UNIVERSITÀ DEGLI STUDI DI MILANO

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI
DIPARTIMENTO DI TECNOLOGIE DELL'INFORMAZIONE

DOTTORATO DI RICERCA IN INFORMATICA (INF/01 INFORMATICA)
SCUOLA DI DOTTORATO IN INFORMATICA, XXIII CICLO

Online Hierarchical Models for Surface Reconstruction

TESI DI DOTTORATO DI RICERCA DI
Francesco Bellocchio

RELATORE
Prof. Vincenzo Piuri

CORRELATORE
Prof. Alberto Borghese

CORRELATORE
Dott. Stefano Ferrari

DIRETTORE DELLA SCUOLA DI DOTTORATO
Prof. Ernesto Damiani

Anno Accademico 2009/10

Abstract

Applications based on three-dimensional object models are today very common, and can be found in many fields as design, archeology, medicine, and entertainment. A digital 3D model can be obtained by means of physical object measurements performed by using a 3D scanner. In this approach, an important step of the 3D model building process consists of creating the object's surface representation from a cloud of noisy points sampled on the object itself. This process can be viewed as the estimation of a function from a finite subset of its points. Both in statistics and machine learning this is known as a regression problem. Machine learning views the function estimation as a learning problem to be addressed by using computational intelligence techniques: the points represent a set of examples and the surface to be reconstructed represents the law that has generated them. On the other hand, in many applications the cloud of sampled points may become available only progressively during system operation. The conventional approaches to regression are therefore not suited to deal efficiently with this operating condition.

The aim of the thesis is to introduce innovative approaches to the regression problem suited for achieving high reconstruction accuracy, while limiting the computational complexity, and appropriate for online operation. Two classical computational intelligence paradigms have been considered as basic tools to address the regression problem: namely the Radial Basis Functions and the Support Vector Machines. The original and innovative aspect introduced by this thesis is the extension of these tools toward a multi-scale incremental structure, based on hierarchical schemes and suited for online operation. This allows for obtaining modular, scalable, accurate

and efficient modeling procedures with training algorithms appropriate for dealing with online learning. Radial Basis Function Networks have a fast configuration procedure that, operating locally, does not require iterative algorithms. On the other side, the computational complexity of the configuration procedure of Support Vector Machines is independent from the number of input variables. These two approaches have been considered in order to analyze advantages and limits of each of them due to the differences in their intrinsic nature.

Acknowledgements

It is a pleasure to thank the people who made this thesis possible.

First of all, I would like to acknowledge my advisor, Prof. Vincenzo Piuri, for his constant support and important advices.

I would like to thank Prof. Alberto Borghese for introducing me to scientific research and for providing valuable suggestions that improved the quality of this study.

A big thank you to Prof. Stefano Ferrari who patiently answered to all my questions. He provided me with many helpful suggestions, important advices and constant help.

I would like to thank Prof. Gerard Dreyfus, Prof. Pau-Choo Chung and Prof. Emil Petriu, the referees of this thesis, for their efforts and precious suggestions.

Finally, I would like to express special thanks to my girlfriend, Gloria, for her love and patience.

Contents

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

1

Introduction

1.1 From real object to digital model

A digital three-dimensional (3D) model is a numerical representation of the *visual* appearance of the object. From the digital model, a realistic representation of the object as a two-dimensional image can be computed. By using techniques such as perspective and shading, the human eye perception can be emulated in this image, giving a realistic representation of the object three-dimensionality. A 3D visualization system is therefore, generally, based on two key elements: the *scene*, a mathematical representation of the three-dimensional objects, and the *rendering*, the technique used for computing the 2D images of the scene.

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. Many of these applications have been developed in a wide variety of fields, encompassing, e.g., design, archeology, medicine, and entertainment. The use of digital 3D modeling in these applications provides various advantages. In particular, the model can be used for digital simulation or to create an easily modifiable digital description of the object. Besides, modeling allows for using real objects, people and environments to create their virtual representation, suited for further computer-based manipulation.

Typical applications that use 3D modeling are, for instance, the following:

- Archeology and the arts need —from one side— to preserve artworks, while —from another side— aim to expose them to as many people as possible in order to support knowledge dissemination and cultural promotion. Virtual museum allows for a larger public access than a real museum without any risk for the exposed objects and, at the same time, it can be a promotional way to attract visitors to the real museum. People interested in a single artwork have the possibility of explore, directly and in a more detailed way, its virtual representation. If an artwork is placed in a theca, the field of view can be strong limited, while its 3D model can be observed from any point of view and at different scales, so that every detail can be appreciated from its realistic virtual copy. Furthermore, 3D modeling can improve both the study of an artwork and the accuracy of its cataloging.

- Fashion design, production, and marketing may greatly exploit the use of 3D models of the human body or its parts. In the virtual fashion the customer model can be acquired for accurately evaluating his/her size as well as possible specific body characteristics. This information can greatly enhance clothing tailoring by specifically taking into account the body peculiarities for customized solutions. The model can be dressed to assist the customer in shopping by offering a virtual view and virtual cloth trying.

- Quantification of features can be exploited to compare objects and classify them. This concept is applied in different contexts. For example, in the industry it is applied to quality control, while in security it is used for identity identification by means biometric measures. 3D digitization may simplify the comparisons by extracting the features from a continuous model instead of considering the clouds of points sampled on the object surface.

- In medical applications, 3D models of human parts and organs can offer a virtual view of the body to physicians for observation, diagnosis support, and therapy monitoring. For instance, the 3D ultrasonography can be used to check the fetal morphology, while 3D tomography helps in vascular and cancer observation, diagnosis, and monitoring.

- Virtual environments are very important for training in correctly executing 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. Furthermore, surgeons can practice operations several times without using resources which are rather limited or constrained, as cadavers or animals. Recently, robotized surgery has been also developed in virtual environments to optimize and tailor the procedure on the specific patient, having then the actual operation performed by robots. Another instance of virtual training was flight simulation, used both for training and evaluating pilots. Often, in these applications the 3D models are combined with haptic devices, in order to enrich the virtual reality experience with tactile sensations.

- Design and reverse engineering are greatly helped by 3D models. Some designers, especially in the architecture field, prefer to create physical prototypes first (e.g., using clay) and then digitize them. The 3D models can be included in simulations and presentations. Furthermore, existing objects whose digital representation is not available may have to be reengineered or included in new projects. For these cases, as well as for reverse engineering, the use of 3D digitization for reproducing the objects is usually less expensive and more accurate than the manual modeling.

- The entertainment industry is increasingly using opportunities offered by 3D modeling. 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 for avoiding complex, expensive and time-consuming makeup sessions to create particular physical features, as well as the use of stuntmen in dangerous scenes. 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 are nowadays inspired to real persons, like in sport games. 3D scanning can boost the realism of the game by significantly increasing the fidelity of the avatars.

Different applications may have very different requirements. For example, reconstruction in virtual archeology needs a good accuracy and a low invasiveness, but generally time is not an important constraint. On the contrary, in videoconference applica-

tions, 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 mimic are of interest, the acquisition and reconstruction techniques can be based on a face model to achieve better results. Similarly, in industrial quality control it is important to implement fast reconstruction with a low cost, since the same operations are repeated for many objects of the same type.

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 with radius r. The Constructive Solid Geometry (CSG) has been introduced to create more complex objects by combining simple solid objects (e.g., cube, cone, sphere) by means of union, intersection, and difference operators (e.g., a tube can be seen as the difference between two cylinders with different radiuses). Unfortunately, this method is not suitable to describe a large class of complex objects, especially when the surface is very uneasy as in fig. 1.1, and therefore its use is limited to Computer Aided Design (CAD) modelling.

The digital 3D model can be created in two different ways: it can be produced by drawing the virtual description within a CAD system or by digitization based on measuring the physical object. Nowadays, with CAD systems operators are able of creating very complex models by using, typically, the Non-Uniform Rational B-Spline (NURBS) [133], a mathematical model that allows for generating curves and surfaces with great flexibility and precision. The NURBS is suitable for handling both analytic and free-form shapes.

Digitization by physical object measurement is a process that allows for obtaining the 3D model in a semi-automatic way by measuring the geometric features of the object as well as its visual features (e.g., color and texture) and, then, by identifying the most appropriate surface representing the measured points.

In the CAD systems the operator must shape the surface representing the object, while in digitization the surface is captured by the measurement devices. Digitization is generally faster than the use of a CAD system and achieves a higher (or, at least, measurable) level of accuracy with respect to the object to be modeled. Furthermore,

Figure 1.1: The Michelangelo's David is an example of large 3D object. It has been digitized in the project described in [105]

digitization, being essentially a measurement process, does not require any artistic ability of the operator.

1.2 The reconstruction of a 3D model

The systems used for digitization of real objects have very different characteristics because the applications are very variegated. The 3D digitization systems are characterized by different performances and the acquisitions techniques used are based on different physics principles. Despite of these differences, the digitization process can be described independently from the particular system used.

3D model construction is a modular process composed by different steps (fig. 1.2). Acquisition systems are used in the first step to collect information (geometric and visual) from physical objects. The sensors of the systems acquire a finite set of infor-

mation, such as 3D coordinates of points belonging to the object surface, colors, and textures. Then, this information is elaborated for obtaining a generalized description. In the regions where samples have not been acquired, the surface has to be estimated by the generalization. Moreover, as the acquisition is realized by means of a sequence of measurements, it is affected by measurement errors. Hence, the reconstruction procedure has to consider strategies for data noise filtering.

In the reconstruction of a complex shape, which cannot be surveyed by a single point of view, many sensors have to be used or many acquisitions have to be performed. Information coming from different sensors (or from different acquisitions) has to be consistently merged. This implies two processing steps: registration, where the data are referred to a single reference system and merging, where the overlapping regions are combined.

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

In fig. 1.2 a high level scheme of digitization process is reported:

- A set of sensors is used to capture information regarding geometrical and chromatic features of the scene.
- The data obtained are used to measure scene features, such as 3D position of points, color, and light position.
- From these data, a generalization of the acquired features, even where the data are scarcely available, is computed.
- The information from different sensors or from different acquisition sessions can be merged.
- The previous steps can be iterated in order to improve the quality of model (e.g., for a denser sampling in some regions).
- The representation can be transformed and optimized using a better paradigm for a specific application.

