduced by permission of IEEE).

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. Grid support has been also adopted by [207][160]; however, in their approach global optimization is used, that makes the configuration procedure computationally heavy.

Finally, growing strategy implicitly implements an active learning procedure [117] as the data points that participate in the configuration of the new Gaussians are only those that carry an over-threshold error.

Chapter 6
Hierarchical Support Vector Regression

In the previous chapter the RBFN model and the advantages of a hierarchical version for surface reconstruction has been presented. In a similar way in this chapter another paradigm, Support Vector Regression (SVR), and its hierarchical version, Hierarchical Support Vector Regression (HSVR) that allows an efficient construction of the approximating surface, are introduced. Thanks to the hierarchical structure, the model can be better applied to 3D surface reconstruction giving a new, more robust and faster configuration procedure.

6.1 SVM Origin

The SVM approach has been developed by Vapnik and co-workers from 1960's to 1990's who, starting from the non-linear generalization of the *Generalized Portrait* model [238], extensively analyzed the conditions under which a minimum of the generalization error corresponds to the minimum of the training error. This has led to the foundations of the statistical learning theory [236][237], called also Vapnik Chervonenkis (VC) theory. This has led to defining a cost function that has the shape of a regularizer.

Initially, this approach was applied only to classification problems. In particular the first works were focused on optical character recognition (OCR) [24]. The classification problem is formulated as a convex optimization problem in which the solution is unique and it can be found using standard optimization software. Furthermore, the computational complexity of the procedure used to find the solution is independent from the input dimension. Thanks to the good performances with respect to other methods, SVMs rapidly became popular for all areas where statistical classification required [53][54].

SVMs have been more recently extended to regression [215], domain in which this approach has been called Support Vector Regression (SVR). The problem is formulated, again, as convex optimization problem whose computational complexity is

F. Bellocchio et al., *3D Surface Reconstruction: Multi-Scale Hierarchical Approaches*,
DOI 10.1007/978-1-4614-5632-2_6, © Springer Science+Business Media, LLC 2013

independent from the data input dimensionality and it has shown good performances for the solution of many applicative problems [84][171][217], among which 3D scanning is one of the most suitable.

The quality of the solution computed by the SVR paradigm depends on a few hyperparameters, generally, selected using a trial and error strategy. When the data are characterized by different frequency content over the input domain, a combination of parameters that produces a good solution over the entire input domain may not be found. A hierarchical SVR structure, that realizes a reconstruction locally at different scales, addresses this problem and it is described here: core of this chapter is an innovative paradigm based on SVR allowing multi-scale reconstruction and its use for 3D surface reconstruction.

In order to better understand the working principles of the Hierarchical Support Vector Regression (HSVR) model, the basic SVM approach is initially considered. Starting from the linear classification problem, the explanation is extended to non-linear case and then to the regression problem. In Sect. 6.2, the hierarchical approach is treated in its different aspect and the HSVR regression is described along as the results of its application to 3D scanning.

6.1.1 Linear classification

A binary classification problem [237] can be defined as follows.

Let $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a training set, where $x_i \in X \subseteq \mathbb{R}^D$, $y_i \in Y = \{-1, 1\}$. The objective is to find a decision function (or hypothesis) $\hat{f}(x) : X \to Y$ able to correctly classify *new examples*.

The decision function can be chosen among many kinds of functions. The selection of the best kind of function is a well known problem treated in depth in Statistical Learning Theory [236] and it is known as model selection problem. In the classical SVM paradigm, $\hat{f}(x)$ has the form of $\Theta(h(x))$, where $h(x)$ is a linear function of x:

$$\hat{f} = \Theta(h(x)) = \Theta(\omega \cdot x + b) \tag{6.1}$$

$$\Theta(z) = \begin{cases} -1 \text{ if } z < 0 \\ 1 \text{ if } z \geq 0 \end{cases} \tag{6.2}$$

where $\omega \in \mathbb{R}^D$ and $b \in \mathbb{R}$. For sake of conciseness, $\hat{f}$, is called linear decision function. A linear decision function divides the input space R^D in two regions separated by the hyperplane $\omega \cdot x + b = 0$. The region in which the point lies determines the classification of the point itself (Fig. 6.1).

The classification problem can then be formulated as the searching of the parameters, ω and b, of the separator hyperplane. If the examples are linearly separable, the separator hyperplanes can be of infinite number and the problem is ill-posed: the solution is not unique. The SVM algorithm finds the hyperplane that maximizes

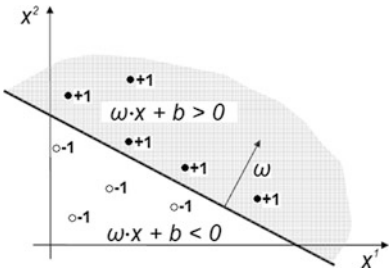

Fig. 6.1 Binary linear classifier. x^i denotes the i-th input variable of the multidimensional datum $\boldsymbol{x}$

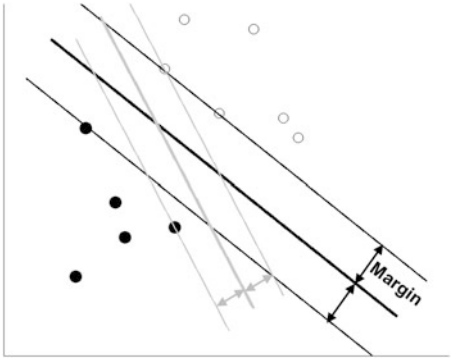

Fig. 6.2 The separator hyperplanes are in infinite number. SVM algorithm finds that one that maximizes the margin

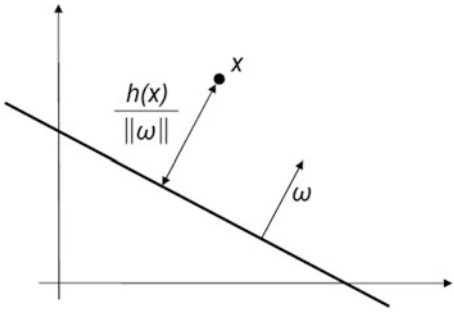

Fig. 6.3 Distance between the hyperplane and a point x

the margin, making the problem well-posed. This is defined as the distance between the hyperplane and the closest examples (Fig. 6.2). The maximization of the margin allows finding the best solution from a statistical point of view, i.e., the hyperplane that minimizes the *statistical risk* of misclassification.

The computation of the hyperplane that maximizes the margin is shown in the following. Let $h(x) = \omega \cdot x + b = 0$ be the equation of the hyperplane. The signed distance between the hyperplane and a generic point x is $\frac{h(x)}{||\omega||}$ (Fig. 6.3). The distance can be obtained multiplying the signed distance by the label y_i (6.2). Every separator hyperplane satisfies the following:

$$\frac{y_i h(x_i)}{||\omega||} \geq M > 0, \forall\, i = 1, \ldots, n \tag{6.3}$$

where M represent the distance between the hyperplane and the closest example. The maximal margin is then:

$$M^* = \max_{||\omega||=1,b} \min_i \; y_i h(x_i) \tag{6.4}$$

Given an hyperplane of margin M with respect to a training set, it is possible to choose, from its infinite representations, the one such that $||\omega|| = \frac{1}{M}$. The hyperplanes in this form, called *canonical form*, are such that:

$$\min_i \; y_i h(x_i) = 1 \tag{6.5}$$

If only canonical hyperplanes are considered, maximizing the margin has the same effect of minimizing $||\omega|| = \frac{1}{M}$. The hyperplane (ω, b) that solves the optimization problem:

$$\begin{aligned} &\min_{\omega,b} \; \frac{1}{2}||\omega||^2 \\ &\text{subject to } y_i(\omega \cdot x + b) \geq 1, \; i = 1, \ldots, n \end{aligned} \tag{6.6}$$

realizes the maximal margin hyperplane with geometric margin $M = \frac{1}{||\omega||}$.

The optimization problem can be solved transforming it into its corresponding dual problem, because the latter is, generally, easier to solve than the primal one. The dual problem is obtained by the Lagrangian form of the primal problem (6.6):

$$L(\omega, b, \alpha) = \frac{1}{2}||\omega||^2 - \sum_{i=1}^{n} \alpha_i (y_i(\omega \cdot x_i + b) - 1) \tag{6.7}$$

where α_i are the Lagrangian multipliers. If the Lagrangian form is maximized with respect to the multipliers $\alpha_i \geq 0$ and minimized with respect to ω and b:

$$\min_{||\omega||,b} \max_{\alpha} \; L(\omega, b, \alpha) \tag{6.8}$$

it can be shown that the solution of this problem is the same of the solution of the primal problem (6.6).

Let (ω^*, b^*) be a pair of values for the problem (6.6). If (ω^*, b^*) do not satisfy all the constraints of (6.6) then $\max_\alpha \; L(\omega, b, \alpha)$ tends to infinite and hence (ω^*, b^*) is not a solution of (6.8). If (ω^*, b^*) satisfy all the constraints of (6.6) then $\max_\alpha \; L(\omega, b, \alpha) = \frac{1}{2}||\omega||^2$, hence the solution of (6.8) is equal to the solution of (6.6).

Necessary conditions for a point (ω, b) to be a minimum of the primal problem (6.6) are the following:

$$\frac{\delta L(\omega, b, \alpha)}{\delta b} = 0 \Rightarrow \sum_{i=1}^{r} \alpha_i y_i = 0$$
$$\frac{\delta L(\omega, b, \alpha)}{\delta \omega} = 0 \Rightarrow \omega = \sum_{i=1}^{r} \alpha_i y_i x_i \tag{6.9}$$

The Lagrangian (6.7) can be rewritten as:

$$L(\omega, b, \alpha) = \frac{1}{2}(\omega \cdot \omega) - \sum_{i=1}^{r} \alpha_i (\omega \cdot x_i) - b \sum_{i=1}^{r} \alpha_i y_i + \sum_{i=1}^{r} \alpha_i \tag{6.10}$$

Using the strong duality theorem [82]:

$$\min_{||\omega||, b} \max_{\alpha} \; L(\omega, b, \alpha) = \max_{\alpha} \min_{||\omega||, b} \; L(\omega, b, \alpha) \tag{6.11}$$

Substituting the conditions (6.9) in the right part of (6.11), the dual problem is then obtained:

$$\begin{aligned} &\max_{\alpha} \; W(\alpha) = \frac{1}{2} \textstyle\sum_{i,j} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \\ &\text{s.t.} \\ &\alpha_i \geq 0, \forall \; i = 1, \ldots, r \end{aligned} \tag{6.12}$$

Therefore, the minimum of the primal problem (6.6) coincides with the maximum of the dual problem (6.12). The latter is also a quadratic programming problem (convex quadratic functional and linear constraints) with a unique minimum that can be found using standard optimization software.

Let α^* be the solution of the dual problem. The second condition of (6.9) is $\omega^* = \sum_{i=1}^{r} \alpha_i^* y_i x_i$ hence ω^* is a linear combination of training set points. Furthermore, from the Kuhn-Tucker theorem [141] it is known that the solution has to satisfy:

$$\alpha^* (y_i (\omega^* \cdot x_i + b^*) - 1) = 0, \quad \forall \; i = 1, \ldots, n \tag{6.13}$$

This relation, known as Karush-Kuhn-Tucker (KKT) complementary condition, is a necessary condition for the optimum. It determines that for inactive constraints $\alpha_i^* = 0$ and for active constraints $\alpha_i^* \geq 0$.

The points x_i that correspond to non zero Lagrangian multipliers are called Support Vectors (SV), $SV = \{x_i : \alpha_i^* > 0\}$. Since for each SV the corresponding constraint is active ($y_i(\omega^* \cdot x_i + b^*) = 1$), their distance from the hyperplane is equal to the margin (Fig. 6.4).

The vector ω^* that determines the slope of the hyperplane is computed as linear combination of SVs:

$$\omega^* = \sum_{SV} y_i \alpha_i^* x_i \tag{6.14}$$

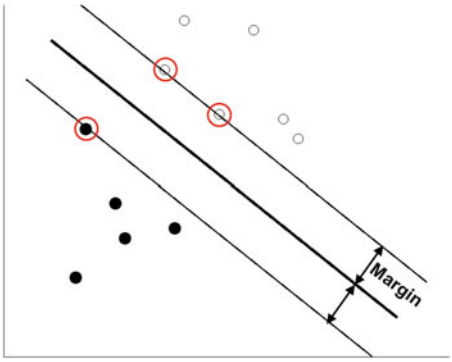

Fig. 6.4 The circled points are the SVs that determine the hyperplane

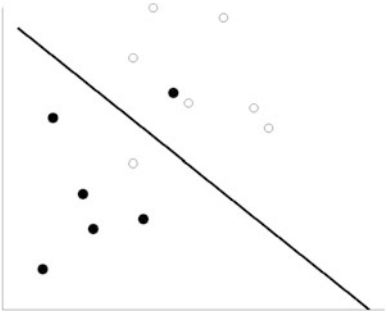

Fig. 6.5 Example of a nonlinearly separable set

The b^* can be computed using the KKT corresponding to just anyone of the support vectors:

$$y_i(\omega^* \cdot x_i + b^*) = 1 \Rightarrow b^* = y_i - \sum_{SV} y_j \alpha_j^* (x_j \cdot x_i) \tag{6.15}$$

Substituting in the decision function the expression of ω^*, we obtain:

$$\hat{f}(x) = \Theta\left(\left(\sum_{SV} y_i \alpha_i^* x_i\right) \cdot x + b^*\right) = \Theta\left(\sum_{SV} y_i \alpha_i^* (x_i \cdot x) + b^*\right) \tag{6.16}$$

6.1.2 Soft margin classification

If the training set is non-linearly separable (Fig. 6.5), there is no hyperplane that can correctly classify all the points. However, it is clear that some hyperplanes are preferable than others for this task. A possible strategy consists in the minimization of the misclassification error, that is the number of points incorrectly classified, and in the maximization at the same time, of the margin for the points correctly classified.

In order to realize that strategy, called *soft margin*, the constraints are relaxed by means of the introduction of the slack variables, $\xi_i \geq 0$. The constraints in (6.6) are reformulated as:

$$y_i(\omega \cdot x_i + b) \geq 1 - \xi_i, \forall\, i = 1, \ldots, n \tag{6.17}$$

The classification error for the training set can be measured as $\sum_{i=1}^{n} \xi_i^p$, where $p \in \mathbb{R}$. The minimization problem can be expressed as:

$$\min \frac{1}{2}(\omega^T \cdot \omega) + C \sum_{i=1}^{n} \xi_i^1 \tag{6.18}$$

where p has been chosen equal to one. C is a regularization constant that determines the trade-off between misclassified samples and the maximization of the margin. The corresponding Lagrangian is:

$$L(\omega, b, \xi, \alpha, q) = \frac{1}{2}(\omega \cdot \omega) + C \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i \left(y_i(\omega \cdot x_i + b) - 1 + \xi_i\right) - \sum_{i=1}^{n} q_i \xi_i \tag{6.19}$$

with $\alpha_i \geq 0$ and $q_i \geq 0$. The dual form is found by differentiating with respect to ω, ξ, and b and imposing stationarity:

$$\begin{aligned} \frac{\delta L(\omega, b, \xi, \alpha, r)}{\delta \omega} &= \omega - \sum_{i=1}^{r} y_i \alpha_i x_i = 0 \\ \frac{\delta L(\omega, b, \xi, \alpha, r)}{\delta \xi_i} &= C - \alpha_i - q_i = 0 \\ \frac{\delta L(\omega, b, \xi, \alpha, r)}{\delta b} &= \sum_{i=1}^{r} y_i \alpha_i = 0 \end{aligned} \tag{6.20}$$

As in the separable case, from substituting the previous conditions in the Lagrangian, the following dual problem is found:

$$\begin{aligned} &\max_{\alpha} W(\alpha) = \tfrac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \\ &\text{s.t.} \\ &\sum_{i=1}^{r} \alpha_i y_i = 0, \\ &0 \leq \alpha_i \leq C, \forall\, i = 1, \ldots, r \end{aligned} \tag{6.21}$$

(6.21) differs from (6.12) only for the constraints on the multipliers. In this case, the KKT conditions are:

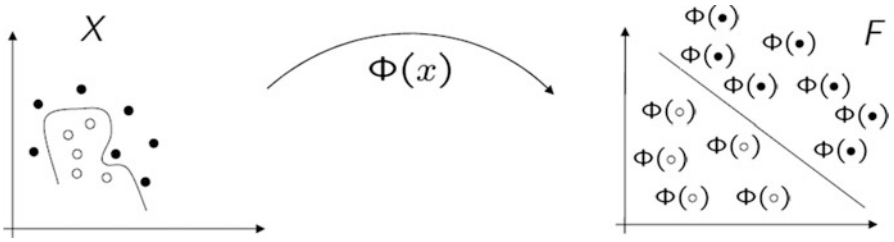

Fig. 6.6 The function ϕ maps the data in another space called feature space

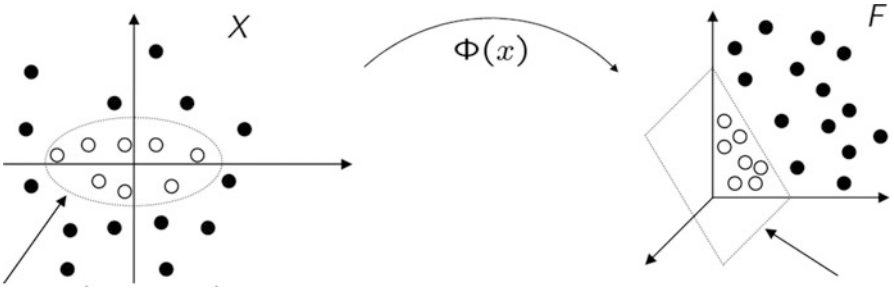

Fig. 6.7 In the feature space the data become linearly separable, and the SVM algorithm can be applied

$$\begin{aligned} \alpha_i \left(y_i(\omega \cdot x_i + b) - 1 + \xi_i\right) &= 0 \\ \xi_i(\alpha_i - C) &= 0 \end{aligned} \tag{6.22}$$

As in the separable case the solution is sparse. The points for which $\alpha_i^* = C$ are called *bounded support vectors* and they have non-zero associated slack variables (second KKT condition). The points for which $0 \leq \alpha_i^* < C$ are called *unbounded support vectors* and they have null slack variables. The decision function is equal to (6.16).