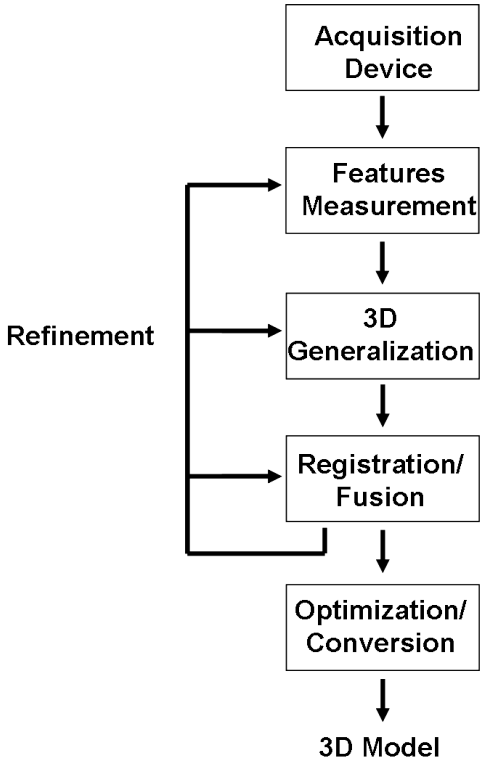


Figure 1.2: The scheme shows the main steps of the 3D objects digitization process.

Each step of digitization process can be improved exploiting a priori knowledge about the scene or about the acquisition technique. Techniques developed for general cases can be improved and customized for particular cases. The possibility of exploiting a priori knowledge allows an improvement of the final 3D model both in terms of quality and computational complexity.

1.3 The 3D surface reconstruction from sampled points

The aim of generalization phase is the computation of the complete description of the model, even in regions where the information has not been gathered.

If just the geometrical information of the physical object is considered, this phase consists, generally, in transforming the points cloud sampled on object surface in a representation from which it is possible to give a realistic graphical depiction. Hence, the object description, based on 3D coordinates points after the sampling, has to be

transformed in a description based on surfaces. In this transformation 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, information about the original surface or about the sampling process (e.g., sample density and noise magnitude). This problem is solved in literature in two distinct ways, often referred as *surface reconstruction* and *function reconstruction* [83].

In the *surface reconstruction* approach, the goal is to find a smooth function $f : \mathbb{R}^3 \to \mathbb{R}$ such that the input points $\{k_1, ..., k_n\} \subset \mathbb{R}^3$ are close to the zero set, $Z(f)$. The zero set, $Z(f) = \{x \in \mathbb{R}^3 | f(x) = 0\}$, represents the estimation of the surface. In a second stage, a contouring algorithm can be used to obtain a simple surface that approximate $Z(f)$.

The objective of *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 $f : \mathbb{R}^2 \to \mathbb{R}$ such that $f(x_i) \approx z_i$.

The first approach allows complete 3D reconstruction, but the surface is described in terms of distance function. The input domain represents the 3D coordinates of points and the codomain is a measure of the distance between the points and the surface 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 this approach the description of a complete 3D model cannot be realized using a single function, as a single function is able to describe just 2.5D surface (i.e., similar to a bas-relief representation). A further step for merging the different functions is needed.

In function reconstruction approach, the problem of surface reconstruction is seen as a problem of function approximation from points belonging to function itself. Problems of this kind occur in many branches of applied mathematics, and computer science. Many techniques have been developed to face them, such as interpolation, extrapolation, regression analysis, and curve fitting.

The function approximation problem is very general and found applications in many real situations. In the last decade, a rigorous theoretical treatment, called Statistical Learning Theory [167], has determined the developing of many efficient paradigms for solving these kinds of problems. Within this approach, the problem, called *regression problem* in this context, is seen as a learning problem, where the two-dimensional vector coordinates of the single point is an input instance, while the third coordinate is seen as an output label. The approximation function is the rule that identifies how to obtain labels from instances.

In supervised learning approaches the calculation of the solution is typically divided in two phases: in the first one, called training, examples (instances and labels) are used to configure the model, namely the parameters of model are determined, while the second one consists in testing the configured model. The testing is structured as follows. New instances are presented to the model and the labels computed by the model are compared with the expected (real) values. The differences between the expected values of labels and those computed by the model are used to evaluate the accuracy of the model itself.

1.4 Objectives of the thesis

For the solution of function reconstruction problem, Supervised Learning approaches, generally, show a good tradeoff between computational complexity, accuracy and robustness of the solution with respect to other methods. In this context, there are many different paradigms able to find the approximation function: Multi-layer Perceptron Networks, Radial Basis Function (RBF) Networks, Support Vector Machines (SVM), etc. In general, there is not a single paradigm better than the others, but each one shows good or bad performances according to the application context.

The Supervised Learning methods are characterized by two possible learning modalities [148]. The first one requires to gather all examples before the configuration of the model. This case is referred as batch learning and is used for stationary problems [22]. The second one, instead, consists in configuring the model using one example by time. This case is referred as online learning [148], and is used for non stationary problems, where the statistical distribution of the input changes with time [145], and for real-time learning [56].

In surface reconstruction problems, the chance of using real-time learning can be very important because the surface can be computed while the acquisition phase is taking place. For instance, in active 3D scanning, where a laser stripe or spot is projected over an artifact to sample data points over its surface [7], a real-time display of the current reconstructed surface would allow driving the laser toward the areas where the details are still missing in the reconstructed surface [27][147]. This largely improves the effectiveness of the scanning procedure.

Another important improvement (in terms of robustness, accuracy, and computational resources saving) can derive by the use of hierarchical models. A hierarchical model is composed by some submodels, typically called layers or levels. The layers are hierarchically organized: each layer realizes a reconstruction up to a certain scale and the output of the model is the sum of the output of each layer. The single layer is generally characterized by a scale parameter, which determines the space/frequency behavior of the layer. The hierarchical models (called, also, multiresolution models) are, then, suitable to analyze the frequency behavior of the surface in the domain. Besides, as the solution space is wide and contains many local minima, the surface reconstruction solution is computationally expensive and may be not robust. Both the efficiency and robustness can be improved utilizing hierarchical techniques.

Furthermore, the observed scene is, typically, composed by objects that can be characterized by different details levels. The chance of utilizing paradigms able to manage information at different scales can simplify the algorithms and increase their computational efficiency. Moreover this approach can be suitably merged with the online learning techniques. In fact the first points acquired from the object can be used for defining the large scale information (i.e., the first layers of the hierarchy). This initial

model can be used to drive the estimation of some initial parameters. Then, the acquisition of new points will determine the adding of new layers and an improvement of the initial model, until a required level of details is reached.

The core of the thesis is the exploring of innovative online learning modalities for hierarchical models that allow robust real-time reconstructions for 3D scanning. In particular two popular supervised learning paradigms, Radial Basis Function Networks and Support Vector Machines, have been considered here.

Hierarchical Radial Basis Function Networks [59] have shown high performances for 3D scanning problems. For this reason, its online configuration can be an important innovative tool for real-time surfaces reconstruction. Beside, Support Vector Machines have became one of the most popular paradigms for both classification and regression problems. With respect to RBF, this approach has the advantage that the computational complexity of the configuration procedure is independent from the dimension of the input space and the number and positions of the units is automatically determined. The RBF approach has the advantage that for problems in low dimensional space the configuration procedure is generally faster. On the other hand, it can be noticed that the nature of the computed solution of SVMs is similar to the RBFs one. Hence, also the SVM paradigm can be extended to a hierarchical version, more suited (but not limited) to 3D scanning problems.

RBFs and SVMs approaches have been chosen because, by means of online configuration and hierarchical organization, they can represent an important improvement of techniques already consolidated for 3D scanning, in particular for the real-time cases. These two approaches have been considered in order to analyze advantages and limits of each of them due to the differences in their intrinsic nature. In particular, the innovative contributions of the thesis can be summarized in the following points:

- Design and analysis of online configuration of Hierarchical Radial Basis Function Network models (HRBF)
- Design and analysis of Hierarchical Support Vectors Machines for Regression models (HSVR)
- Comparison of HRBF and HSVR models

- Analysis for a future development of online configuration of HSVR models

This work can be seen, also, as a starting point for studying online configured hierarchical models for applications different from the 3D scanning. In fact, both the paradigms considered are used for many other applications. For instance, SVMs are largely applied for pattern recognition problems. Hence, the study of hierarchical SVM models for classification can be a possible research direction for the future.

1.5 Thesis structure

The thesis is organized in six chapters. The first three chapters are dedicated to the introduction and detailed description of the problems, while in the remaining chapters, the innovative contributions of the work are explained, and the results are presented and discussed. In particular:

- In chapter 2 the problem of real object digitization is discussed. A survey about 3D shape reconstruction techniques and a description of the three-dimensional reconstruction problems is presented. The aim of the chapter is to give a brief overview of the typical approaches used for obtaining the digitization of a physical object.

- In chapter 3 the problem of function approximation from points for the case of supervised learning approach is discussed. This chapter contains the concepts underlying to the paradigms object of the study.

- In chapter 4 the HRBF model is presented. The RBF network model is explained and the hierarchical paradigm working principles are discussed with particular attention to the online configuration of the model.

- In chapter 5 an innovative multi-scale incremental structure for SVMs is presented. Advantages and limits of the model are explained and the experimental results are reported.

- In chapter 6 the results obtained for the two approaches are compared, the conclusions and the possible future research directions are presented.

2

The digitization process

In the previous chapter, the procedure used to obtain a digital copy of a real object has been described omitting many details, for sake of conciseness. In this chapter, the single steps which compose the digitization are described in depth and the approaches used in literature for implementing them are discussed.

The digitization process can be divided in three main phases: acquisition, reconstruction and optimization.

During the acquisition the real object is analyzed by means of devices that measure some characteristics. This information is used during the reconstruction phase for the creation of a three-dimensional model of the real object. Then the model can be optimized: its representation is transformed in an equivalent one, more suitable for the final application that will use the digital copy.

Particular attention is given here to the reconstruction phase: in fact this is the core of the thesis. However, understanding the other phases is important because, even if each phase can be described individually, there is a certain level of dependency among each other.

Initially, an overview about the different techniques used to realize the geometrical measure of a real object is presented (Section 2.1). After this overview, in Section 2.2 it is showed how the collected data can be used to obtain a general description of the digital model. In the last part of the chapter, Section 2.3, some possible post-processing procedure are described. They can be applied to the model in order to adapting it to a particular application.

2.1 Acquisition

An acquisition system is composed by a set of devices and procedures able to capture the shape and the appearance of an object. Generally, the acquisition systems realize a sampling of the acquired object surface features. The information collected during the sampling is used in the reconstruction phase to obtain the object model. The different systems can be evaluated considering their performances in terms of:

Accuracy The accuracy is an index that describes how much the measurements evaluated by a system are close to their *real values*. Different techniques can be utilized 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 with respect to the optimal plane).

Resolution The resolution measures the density of details 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 employed for measuring a feature (for example, the points per second that the system can sample). Obviously, it is important to consider also the kind of feature measured (e.g., a line contains more information than a point).

Flexibility The flexibility of a system is the capacity of acquiring a wide class of objects (considering materials and dimensions). This feature depends on the kind of sensors used and the size of the acquisition field.

Invasiveness The invasiveness is the effect that the acquisition procedure can cause to the object. The measure can modify the object in different way (light sensitivity, fragileness). In practice, this limits the use of the system for a certain class of objects.

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 features can be easily quantified, while other features are just qualitative. There are features dependent on some components of the system. For example, the physical principle used for obtaining the measurement determines the invasiveness of the system. Some features are quite incompatible. For instance, generally the usability cannot coexist with a high level of flexibility and a high level of accuracy, as both the last two features 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 used characteristics for categorizing the 3D scanning systems, as sketched in fig. 2.1. In the following sections, the devices that belong to each category are described, their working principles are explained, and their advantages and disadvantages are consequently discussed.

2.1.1 Contact 3D Scanners

In contact 3D scanners the surface of the object is probed by the physical touch. There are mainly two types of these systems: Coordinate Measuring Machine (CMM)(fig. 2.2-b) and Joined Arm (fig. 2.2-a).

The first is composed by a tactile probe attached to a vertical arm, which can be moved along the horizontal plane. The movement of the probe is allowed by the three orthogonal axes in a typical three-dimensional coordinate system. The probed coordinates result directly from the displacement of the actuators along each axis. The object is placed on a reference plane where the probe can explore it. The movement of the probe can be both automatically and manually operated.

Generally, these systems enjoy a good accuracy (Helmel Checkmaster 112-102 allows for an accuracy of 9 μm) and they are used mostly in manufacturing. The disadvantages are that the working volume is bounded by the structure and that the acquisition direction is only vertical.

The Joined Arm is composed by a chain of articulated links with a probe mounted at the end of the last link. The 3D coordinates of the probe result as the composition

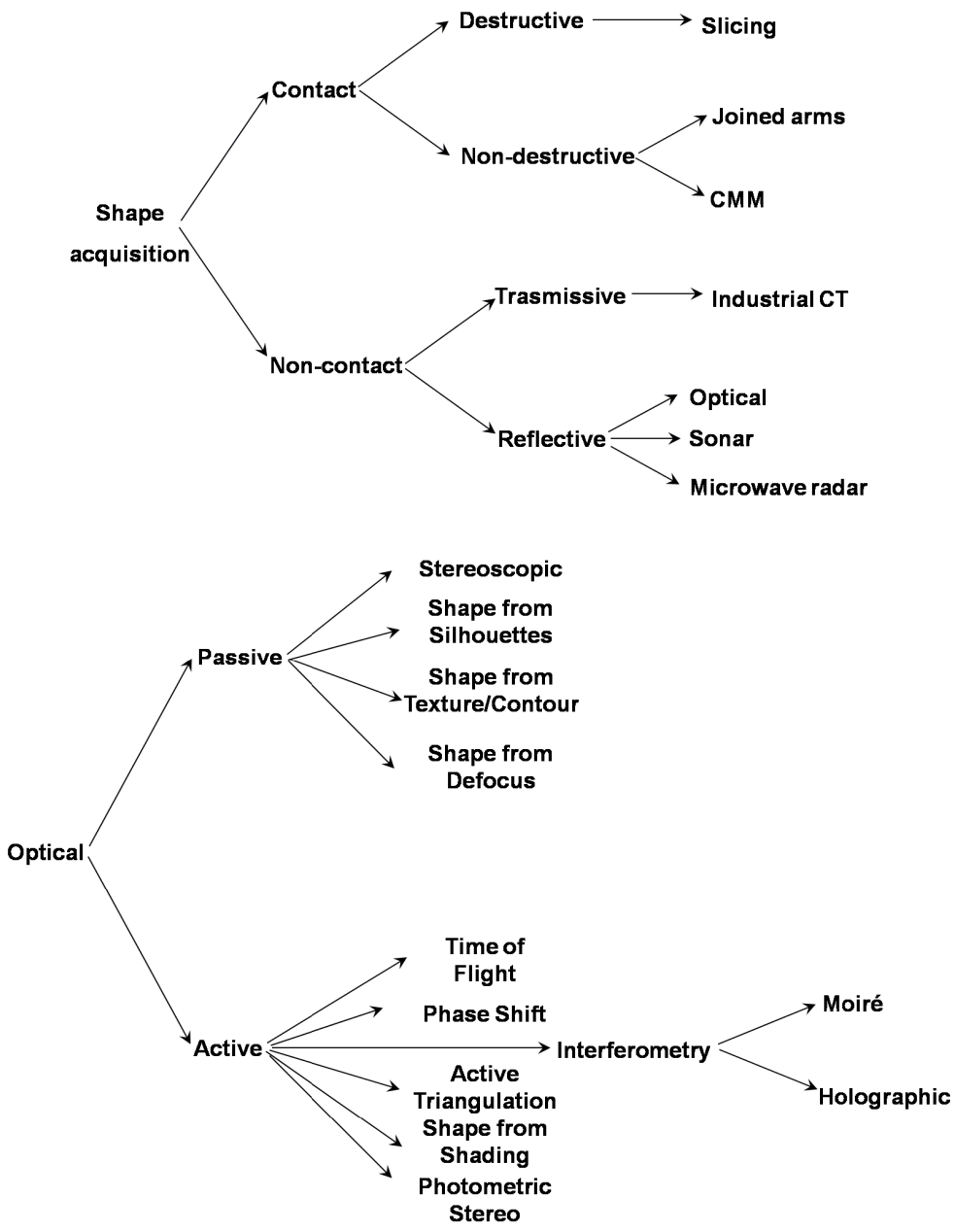
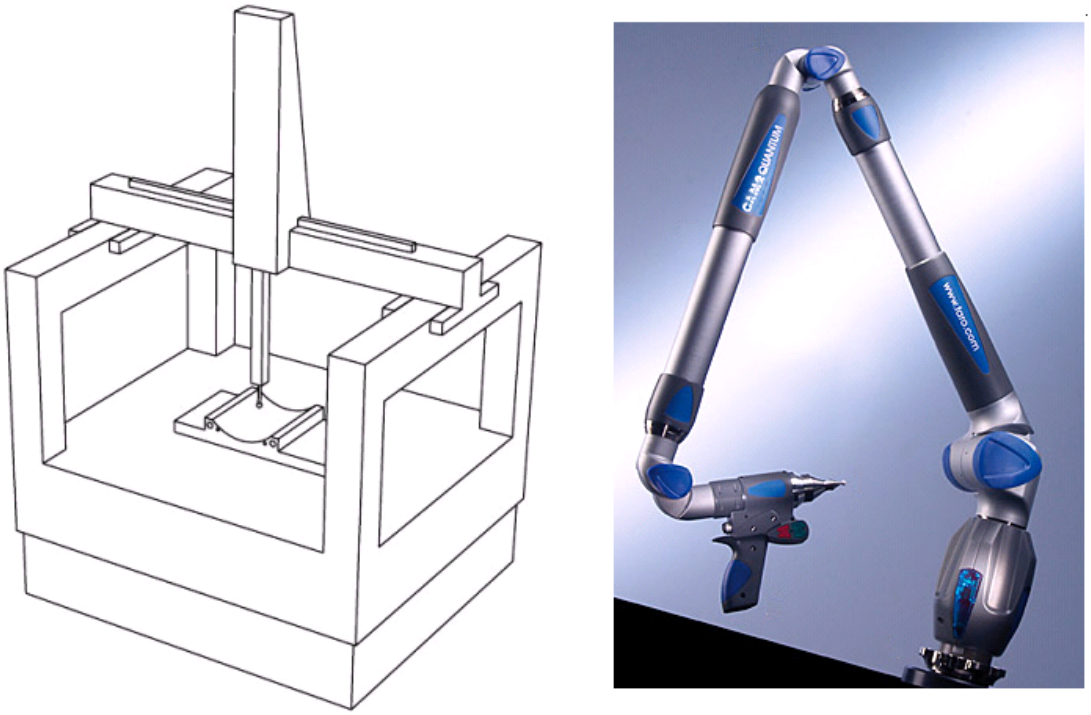


Figure 2.1: 3D scanner taxonomy

of the rototranslations operated by each link. Since, as for the CMM, the point coordinates are computed by the position of the 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 mechanical technology to be realized.

The main differences between CMM and Joined arm are in the mechanical structure. Since generally the arms have a greater degree 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 [2]), but they are relatively slow compared to the other scanner systems. Furthermore, these methods are invasive and so they are not suitable for delicate object (e.g. archaeological artifacts). Another disadvantage is the price, as these systems are generally not cheap. It should be noticed that the tactile probe of both these systems can be substituted with another kind of sensor. In this case the systems are not longer belonging to the class of contact 3D scanners.



(a) (b)

Figure 2.2: (a) Coordinate Measuring Machines, (b) Joined Arm

2.1.2 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.

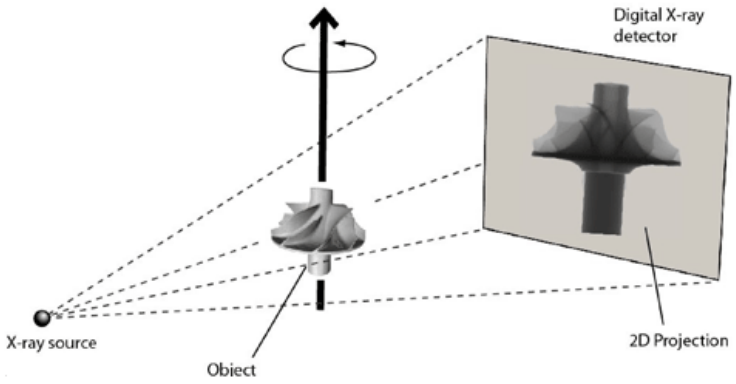


Figure 2.3: Computer tomography

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) (fig. 2.3). 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 photons emitted by the collision penetrate the target object, and are captured by a 2D detector as a digital radiographic image. The 3D model is 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 series of 2D radiographs using, generally, the back-projection algorithm [57] it is possible to compute a 3D voxel model.