6.1.3 Non-linear classification

Since SVMs are linear machines, they are able to compute only hyperplanes. Hence, they perform poorly on classification problems where the data are not linearly separable. The strategy used to realize a non linear classification with a linear machine is based on the idea that the data can be mapped in another space, called *feature space* (Fig. 6.6). Since, generally, the feature space has higher dimension, the data in this space can become linearly separable, which allows the use of the SVM algorithm (Fig. 6.7).

Having the $\phi : X \rightarrow F$ mapping function, the decision function of the primal problem becomes:

$$\hat{f}(x) = \Theta(\omega \cdot \phi(x) + b) = \Theta\left(\sum_{j=1}^{D} \omega_j \phi_j(x) + b\right) \tag{6.23}$$

where D is the dimension of the feature space ($\omega \in \mathbb{R}^D$). The primal problem becomes an optimization problem in a D-dimensional space. If D is large, computation problems arise both for the optimization and for the evaluation of the decision function. The dual formulation becomes:

$$\begin{aligned} &\max_\alpha W(\alpha) = \tfrac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (\phi(x_i) \cdot \phi(x_j)) \\ &\text{s.t.} \\ &\textstyle\sum_{i=1}^{n} \alpha_i y_i = 0, \\ &0 \leq \alpha_i \leq C, \forall\ i = 1, \ldots, n \end{aligned} \tag{6.24}$$

and the relative decision function is:

$$\hat{f}(x) = \Theta\left(\sum_{i=1}^{n} y_i \alpha_i (\phi(x_i) \cdot \phi(x)) + b\right) \tag{6.25}$$

We note that the elements $\phi(x_i)$ appear only in dot products in both dual problem and in the decision function. Computing directly the results of these dot products is possible to avoid the explicit computation of ϕ function. The *kernel* function, introduced in the next section, is the tool that allows the direct computation of the dot products.

6.1.3.1 Kernel

A *kernel* is a function $k : X \times X \rightarrow \mathbb{R}$ such that:

$$k(x, x') = \phi(x) \cdot \phi(x'), \qquad \forall\ x, x' \in X \tag{6.26}$$

where ϕ is a map from the input space X to space F endowed with the dot product operation. The kernel defines implicitly the map ϕ and then can be used to find the optimal hyperplane in the space F. Hence, the explicit computation of $\phi(x)$ can be avoided using the kernel (this technique is also known as the "kernel trick").

The kernel can be found from the mapping function ϕ but generally it is set a priori and the function ϕ can remain unknown. As the dot product is commutative, the kernel has to be symmetric:

$$k(x, z) = k(z, x), \forall\, x, z \in X \tag{6.27}$$

Let X an input domain, it can be proved that a symmetric function $k(\cdot, \cdot) : X \times X \rightarrow \mathbb{R}$ is a kernel if and only if the matrix:

$$K = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_r) \\ \vdots & \vdots & \vdots & \vdots \\ k(x_r, x_1) & k(x_r, x_2) & \cdots & k(x_r, x_r) \end{bmatrix} \in \mathbb{R}^{r \times r} \tag{6.28}$$

is positive semidefinite for any subset $\{x_1, \ldots, x_r\} \subset X$. In the SVM context, K contains the values of the kernel for any combination of the input point pairs and it is called *kernel matrix*. This information is used in the dual optimization problem. By using the kernel the classification problem is formulated as:

$$\begin{aligned} &\max_{\alpha} W(\alpha) = \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \\ &\text{s.t.} \\ &\sum_{i=1}^{r} \alpha_i y_i = 0, \\ &0 \leq \alpha_i \leq C, \forall\, i = 1, \ldots, r \end{aligned} \tag{6.29}$$

The decision function becomes:

$$\hat{f}(x) = \Theta \left(\sum_{i=1}^{r} y_i \alpha_i k(x_i, x) + b \right) \tag{6.30}$$

6.1.4 Support Vector Regression

6.1.4.1 Linear regression

As illustrated more formally in Chap. 4, the regression problem can be formulated as the search of a function $\hat{f}$, given a training set $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, $x_i \in X \subseteq \mathbb{R}^D$, $y_i \in Y \subseteq \mathbb{R}$, such that $\hat{f}(x) : X \rightarrow Y$ is able to approximate the given samples, $\hat{f}(x_i) \approx y_i$. The solution is searched optimizing a *loss function* that measures the difference between the prediction and the real value of the samples. Two common loss functions are the following:

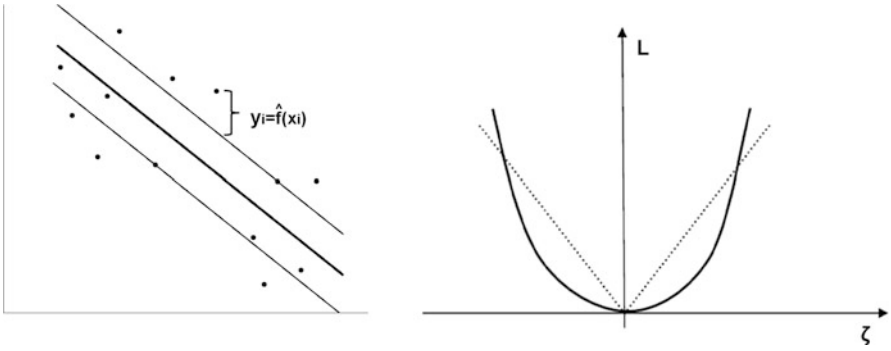

Fig. 6.8 The goodness of an approximation, $\hat{f}$, is evaluated on the distance between a point and its approximation on the surface (left panel). This distance is evaluated by means of a loss function, which, for instance, can be quadratic or linear (right panel)

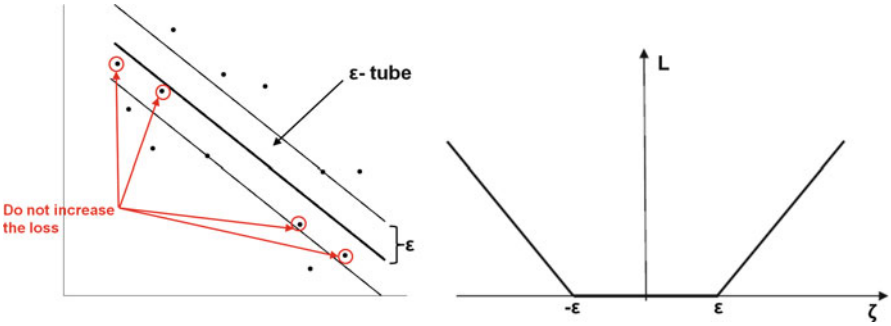

Fig. 6.9 The points inside the ε-tube do not contribute to the training error (on the left), while the points outside the ε-tube contribute linearly to the error due to the shape of the loss function (on the right)

- Quadratic loss, $L(\zeta) = \zeta^2$
- 1-norm loss, $L(\zeta) = |\zeta|$

where $\zeta = y - \hat{f}(x)$ (Fig. 6.8).

In classification problems only the points that lie close to the hyperplane for less than the margin contribute to the loss function; this guarantees the sparsity of the solution. In order to guarantee this property also in the regression case Vapnik introduced a family of loss function called ε-*insensitive*. In the case of 1-norm loss (Fig. 6.9) the ε-insensitive version results:

$$L_1^{\varepsilon}(\zeta) = \begin{cases} 0 & \text{if } |\zeta| \leq \varepsilon \\ |\zeta| - \varepsilon & \text{otherwise} \end{cases} \tag{6.31}$$

The SVR problem can be formulated (1-norm) as follows:

$$\min_{\omega,b} \frac{1}{2}(\omega \cdot \omega) + C\frac{1}{n}\sum_{i=1}^{n} L_1^{\varepsilon}(|y_i - (\omega \cdot x_i) - b|) \tag{6.32}$$

The cost function can be seen as composed of two terms: the first, $\frac{1}{2}(\omega^T \cdot \omega)$, controls the slope of the solution, while the second, $\frac{1}{n}\sum_{i=1}^{n} L_1^{\varepsilon}(|y_i - (\omega \cdot x_i) - b|)$, controls the approximation error.

By the absorption of the term $\frac{1}{n}$ in the constant C and introducing the slack variables the problem becomes:

$$\begin{aligned} &\min_{\omega,b} \frac{1}{2}(\omega \cdot \omega) + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \\ &\text{s.t.} \\ &y_i - (\omega \cdot x) - b \leq \varepsilon + \xi_i, \forall\, i = 1, \ldots, n \\ &(\omega \cdot x) + b - y_i \leq \varepsilon + \xi_i^*, \forall\, i = 1, \ldots, n \\ &\xi_i, \xi_i^* \geq 0, \forall\, i = 1, \ldots, n \end{aligned} \tag{6.33}$$

The constant C is the regularizer parameter and it controls the trade-off between the slope of the hyperplane and the training error measured according to (6.31).

As for the classification case, the Lagrangian of the primal problem is formulated introducing the multipliers $\alpha_i, \alpha_i^*, \eta_i, \eta_i^*$:

$$\begin{aligned} L(\omega, b, \alpha, \alpha^*, \eta, \eta^*) = &\frac{1}{2}||\omega||^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \\ &- \sum_{i=1}^{n}\alpha_i(\varepsilon + \xi_i - y_i + (\omega \cdot x_i + b)) \\ &- \sum_{i=1}^{n}\alpha_i^*(\varepsilon + \xi_i^* + y_i - (\omega \cdot x_i - b)) \\ &- \sum_{i=1}^{n}(\eta_i\xi_i + \eta_i^*\xi_i^*) \end{aligned} \tag{6.34}$$

The minimum conditions result:

$$\frac{\delta L}{\delta b} = 0 \Rightarrow \sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0$$

$$\frac{\delta L}{\delta \omega} = 0 \Rightarrow \omega = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)x_i$$

$$\frac{\delta L}{\delta \xi_i} = 0 \Rightarrow \eta_i = C - \alpha_i$$
$$\frac{\delta L}{\delta \xi_i^*} = 0 \Rightarrow \eta_i^* = C - \alpha_i^* \tag{6.35}$$

From the substitution of the previous conditions in the Lagrangian, the dual problem is obtained:

$$\max_{\alpha_i, \alpha_j} \sum_{i,j=1}^{n} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)(x_i \cdot x_j) - \varepsilon \sum_{i=1}^{n} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} y_i(\alpha_i + \alpha_i^*)$$

s.t. (6.36)

$$\sum_{i=1}^{n} (\alpha_i - \alpha_i^*) = 0$$

$$\alpha_i, \alpha_i^* \in [0, C], \forall\, i = 1, \ldots, n$$

The problem is, again, a maximization of a convex quadratic functional with linear constraints. Hence it is a quadratic optimization problem and then the solution is unique. For this problem, the KKT conditions result:

$$\begin{aligned} \alpha_i(\varepsilon + \xi_i - y_i + (\omega \cdot x_i) + b) &= 0 \\ \alpha_i^*(\varepsilon + \xi_i^* + y_i - (\omega \cdot x_i) - b) &= 0 \\ (C - \alpha_i)\xi_i &= 0 \\ (C - \alpha_i^*)\xi_i^* &= 0 \end{aligned} \tag{6.37}$$

The first two conditions assure that the Lagrangian multipliers are zero for each point lying in the ε-tube. Then the solution is sparse. The last two conditions say that each point lying outside the ε-tube is a bounded support vector, namely $\xi_i > 0 \Rightarrow \alpha_i = C$. Each point lying on the margin ($y_i - (\omega \cdot x_i) + b = \varepsilon, \xi = 0$), has Lagrangian multipliers with value in the range $[0, C]$, $0 \leq \alpha_i \leq C$ (Fig. 6.10).

The solution can be expressed as:

$$\hat{f}(x) = \sum_{SV} (\alpha_i - \alpha_i^*)(x_i \cdot x) + b \tag{6.38}$$

The value of b is computed using the KKT conditions. In Fig. 6.11, an example of linear regression is shown for several value of ε. The training set is obtained sampling a linear function (Fig. 6.11a) and a random uniform quantity has been added to the samples.

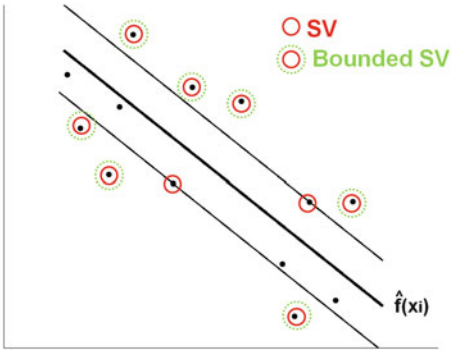

Fig. 6.10 Only the points that do not lie inside the ε-tube are SVs. Every point outside the ε-tube is a bounded SV, and the value of their α_i is C

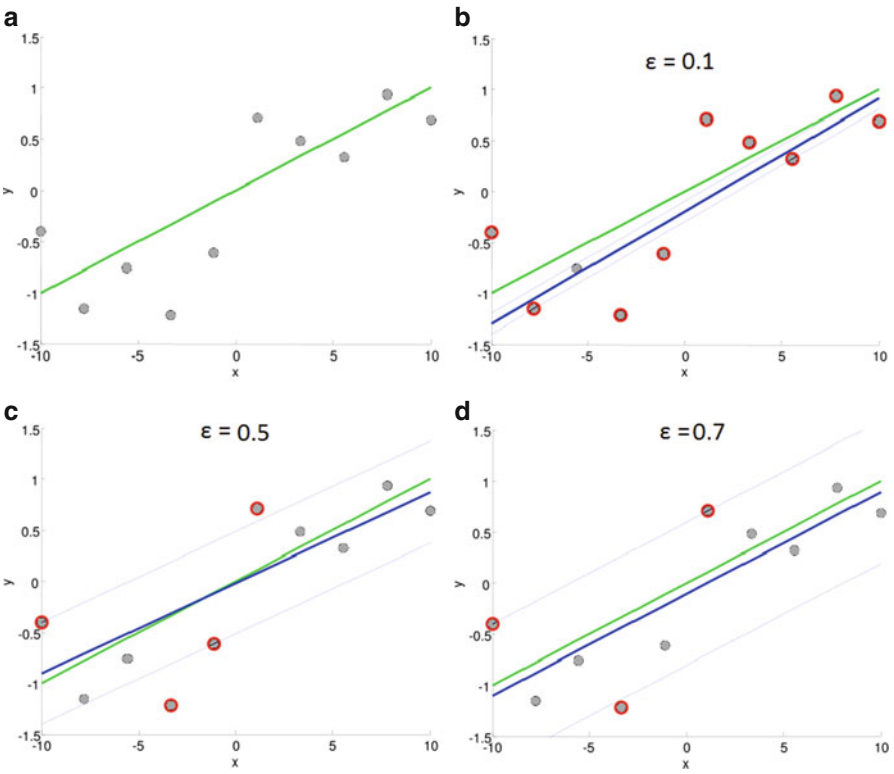

Fig. 6.11 In (**a**), a training set is obtained sampling a linear function in 10 points and adding a random uniform quantity to the points. In (**b**), (**c**), and (**d**) the SVR approximation obtained with ε-tube respectively of 0:1, 0:5, and 0:7 is shown by the darker line. The circled points are the SVs

6.1.4.2 Non-linear regression

Since the SVR are linear machines, they are able to compute just linear regression. As for the classification problem, in order to obtain a non-linear solution the training set is mapped to another space (characterized by higher dimension) in which the problem becomes linear. The solution is then computed in that space. Since the feature space can have infinite dimensions, the computation of the mapping function can be unfeasible. Fortunately, by the use of the kernel, also in this case, the explicit computation of this function can be avoided.

First of all, it can be noted that both in the dual regression problem (6.36) and in the decision function (6.38) the training points appear only as the dot product of pairs of them. The kernel function can be used to compute the result of these dot products. By using the kernel, the regression optimization problem has the form:

$$\begin{aligned} &\max_{\alpha_i,\alpha_j} \sum_{i,j=1}^{n}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)k(x_i,x_j) - \varepsilon\sum_{i=1}^{n}(\alpha_i+\alpha_i^*) + \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \sum_{i=1}^{n} y_i(\alpha_i+\alpha_i^*) \\ &\text{s.t.} \\ &\sum_{i=1}^{n}(\alpha_i-\alpha_i^*)=0 \\ &\alpha_i,\alpha_i^* \in [0,C], \forall\, i=1,\ldots,n \end{aligned} \tag{6.39}$$

The decision function becomes:

$$h(x)=\sum_{SV}(\alpha_i-\alpha_i^*)k(x_i,x)+b \tag{6.40}$$

Another form of the dual problem is, generally, preferred. Substituting $\alpha-\alpha^*$ with β and taking into account that $\alpha_i\alpha_i^*=0$ the following alternative form is obtained:

$$\begin{aligned} &\max_{\beta} \sum_{i,j=1}^{n}\beta_i\beta_j k(x_i,x_j) - \quad \varepsilon\sum_{i=1}^{n}|\beta_i| + \sum_{i=1}^{n} y_i\beta_i \\ &\text{s.t.} \\ &\sum_{i=1}^{n}\beta_i=0 \\ &-C\le\beta_i\le C, \forall\, i=1,\ldots,n \end{aligned} \tag{6.41}$$

The decision function becomes:

$$h(x)=\sum_{SV}\beta_i k(x_i,x)+b \tag{6.42}$$

As the solution satisfies the KKT conditions, the following conditions on β_i hold:

$$|\beta_i| = \begin{cases} 0, & |y_i - f(x_i)| < \varepsilon \\ [0, C], & |y_i - f(x_i)| = \varepsilon \\ C, & |y_i - f(x_i)| > \varepsilon \end{cases} \tag{6.43}$$

Due to numerical approximation of the solution, a tolerance threshold δ, is introduced in practice, and (6.43) becomes:

$$|\beta_i| = \begin{cases} 0, & |y_i - f(x_i)| < \varepsilon - \delta \\ [0, C], & \varepsilon - \delta \leq |y_i - f(x_i)| \leq \varepsilon + \delta \\ C, & |y_i - f(x_i)| > \varepsilon + \delta \end{cases} \tag{6.44}$$

The Gaussian function is one of the most used kernels. When used, (6.42) assumes exactly the form of the function computed by the RBF network (5.2). Hence, both the SVRs and the RBF networks are characterized by a solution that is a linear combination of kernel functions.