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. This kind of system allows the reconstruction of both external and internal surfaces and the method is unaffected by certain object characteristics (dark, reflective or transparent surfaces). The structure of the hardware makes the system suitable for only relatively small objects. It should be noted that the density and the thickness of the object affect the energy collected by the X-rays detector. Furthermore the reconstruction of the model from the 2D radiographic images is computationally intensive.

Reflective Systems

The reflective systems exploit the radiation reflected by the object surface for estimating the position of the points of the surface. They can be classified from the type of radiation they use. In particular, optical systems use optical radiation (wavelength between 100 nm and 300 μm), while non-optical systems use sound or non-optical electromagnetic radiation 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 composed by devices based on radar and sonar systems. Although the radiations exploited are very different (the radar uses electromagnetic microwaves, the sonar uses sound or ultrasound waves), both of them are based on the principle of measuring the time-of-flight of the emitted radiation: from the time required for the wave to reach the object and return to the system, knowing the speed of the utilized radiation, it is possible to estimate the distance covered by the radiation, which can be considered equal to the double of the distance of the object from the scanning device. As this principle is used also for a class of optical scanners, it will better explained in the next section.

Due to the use of microwave radiations, radar systems have a very large depth of field, up to 400 km, and can perform ground penetrating reconstructions (fig. 2.4). A typical application is for air defense. These systems are quite expensive and generally have a low accuracy. When a sonic wave is used, as in sonar systems, the measurement is insensitive to the optical properties of the object and can be applied for the reconstruction in environments where the optical radiation would be distorted or too

much attenuated (due to absorption), as the underwater setting. These systems are characterized by a low accuracy (fig. 2.5) due to low signal to noise ratio.

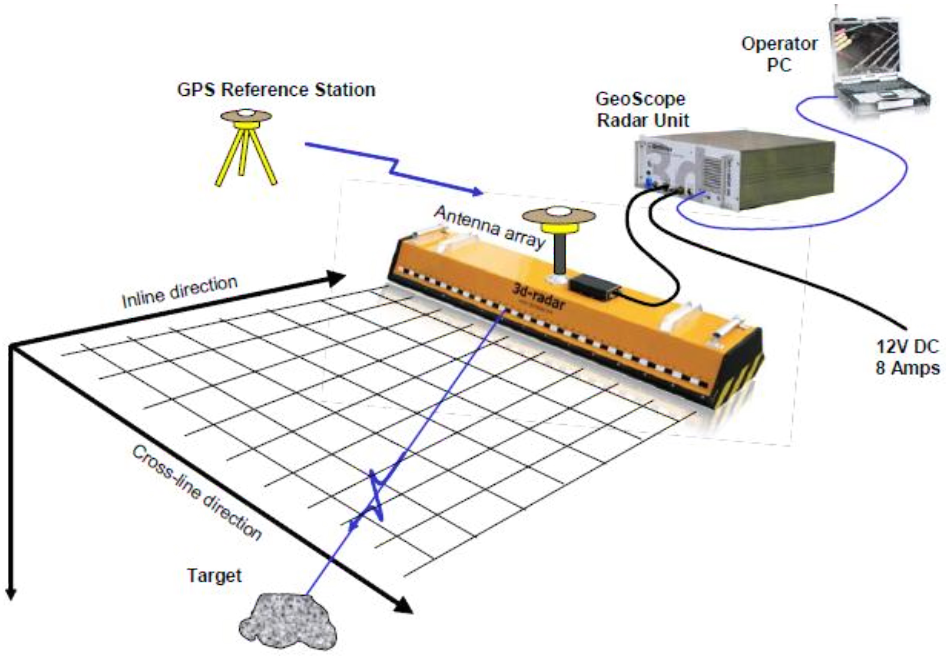


Figure 2.4: Ground penetrating radar system

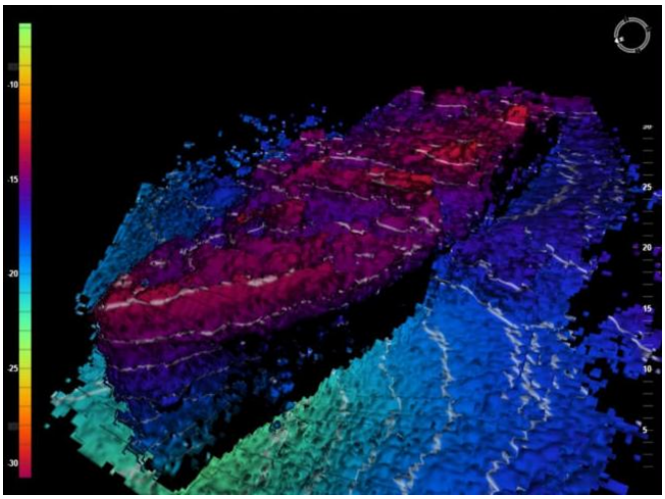


Figure 2.5: A range image acquired by a sonar system

2.1.3 Optical 3D Scanners

The ability of reconstructing an object without physically touching has important advantages: it is applicable to delicate objects (e.g., archeology artifacts), the use of radiation allows, generally, high speed of acquisition and wide range reconstruction (e.g., landscape reconstruction). Furthermore, very low cost systems can be realized. Depending on the source of the radiation (device emitted or environmental), these systems can be divided in two sub-categories: passive and active systems.

2.1.3.1 Passive Systems

The passive systems do not emit any kind of radiation themselves; they usually use the reflected ambient radiation. Generally, they are based on the use of Charge-Coupled Devices (CCDs), the classical sensors that are embedded in the commercial digital cameras. The sensors collect images of the scene, eventually from different points of view or with different optical setups. Then, the images are analyzed in order to compute the 3D coordinates of some points in the scene. The passive scanner can be very cheap; normally, they do not need particular hardware but typically do not yield dense and highly accurate digitization. Often, with these scanners the 3D points computation is not easy and a heavy computational effort can be required. Although they share the same sensor technology, different families of passive optical scanner can be found in the literature [176], which are characterized by the principle used to estimate the surface coordinates: stereoscopic, silhouettes, texture (or contour) and defocus.

Stereoscopic

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 as their 2D projection in the taken images. The first task of the reconstruction algorithm consists in identifying pairs of 2D points in different images that correspond to the same 3D point in the scene. If the corresponding 2D points are found, their projected rays can be estimated and the 3D positions of the points can be recovered as the intersection of the projection rays (fig. 2.6 and fig. 2.7). This reconstruction method is known as triangulation. It should be noticed that this method

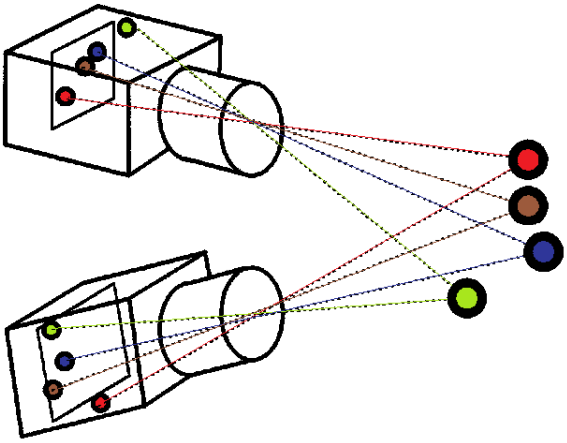


Figure 2.6: The intersection of the projected rays gives the 3D coordinates of the points.

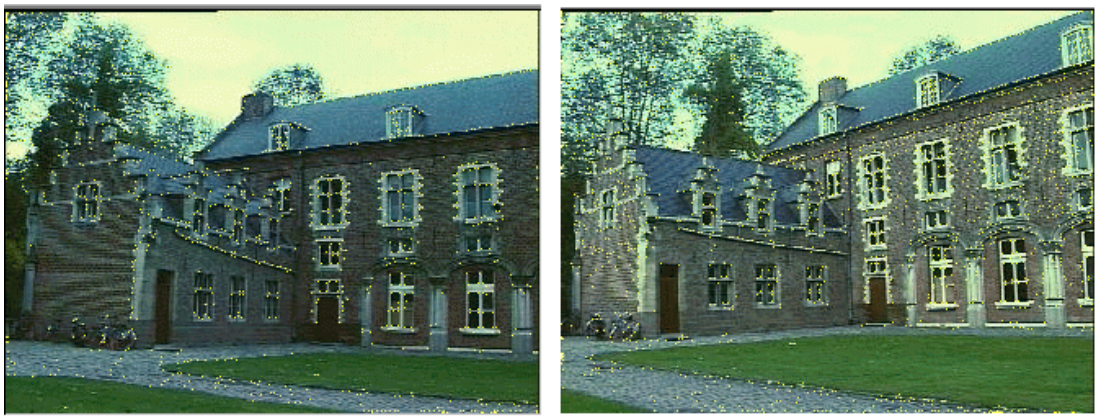


Figure 2.7: Example of real pair of images where peculiar points are highlighted

requires complete knowledge of the camera parameters: their (relative) position and orientation, but also their internal parameters (i.e., focal length, optical center, CCD size, and distortion parameters). The camera parameters are determined during a phase called calibration. Generally, this phase is performed before the scanning session, using particular known scene, as a chessboards or simple objects, where the correspondence problem (i.e., the matching between the projections of the same points in the 3D space on the acquired images) can be easily solved. An estimation of the calibration parameters can be also computed directly from the images of the object (exploiting some additional constraints on the acquired model) [114].

The main problem of this kind of system is the computation of the correspondence pairs of the 3D points. For this reason, the stereoscopic technique is generally used for the reconstruction of particular objects in which the correspondence problem can be solved easily. Since using standard image processing techniques it is relatively simple to extract peculiar points such as the corners of an object from an image, these methods are applied for the reconstruction of building or, in general, of objects in which the edges are evident (fig. 2.7).

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. However, it should be considered that in this case the complexity of the estimation of the calibration parameters can increase.

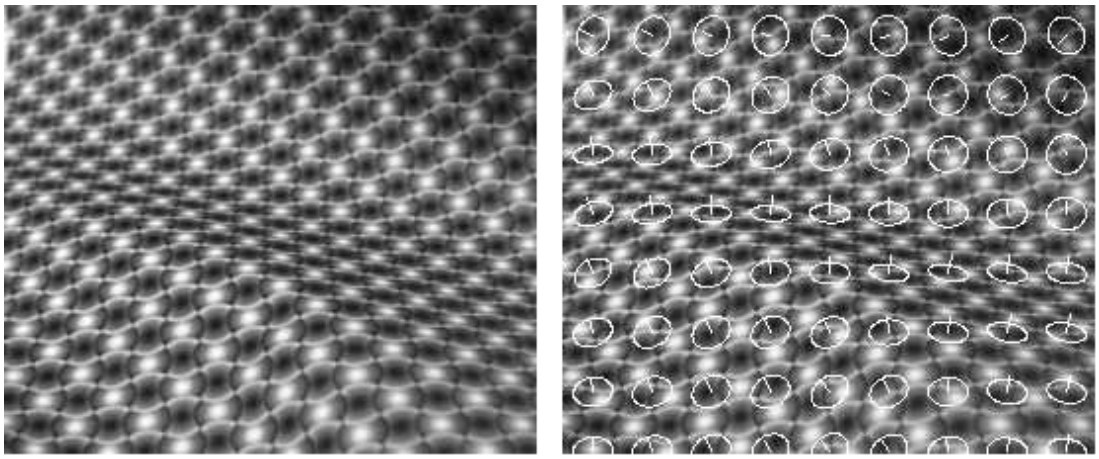
The main advantages of these techniques are the potential low cost of the hardware needed and the non-invasiveness of the method. The generally low accuracy and the sensibility to the calibration phase limit the diffusion of these systems in real applications.

Figure 2.8: Shape-from-silhouettes system.

Shape-from-silhouettes

The silhouette systems [137] [166] compute the model as composition of the contours of the object taken from different points of view. To this aim, the typical scanner of this category is composed by a turn table (where the object is placed on, fig. 2.8), a flat 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. Each contour can be seen as a cone of projected rays that contains the object. The intersection of these cones determines the approximate shape of the object.

This system has the benefit that is realizable easily and with low cost hardware, but has 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 then they cannot be reconstructed, which limit the use of these systems in many real applications.



(a) (b)

Figure 2.9: In a surface textured with a repeated pattern, (a), the shape of the surface causes distortions in the texture that can be analyzed for estimating the surface normals, (b).

Shape-from-texture and shape-from-contour

Techniques that extract information about the objects 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 for real 3D scanners. In fact, these techniques are not able to compute the 3D coordinate of object points, 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 (fig. 2.9). The curvature of the surface can be computed analyzing the distortion of the texture. The surface normals are estimated from the analysis of the local inhomogeneities [11]. 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 planar object. 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, it is possible to estimate the surface orientation that realizes this distortion.

Figure 2.10: Three images captured with different focal length

Shape-from-defocus

In the shape-from-defocus systems [104] the defocus produced by a lens is driven to allow the extraction of depth information (fig. 2.10). In these scanners, a conventional camera captures several images of the same scene using different focal lengths. Generally, this method can make use of a single camera that records all the images for the different focus setups. 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 is determined,

then knowing the region on focus for a given focal length gives the distance of that region from the camera too.

Typically, these systems are not able to make very precise reconstruction, as the accuracy depends on the setup (depth field). 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.

2.1.3.2 Active Systems

The active systems emit some kind of radiation, and the interaction between the object and the radiation is captured by a sensor. From the analysis of the captured data, knowing the features of the emitted radiation, the points' coordinates can be obtained. As a matter of fact, they are the most common scanner systems. Among the several kinds of scanners that belong to this category, the most exploited principles are: time of flight (ToF), phase shift and active triangulation. However, interferometry scanners found application for specific problems, such as the digitization of very small objects, while illuminant-based techniques have theoretic interest especially for applications where the color of the object have to be captured.

Time-of-flight

Time-of-flight (ToF) systems measure the distance from scanner to surface points through the measurement of the time employed by the radiation to reach the object and come back to the scanner. Knowing the speed of radiation and the round-trip time, it is possible to compute the distance and (knowing the direction of the emitted radiation) the 3D points coordinates [73] [1]. Hence, changing the direction of the emission, the system 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 of low frequency) and sonar (acoustic waves). The optical signal based systems are the most used type. Such systems are sometimes referred to as LIDAR (LIght Detection And Ranging) or LADAR (LAser Detection And Ranging). These systems are characterized by a relatively high speed acquisition ($10,000 \div 100,000$ points per second) and their depth of view can reach some kilometers.

Generally, the optical ToFs accuracy is limited because the high speed of radiation used. In fact, for measuring the distance with 1 mm accuracy, it is necessary to be able to measure a time range in the order of picoseconds. Hence, these systems are generally applied in long-range 3D measurement of large object, such as building 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 to geographic reconstruction; the 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 the scanner position for each acquired point. Hence, each point can be referred to the same reference system and the acquired points (which can form a dense cloud of points) can be related to a cartographic reference frame, for an extremely detailed description of the covered surface [170].

ToF scanners are often used in environment digitization. A relatively recent application is the digital crime scene reconstruction; using the digital model, the police are helped in the scene analysis task.

For this aim, the typical scanner model is composed by 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 [3].

Another kind of ToF system is the Zcam (produced by 3DVSystems) which provide 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 grey level image, where the grey value correlates to the relative distance.

Phase shift

Phase shift systems use a laser beam which power is sinusoidally modulated over the time [5]. From the phase difference between the emitted and reflected signal, it is

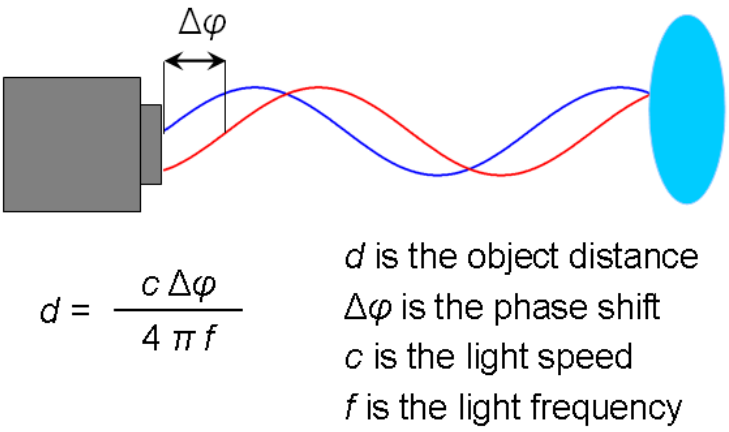


Figure 2.11: Phase shift working principle.

possible to compute the round-trip distance. In fact, the phase difference between the emission and reflection signal is proportional to the traveled distance (fig. 2.11). Since the phase can be distinguished only within the same period, the periodicity of the signal creates ambiguity. To resolve this ambiguity, multiple frequency signals are used. This method has performances quite similar to the ToF method, but can reach a higher acquisition speed (500,000 points per second).

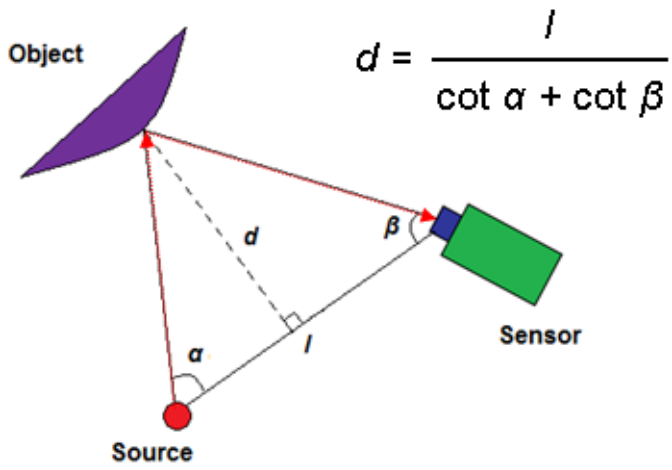


Figure 2.12: A scheme of active triangulation system.

Active Triangulation

In active triangulation systems the scene is illuminated by a coherent light source from one direction and viewed from another. These systems primarily differ in the light structure used (single spot, laser sheet beam, or coded light) and the scanning method (moving the object or moving the scanning mechanism). If the source is a low-divergence laser beam, the interaction of this radiation with the surface object will produce a spot, which can be detected by a sensor (typically a CCD). 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. As the laser line is known, the 3D point will result as the intersection point between the camera line and the laser line. Hence, the point 3D coordinates can be calculated by triangulation (fig. 2.12).

If the laser orientation and position are not known, it is possible to calculate the coordinates using two or more cameras as in stereoscopic method. In this way, the system acquires one point per frame, while using a different light source (such as a laser sheet or a matrix spot) more points per frame can be captured. As the laser sheet illuminates a plane in the space, the camera captures the contour resulting from the intersection of this plane and the object surface. Then, for any image pixel on the contour, the corresponding 3D point on the object surface is found by intersecting the ray passing through the pixel and the laser 3D plane equation. The use of a matrix spot allows to sample a region instead of a line (as done by the laser sheet), and it can be potentially the fastest solution for surface acquisition. However, the problem of matching each beam with its projected point acquired by the camera is more complex than in the single beam case.

The use of a matrix spot would be required, with strong constraints on the speed of acquisition, whenever a moving object has to be acquired. As the problem of determine the pairs of points on the images is hard, this technique is not generally used. The typical approach used in this case is instead the projection of a structured light pattern. There are many different techniques based on the projection of structured pattern, and generally they make use of a calibrated camera-projector pair [15]. The aim of these techniques is to characterize each point by projecting a different light pattern on a different direction. Hence, the illumination is used like a code, allowing the correct

identification of each direction. The encoding is realized using different strategies as colored stripes [178] or time-coded stripes [147]. The colored stripes encoding presents an important problem: both the surface color of the object and the ambient light influenced the color of the reflected light. For this reason, to reconstruct a colored (or textured) object, other kinds of coding are preferred.

In [147], a structured-light range-finder scanner using temporal stripe coding is proposed. Using a projector and a camera synchronized at 60 Hz, four successive frames are exploited to acquire a 115×77 matrix points. For each frame, a set of black/white stripes is projected. Observing as a pixel changes its color (from white to black and from black to white) in different frames, it is possible to compute which stripe is illuminating the pixel and then, by the triangulation, the 3D position of the point. Actually, the entities that carry 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 have a stripe on the left white and on the right black, and so on).