6.2 Multi-scale hierarchical SVR

In the following sections the SVM regression using kernel functions is considered and an approach based on a hierarchical structure is presented. In order to support the explanation, some experimental results are reported.

6.2.1 Regression using a single kernel

A single kernel function is used in the standard SVR approach. This features a predefined shape characterized by a set of parameters and therefore has a predefined frequency content. The accuracy of the solution is strongly dependent on the choice of such kernel and the choice is often not straightforward [142]. Generally, the kernel is chosen through trial and error, considering a priori knowledge on the problem or using heuristic strategies.

There are datasets for which the use of a single kernel does not produce accurate solutions: for example when the data are characterized by a frequency content that

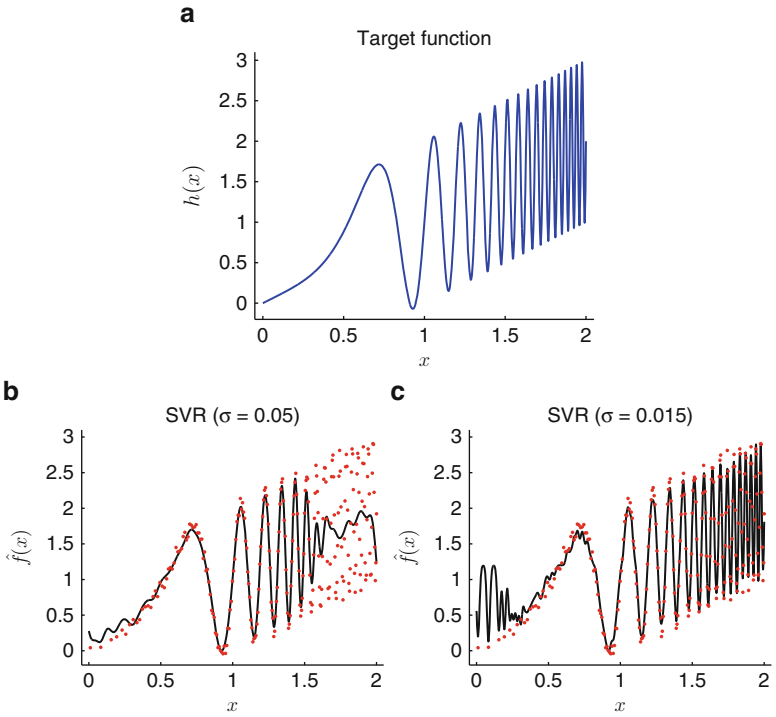

Fig. 6.12 (**a**) A function with non-stationary frequency content, and (**b**)–(**c**) two SVR using a single Gaussian kernel with two different scale parameters, σ. (**b**) A large scale kernel provides a smooth solution, but is unable to reconstruct the details, while (**c**) a small scale kernel suffers of overfitting, providing poor generalization. Originally published in [31] (reproduced by permission of IEEE)

varies in the space. Multiple kernels approaches have been introduced [195] [242] to cover these cases. In order to improve the accuracy of the solution, the kernel is defined as a linear combination of basic kernels and the coefficients of this combination are determined during the computation of the solution. With these approaches the number and the type of the basic kernels have to be chosen a priori. The form of the solution becomes:

$$\hat{f}(x) = \sum_{i=1}^{n} (\beta_i \sum_{j=1}^{m} \mu_j k_j(x, x_i)) + b \tag{6.45}$$

where m is the number of basic kernels and μ_j are the coefficients that have to be determined in the optimization phase. The solution is a linear combination of the same multiple kernels function.

An example of using a SVR with a single kernel is shown in Fig. 6.12. The data points have been sampled on the curve $f(\cdot)$

$$f(x) = \sin(2\pi x^4) + x \tag{6.46}$$

whose local frequency content increases with x (Fig. 6.12a). The sampling step is decreased with the local frequency according to $\frac{1}{120x}$. A large scale kernel determines a solution with large error in the high frequency regions (Fig. 6.12b). Vice versa, a small scale kernel determines an inaccurate solution in the low frequency regions (Fig. 6.12c).

As for the HRBF model, using a set of different layers, each one featuring a reconstruction at a certain scale, it is possible to cope with the problem shown before. In particular, each layer will be characterized by the same kernel function, but in the different layers different kernel function will be used. This model has been named Hierarchical Support Vector Regression (HSVR).

6.2.2 *Support vector regression with a hierarchy of kernels*

As the HRBF model, the HSVR model is composed of a set of layers organized in a hierarchical stack. The output of the model is computed as the sum of the outputs of each layer, $\{a_l(\cdot)\}$. Each layer is a SVR and it is characterized by the scale of its kernel. In case of the Gaussian kernel, these parameters are the Gaussian width. More formally, the solution computed by the HSVR model has the following form:

$$\hat{f}(x) = \sum_{l=1}^{L} a_l(x, \sigma_l) \tag{6.47}$$

where L is the number of layers and σ_l determines the scale of the kernel of the l-th layer. The scale of each layer decrease with the increase in the number of layers, $\sigma_l \geq \sigma_{l+1}$. One of the most used kernels is the Gaussian function; in this case the output of each layer has the following form:

$$a_l(x; \sigma_l) = \sum_{k=1}^{M_l} \beta_{l,k}\, G(||x - x_{l,k}||; \sigma_l) + b_l = \sum_{k=1}^{M_l} \beta_{l,k}\, e^{((x-x_{l,k})^2/\sigma_l^2)} + b_l \tag{6.48}$$

where M_l is the number of SVs, $\beta_{l,k}$ is the coefficient of the k-th SV and b_l is the bias of the l-th layer. The σ_l parameter determines the reconstruction scale of the l-th layer. The configuration of the HSVR model is realized layer by layer starting from that one with the largest σ, σ_1.

The configuration of first layer is realized using the dataset. For all the other layers, the configuration is performed considering the residual r_l at the previous

layer, namely the difference between the dataset and the output of the model configured up to l-layer. The residual r_l is computed as:

$$r_l(x_i) = r_{l-1}(x_i) - a_l(x_i) \tag{6.49}$$

The l-th layer will be configured with the training set, S_l, defined as:

$$S_l = \{(x_1, r_{l-1}(x_1)), \ldots, (x_n, r_{l-1}(x_n))\} \tag{6.50}$$

and $r_0(x_i) = y_i$ is assumed.

As for the HRBF, the value of scale parameter of the first layer, σ_1, could be chosen proportional to the size of the input data bounding box. A rule to decrease σ_l from one layer to the next should be also chosen. Halving the value of σ for each new layer seems a natural choice. The hierarchical structure could grow adding new layers until the reconstruction error does not decrease anymore. This error could be computed using a validation dataset, implementing a cross-validation like strategy.

Each SVR is defined also by two other parameters: ε, that determines the radius of the insensitivity circular region around the regression function and C_l, that controls the trade-off between the training error and the smoothness of the solution.

The value of C_l is generally found by trial and error strategy, although other strategies have been proposed [224, 240]. For the HSVR model, the setting of C_l has been related to the standard deviation of the residual of each layer with the following relationship:

$$C_l = J \,\text{std}(\, r_{l-1}(x_i)) \tag{6.51}$$

This stems from two different considerations. The first is that C_l represents the maximum weight of each Gaussian in (6.42). If there are regions in the input domain where the overlap among Gaussians is not significant, C_l represents also, approximately, the maximum value of the solution. Therefore, it should be set as large as the maximum value of the input data. The second consideration is that a too large value of C_l could produce overfitting. From these two considerations and experimental trials on different datasets it has been found that the interval $(0,\, 5]$ is an effective range for J. However, as shown in Sect. 6.2.4, the accuracy is pretty independent on the J value.

ε is the same for all the layers and, as in standard SVR, cannot be determined from the dataset. In general, it is linearly related to data noise amplitude [214].

6.2.3 Training set reduction

Although the previous scheme does solve the problem of reconstructing a function with different frequency content in different input regions, it presents the drawback of requiring a large number of SVs. This is due to to the fact that the number of SVs is strongly related to ε and the HSVR model uses the same ε for all the layers. The total number of SVs is then generally close to the number of layers multiplied

by the number of SVs of the first layer. As the first layers compute a very smooth approximation of the data it seems unreasonable that they required a large number of SVs with large scale parameter. This is due to the form of the solution computed by SVR: all the points outside the ε-tube are selected as SVs (6.43).

This problem is faced configuring each layer two times. The first time the regression is computed using all the data to select few points that are meaningful for that layer. The second time, the regression can be computed using only a subset of selected points. We note that the points that lie close to the regression function are those that are better explained by the approximation itself while those that lie far from the function can be regarded as outliers. For this reason, the configuration of the layer using just few meaningful points produces an approximation similar to that computed using all the points with a substantial decrease in the total number of SVs. For each layer the error introduced using the selected set of points is, generally, small (cf. Fig. 6.13). On the contrary, the number of SVs decreases significantly. The selected set of points used in the second step for configuring a layer is defined as:

$$S'_l = \left\{ (x_i,\, r_{l-1}(x_i)) \text{ s.t. } ||r_l(x_i)| - \varepsilon| < \delta \vee |r_l(x_i)| < \frac{\varepsilon}{2} \right\} \tag{6.52}$$

where δ is the numerical approximation tolerance threshold (6.44). This set is composed of the points that lie on the border of ε-tube and those that lie far from the approximation less of $\varepsilon/2$. It is remarked that, for each layer, the approximation is computed in two different steps. In the first step an approximation of the residual is computed using all the training points. The aim of this phase is the computation of S'_l. In the second step, using only the points in S'_l, the final regression for that layer is obtained. As in the second step the density of the points is significantly less than in the first step, experimental results have suggested to increase, for the second step, the value of C_l proportionally to the change in data density:

$$C'_l = C_l \frac{|S_l|}{|S'_l|} = J \,\text{std}(\, r_{l-1}(x_i)) \frac{|S_l|}{|S'_l|} \tag{6.53}$$

It should be remarked that, similarly to the HRBF model [97] [91], the error introduced by the reduction strategy is not critical because it will be contained in the residual taken care in the next layer where it is recovered. Other reduction procedures for SVR [152] [175] have been proposed in the literature and they will be compared and discussed in Sect. 6.2.7.

6.2.4 Experimental results

In order to evaluate the performance of the HSVR model with respect to the standard SVR model [215] for 3D scanning, an experimental comparison has been

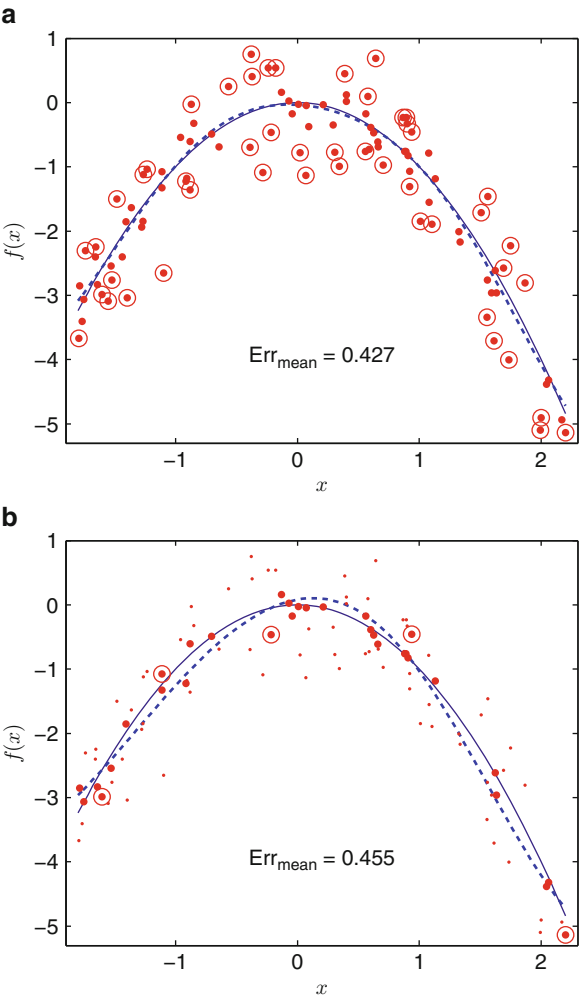

Fig. 6.13 Data points reduction. A smooth function is shown in thin line in both panels. A cloud of 100 points have been obtained randomly sampling the function and adding random Gaussian quantity to the points sampled. These points are displayed as dots. Thick dots represent all the points used by the optimization engine to determine the regression represented as a thick dashed line. Circled dots represent the SVs. Err_{mean} is computed as $1/n \sum_{i=1}^{n} |\hat{f}(x_i) - y_i|$. In panel (**a**) the SVR curve obtained through standard SVR ($\varepsilon = 0.416$, $C = 9.67$, $\sigma = 1.66$, $\text{Err}_{\text{mean}} = 0.427$) and in panel (**b**) the solution obtained considering only the points in S_l' (6.52). Note that in the latter case only 32 points are used in the optimization procedure (the unused points are shown as small dots). The number of SVs drops from 49 to 5. Both solutions are contained inside an ε-tube around the real function. Originally published in [31] (reproduced by permission of IEEE)

carried out. The experiments have been performed on simulated and real data. The obtained surface has been evaluated from a qualitative point of view analyzing the surface aspect and from a quantitative point of view using the following indices: the root mean square error (RMSE), the mean absolute error (Err_{mean}) and its standard deviation (Err_{std}). The datasets have been partitioned in three subsets: training set, validation test and test set. The first one is used to configure the models, the second to select, with respect to the accuracy, the best combination of hyperparameters (ε, σ, and C for the standard SVR model, and ε and C only for the HSVR model) and the last to evaluate the selected model.

For the HSVR model the validation set is used also to decide when the insertion of new layers has to be stopped. The σ of the first layer is taken equal to the size

of the input domain. The value of C of each layer is set as indicated in (6.51) and (6.53).

The LibCVM Toolkit Version 2.2 [231] has been used to solve the optimization problem of the both HSVR and standard SVR models. This software has been chosen because it shows the accuracy of SVM^{light} [132] (one of the most popular C implementation for SVM) with sensitive saving of computational time. The machine used for the experiments was equipped with an Intel Pentium 4, at 2.40 GHz, 512 KB cache, and 512 MB of memory.

6.2.5 *Regression on synthetic data*

The first experiment has been carried out using a 2D synthetic dataset to better appreciate the regression properties. The set of points has been obtained sampling the univariate function defined in (6.46). This function, having different frequency content in different input domain regions, allows for emphasizing the limits due to the use of a single scale kernel. The training set has 252 points, sampled with a varying step proportional to the frequency content. The values of a random uniform variable in the range of $[-0.1,\ 0.1]$ have been added to the data to simulate the measurement noise. The validation and test set have been obtained sampling uniformly the target function (6.46) in 500 points.

The trials on standard SVR have been performed on all the possible combinations of the following values of the hyperparameters, ε, σ, and C:

$$\varepsilon \in \{0.0, 0.005, 0.01, 0.025, 0.05, 0.075, 0.1, 0.2\} \tag{6.54}$$

$$\sigma \in \{0.015, 0.022, 0.0313, 0.0625, 0.125, 0.25\} \tag{6.55}$$

$$C \in \{0.5, 1, 1.5, 2, 5, 10, 20\} \tag{6.56}$$

The combination that determines the lowest validation error is selected as the best one. The hyperparameters of the best case are: $\varepsilon = 0.05$, $\sigma = 0.022$, and $C = 20$ and the approximation function computed is reported in Fig. 6.14. The poor accuracy of the solution is evident. Better approximations instead are computed by the HSVR models, as it is shown in Figs. 6.15.

In Table 6.1 the quantitative data for the different approaches are reported. The better accuracy of the hierarchical models is confirmed by these data. The last column contains the computational time required to configure the three models. For the hierarchical cases, this time concerns the configuration of all the layers. While for the standard SVR the computational time regards only the configuration of the model with the best combination of hyperparameters. The configuration time of all the hyperparameters combinations is 43.8 s for the standard SVR (336 trials), and 3.27 s and 4.49 s for HSVR without and with reduction (8 trials).