Another very efficient scanner system is proposed in [87]. This system uses three phase-shifted sinusoidal gray-scale fringe patterns, to provide pixel-level resolution. A projector and a camera are synchronized at 120 Hz with a resolution of 532×500 points per frame, achieving a system accuracy of 0.05 mm. For each pixel, the phase from the three pattern intensities is calculated. This phase information determines the correspondence between the image field and the projection field. The phase map calculated from the three camera images can be converted to the depth map by a phase-to-height conversion algorithm based on triangulation. With this system it is possible to realize a real-time reconstruction. For example, the system is able to measure human faces, capturing 3D dynamic facial changes. In order to provide a high definition real-time reconstruction it is employed a GPU (Graphics Processing Unit)[4] to compute the 3D coordinates points. These devices have a highly parallel structure that makes them more effective than typical CPUs, for a range of complex algorithms [183]. GPUs are very useful in the reconstruction problems because typically these problems are characterized by parallelizable computation.

The active triangulation systems, generally, are characterized by a good accuracy and are relatively fast. The strong limitation of these systems is the size of scanning field. As the coordinates are computed by means of triangulation, the accuracy of

these systems depends by the angle formed by the emitted light and the line of view, or equivalently, by the ratio between the emitter/sensor displacement and the distance of the object (fig. 2.12). In the optimal set-up, the considered angle should be 45°. Moreover, the resolution of CCD, the resolution and the power of source of light limit further the active triangulation systems to a depth of field of few meters. Hence, these systems are not usable for digitization of large objects. Furthermore objects color and ambient illumination may interfere with the measurement.

Figure 2.13: Example of the shape information contained in the shading.

Shape-From-Shading and Photometric Stereo

The shape-from-shading problem consists in the estimation of the three-dimensional shape of a surface from the brightness of an image of that surface. The first formulation of this problem was proposed in the 70s [85]. 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 [29]. 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. Moreover, this kind of ambiguity can be widely generalized. In [16], it is showed that, given the illuminant direction and the Lambertian reflectance (albedo) of the surface, the same image 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 lighting with known direction and intensity. But, again, this simplification is not enough and, in order to solve the problem, knowledge about reflection properties of the surface of the object 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 values of the normals in a neighborhood of the considered point and making the assumption that the surface is smooth.

When a photometric stereo technique is used, the problem is simplified by illuminating the scene from different positions [82] [80]. With this technique, introduced in [177], it is possible to estimate the local surface orientation by using several images of the same surface taken from the same viewpoint, but under illumination that comes from different directions. The light sources are ideally point sources, which positions are known with respect to the reference system, oriented in different directions. The lights are activated one at a time, for each captured frame, so that in each image there is a well-defined 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 estimation of the color of the surface.

Moiré Interferometry

The Moiré interferometry is a technique that allows to detect and measure deformations in a quasi-planar surface. The method utilizes interference effect between some form of specimen grating and reference grating [88]. 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. 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 fringe distances obtained from the superimposition of the grating projected with the grating observed and knowing the projection and observation angles.

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

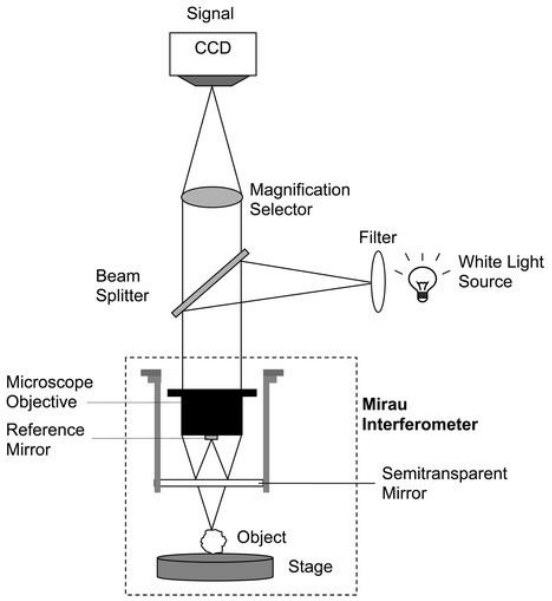


Figure 2.14: A scheme of a holographic system

Holographic Interferometry

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 (fig. 2.14). The holographic interferometry is a technique for measuring vertical displacement by comparing the holograms of the same object at different states. In particular, vertical displacements can be estimated comparing images taken while the object is moved along the vertical axis. The images are analyzed to detect the peak of the interference pattern for each pixel, which allows computing the height of the considered pixel. The systems based on this technique are quite expensive, but allows sub-nanometer measurement. Since the field of view is very small, generally, the method is applied for objects of size of few millimeters.

2.1.4 Hybrid Techniques

In the previous pages, an overview of many techniques characterized by complementary strength and weakness has been presented. Many real systems implements more than one technique for exploiting the advantages of each approach. For instance CAM2 Laser ScanArm V3 is a Joined 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 Joined 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.

Furthermore an optical 3D scanner system can be enriched with a couple of emitter and receiver that exploit another kind of radiation (such as sonic, microwave radiation) for the objects or environments in which the optical radiation cannot be applied.

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

Picture	Model	Measuring Method	Scan Range (depth of field)	Accuracy	Input Time
	Creaform Exascan	Laser triangulation	30 cm	±0.04mm	25,000 points/s
	Leica HDS6000	Tof, phase-based	100 m	±2 mm (25 m)	500,000 points/s
	I-Site 4400LR	Tof, pulsed range finder	600 m	±20 mm	4,400 points/s
	Metris MCA	Phisical contact	2 m	±0.028 mm	
	Metris K-Robot	Laser triangulation	17 m^3	±0.1 mm	1,000 points/s

Figure 2.15: Some types of 3D scanners

2.1.5 Evaluation of 3D systems

The systems have been classified in a taxonomy that privileges, as criterion for the classification, the physical principle exploited to extract the 3D information. However, other properties of the scanning systems can be used as classification key. For instance, among other, the accuracy, the resolution, or the speed of acquisition can characterize a scanner system, but these properties are more related to an actual implementation of the systems than to a class of scanners and hence are not suited for a structured treatment of the subject. 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. Obviously, there is no way for indicating a scanner system as the best one, because each model has been designed for a specific field of application (fig. 2.15).

In [35] a method for evaluating the 3D scanners is suggested. It considers some important features (e.g. field of view, accuracy, physical weight, scanning time) and for each feature it associates a weight. By giving a score for each feature of each scanners

considered, a single final score can be computed as the sum of each score.

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

An important aspect to note about every 3D scanner is the calibration procedure. Generally 3D scanners have different setups and the point cloud reconstruction is possible only if the setup parameters are known. The aim of calibration phase is the estimation of setup parameters. This phase is critical for many types of scanners because the precision of the system is, typically, strongly connected to the quality of calibration performed. Moreover the time employed to realize the calibration can be very long. The procedure can spend several minutes, even though the scanning session can require just some seconds.

However, as the technological advances improve the computational power and the performances of the devices, more attention is paid by the scanner designers for making the systems more user friendly. In fact, in the last decade many research works are related to the estimation of the calibration parameters without a proper calibration stage. In this track, an interesting approach, mainly oriented to the stereoscopic techniques, is the passive 3D reconstruction, which allows the estimation of the calibration parameters after the acquisition session.

Since devices for the reproduction of 3D contents 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 small, point-and-click optical devices.

2.2 Reconstruction

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 charac-

terize this problem: the a priori knowledge about the object and the kind of information collected by the acquisition phase.

In order to solve this problem, a paradigm for surface representation is needed. This paradigm can be used to encode a priori knowledge about the problem. For example, if the acquisition is concentrated to a certain class of objects, a parametric model can be used to represent the objects of that class. The reconstruction can be simplified by searching just the parameters that best fit the acquired data.

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

The problem of surface reconstruction can be formulated as an optimization problem. In general, a model based approach is more robust [48]. However, the general case where the a priori knowledge is strongly limited is here examined.

Similar considerations can be applied for the data acquired. Some systems are able to acquire lines, edges or other features with a higher informative content, but the reconstruction from points is a more general problem and it is considered here. Furthermore, in reconstruction phase, problems connected to the data uncertainty should be considered, such as: noise, missing or ambiguous data, and the missing information about the surface topology.

2.2.1 Reconstruction from points

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

The hand surface reconstruction can be considered in order to clarify the difference between geometry and topology of a set of data and their influence in the reconstruction problem. Points placed on the tip of two fingers of the same hand can be geometrically very close, but topologically are very far from each other. In fact, the geometric (Euclidean) distance is defined as the shortest line segment that connects the two points, while the topological distance is defined considering only those segments that belong to the surface: the two distances can be very different. The geometric and topological

distance is similar for points in the region at the base of the fingers, close to the palm: there, points belonging to different fingers can be confused due to the presence of noise.

The difference between topological and geometric distance becomes smaller and smaller if the surface region containing the points becomes similar to a plane. The knowledge on the topology of the surface simplifies the reconstruction problem. For this reason, the a priori knowledge about the object can be very useful. Some surface transformations do not change the features related to the topology.

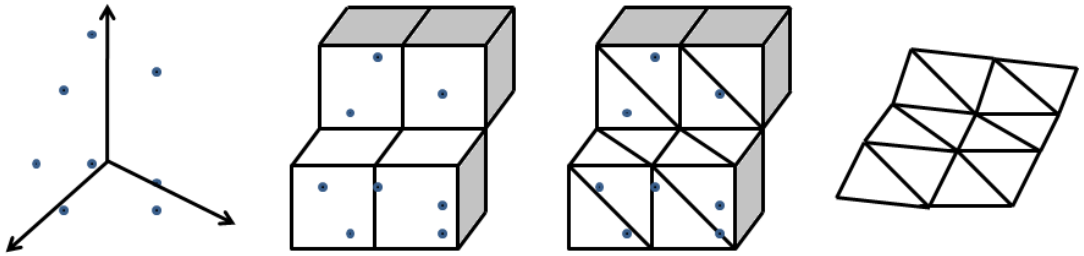
If the reconstruction is based only on the 3D coordinates of the points, the topology information can be unknown. The simplest way to determine the topological distance among the points is to consider it equal to the Euclidean one. Hence, points geometrically close will be also considered topologically close.

In practical cases, the problem of 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 found. This model is computed by means local planar approximation and it is used as topological reference. In the second phase the positions of the points is transformed on the base of their displacement with respect to the polygonal model computed in the first phase. The results of the two phases are used to reconstruct the detailed surface.

The problem of surface reconstruction from points has been largely studied in the last years. The techniques to address it can be divided into two classes:

- volumetric techniques, which process the data without any additional information about the topology and are based on the search of the volume occupied by the object;
- function approximation techniques, which consider the surface as a function on a given domain.

There are also techniques based on the combination of the two classes. For example it is possible to use a volumetric technique to find a rough reconstruction of the object surface (that will belong to the same topological class of the object surface) and use this reconstruction as the domain of a function that realizes a detailed approximation of the object surface.



(a) (b) (c) (d)

Figure 2.16: 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.

Another feature that characterizes the reconstruction techniques is the kind of solution: interpolation or approximation of the acquired points. As, generally, the points are the result of a measurement process, an interpolation cannot determine a good solution, due to measuring error. Despite this consideration, an interpolation can be used as first approximation that can be improved in next processing phases.

A more detailed classification of the reconstruction techniques can be found in [115].

There are definitions and methods well known in the field of surface reconstruction, as: convex hull, Delaunay triangulation, Voronoi diagram, α-shapes, octree, marching cubes. The full explanation of these concepts goes beyond the scope of this thesis and can be found in [139]. For reference, a short definition of these concepts is reported in the Glossary (6.2.3).

2.2.2 Volumetric methods

Generally, for the computation of the object topology, volumetric methods are used. The algorithms of this class are composed of the following steps [115]:

1. Point space decomposition in cells

2. Selection of the cells crossed by the surface

3. Surface computation from the selected cells

In [10] the bounding box of the points is partitioned into cubes of the same edge (fig. 2.16-a–b). Only the cubes that have at least an internal data point are considered (fig. 2.16-b). 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. 2.16-c). 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. 2.16-d).

In [84] the *signed distance function* concept is used for the approximation of the surface topology. The signed distance function of the surface assumes a positive value in external points and negative value in internal points. The isosurface (i.e., the region of the space where the signed distance function is zero) is then the target surface. The space occupied by the points is divided in a regular lattice and for each vertex the value of the signed distance function is estimated. Then, the cubes which have vertices of opposite sign are selected. This is equivalent to select the cubes through which the surface passes. Then the *marching cubes* algorithm is applied to the selected cubes for improving the surface resolution.

A similar approach is proposed in [144], but with a different technique for the computation of the signed distance function. 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.

Another critical factor of the volumetric methods based on a regular lattice is the size of the cells. In [14] tetragonal cells are used. In the first phase, there is a single cell that includes all the points, which is then divided in four parts adding the barycenter at the set of the vertices. Then, the signed distance for all the vertices is computed. The tetrahedrons whose vertices have the same sign are deleted. For each of the remaining tetrahedrons an approximation of the surface that passes through it (a Bernstein-Bézier polynomial) 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.

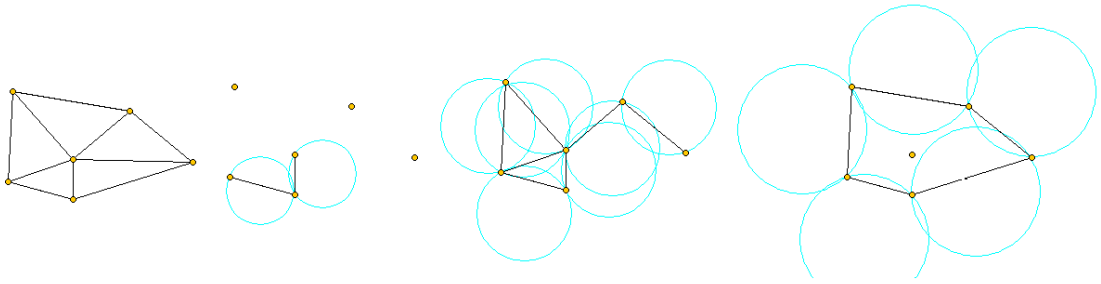


Figure 2.17: Example of α-shape computation in 2D space. In panel (a) the Delaunay triangulation for a set of six points is shown, in panel (b), (c), and (d) the results of the edges elimination based on three different values of α.

In [130] the set of points is partitioned by means a clustering algorithm (*k-means*). Each cluster is represented by a polygonal approximation and the model obtained is used as 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 the clustering.

There are some methods based on the deletion of elements. This approach is called *sculpturing* or *carving*. In [54] it is proposed a reconstruction algorithm based on α-shape. The Delaunay triangulation is first computed and then every triangle which cannot be circumscribed by a sphere of radius α is deleted: the remaining triangles compose the α-shape of the dataset. In fig. 2.17-a–d the working principle for the 2D case is represented. The Delaunay triangulation is first computed (fig. 2.17-a) and the every edge which cannot be circumscribed by a circle of radius α is deleted (fig. 2.17b–d).

For the computation of the faces composing the surface, the following criterion is adopted: 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 points 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.

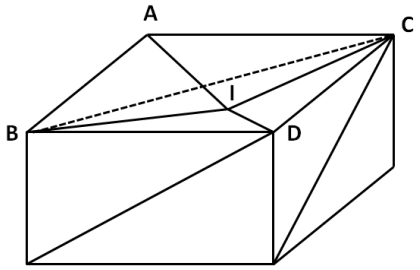
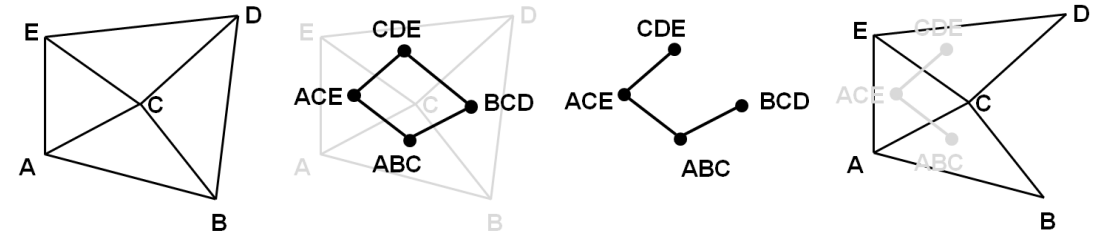


Figure 2.18: Example of tetrahedrons elimination [24]. The tetrahedrons $ABCI$ and $BCDI$ are eliminated because they have the following elements toward the outside of the object: single face, three edges and three points. In this way, the cavity determined by I can be visible.

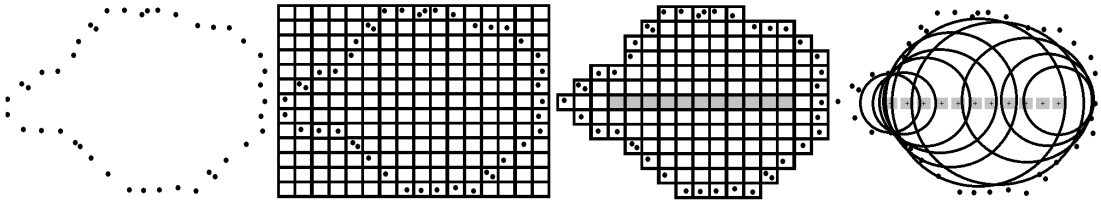
In [24] it is proposed a method that, from the set of tetrahedrons obtained from the Delaunay triangulation, computes the elimination of tetrahedrons considered external to the object. The external tetrahedrons are selected using a heuristic (the tetrahedrons that have the following elements toward the outside of the object: two faces, five edges and four points or a single face, three edges and three points) (fig. 2.18). The aim of the procedure is obtaining a polyhedron of genus 0 which have all the data points as its vertices (in other words, all the data points are on the surface). This is obtained by iteratively eliminating a 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 to sort the tetrahedrons. The tetrahedrons with the largest values will be eliminated first. The decision value is defined as the maximum distance between the faces of the tetrahedron and the points that lies on the sphere circumscribing the tetrahedron self. Hence, tetrahedrons that are large and short will be eliminated first. These tetrahedrons, generally, hide surface details. This algorithm does not allow the reconstruction of surfaces with non-zero *genus* (see Glossary 6.2.3).

An alternative approach [150] is based on the duality between Delaunay triangulation and Voronoi diagram. The minimum *spanning tree* of the Voronoi diagram (where each node corresponds to the center of the thetraedron 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. 2.19). The elimination of a node in the *spanning tree* corresponds to elimination of the tetrahedron associated in the triangulation (fig. 2.19-d).



(a) (b) (c) (d)

Figure 2.19: Example of spanning tree computation in 2D space. In panel (a) the triangulation for a set of five points is shown. In panel (b) the graph associated to the triangulation; the nodes represent the triangles and the edges represent the edges in common between two triangles. In panel (c) the minimum spanning tree is depicted. In panel (d) the triangulation resulted from the pruning of the minimum spanning tree.



(a) (b) (c) (d)

Figure 2.20: Example of medial axis approach in 2D space. In panel (a) the point cloud. In panel (b) the input points partitioned by a regular lattice. In panel (c) external squares are eliminated and the squares with the maximal distance are selected. In panel (d) the circles with radius equal to the minimum distance to the points are represented.

In [19], the principle of *sculpturing* of an initial triangulation obtained from the α-shape is used. The elimination criterion is different from the previous method, 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. As in [84], the surface can be defined as a signed distance function. This function, as the surface, is unknown. An approximation is computable and can be used as estimation of the surface (for example by means the *marching cubes* algorithm). The signed distance function should be reasonable just in the regions close to the surface.

In [23], a different estimation of the distance from the surface with respect to the signed distance function is introduced. It is called *medial axis*, 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. 2.20-d). An approximation of the medial axis of a data set is computed from the *bounding box* of the input points partitioned by a regular lattice (fig. 2.20-b). 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. 2.20-c). 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. 2.20-d). 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, then they are noise affected. For this reason, a noise filtering phase can be needed for the vertices of the reconstructed surface.

2.2.3 Function approximation

In [119], a mesh deforming procedure is proposed. 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.

A different approach is described in [160], where the adaptation is obtained by the use of *oriented particles*. Each particle has some parameters that are modified during

the adaptation phase. The modeling of the interactions strength (attractive and repulsive) among particles determines the surface.

The physic metaphor for surface reconstruction is used, also, in other works. In [10] 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 model can be expressed as a differential linear equation system and the solution can be found iteratively. After the computation of the equilibrium state, the physical model is transformed in a geometrical model. The solution represents a tradeoff between smoothness of the surface and data fitting. The trade off is determined by the chosen physical parameters.

The physical modeling approach is used, also, in [163]. 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 the point cloud data. The ellipsoids are used to incorporate the global shape and the local features are represented by means of splines. The deformations are determined by dynamic rules, expressed as a set of Lagrangian motion equations. The numerical simulation of the motion equations determines the evolution of the 3D model under the action of forces and constraints. 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.