As the best case for SVR is that with the maximum tried value of C, other experiments with larger values of C have been performed. When C grows up to 100,000 the reconstruction error of SVR decrease to 0.0517. Although the

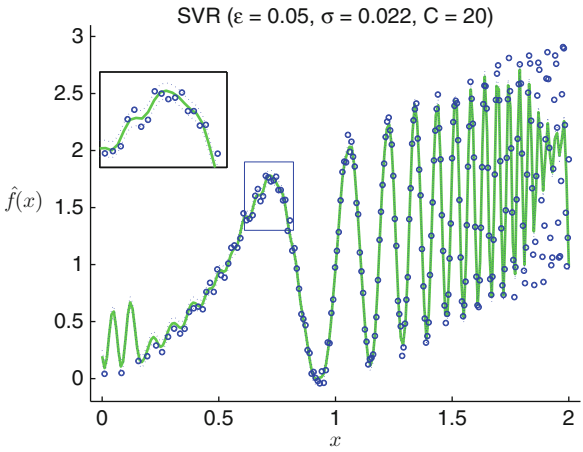

Fig. 6.14 Reconstruction provided by standard SVR with the best hyperparameters ($\varepsilon = 0.05$, $\sigma = 0.022$, $C = 20$). Note the poor approximation on the right side of the curve and spurious oscillations on the left. Originally published in [31] (reproduced by permission of IEEE)

computational time grows enormously to 3,129 s, also in this case the error for the SVR is larger than that one of HSVR.

When the approximation error is under the value of ε, the points, on average, lie inside the ε-tube. This happens for the SVR only when the value of ε is quite large (for $\varepsilon > 0.1$). Instead for the HSVR model the error is smaller than ε for $\varepsilon > 0.05$ and therefore also for the optimal value of 0.075 (cf. Fig. 6.16). The different frequency contribution of the HSVR layers is shown in Fig. 6.17. Looking at this figure it is clear as the different layers act at different level of detail. The approximations with and without reduction are very close each other and they have a similar time course. The effect of reduction for each layer is depicted in Fig. 6.18. The error for HSVR model without and with reduction after the first and second configuration step is represented with respect to the number of layers. The figure shows as the final error is almost coincident for the three cases. A more detailed description of the approximation error for the two HSVR models is summarized in Table 6.2.

6.2.6 Regression on 3D scanner data

The Fig. 6.19 represents a typical points cloud obtained by a 3D scanner. This dataset has been collected from a real artifact (a Panda mask) by means of the same 3D scanner used for Sect. 5.3.4 [91]. The dataset considered contains 22,000 points. The experiments have been carried out choosing randomly 18,000 points for training, 2,000 for validation, and 2,000 for testing. The coordinates of the points have been scaled in the range $[-1, 1]$. Similarly to Sect.5.3.4, only points that lie far from the boundary by more than 0.1 are considered for validation and test error as the border regions are generally characterized by a larger reconstruction error due to the lack of points.

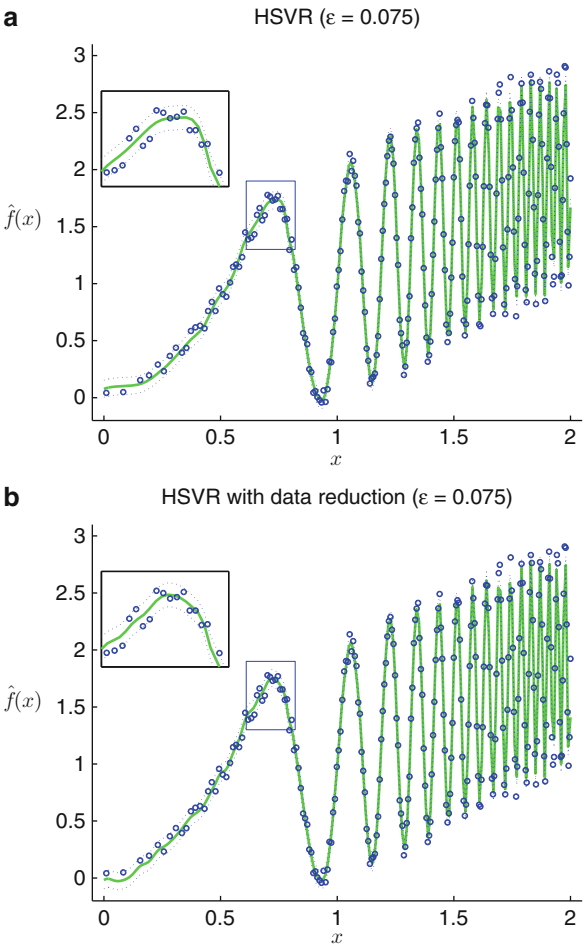

Fig. 6.15 Reconstruction provided by HSVR (**a**) and HSVR with data points reduction (**b**) with $\varepsilon = 0.075$. The dotted lines clearly visible in the boxes mark the border of the ε-insensitive region (i.e., the data points that lie inside this region do not increase the loss function value (6.32)). Originally published in [31] (reproduced by permission of IEEE)

Table 6.1 The reconstruction accuracy on the synthetic dataset of the models in Figs. 6.14 and 6.15

	Err_{mean}	Err_{std}	RMSE	#SVs	Time [s]
HSVR	0.0282	0.0262	0.0385	1545	0.308
HSVR (red.)	0.0313	0.0338	0.0460	243	0.382
SVR	0.0816	0.167	0.186	149	0.451

Published in [31] (reproduced by permission of IEEE).

Different surfaces have been created through SVR, using all the possible combinations of the following values of the hyperparameters, ε, σ, and J:

$$\varepsilon \in \{0, 0.0025, 0.005, 0.01, 0.02\} \tag{6.57}$$

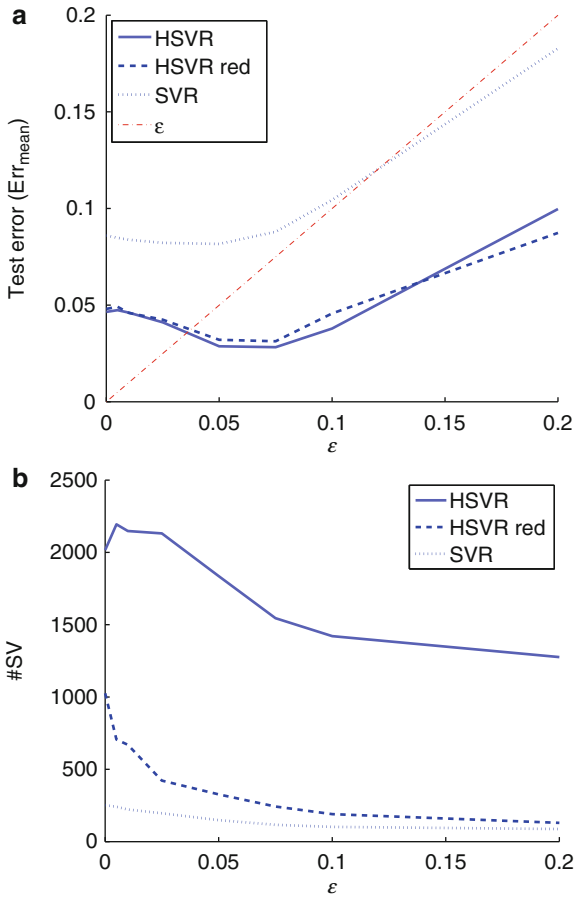

Fig. 6.16 (**a**) Mean test error and (**b**) number of SVs used, as a function of ε. For reference, in panel (**a**) the value of ε has been reported as a dot-dashed line. Originally published in [31] (reproduced by permission of IEEE)

$$\sigma \in \{0.188, 0.0938, 0.0469, 0.0234\} \tag{6.58}$$

$$J \in \{0.5, 1, 2, 5\} \tag{6.59}$$

C was set similarly to (6.51), namely:

$$C = J \,\mathrm{std}(y) \tag{6.60}$$

while only the different combinations of ε and J where considered to configure the HSVR model.

The surfaces associated to the lowest test error are shown in Fig. 6.20a–c. However, they have not the best appearance because of the presence of residual noise. Better surfaces can be obtained for the HSVR models, discarding some of the last layers, Fig. 6.20e–f. This is possible thanks to the hierarchical structure of the HSVR models and cannot be applied to SVR model. For SVR model we analyzed

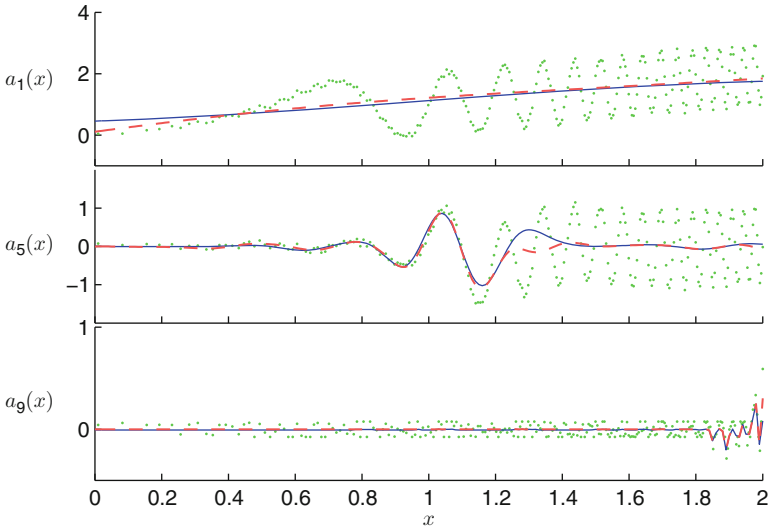

Fig. 6.17 Reconstruction operated by the 1-th, 5-th and 9-th layer of the HSVR model when all the data points are considered (dashed line) and when only the points in S_l' are considered (continuous line). The residual of each layer (i.e., the training points for that layer) are reported as dots. A small difference between the solution obtained with and without data point reduction can be observed. This difference becomes smaller and smaller and in the last layer, the two curves are almost coincident. Originally published in [31] (reproduced by permission of IEEE)

one by one by hands all the surfaces configured and chose the one with the best appearance, that is reported in Fig. 6.20.

The test error and the number of SV with respect to ε for the three models is reported in Fig. 6.21 for the best, the average and the worst case. These results have been obtained as the average over five randomizations of the data set. A quantitative representation of the best case for the three models is reported in Table 6.3. The best case is very similar from the accuracy point of view. This is true also for all the best models for each value of ε (Fig. 6.21a). An important difference is clear for the average and worst case. In fact the averaged (worst) test error is 0.0113 (0.012), 0.0116 (0.015), and 0.0140 (0.019) for respectively HSVR, HSVR with reduction, and SVR.

Another important difference is the configuration time required to obtain a good solution. This is 682 s for HSVR, 1,104 s for HSVR with reduction, and 382 s for SVR for the best case and of 1024 s, 1241 s, and 1093 s on average for all the surfaces created. However, we explicitly note that we have to consider also the time required to explore the hyperparameters space besides the configuration time. In particular, as for the HSVR the σ does not need to be selected in advance, the hyperparameters space has one dimension less than the SVR case. In practice, we tried 20 combination of J and ε to find the best case for HSVR without and with reduction for a total configuration time of 16,845 s and 22,168 s respectively. Instead

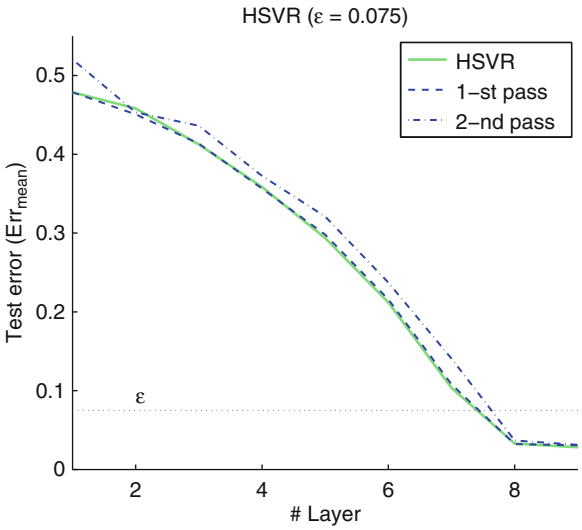

Fig. 6.18 The test error (Err_{mean}) of the HSVR models as new layers are inserted. The continuous line represents the error of the HSVR model with no data reduction. The dashed line represents the error of the model when data reduction was applied in all the previous layers, but not in the current one in which all the data points are passed to the optimization procedure (1-st optimization step). The dot dashed line represents the error of the HSVR model when data reduction is applied to all the layers included the current one (2-nd optimization step is applied to the configuration of all the layers). Originally published in [31] (reproduced by permission of IEEE)

for the SVR we tried 80 combinations of J, ε and σ for a total time configuration time of 87,410 s.

The total number of SVs is another critical aspect of these models. The HSVR uses a significant large amount of SVs (100,448) with respect to the standard SVR (Fig. 6.21 and Table 6.3). However, after pruning, the number of SVs drops to 11,351 saving 9.61% of SVs (Table 6.3) with respect to standard SVR.

6.2.7 Experimental results discussion

The HSVR model presented here has been developed for mainly three reasons: solve the problem connected to the use of a single scale kernel function, allow a multi-scale surface reconstruction, and speed-up the exploration of the hyperparameters space.

The standard version of SVR [215] is characterized by the use of a single kernel function. This has to be carefully chosen to obtain good results for a given data set. Moreover, poor approximation is still obtained for data with a different frequency content in different domain regions: Fig. 6.14 and Table 6.1 show that in this case the SVR approach is not able to compute an acceptable solution. In particular the best

Table 6.2 Configuration Data of the HSVR model ($\varepsilon = 0.075$)

	HSVR					HSVR red.				
# Layer	Err_{mean}	Err_{std}	RMSE	#SVs (tot.)	time [s] (tot.)	Err_{mean}	Err_{std}	RMSE	#SVs (tot.)	time [s] (tot.)
1	0.479	0.333	0.583	237 (237)	0.017 (0.017)	0.520	0.302	0.602	3 (3)	0.017 (0.017)
2	0.458	0.337	0.569	240 (477)	0.02 (0.037)	0.453	0.341	0.567	4 (7)	0.018 (0.035)
3	0.412	0.368	0.552	227 (704)	0.019 (0.056)	0.436	0.361	0.566	6 (13)	0.02 (0.055)
4	0.359	0.366	0.511	220 (924)	0.02 (0.076)	0.373	0.380	0.532	9 (22)	0.021 (0.076)
5	0.294	0.367	0.470	201 (1125)	0.047 (0.123)	0.321	0.385	0.501	15 (37)	0.052 (0.128)
6	0.212	0.329	0.391	171 (1296)	0.078 (0.201)	0.237	0.344	0.418	35 (72)	0.102 (0.23)
7	0.104	0.221	0.244	131 (1427)	0.065 (0.266)	0.141	0.273	0.307	51 (123)	0.073 (0.303)
8	0.0328	0.0390	0.051	82 (1509)	0.036 (0.302)	0.0367	0.0446	0.0577	79 (202)	0.067 (0.37)
9	0.0282	0.0262	0.0385	36 (1545)	0.006 (0.308)	0.0313	0.0338	0.046	41 (243)	0.012 (0.382)

Published in [31] (reproduced by permission of IEEE).

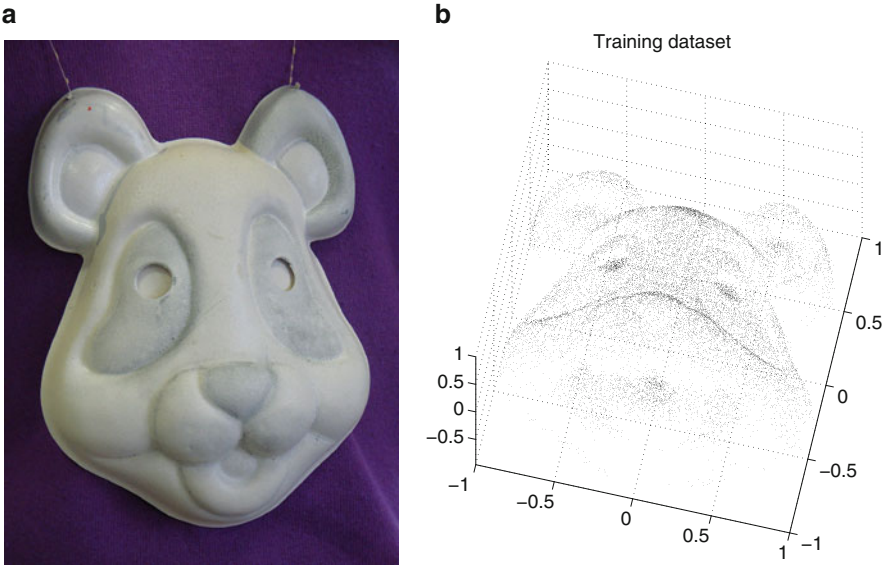

Fig. 6.19 Panel (**a**) shows the artifact (a Panda mask) that has been digitized obtaining the cloud of data reported in panel (**b**). Originally published in [31] (reproduced by permission of IEEE)

kernel function produces rapid oscillations in the smoothest region and it is not able to reconstruct the high frequency oscillations in the high frequency region. Thanks to the introduction of multiple scales, HSVR, is able to compute an accurate solution also in this case. As shown in Fig. 6.17 the smooth region is reconstructed by kernels with a large scale (first layers) and the high frequency regions are reconstructed by the kernels with a small scale parameter (last layers).

Moreover, a more accurate surface is obtained by the HSVR model for a wide range of ε as shown in Figs. 6.16a and 6.21a–c. The drawback of HSVR is the large amount of SVs used. The reduction procedure can effectively solve this problem (cf. Fig. 6.18), allowing a similar accuracy with a large saving in the number of SVs. This means that a large amount of SVs are not actually needed for the reconstruction; these are the points that lie far from the actual surface. We also remark that the reduction procedure can be applied because the hierarchical structure is able to recover in the next layers the small error introduced in one layer by this procedure as shown in Fig. 6.18.

The aim of the reduction strategy is the optimization of the trade-off between the number of points discarded and the increase of the error. The experimental results have shown that the rule described by (6.52) allows obtaining good results. By a closer analysis (6.13), the points passed to second configuration step can be be grouped in two kinds: the most internal to the ε-tube are used to constraint the function close to the target but are not SVs, those that lie on the border of the ε-

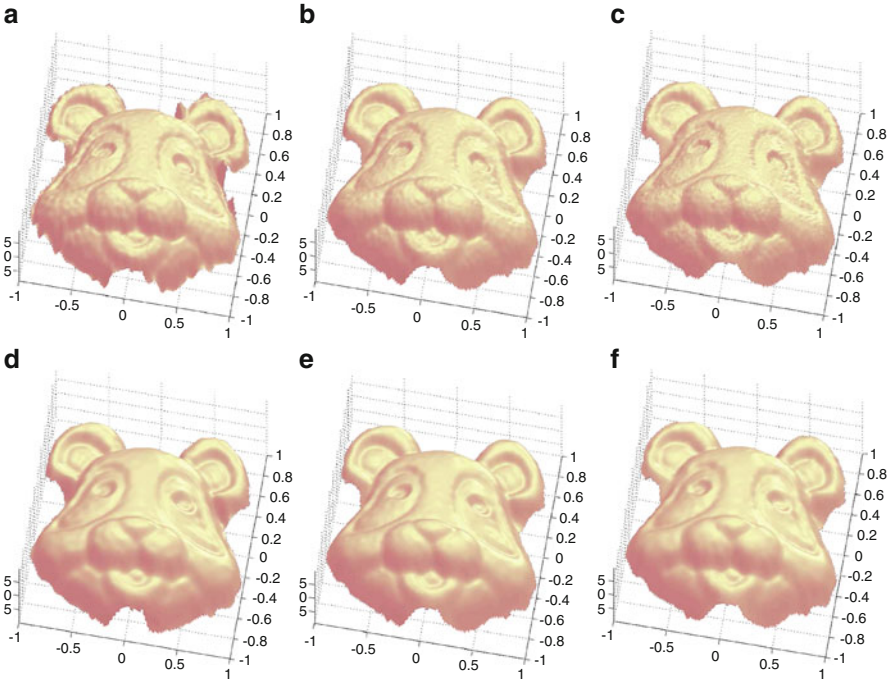

Fig. 6.20 Panels (**a**), (**b**), and (**c**) show the surfaces that determine the lowest test error for SVR, HSVR and HSVR with reduction. The parameter used were respectively $J = 0.5$, $\varepsilon = 0.005$ and $\sigma = 0.0469$ for SVR, $J = 0.5$ and $\varepsilon = 0.01$ for HSVR, and $J = 1$ and $\varepsilon = 0.01$ for HSVR with reduction. Although these surfaces are optimal in terms of the test error, their visual appearance is not of good quality. A better result is shown in panels (**d**), (**e**), and (**f**), for SVR, HSVR, and HSVR with reduction, respectively. In (**d**) the surface obtained through SVR with a suboptimal set of parameters ($J = 5$, $\varepsilon = 0.005$, and $\sigma = 0.0938$) is shown. In panels (**e**) and (**f**) the surface from the same HSVR models for (**b**) and (**c**) are used, but discarding some of the last layers (one of seven for (**e**) and three of ten for (**f**)). Originally published in [31] (reproduced by permission of IEEE)

tube, are SVs and support the surface. In the second step, a subset of both types of points will become the SVs of the current layer.

The reduction strategy produces a more sparse solution with respect to the first configuration step and the standard configuration strategy and it is particularly efficient when the input dataset is dense. For sparse dataset the use of this strategy could turn out not so advantageous. Sparse solutions have been investigated in several papers. The methods used can be divided in two families. The first family includes in the optimization problem a strategy to control the number of SVs used in the solution [109] [135]. The second family computes the solution in the standard way and then applies a strategy to select which SVs have to be discarded [175]. The strategy proposed here belongs to the second family. In general, the pruning methods consider very carefully the error introduced eliminating or limiting the

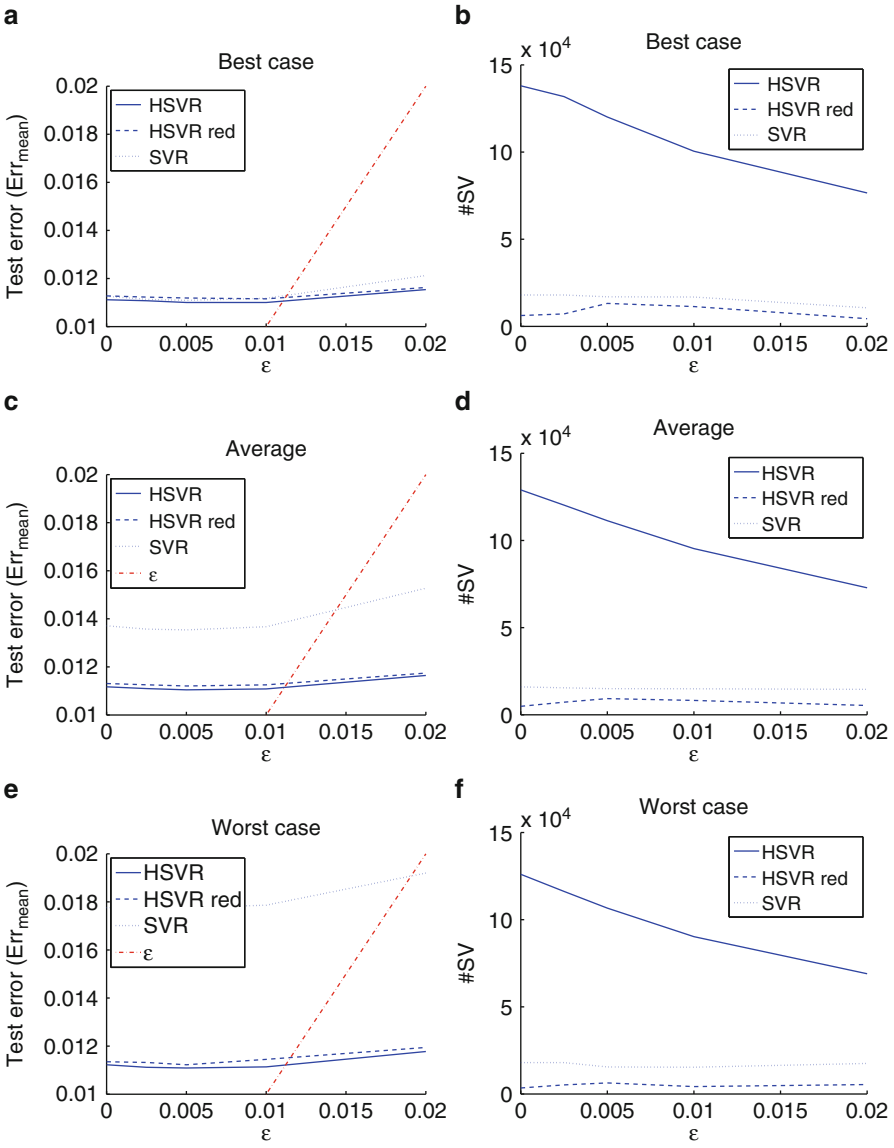

Fig. 6.21 Test error (**a, c, e**) and number of SVs Lettering in panels (**b, d, f**) seems smaller than lettering in panels (**a, c, e**) used by the SVR and HSVR model for the Panda dataset as a function of ε are reported. Results for the best, average and worst cases are plotted. For reference, the value of ε has been reported as a dot-dashed line. Originally published in [31] (reproduced by permission of IEEE)

number of SVs. The strategy proposed here can relax this constraint because the error introduced will flow into the residual input to the next layer where it can

Table 6.3 Results for "Panda mask" data set

	Err_{mean}	Err_{std}	RMSE	#SVs	time [s]
HSVR	0.0110	0.0115	0.0160	100 448	682
HSVR (red.)	0.0112	0.0119	0.0163	11 351	1 104
SVR	0.0111	0.0117	0.0161	12 442	382

Published in [31] (reproduced by permission of IEEE).

be recovered. Therefore the error introduced by the reduction is not so critical and the time spent to analyze it can be saved. For this reason this strategy is, generally, more efficient than the other reduction strategies proposed.

The error introduced by the reduction strategy is shown in Fig. 6.13a, b: the difference between the functions is due to the absence of some SVs. This is added to the residual of the layer and it is taken care by the next layer, where it is recovered as it is evident in Fig. 6.17. Furthermore the error due to the reduction becomes smaller as new layers are added, in the last layers the difference tends to vanish: comparing Fig. 6.20 the difference between the surface computed with and without reduction is not appreciable. This is confirmed also by the test error in Fig. 6.21. The drawback of the reduction strategy is the increase of the computational time due to the second step of configuration (Table 6.3). For the Panda dataset the time increases of 61.9% and the number of SVs decrease of 88.7%. Furthermore the HSVR with reduction compares well with standard SVR from the points of view of computational time and number of SVs.

Computational time and number of SVs, as well as the accuracy, are strongly related to the number of layers of the model. The strategy chosen to stop the growth of the HSVR model considered here is based on monitoring the validation error: when it does not decrease anymore the configuration procedure is stopped. Other criteria are possible. For example, if the level of noise on the data is known, configuration could be stopped when the training error goes under the noise level over the entire input domain [91]. Alternatively, the visual appearance of the surface could be monitored by the user, who can decide when stopping the configuration procedure. This would not be possible with the classical SVR approach.

The hyperparameters choice remains a critical aspect of all SVR approaches. In practice, the common strategy is based on trial and error. This means that the exploration of the hyperparameters space, constituted by C, σ and ε, is time consuming. HSVR allows reducing the dimensionality of the search space to two dimensions as optimal value of σ has not to be searched. Starting with a large σ and halving its value for each new layer generally assures that a scale locally adequate to the data will be automatically found (Fig. 6.17). Moreover, C can be set through a secondary parameter, J (6.51) that depends on the range of the output data. We have experimentally found that low values of J (in the range $(0, 5]$) are usually enough to obtain accurate results. Furthermore, as Fig. 6.21a–c shown that the HSVR approach is robust also with respect to different values of J as best, average and worst cases produce similar results.

Chapter 7
Conclusion

We review here the main characteristics of the HRBF and HSVR models. Possible future developments based on parallelization and GPU implementation are described.

7.1 Modeling through HRBF and HSVR

Modeling is an essential activity in a broad variety of the scientific and real-life problems. Modeling is directed to create a simplified representation of phenomena, objects, or physical processes which captures the essential relationships among the quantities and variables of interest.

3D models are used in many application fields: design, archeology, medicine, engineering, architecture, physics and entertainment. The use of a 3D digital model has various advantages. For example, in medicine the 3D models of internal organs are used for training and for planning minimal invasive surgery. Entertainment is the field in which digital models are most known. Nowadays, almost all video games are based on 3D models of the characters and environments, and in the film industry digital sets are seamlessly integrated with real sets. In engineering, the use of 3D models make it possible to simulate in real-time the interaction between the structures and the environment; similarly in architecture, the environmental impact of new buildings can be evaluated during the design phase.

3D models can be generated in two ways: CAD modeling, in which the model is built manually by a user, and physical object measurement, in which models are obtained by 3D scanners as described in this book. The output of 3D scanners is typically a cloud of points sampled on the object surface where the points are affected by measurement noise. From such cloud, a continuous description of the surface can be obtained. This can be regarded as a function approximation problem in the analytical domain and as a supervised learning problem in the connectionist/machine learning domains. This book describes several innovative models to achieve this.

F. Bellocchio et al., *3D Surface Reconstruction: Multi-Scale Hierarchical Approaches*,
DOI 10.1007/978-1-4614-5632-2_7,

In particular two kinds of paradigms for surface reconstruction have been considered:

On-line models. On-line models are particularly important for surface reconstruction because they allow reconstructing the surface in real-time while sampling points on the object's surface. Such paradigm enables fully interactive scanning as the user can drive the probe selectively in the regions where the details are still missing in the reconstructed surface. This will improve the effectiveness of the scanning procedure.

Hierarchical models. A hierarchical model is composed of a pool of sub-models. Each sub-model realizes the reconstruction up to a certain scale and the output of the whole hierarchy is computed as the sum of the output of each sub-model. Hierarchical models are important tools for surface reconstruction because they generally provide a robust and accurate model. It enables in a natural way LOD models, that are models with different Level Of Details.

The starting point is represented by two approaches: Radial Basis Function networks (Sect. 5.1) and Support Vector Machines (Sect. 6.1). The first one is a local method since surface approximation is based on local fitting of the data, while the second one is a global method in which the surface is viewed as the result of the optimization of a cost function computed over all the data points.

Both methods have been extended to provide a multi-scale incremental reconstruction leading to the Hierarchical Radial Basis Function model (HRBF, [46][97]) and to the Hierarchical Support Vector Regression model (HSVR, [31]). Moreover, an on-line version of the HRBF model [91], with real-time surface reconstruction has been developed and described in Sect. 5.3. HSVR has been complemented with an effective pruning of the number of Support Vectors as described in Sect. 6.2.

The results on simulated and real data show that the on-line version of the HRBF model can effectively perform a real-time reconstruction and the accuracy of the solution tends to the accuracy of the batch HRBF model after the acquisition of a reasonable amount of points (Figs. 5.9 and 5.10). The HSVR model allows a more accurate and robust reconstruction with respect to SVR, as kernels at different scales are automatically selected. Furthermore, the time required for identifying an adequate value for the hyperparameters is largely reduced. Pruning, on the other side, does not decrease the accuracy while it allows a large saving in the number of SVs.

7.2 Comparison of the HRBF and HSVR approaches

The HRBF, like all the local approaches, has the advantage that being based on local fitting it does not need iterative configuration procedures. In practice, this means that the configuration of a HRBF network is less computational demanding than configuring global models. This is confirmed by data reported in Table 7.1: the configuration time of a HRBF model is only the 7.2% (4.44%) of the time required to configure a HSVR without (with) pruning.

Table 7.1 Comparison of HRBF and HSVR models on the "panda mask" dataset

	Err_{mean}	Err_{std}	RMSE	#SVs	time [s]
HRBF	0.0112	0.0111	0.0158	14 679	49
HSVR [31]	0.0110	0.0115	0.0160	100 448	681
HSVR [31]	0.0112	0.0119	0.0163	11 351	1 104

Partially published in [31] (reproduced by permission of IEEE).

On the other hand, local approaches are more sensitive to outliers, which can strongly influence the local statistics while they have little influence on a global cost function, like the one used as training error in (6.31). The local approaches are also more sensitive to the sparseness of the data. The sparser the data, the fewer are the elements used to locally fit the surface. This means a less robust estimate. This is the reason for which, generally, local approaches do not show good performance with data that lie in spaces of high dimensionality as data sparseness increases exponentially with the number of space dimensions. The SVR (and HSVR) model has the advantage that the computational complexity of the configuration phase does not depend on the number of dimensions of the input space as this is hidden by the use of the kernel function. Moreover, in the 3D case, the influence region of each Gaussian of HRBF is assumed a square (close neighborhood, Sect. 5.3). Only the points that lie in this region are used to estimate the unit's weight. The approximation of the neighborhood of a point with a hypercube becomes less and less accurate with the increase of data dimensionality and many points further from the Gaussian center than points in its influence region do participate in the computation of the Gaussian's weight.

In the 3D space with sufficiently dense data sets, both methods reach the same level of accuracy (cf. Table 7.1). Even though the test error is very similar for these two approaches, the visual appearance of the surface computed by the HRBF model is better than that computed by HSVR (Fig. 7.1). This is due to the strategy used for placing the units. In the HSVR approach the position of the units is determined by the position of the training points, while in HRBF the units are placed on regularly spaced grid for each layer. Hence, in HRBF the input space is covered uniformly by the units, and this, generally, improves the smoothness of the reconstruction.

The position of the units of the different layers that contribute to the reconstruction of the function defined in (6.46) and shown in Fig. 6.12a is shown in Fig. 7.2a for HRBF and in Fig. 7.2b for HSVR.

As can be seen the SVs are somewhat more sparse on the input domain in all the layers, although their density increases on the right (high frequency) with the number of layers. HRBF model appears instead more selective and Gaussians at smaller scales are added only in the regions on the right, those of higher frequency content.

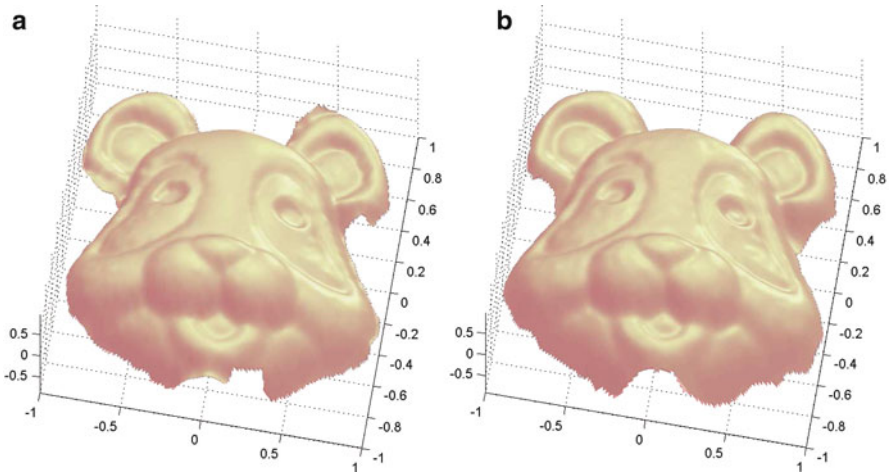

Fig. 7.1 (**a**) The surface computed by the HRBF model. (**b**) The surface computed by the HSVR model [31] (reproduced by permission of IEEE)

7.3 Future developments

7.3.1 On-line HSVR

When a new data point is added to a data set, the HSVR model should be reconfigured from scratch. To avoid this and to allow a continuous adaptation to new data points of the already configured model, on-line configuration is required. This modality has been explored for SVM models used for classification [73][223]. More recently on-line configuration of SVR models has also been developed [156][258]: the weight associated to a new point is computed in a finite number of discrete steps until it meets the KKT conditions, while ensuring that the existing data continue to satisfy the KKT conditions at each step. However, once the first HSVR layer has been modified, all the next layers have to be reconfigured as the update of the first layer implies changing the residual used to configure all the next layers.

7.3.2 Implementation on parallel Hardware

Parallel computing is the simultaneous use of multiple computational resources to solve a computational problem. To achieve this, the task should be partitioned into smaller independent subproblems, that can be solved concurrently. Such decomposition allows reducing the computational time by a factor equal to the number of the parallel processing units available.

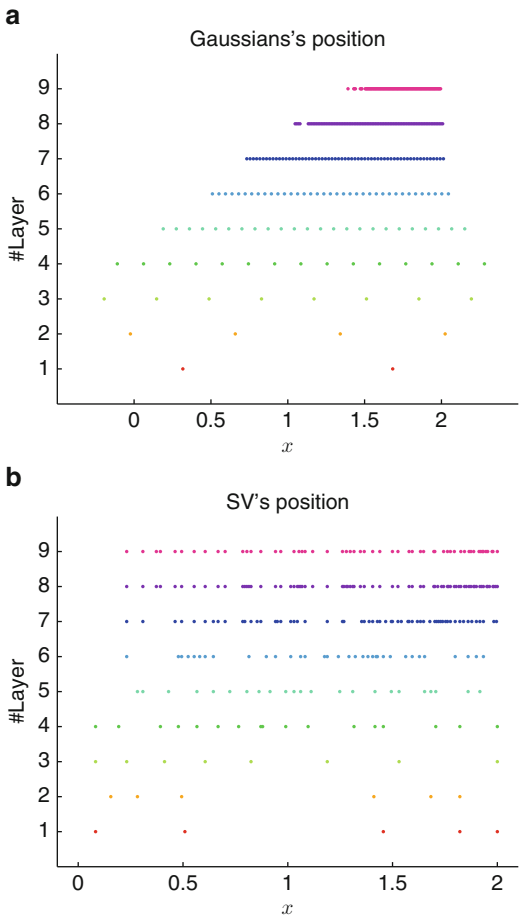

Fig. 7.2 Reconstruction of the function defined in (6.46): the position of the Gaussians is shown in panel (**a**) and the position of the SVs is shown in panel (**b**) for the different layers

Recently, the always increasing hunger of number crunching by the digital entertainment industry has pushed the producers to provide the PCs with high performance graphics cards. These were designed with a highly parallel architecture that was soon recognized useful in many domains besides the processing of the graphics pipeline. Such cards were named Graphical Processing Units (GPU, [155]) and API and a specific language for their programming have been developed first by Nvidia (the CUDA language) and then by other producers. GPU have are being heavily used in many areas, encompassing high-energy physics, image processing, multimedia, and life science. Nowadays, they are the most powerful computational hardware per dollar available on the market [179][180]. GPU's rapid increase in both programmability and capability has therefore brought the research community to use these systems to solve an increasing number of problems by transforming sequential algorithms into parallel ones.

Such architectures are a good match for both HRBF and HSVR models. Moreover, GPUs can be used also for rendering in real-time the surface computed by the models. The HRBF configuration procedure, both on-line and batch, is characterized by local operations, which can be performed in parallel over different units. The configuration procedure of the SVR consists of minimizing a convex problem defined on the coefficients space of the input points. This process is generally realized using a Sequential Minimal Optimization (SMO) that performs optimization by considering two coefficients at a time and by freezing the others [72]. Even though this formulation is not suited for parallel hardware, some variants of the SMO algorithm have been proposed for parallel hardware [56][58]. For instance, the solution presented in [56] has been realized on CUDA architecture and shows a speed-up of 10 to 70 times over libSVM. This would allow a very short configuration time.

Very recently GPUs have been introduced in smart phones: new Nvidia Tegra processor line integrates an ARM CPU, a GPU, a memory controller and the bus controllers, northbridge and southbridge, onto one card delivering high performance with low power consumption to make these architectures particularly suitable to mobile computing [10]. This enables the users not only to get 2D images and videos, but also to acquire 3D models directly on their portable systems. Moreover, since devices for reproducing 3D content are becoming widely available to the consumer market, compact and easy-to-use devices for capturing 3D contents are likely to be proposed soon. Hence, besides current 3D digital cameras which capture only a pair of displaced images, it can be envisioned that the miniaturization of components such as CCD sensors or pico-projectors will be exploited for implementing very small, point-and-click active optical devices for 3D scanning.

Glossary

α-shape The α-shape is a generalization of Delaunay triangulation [25]. The α-shape of a set of points P is obtained from its Delaunay triangulation by removing the elements (edges, triangles and tetrahedrons) that cannot be inscribed in a sphere of radius α.

Convex hull The *convex hull* of a set of points $P \subset R^D$ is the intersection of convex subsets of R^D that contain P. For a finite set of points in R^2, the convex hull is the convex polygon of minimum area that contains the points. The vertices of the convex hull are a subset of P.

Delaunay triangulation The Delaunay triangulation of a set of points P, is composed by triangles (tetrahedrons in three dimension) that have as vertices the points of P, such that there are not vertices lying in the spheres circumscribed to the triangles.

Genus A topologically invariant property of a surface defined as the largest number of non intersecting simple closed curves that can be drawn on the surface without separating it. Roughly speaking, it is the number of holes in a surface.

Manifold A manifold is a topological space which is locally Euclidean (i.e., around every point, there is a neighborhood which is topologically equivalent to the open unit ball in R^n).

Marching Cubes The *Marching Cubes* [154] is an algorithm for the approximation of isosurface in volumetric data. Each cube is characterized by the values of the vertices that represent the voxel. For a user defined threshold, if a cube has some vertices over threshold and some others under threshold, the isosurface of value equal to threshold pass through the cube. Determining the vertices of the cube that belong to the isosurface, it is possible to create triangular patches that divide the cube in regions internal and external to the isosurface. The representation of the surface is generated connecting the obtained triangular patches.

F. Bellocchio et al., *3D Surface Reconstruction: Multi-Scale Hierarchical Approaches*,
DOI 10.1007/978-1-4614-5632-2, © Springer Science+Business Media, LLC 2013

Octree An *octree* is a tree data structure where each node has eight children. Each node of the octree represents a cube in the three-dimensional space. Each child represents an eighth of the volume of the father node. This data structure is used for the search of points belonging to a region of the space, since it allows a logarithmic access time/cost to the data of the set.

Voronoi diagram The Voronoi diagram of a set of points $P \subset R^D$ is a partition in regions of the space determined by P. For each $p \in P$, the corresponding region is determined as the collection of the points in R^D that are closer to p than to any other point in P.

References

1. Blender Foundation's Peach Open Movie project. URL http://www.bigbuckbunny.org/index.php/about/
2. Blender.org. URL http://www.blender.org/
3. Checkmaster manual CMMs. URL http://www.helmel.com/Checkmaster.htm
4. Digital Michelangelo project. URL http://graphics.stanford.edu/projects/mich/
5. FARO Photon laser scanner. URL http://www.faro.com/contentv2.aspx?cid=1&content=misc&item=1210&ct=us
6. FARO UK — measuring arms. URL http://www.faro.com/quantum
7. Leica HDS6200 — Advanced, affordable ultra-high speed, phase scanner. URL http://hds.leica-geosystems.com/en/LeicaHDS6200_64228.htm
8. Leica ScanStation C10 — The all-in-one laser scanner for any application. URL http://hds.leica-geosystems.com/en/Leica-ScanStation-C10_79411.htm
9. Maptek - I-Site 8800 3D laser scanner. URL http://www.maptek.com/products/i-site/i-site_8800.html
10. NVIDIA Tegra APX applications processors. URL http://www.nvidia.com/object/product_tegra_apx_us.html
11. QtMorph: Software for morphing images. URL http://borghese.di.unimi.it/Research/Software/index.html
12. REWIRE project — Rehabilitative Wayout In Responsive home Environments. URL http://www.rewire-project.eu/
13. VIVID 9i 3D laser scanner. URL http://sensing.konicaminolta.us/products/vivid-9i-3d-laser-scanner
14. 3D real-time reconstruction using hrbf on-line networks (2009). URL http://www.dti.unimi.it/ferrari/hrbf_online/hrbf_online.wmv
15. Aha, D., Albert, D.K.M.: Instance-based learning algorithms. Machine Learning pp. 37–66 (1991)
16. Al-Temeemy, A.A.: Three dimensional LADAR imaging system using AR-4000LV laser rangefinder. In: Proceedings of the Microwaves, Radar and Remote Sensing Symposium, pp. 263–266 (2011)
17. Alexander, O., Rogers, M., Lambeth, W., Chiang, J.Y., Ma, W.C., Wang, C.C., Debevec, P.: The digital emily project: Achieving a photorealistic digital actor. IEEE Computer Graphics and Applications **30**(4), 20–31 (2010)
18. Alexandridis, A., Sarimveis, H., Bafas, G.: A new algorithm for online structure and parameter adaptation of RBF networks. Neural Networks **16**(7), 1003–1017 (2003)
19. Algorri, M.E., Schmitt, F.: Surface reconstruction from unstructured 3D data. In: Computer Graphics Forum, vol. 15, pp. 47–60 (1996)

DOI 10.1007/978-1-4614-5632-2

20. Aloimonos, Y.: Detection of surface orientation from texture I: the case of plane. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 584–593 (1986)
21. Alpern, B., Carter, L.: The hyperbox. In: Proceedings of the IEEE Conference on Visualization, pp. 133–139, 418 (1991)
22. Atkenson, G.C., Moore, A.W., Schaal, S.: Locally weighted learning. Artificial Intelligence Review **11**, 11–73 (1997)
23. Baader, A., Hirzinger, G.: A self-organizing algorithm for multisensory surface reconstruction. In: Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems, vol. 1, pp. 81–89 (1994)
24. Bahlmann, C., Haasdonk, B., Burkhardt, H.: On-line handwriting recognition with support vector machines — a kernel approach. In: Proceedings of the 8th International Workshop on Frontiers in Handwriting Recognition, pp. 49–54 (2002)
25. Bajaj, C.L., Bernardini, F., Xu, G.: Automatic reconstruction of surfaces and scalar fields from 3D scans. In: Proceedings of the ACM SIGGRAPH Conference on Computer graphics and interactive techniques, pp. 109–118 (1995)
26. Barhak, J., Fischer, A.: Parameterization and reconstruction from 3D scattered points based on neural network and PDE techniques. IEEE Transactions on Visualization and Computer Graphics **7**(1), 1–16 (2001)
27. Batlle, J., Mouaddib, E., Salvi, J.: Recent progress in coded structured light as a technique to solve the correspondence problem: a survey. Pattern Recognition **31**(7), 963–982 (1998)
28. Belhumeur, P.N., Kriegman, D.J., Yuille, A.L.: The bas-relief ambiguity. International Journal of Computer Vision pp. 33–44 (1999)
29. Bellocchio, F., Borghese, N.A., Ferrari, S., Piuri, V.: Kernel regression in HRBF networks for surface reconstruction. In: Proceedings of the 2008 IEEE International Workshop on Haptic Audio and Visual Environments and Games, pp. 160–165 (2008)
30. Bellocchio, F., Ferrari, S.: Depth Map and 3D Imaging Applications: Algorithms and Technologies, chap. 3D Scanner, State of the Art, pp. 451–470. IGI Global (2011)
31. Bellocchio, F., Ferrari, S., Piuri, V., Borghese, N.A.: A hierarchical approach for multi-scale support vector regression. IEEE Transactions on Neural Networks and Learning Systems **23**(9), 1448–1460 (2012)
32. Bellocchio, F., Ferrari, S., Piuri, V., Borghese, N.: Online training of hierarchical RBF. In: Proceedings of the 2007 IEEE-INNS International Joint Conference on Neural Networks, pp. 2159–2164 (2007)
33. Berg, A.B.: Locating global minima in optimisation problems by a random-cost approach. Nature **361**, 708–710 (1993)
34. Bernardini, F., Bajaj, C.L., Chen, J., Schikore, D.R.: Automatic reconstruction of 3D CAD models from digital scans. International Journal of Computational Geometry and Applications **9**(4-5), 327–370 (1999)
35. Besl, P., McKay, N.: A method for registration of 3-D shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence **17**(8) (1992)
36. Billings, S.A., Zheng, G.L.: Radial basis function network configuration using genetic algorithms. Neural Networks **8**(6), 877–890 (1995)
37. Bishop, C.M.: Neural networks for pattern recognition. Oxford University Press (1995)
38. Bittar, E., Tsingos, N., Gascuel, M.P.: Automatic reconstruction of unstructured 3D data: Combining medial axis and implicit surfaces. Computer Graphics Forum **14**(3), 457–468 (1995)
39. Blanz, V., Vetter, T.: Face recognition based on fitting a 3D morphable model. IEEE Transactions on Pattern Analysis and Machine Intelligence **25**(9), 1063– 1074 (2003)
40. Boissonat, J.D.: Geometric structures for three-dimensional shape representation. ACM Transactions on Graphics **3**(4), 266–286 (1984)
41. Borghese, N., Cerveri, P.: Calibrating a video camera pair with a rigid bar. Pattern Recognition **33**(1), 81–95 (2000)

42. Borghese, N., Ferrari, S., Piuri, V.: A methodology for surface reconstruction based on hierarchical models. In: Proceedings of the 2003 IEEE International Workshop on Haptic Virtual Environments and Their Applications, pp. 119–124 (2003)
43. Borghese, N., Ferrari, S., Piuri, V.: Real-time surface meshing through HRBF networks. In: Proceedings of 2003 IEEE-INNS-ENNS International Joint Conference of Neural Networks, vol. 2, pp. 1361–1366 (2003)
44. Borghese, N., Ferrigno, G.: An algorithm for 3-D automatic movement detection by means of standard TV cameras. IEEE Transactions on Biomedical Engineering **37**(12), 1221–1225 (1990)
45. Borghese, N.A., Colombo, F.M., Alzati, A.: Computing camera focal length by zooming a single point. Pattern Recognition **39**(8), 1522 – 1529 (2006)
46. Borghese, N.A., Ferrari, S.: Hierarchical RBF networks and local parameter estimate. Neurocomputing **19**(1–3), 259–283 (1998)
47. Borghese, N.A., Ferrari, S.: A portable modular system for automatic acquisition of 3D objects. IEEE Transactions on Instrumentation and Measurement **49**(5), 1128–1136 (2000)
48. Borghese, N.A., Ferrigno, G., Baroni, G., Ferrari, S., Savaré, R., Pedotti, A.: AUTOSCAN: a flexible and portable 3D scanner. IEEE Computer Graphics and Applications **18**(3), 38–41 (1998)
49. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and regression trees. Wadsworth (1984)
50. Brooks, M.: Two results concerning ambiguity in shape from shading. In: Proceedings of the AAAI Third National Conference on Artificial Intelligence, pp. 36–39 (1983)
51. Broomhead, D.S., Lowe, D.: Multivariable functional interpolation and adaptive networks. Complex Systems **2**, 321–355 (1988)
52. Broyden, C.G.: The convergence of a class of double-rank minimization algorithms. Journal of the Institute of Mathematics and Its Applications **6**, 76–90 (1970)
53. Burges, B.S.C., Vapnik, V.: Extracting support data for a given task. In: Proceedings of the First International Conference on Knowledge Discovery and Data Mining. AAAI Press (1995)
54. Burges, B.S.C., Vapnik, V.: Incorporating invariances in support vector learning machines. In: Proceedings of the International Conference on Artificial Neural Networks, vol. 1112, pp. 47–52 (1996)
55. Cai, Z.: Weighted Nadaraya-Watson regression estimation. In: Statistics and Probability Letters, pp. 307–318 (2001)
56. Carpenter, A.: cuSVM: a CUDA implementation of support vector classification and regression (2009). URL http://patternsonascreen.net/cuSVM.html
57. Catalan, R.B., Perez, E.I., Perez, B.Z.: Evaluation of 3D scanners to develop virtual reality applications. In: Proceedings of the Fourth Congress of Electronics, Robotics and Automotive Mechanics, pp. 551–556 (2007)
58. Catanzaro, B., Sundaram, N., Keutzer, K.: Fast support vector machine training and classification on graphics processors. In: Proceedings of the 25th International Conference on Machine learning, pp. 104–111. ACM (2008)
59. Catapano, I., Crocco, L., Krellmann, Y., Triltzsch, G., Soldovieri, F.: A tomographic approach for helicopter-borne ground penetrating radar imaging. Geoscience and Remote Sensing Letters, IEEE **9**(3), 378–382 (2012)
60. Catmull, E., Clark, J.: Recursively generated B-spline surfaces on arbitrary topological meshes. Computer-Aided Design **10**(6), 350–355 (1978)
61. Cerveri, P., Ferrari, S., Borghese, N.A.: Calibration of TV cameras through RBF networks. In: Proceedings of SPIE Conference on Applications of Soft Computing, pp. 312–318 (1997)
62. Cetin, B., Barhen, J., Burdick, J.: Terminal repeller unconstrained subenergy tunnelling (trust) for fast global optimization. Journal of Optimization Theory and Application **77**(1), 97–126 (1993)
63. Chan, Y.T.: Wavelet Basics. Kluwer Academic Publishers (1995)
64. Chen, S., Hong, X., Harris, C.J.: Sparse kernel density construction using orthogonal forward regression with leave-one-out test score and local regularization. IEEE Transactions on System, Man, and Cybernetics, Part B: Cybernetics **34**(4), 1708–1717 (2004)

65. Clark, P., Niblett, T.: The CN2 induction algorithm. Machine Learning **3**, 261–283 (1989)
66. Colombo, C., Bimbo, A.D., Del, A., Pernici, F.: Metric 3D reconstruction and texture acquisition of surfaces of revolution from a single uncalibrated view. IEEE Transactions on Pattern Analysis and Machine Intelligence **27**, 99–114 (2005)
67. Colombo, C., Bimbo, A.D., Persini, F.: Metric 3D reconstruction and texture acquisition of surfaces of revolution from a single uncalibrated view. IEEE Transactions on Pattern Analysis and Machine Intelligence pp. 99–114 (2005)
68. Cordier, F., Seo, H., Magnenat-Thalmann, N.: Made-to-measure technologies for an online clothing store. IEEE Computer Graphics and Applications **23**(1), 38–48 (2003)
69. Costin, M., Ignat, A., Baltag, O., Bejinariu, S., Stefanescu, C., Rotaru, F., Costandache, D.: 3D breast shape reconstruction for a non-invasive early cancer diagnosis system. In: Proceedings of the 2nd International Workshop on Soft Computing Applications, pp. 45–50 (2007)
70. Cover, T., Hart, P.E.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory pp. 21–27 (1967)
71. Craven, P., Wahba, G.: Smoothing noisy data with spline functions. Estimating the correct degree of smoothing by the method of generalized cross-validation. Numerische Mathematik **31**(4), 377–403 (1978/79)
72. Cristianini, N., Shawe-Taylor, J.: An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press (2000)
73. Csato, L., Opper, M.: Sparse representation for gaussian process models. Advances in neural information processing systems **13**, 444–450 (2001)
74. Dai, L.: The 3D digital technology of fashion design. In: Proceedings of the International Symposium on Computer Science and Society, pp. 178–180 (2011)
75. Dasarathy, B.: Nearest Neighbor (NN) norms: NN pattern classification techniques. IEEE Press (1991)
76. Daubechies, I.: Orthonormal bases of compactly supported wavelets. Communications on Pure and Applied Mathematics **41**, 909–996 (1988)
77. Daubechies, I.: Ten lectures on Wavelets. SIAM (1992)
78. Debevec, P.E., Taylor, C.J., Malik, J.: Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In: Proceedings of the ACM SIGGRAPH Conference on Computer graphics and interactive techniques, pp. 11–20 (1996)
79. Dick, C., Burgkart, R., Westermann, R.: Distance visualization for interactive 3D implant planning. IEEE Transactions on Visualization and Computer Graphics **17**(12), 2173–2182 (2011)
80. Donoho, D.L., Johnstone, I.M.: Ideal spatial adaptation by wavelet shrinkage. Biometrika (1993)
81. Dorai, C., Wang, G., Jain, A.K., Mercer, C.: Registration and integration of multiple object views for 3D model construction. IEEE Transactions on Pattern Analysis and Machine Intelligence **20**(1), 83–89 (1998)
82. Dorn, W.S.: Duality in quadratic programming. Quarterly of Applied Mathematics **18**(2), 155–162 (1960)
83. Dreyfus, G.: Neural networks: methodology and applications. Springer (2005)
84. Drucker, H., Burges, C., Smola, L.K.A., Vapnik, V.: Support vector regression machines pp. 151–161 (1997)
85. Edelsbrunner, H., Mucke, E.P.: Three-dimensional alpha shapes. ACM Transactions on Graphics **13**(1), 43–72 (1994)
86. English, C., Zhu, S., Smith, C., Ruel, S., Christie, I.: Tridar: A hybrid sensor for exploiting the complimentary nature of triangulation and LIDAR technologies. In: B. Battrick (ed.) Proceedings of the 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space (2005)
87. Evans, F., Skiena, S., Varshney, A.: Optimizing triangle strips for fast rendering. In: Proceedings of the IEEE Visualization Conference, vol. 2, pp. 319–326 (1996)
88. Feldkamp, L.A., Davis, L.C., Kress, J.: Practical conebeam algorithm. Journal of the Optical Society of America A: Optics, Image Science, and Vision **1**, 612–619 (1984)

89. Ferrari, S., Bellocchio, F., Borghese, N., Piuri, V.: Refining hierarchical radial basis function networks. In: Proceedings of the 2007 IEEE International Workshop on Haptic Audio and Visual Environments and Games, pp. 166–170 (2007)
90. Ferrari, S., Bellocchio, F., Piuri, V., Borghese, N.: Multi-scale support vector regression. In: Proceedings of the 2010 IEEE-INNS International Joint Conference on Neural Networks, pp. 2159–2164 (2010)
91. Ferrari, S., Bellocchio, F., Piuri, V., Borghese, N.A.: A hierarchical RBF online learning algorithm for real-time 3-D scanner. IEEE Transactions on Neural Networks **21**(2), 275–285 (2010)
92. Ferrari, S., Borghese, N.A., Piuri, V.: Multi-resolution models for data processing: an experimental sensitivity analysis. In: Proceedings of the 2000 IEEE Instrumentation and Measurement Technology Conference, vol. 2, pp. 1056–1060 (2000)
93. Ferrari, S., Borghese, N.A., Piuri, V.: Multiscale models for data processing: an experimental sensitivity analysis. IEEE Transactions on Instrumentation and Measurement **50**(4), 995–1002 (2001)
94. Ferrari, S., Ferrigno, G., Piuri, V., Borghese, N.A.: Reducing and filtering point clouds with enhanced vector quantization. IEEE Transactions Neural Networks **18**(1), 161–177 (2007)
95. Ferrari, S., Frosio, I., Piuri, V., Borghese, N.: The accuracy of the HRBF networks. In: Proceedings of 2004 IEEE Instrumentation and Measurement Technology Conference, pp. 482–486 (2004)
96. Ferrari, S., Frosio, I., Piuri, V., Borghese, N.A.: Automatic multiscale meshing through HRBF networks. IEEE Transactions on Instrumentation and Measurement **54**(4), 1463–1470 (2005)
97. Ferrari, S., Maggioni, M., Borghese, N.A.: Multi-scale approximation with hierarchical radial basis functions networks. IEEE Transactions on Neural Networks **15**(1), 178–188 (2004)
98. Ford, C., Etter, D.: Wavelet basis reconstruction of nonuniformly sampled data. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing **45**(8), 1165–1168 (1998)
99. Forsey, D.R., Bartels, R.H.: Hierarchical B-spline refinement. ACM SIGGRAPH Computer Graphaphics Newsletter **22**(4), 205–212 (1988)
100. Forsey, D.R., Wong, D.: Multiresolution surface reconstruction for hierarchical B-splines. Graphics Interface pp. 57–64 (1998)
101. Fougerolle, Y., Gribok, A., Foufou, S., Truchetet, F., Abidi, M.: Boolean operations with implicit and parametric representation of primitives using r-functions. IEEE Transactions on Visualization and Computer Graphics **11**(5), 529–539 (2005)
102. Friedman, J.: Multivariate adaptive regression splines. Annals of Statistics **19**(1), 1–141 (1991)
103. Friedman, J.H., Stuetzle, W.: Projection pursuit regression. Journal of the American Statistical Association **76**, 817–823 (1981)
104. Fritzke, B.: Growing cell structures — A self-organizing network for unsupervised and supervised learning. Neural Networks **7**(9), 1441–1460 (1994)
105. Fritzke, B.: Growing grid — A self-organizing network with constant neighborhood range and adaptation strength. Neural Processing Letters **2**(5), 9–13 (1995)
106. Frosio, I., Alzati, A., Bertolini, M., Turrini, C., Borghese, N.A.: Linear pose estimate from corresponding conics. Pattern Recognition **45**(12), 4169–4181 (2012)
107. Frosio, I., Borghese, N.A.: Optimized algebraic local tomography. In: Proceedings of the GIRPR Conference (2010)
108. Frosio, I., Mainetti, R., Palazzo, S., Borghese, N.A.: Robust kinect for rehabilitation. In: Proceedings of the GIRPR Conference (2012)
109. Fung, G.M., Mangasarian, O.L., Smola, A.J.: Minimal kernel classiers. Journal of Machine Learning Research **3**, 2303–321 (2002)
110. Gambino, M.C., Fontana, R., Gianfrate, G., Greco, M., Marras, L., Materazzi, M., Pampaloni, E., Pezzati, L.: A 3D scanning device for architectural relieves based on Time-Of-Flight technology. Springer (2005)

111. Geman, S., Bienenstock, E., Doursat, R.: Neural networks and the bias/variance dilemma. Neural Computation **4**(1), 1–58 (1992)
112. Girosi, F., Jones, M., Poggio, T.: Regularization theory and neural networks architectures. Neural Computation **7**(2), 219–269 (1995)
113. Gorse, D., Sheperd, A., Taylor, J.: Avoiding local minima by a classical range expansion algorithm. In: Proceedings of the International Conference on Artificial Neural Networks (1994)
114. Hall, P.: On projection pursuit regression. Annals of Statistics **17**(2), 573–588 (1989)
115. Harman, P.: Home based 3D entertainment-an overview. In: Proceedings of the International Conference on Image Processing, pp. 1–4 vol.1 (2000)
116. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision, second edn. Cambridge University Press (2004)
117. Hasenjäger, M., Ritter, H.: New learning paradigms in soft computing. In: Active learning in neural networks, pp. 137–169. Physica-Verlag GmbH (2002)
118. Heckbert, P.S., Garland, M.: Optimal triangulation and quadric-based surface simplification. Computational Geometry **14**, 49–65 (1998)
119. Heikkinen, V., Kassamakov, I., Haggstrom, E., Lehto, S., Kiljunen, J., Reinikainen, T., Aaltonen, J.: Scanning white light interferometry, x2014; a new 3D forensics tool. In: Proceedings of the 2011 IEEE International Conference on Technologies for Homeland Security, pp. 332–337 (2011)
120. Hernandez, C., Vogiatzis, G., Cipolla, R.: Multiview photometric stereo. IEEE Transactions on Pattern Analysis and Machine Intelligence **30**(3), 548–554 (2008)
121. Hertz, J., Krogh, A., Palmer, R.G.: An introduction to the theory of neural computation. Addison Wesley (1991)
122. Higo, T., Matsushita, Y., Joshi, N., Ikeuchi, K.: A hand-held photometric stereo camera for 3D modeling. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1234–1241 (2009)
123. Hoppe, H.: Surface reconstruction from unorganized points. PhD Thesis, Dept. of Computer Science and Engineering, University of Washington (1994)
124. Hoppe, H., DeRose, T., Duchamp, T., Mc-Donald, J., Stuetzle, W.: Surface reconstruction from unorganized points. In: Proceedings of the ACM SIGGRAPH Conference on Computer graphics and interactive techniques), vol. 26, pp. 71–78 (1992)
125. Horn, B.: Obtaining shape from shading information. The Psychology of Computer Vision (1975)
126. Hornik, K.: Approximation capabilities of multilayer feedforward networks. Neural Networks **4**(2), 251–257 (1991)
127. Huang, P., Zhang, S.: Fast three-step phase shifting algorithm. Applied Optics **45**(21), 5086–5091 (2006)
128. Hur, N., Lee, H., Lee, G.S., Lee, S.J., Gotchev, A., Park, S.I.: 3DTV broadcasting and distribution systems. IEEE Transactions on Broadcasting **57**(2), 395–407 (2011)
129. Iddan, G.J., Yahav, G.: 3D imaging in the studio (and elsewhere). In: Proceedings of the SPIE Conference on Three-Dimensional Image Capture and Applications, pp. 48–55. San Jose, CA (2001)
130. Idesawa, M., Yatagai, T., Soma, T.: Scanning moiré method and automatic measurement of 3-D shapes. Applied Optics **16**(8), 2152–2162 (1977)
131. Isler, V., Wilson, B., Bajcsy, R.: Building a 3D virtual museum of native american baskets. In: Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission, pp. 954–961 (2006)
132. Joachims, T.: Making large-scale SVM learning practical. In: B. Schölkopf, C. Burges, A. Smola (eds.) Advances in Kernel Methods — Support Vector Learning, chap. 11, pp. 169–184. MIT Press, Cambridge, MA (1999)
133. Katznelson, Y.: An introduction to harmonic analysis. Dover (1976)
134. Kaufman, L., Rousseeuw, P.J.: Finding groups in data — an introduction to cluster analysis. Wiley (1990)

135. Keerthi, S.S., Chapelle, O., Coste, D.D.: Building support vector machines with reduced classifier complexity. Journal of Machine Learning Research **7**, 1493–1515 (2006)
136. Kibler, D., Aha, D.W., Albert, M.K.: Instance-based prediction of real-valued attributes. Computational Intelligence pp. 51–57 (1989)
137. Kirkpatrick, S., Gelatt, C., Vecchi, M.: Optimisation by simulated annealing. Science **220**, 671–680 (1983)
138. Klinke, S., Grassmann, J.: Projection pursuit regression and neural network. Tech. rep., Humboldt Universitaet Berlin (1998)
139. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, vol. 2, pp. 1137–1143 (1995)
140. Kohonen, T.: Self-Organizing Maps. Springer (1995)
141. Kuhn, H.W., Tucker, A.W.: Nonlinear programming. In: Proceedings of the 2nd Berkeley Symposium, pp. 481–492. Berkeley: University of California Press (1951)
142. Lanckriet, G., Cristianini, N., Bartlett, P., Ghaoui, L.E., Jordan, M.I.: Learning the kernel matrix with semi-definite programming. Journal of Machine Learning Research **5**, 27–72 (2004)
143. Lee, A., Moreton, H., Hoppe, H.: Displaced subdivision surfaces. In: Proceedings of the ACM SIGGRAPH Conference on Computer graphics and interactive techniques, pp. 85–94 (2000)
144. Lee, A.W.F., Dobkin, D., Sweldens, W., Schröder, P.: Multiresolution mesh morphing. In: Proceedings of the 1999 ACM SIGGRAPH Conference on Computer graphics and interactive techniques, pp. 343–350 (1999)
145. Lee, A.W.F., Sweldens, W., Schroder, P., Cowsar, L., Dobkin, D.: MAPS: Multiresolution adaptive parameterization of surfaces. In: Proceedings of the ACM SIGGRAPH Conference on Computer graphics and interactive techniques, pp. 95–104 (1998)
146. Lee, J.-D., Lan, T.-Y., Liu, L.-C., Lee, S.-T., Wu, C.-T., Yang, B.: A remote virtual-surgery training and teaching system. In: Proceedings of the IEEE 3DTV Conference, pp. 1–4 (2007)
147. Lee, S., Wolberg, G., Shin, S.Y.: Scattered data interpolation with multilevel B-splines. IEEE Transactions on Visualization and Computer Graphics **3**(3), 228–244 (1997)
148. Levenberg, K.: A method for the solution of certain non-linear problems in least squares. The Quarterly of Applied Mathematics **2**, 164–168 (1944)
149. Levin, A., Fergus, R., Durand, F., Freeman, W.T.: Image and depth from a conventional camera with a coded aperture. ACM Transactions on Graphics **26**(3), 701–709 (2007)
150. Levoy, M., Rusinkiewicz, S., Ginzton, M., Ginsberg, J., Pulli, K., Koller, D., Anderson, S., Shade, J., Curless, B., Pereira, L., David, J., Fulk, D.: The Digital Michelangelo project: 3D scanning of large statues. In: Proceedings of the 2000 ACM SIGGRAPH Conference on Computer graphics and interactive techniques (2000)
151. Levy, R., Dawson, P.: Reconstructing a Thule whalebone house using 3D imaging. IEEE Multimedia **13**, 78–83 (2006)
152. Liang, X.: An effective method of pruning support vector machine classifiers. IEEE Transactions on Neural Networks **21**(1), 26–38 (2010)
153. Lin, Y., Song, M., Quynh, D.T.P., He, Y., Chen, C.: Sparse coding for flexible, robust 3D facial-expression synthesis. IEEE Computer Graphics and Applications **32**(2), 76–88 (2012)
154. Lorensen, W.E., Cline, H.E.: Marching cubes: a high resolution 3D surface construction algorithm. ACM SIGGRAPH Computer Graphaphics Newsletter **21**(4), 163–169 (1987)
155. Luebke, D., Humphreys, G.: How GPUs work. IEEE Computer (2007)
156. Ma, J., Theiler, J., Perkins, S.: Accurate on-line support vector regression. Neural Computation **15**, 2683–2703 (2003)
157. Mallat, S.: A theory for multiscale signal decomposition: The wavelet representation. IEEE Transactions on Pattern and Machine Intelligence **11**(7), 674–693 (1989)
158. Marquardt, D.: An algorithm for least-squares estimation of nonlinear parameters. SIAM Journal on Applied Mathematics **11**, 431–441 (1963)
159. Martinetz, T., Berkovich, S., Schulten, K.: Neural-gas network for vector quantization and its application to time-series prediction. IEEE Transactions on Neural Networks **4**(4), 558–568 (1993)

160. Martínez, J.I.M.: Best approximation of gaussian neural networks with nodes uniformly spaced. IEEE Transactions on Neural Networks **19**(2), 284–298 (2008)
161. Mckinley, T.J., McWaters, M., Jain, V.K.: 3D reconstruction from a stereo pair without the knowledge of intrinsic or extrinsic parameters. In: Proceedings of the Second International Workshop on Digital and Computational Video, pp. 148–155 (2001)
162. de Medeiros Brito, A., Doria Neto, A., Dantas de Melo, J., Garcia Goncalves, L.: An adaptive learning approach for 3-D surface reconstruction from point clouds. IEEE Transactions on Neural Networks **19**(6), 1130–1140 (2008)
163. Medioni, G., Choi, J., Kuo, C.H., Choudhury, A., Zhang, L., Fidaleo, D.: Non-cooperative persons identification at a distance with 3D face modeling. In: Proceedings of the First IEEE International Conference on Biometrics: Theory, Applications, and Systems, pp. 1–6 (2007)
164. Mencl, R., Muller, H.: Interpolation and approximation of surfaces from three-dimensional scattered data points. In: Proceedings of the Eurographics 98 Conference, pp. 51–67 (1998)
165. Mencl, R., Müller, H.: Interpolation and approximation of surfaces from three-dimensional scattered data points. In: Proceedings of the Dagstuhl Conference on Scientific Visualization, pp. 223–232 (1999)
166. Meng, Q., Lee, M.: Error-driven active learning in growing radial basis function networks for early robot learning. Neurocomputing **71**(7-9), 1449–1461 (2008)
167. Michalsky, R., Mozetic, I., Hong, J., Lavrac, N.: The multi-purpose incremental learning system, AQ15 and its testing application to three medical domains. In: Proceedings of the AAAI Sixth National Conference on Artificial Intelligence, pp. 1041–1045 (1986)
168. Miller, J.V., Breen, D.E., Lorensen, W.E., O'Bara, R.M., Wozny, M.J.: Geometrically deformed models: a method for extracting closed geometric models form volume data. ACM SIGGRAPH Computer Graphaphics Newsletter **25**(4), 217–226 (1991)
169. Moody, J., Darken, C.J.: Fast learning in networks of locally-tuned processing units. Neural Computation **1**(2), 281–294 (1989)
170. Morcin, F., Garcia, N.: Hierarchical coding of 3D models with subdivision surfaces. In: Proceedings of the IEEE International Conference on Image Processing, vol. 2, pp. 911–914 (2000)
171. Müller, K.R., Smola, A., Rätsch, G., Schölkopf, B., Kohlmorgen, J., Vapnik, V.: Predicting time series with support vector machines. In: Proceedings of the International Conference on Artificial Neural Networks, vol. 1327, pp. 999–1004 (1997)
172. Narendra, K.S., Parthasarathy, K.: Gradient methods for the optimization of dynamical systems containing neural networks. IEEE Transactions on Neural Networks **2**(2), 252–262 (1992)
173. Narendra, K.S., Thathachar, M.A.L.: Learning automata — an introduction. Prentice Hall (1989)
174. Newcombe, R.A., Davison, A.J., Izadi, S., Kohli, P., Hilliges, O., Shotton, J., Molyneaux, D., Hodges, S., Kim, D., Fitzgibbon, A.: KinectFusion: Real-time dense surface mapping and tracking. In: Proceedings of the 2011 IEEE International Symposium on Mixed and Augmented Reality, pp. 127–136 (2011)
175. Nguyen, D., Ho, T.: An efficient method for simplifying support vector machines. In: Proceedings of the 22nd international Conference on Machine learning, pp. 617–624 (2005)
176. Nikolov, S., Bull, D., Canagarajah, C., Halliwell, M., Wells, P.: Image fusion using a 3-D wavelet transform. In: Proceedings of the Seventh International Conference on Image Processing And Its Applications, pp. 235–239 (1999)
177. Olhoeft, G.: Applications and frustrations in using ground penetrating radar. Aerospace and Electronic Systems Magazine, IEEE **17**(2), 12–20 (2002)
178. Orr, M.J.L.: Regularization in the selection of radial basis function centers. Neural Computation **7**(3), 606–623 (1993)
179. Owens, J.D., Houston, M., Luebke, D., Green, S., Stone, J.E., Phillips, J.C.: GPU computing. Proceedings of the IEEE **96**, 879–899 (2008)
180. Owens, J.D., Luebke, D., Harris, N.G.M., Kruger, J., Lefohn, A.E., Purcell, T.J.: A survey of general-purpose computation on graphics hardware. In: Proceedings of the Eurographics Conference, pp. 21–25 (2005)

181. Pan, Q., Reitmayr, G., Rosten, E., Drummond, T.: Rapid 3D modelling from live video. In: Proceedings of the 33rd IEEE MIPRO International Convention, pp. 252–257 (2010)
182. Park, K., Yun, I.D., Lee, S.U.: Automatic 3-D model synthesis from measured range data. IEEE Transactions on Circuits and Systems for Video Technology **10**(2), 293–301 (2000)
183. Pastor, L., Rodriguez, A.: Surface approximation of 3D objects from irregularly sampled clouds of 3D points using spherical wavelets. In: Proceedings of the International Conference on Image Analysis and Processing, pp. 70–75 (1999)
184. Perez-Gutierrez, B., Martinez, D., Rojas, O.: Endoscopic endonasal haptic surgery simulator prototype: A rigid endoscope model. In: Proceedings of the 2010 IEEE Virtual Reality Conference, pp. 297–298 (2010)
185. Perona, P., Malik, J.: Scale-space and edge detection using anisotropic diffusion. IEEE Transactions on Pattern Analysis and Machine Intelligence **12**(7), 629–639 (1990)
186. Piegel, L., Tiller, W.: The NURBS book. Springer (1997)
187. Pirovano, M., Mainetti, R., Baud-Bovy, G., Lanzi, P.L., Borghese, N.A.: Self-adaptive games for rehabilitation at home. In: Proceedings of IEEE Conference on Computational Intelligence and Games. Granada (Spain) (2012)
188. Platt, J.: A resource-allocating network for function interpolation. Neural Computation **3**, 213–225 (1991)
189. Poggio, T.: A theory of how the brain might work. Tech. Rep. AIM-1253, Massachusetts Institute of Technology (1990)
190. Poggio, T., Girosi, F.: Network for approximation and learning. Proceedings of the IEEE **78**, 1481–1497 (1990)
191. Potmesil, M.: Generating octree models of 3D objects from their silhouettes in a sequence of images. Computer Vision, Graphics, and Image Processing **40**(1), 1–29 (1987)
192. Praun, E., Hoppe, H., Finkelstein, A.: Robust mesh watermarking. In: Proceedings of the ACM SIGGRAPH Conference on Computer graphics and interactive techniques, pp. 49–56 (1999)
193. Pulli, K.: Multiview registration for large data sets. In: Proceedings of the Second International Conference on 3-D Digital Imaging and Modeling, pp. 160–168 (1999)
194. Qi, Y., Cai, S., Yang, S.: 3D modeling, codec and protection in digital museum. In: Proceedings of the Second Workshop on Digital Media and its Application in Museum Heritages, pp. 231–236 (2007)
195. Qiu, S., Lane, T.: Multiple kernel learning for support vector regression. Tech. rep., Computer Science Department, The University of New Mexico, Albuquerque, NM, USA (2005)
196. Quinlan, R.J.: Learning with continuous classes. In: Proceedings of the 5th Australian Joint Conference on Artificial Intelligence, pp. 343–348. World Scientific, Singapore (1992)
197. Reddy, C.K., Park, J.H.: Multi-resolution boosting for classification and regression problems. In: Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, pp. 196–207. Springer-Verlag, Berlin, Heidelberg (2009)
198. Rejtö, L., Walter, G.: Remarks on projection pursuit regression and density estimation. Stochastic Analysis and Applications **10**, 213–222 (1992)
199. Rigotti, C., Borghese, N.A., Ferrari, S., Baroni, G., Ferrigno, G.: Portable and accurate 3D scanner for breast implants design and reconstructive plastic surgery. In: Proceedings of the SPIE International Symposium on Medical Imaging, pp. 1558–1567 (1998)
200. Rodriguez-Larena, J., Canal, F., Campo, F.: An optical 3D digitizer for industrial quality control applications. In: Proceedings of the 1999 IEEE International Conference on Emerging Technologies and Factory Automation, pp. 1155–1160 (1999)
201. Roosen, C., Hastie, T.: Automatic smoothing spline projection pursuit. Journal of Computational and Graphical Statistics **3**, 235–248 (1994)
202. Rossignac, J.: Edgebreaker: Connectivity compression for triangle meshes. GVU Technical Report GIT-GVU-98-35 (1998)
203. Roth, G., Wibowoo, E.: An efficient volumetric method for building closed triangular meshes from 3-D image and point data. In: Graphics Interface, pp. 173–180 (1997)

204. Rozza, A., Lombardi, G., Rosa, M., Casiraghi, E., Campadelli, P.: IDEA: Intrinsic dimension estimation algorithm. In: Proceedings of the International Conference on Image Analysis and Processing, pp. 433–442 (2011)
205. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. Parallel Distributed Processing: Explorations in the Microstructure of Cognition **1**, 318–362 (1986)
206. Rusinkiewicz, S., Hall-Holt, O., Levoy, M.: Real-time 3D model acquisition. In: Proceedings of the 2002 ACM SIGGRAPH Conference on Computer graphics and interactive techniques, pp. 438–446. ACM Press (2002)
207. Sanner, R.M., Slotine, J.E.: Gaussian networks for direct adaptive control. IEEE Transactions on Neural Networks **3**(6), 837–863 (1992)
208. Sansoni, G., Docchio, F.: 3-D optical measurements in the field of cultural heritage: the case of the Vittoria Alata of Brescia. IEEE Transactions on Instrumentation and Measurement **54**(1), 359–368 (2005)
209. Schnars, U., Jptner, W.P.O.: Digital recording and numerical reconstruction of holograms. Measurement Science and Technology **13**(9), R85 (2002)
210. Schreiber, T., Brunnett, G.: Approximating 3D objects from measured points. In: Proceedings of 30th ISATA Conference (1997)
211. Schroder, P., Sweldens, W.: Spherical wavelets: Efficiently representing functions on the sphere. In: Proceedings of the ACM SIGGRAPH Conference on Computer graphics and interactive techniques, pp. 161–172 (1995)
212. Seyama, J., Nagayama, R.S.: The uncanny valley: Effect of realism on the impression of artificial human faces. Presence: Teleoperators and Virtual Environments **16**(4), 337–351 (2007)
213. Shapiai, M.I., Ibrahim, Z., Khalid, M., Jau, L.W., Pavlovich, V.: A non-linear function approximation from small samples based on Nadaraya-Watson kernel regression. In: Proceedings of the Second International Conference on Computational Intelligence, Communication Systems and Networks, pp. 28–32 (2010)
214. Smola, A., Murata, N., Schölkopf, B., Müller, K.R.: Asymptotically optimal choice of ε-loss for support vector machines. In: Proceedings of the 8th International Conference on Artificial Neural Networks, pp. 105–110. Springer (1998)
215. Smola, A., Schölkopf, B.: A tutorial on support vector regression. Statistics and Computing **14**, 199–222 (2004)
216. Specht, D.F.: Probabilistic neural networks. Neural Networks **3**, 109–118 (1990)
217. Stitson, M., Gammerman, A., Vapnik, V., Vovk, V., Watkins, C., Weston, J.: Support vector regression with ANOVA decomposition kernels. Advances in Kernel MethodsSupport Vector Learning pp. 285–292 (1999)
218. Stone, E., Skubic, M.: Evaluation of an inexpensive depth camera for passive in-home fall risk assessment. In: Proceedings of the 5th International Conference on Pervasive Computing Technologies for Healthcare, pp. 71–77 (2011)
219. Su, J., Dodd, T.J.: Online functional prediction for spatio-temporal systems using time-varying radial basis function networks. In: Proceedings of the 2nd International Asia Conference on Informatics in Control, Automation and Robotics, vol. 2, pp. 147–150 (2010)
220. Sweldens, W.: The lifting scheme: A new philosophy in biorthogonal wavelet constructions. In: Proceedings of the SPIE Conference on Wavelet applications in signal and image processing, pp. 68–79 (1995)
221. Sweldens, W.: The lifting scheme: A custom-design construction of biorthogonal wavelets. Applied and Computational Harmonic Analysis **3**(2), 186–200 (1996)
222. Sweldens, W.: The lifting scheme: A construction of second generation wavelets. SIAM Journal on Mathematical Analysis **29**(2), 511–546 (1997)
223. Syed, N.A., Liu, H., Sung, K.K.: Incremental learning with support vector machines. In: Proceedings of the Workshop on Support Vector Machines at the International Joint Conference on Artificial Intelligence (1999)
224. Tang, Y., Guo, W., Gao, J.: Efficient model selection for support vector machine with Gaussian kernel function. In: proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, pp. 40–45 (2009)

225. Taylan, P., Weber, G.W.: Multivariate adaptive regression spline and continuous optimization for modern applications in science, economy and techology. Tech. rep., Humboldt Universitaet Berlin (2004)
226. Teichmann, M., Capps, M.: Surface reconstruction with anisotropic density-scaled alpha shapes. In: Proceedings of the IEEE Visualization Conference, pp. 67–72 (1998)
227. Terzopoulos, D., Metaxas, D.: Dynamic 3D models with local and global deformations: deformable superquadrics. IEEE Transactions on Pattern Analysis and Machine Intelligence **13**(7), 703–714 (1991)
228. Tirelli, P., Momi, E.D., Borghese, N.A., Ferrigno, G.: An intelligent atlas-based planning system for keyhole neurosurgery. International Journal of Computer Assisted Radiology and Surgery **4**(1), 85–91 (2009)
229. Tomasi, C., Kanade, T.: Detection and tracking of point features. Tech. rep., Carnegie Mellon University (1991)
230. Tsai, R.: A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. IEEE Journal of Robotics and Automation **3**(4), 323–344 (1987)
231. Tsang, I., Kwok, J., Cheung, P.M.: Core vector machines: Fast SVM training on very large data sets. Journal of Machine Learning Research **6**, 363–392 (2005)
232. Turk, G., Levoy, M.: Zippered polygon meshes from range images. In: Proceedings of the ACM SIGGRAPH Conference on Computer graphics and interactive techniques, pp. 311–318 (1994)
233. Ullrich, S., Kuhlen, T.: Haptic palpation for medical simulation in virtual environments. IEEE Transactions on Visualization and Computer Graphics **18**(4), 617–625 (2012)
234. Uysal, I.L., Altay, H., Venir, G.: An overview of regression techniques for knowledge discovery. Knowledge Engineering Review **14**, 319–340 (1999)
235. Vaillant, R., Faugeras, O.: Using extremal boundaries for 3D object modelling. IEEE Transactions on Pattern Analysis and Machine Intelligence **2**(14), 157–173 (1992)
236. Vapnik, V.: Statistical learning theory. Wiley-Interscience (1989)
237. Vapnik, V., Chervonenkis, A.: Theory of pattern recognition. Nauka, Moscow (1974)
238. Vapnik, V., Lerner, A.: Pattern recognition using generalized portrait method. Automation and Remote Control **24**, 774–780 (1963)
239. Visintini, D., Spangher, A., Fico, B.: The VRML model of Victoria square in Gorizia (Italy) from laser scanning and photogrammetric 3D surveys. In: Proceedings of the Web3D 2007 International Symposium, pp. 165–169 (2007)
240. Wang, D., Wu, X.B., Lin, D.M.: Two heuristic strategies for searching optimal hyper parameters of C-SVM. In: Proceedings of the Eighth International Conference on Machine Learning and Cybernetics, pp. 3690–3695 (2009)
241. Wang, L., Chu, C.h.: 3D building reconstruction from LiDAR data. In: Proceedings of the 2009 IEEE International Conference on Systems, Man and Cybernetics, pp. 3054–3059 (2009)
242. Wang, Z., Chen, S., Sun, T.: MultiK-MHKS: A novel multiple kernel learning algorithm. IEEE Transactions on Pattern Analysis and Machine Intelligence **30**(2), 348–353 (2008)
243. Watkins, C.J.C.H., Dayan, P.: Q-learning. Machine Learning Journal **8**(3) (1992)
244. Weiss, S., Indurkhya, N.: Optimized rule induction. IEEE Expert **8**(6), 61–69 (1993)
245. Weiss, S., Indurkhya, N.: Rule-based machine learning methods for functional prediction. Journal of Artificial Intelligence Research **3**, 383–403 (1995)
246. Wohler, C.: 3D computer vision: efficient methods and applications. Springer (2009)
247. Wongwaen, N., Sinthanayothin, C.: Computerized algorithm for 3D teeth segmentation. In: Proceedings of the International Conference On Electronics and Information Engineering, pp. V1–277–V1–280 (2010)
248. Woodham, R.J.: Photometric method for determining surface orientation from multiple images. Optical Engineering **19**(1), 139–144 (1980)
249. Wust, C., Capson, D.W.: Surface profile measurement using color fringe projection. Machine Vision and Applications **4**, 193–203 (1991)

250. Xu, H., Hu, Y., Chen, Y., Ma, Z., Wu, D.: A novel 3D surface modeling based on spatial neighbor points coupling in reverse engineering. In: Proceedings of the International Conference on Computer Design and Applications, pp. V5–59–V5–62 (2010)
251. Xue, B., Dor, O., Faraggi, E., Zhou, Y.: Real-value prediction of backbone torsion angles. Proteins **1**(72), 427–33 (2008)
252. Yang, R., Allen, P.K.: Registering, integrating and building CAD models from range data. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 3115–3120 (1998)
253. Yuille, A.L.: A computational theory for the perception of coherent visual motion. Nature **333**, 71–74 (1988)
254. Zha, H., Hoshide, T., Hasegawa, T.: A recursive fitting-and-splitting algorithm for 3-D object modeling using superquadrics. In: Proceedings of the International Conference on Pattern Recognition, vol. 1, pp. 658–662 (1998)
255. Zhang, F., Du, Z., Sun, L., Jia, Z.: A new novel virtual simulation system for robot-assisted orthopedic surgery. In: Proceedings of the IEEE International Conference on Robotics and Biomimetics, pp. 366–370 (2007)
256. Zhang, Z.: Flexible camera calibration by viewing a plane from unknown orientations. In: Proceedings of the 1999 IEEE International Conference on Computer Vision, pp. 666–673 vol.1 (1999)
257. Zhao, Y., Zhao, J., Zhang, L., Qi, L.: Development of a robotic 3D scanning system for reverse engineering of freeform part. In: Proceedings of the International Conference on Advanced Computer Theory and Engineering, pp. 246–250 (2008)
258. Zhenhua, Y., Xiao, F., Yinglu, L.: Online support vector regression for system identification. Advances in Natural Computation **3611**, 627–630 (2005)