The iterative adaptive structures have been used also in the field of artificial intelligence, as methods for the solution of learning from examples problems. The *Self-Organizing Map* (SOM) [97], also known as *Kohonen features map*, model is composed by a set of units, $\{u_j\}$ organized in a reticular structure, and characterized by a value, $w_j \in \mathbb{R}^D$, in feature 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 feature space, u_k, called *winning unit*, and the units connected to it are adapted with the following rules:

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

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). At the end of the configuration the units closer in the reticular structure will have similar features. When applied to three-dimensional data, the reticular structure will represent the polygonal surface and it will be a smooth approximation of data points.

The SOM presents two kinds of problems. Despite that it can be shown that the initial configuration of the features vector of each unit can be chosen randomly, in practice the performances are greatly affected by the initial position of the units. A good starting configuration allows to save computational time that can be used in order to obtain a better solution. Hence, a preprocessing phase to obtain an initial model close to the data can be useful in this sense.

Some units, called *dead units*, are not used in the adaptation. The final position of these units will be similar to the initial one. This fact can determine not smooth regions of the surface. This situation occurs when these units are too far from the elements of dataset and they never result *winning units* or when the reticular structure is too dense and these units are never chosen for the adaptation. To solve the problem of dead units, in [13] 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 closer input data in feature space. Hence, possible dead units can be brought near to the data and can participate to the fitting.

In [116] it is presented an approach based on the idea of *surface description graph* (SGD). It is defined as an extension of the minimum *spanning tree* of the dataset. Each SGD node is used for the extraction of the surface features in a region. The aim of this procedure is the individuation of the regions with the same morphological features, which can be connected in order to obtain the surface reconstruction.

2.2.4 Multiresolution representation

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. The large scale features are generally

related to the kind of object, 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 possible, few resources would be generally sufficient to describe the behavior at large scale; conversely the configuration of the resources for the small scale features can be performed with an analysis of the data focused in 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. 2.21-a), where the description at low resolution is extracted first, and then the resolution is increased till the details are reconstructed;
- *fine to coarse* (fig. 2.21-b), where the description at the maximum of the resolution is performed first, and then the description at smaller resolution is obtained.

A coarse to fine multiresolution paradigm is presented in [182], where a cloud of points is recursively approximated using superquadrics. In the initial phase, the superquadric that best approximates the data is estimated (fig. 2.22-b). For each point, the residual of the fitting (i.e., the distance between the point and its approximation) is computed (fig. 2.22-c) and a plane is fitted to data accounting for large error (fig. 2.22-d). 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. 2.22-e). 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 at different resolutions, is determined. This method does not determine a hierarchy, because the reconstruction at a scale does not use the model at the previous resolution.

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 a scale of reconstruction. The reconstruction for a certain scale is obtained

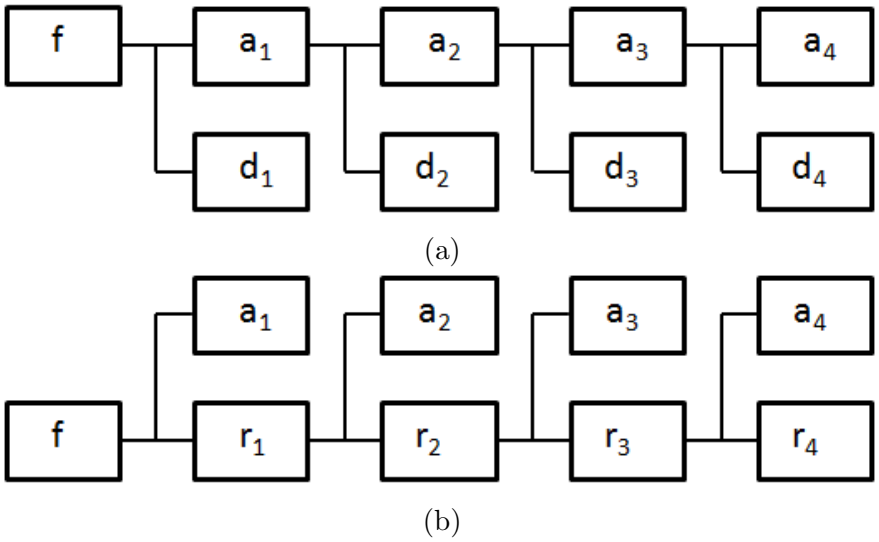


Figure 2.21: The two approaches to obtain a multiresolution representation of the target surface f. In the fine to coarse approach, panel (a), the scale of approximation increases with the layers. The function a_i $(0 \leq i \leq 4)$ represents the approximation of a_{i-1}, where $a_0 = f$, obtained at layer i, while d_i represents the error of the approximation of the layer i: $d_i = a_{i-1} - a_i$. In the coarse to fine approach, panel (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 error of the approximation of the layer i: $r_i = r_{i-1} - a_i$.

using all the levels at scale greater or equal to the scale chosen. Then, the representation presents a hierarchical structure.

The Hierarchical Radial Basis Functions (HRBF) networks [25] [26], treated in depth in chapter 4, are an example of a *coarse to fine* hierarchical paradigm. The HRBF model is composed of a pool of subnetworks, called layers, and its output is computed as the sum of the outputs of its layers. The output of each layer is computed as a linear combination of Gaussian. For each layer, the Gaussians are regularly spaced on the input domain and they have the same standard deviation.

The configuration of the weights 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 (the standard deviation of the Gaussians) is then decreased

(a) (b) (c) (d) (e)

Figure 2.22: Panel (a) shows a 3D point cloud. In panel (b) the first approximation computed using a superquadric. In panel (c) the grey level represents the residual of the points. In panel (d) the dividing plane for the region with large residual is shown. In panel (e) the second approximation on the two subset is depicted.

and a new layer is inserted to approximate the detail. The process can be iterated till the maximal possible resolution is reached (fig. 2.23).

The Hierarchical Support Vector Regression (HSVR) model works in a very similar way; it is explained in depth in chapter 5. The idea is, again, 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 solutions of an optimization problem.

In a similar way, the *Multilevel B-splines*, proposed in [102], 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, [66] works only with regularly spaced data. For this case, a fine to coarse algorithm has been proposed [67], where the multiresolution representation is derived from a initial B-spline representation at the maximal resolution.

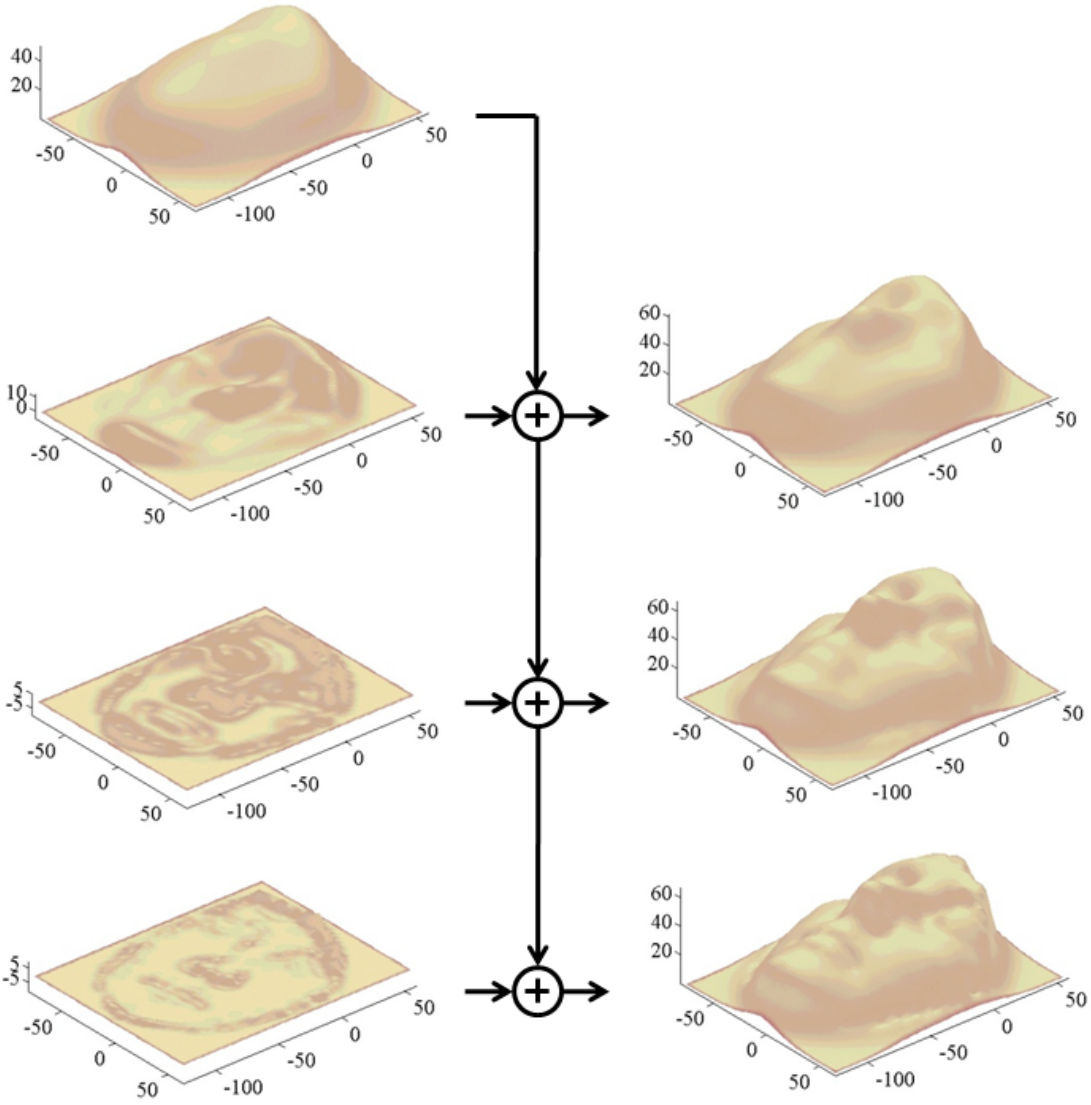


Figure 2.23: The outputs of four layers of a HRBF model are depicted in the left. The sum of the single layers output gives the multiscale approximation, in the right.

As the previous method, the *Multiresolution Analysis* based on the wavelet transformation [110][38] works with regularly spaced data. An MRA is characterized by a pair of functions, called scaling function and wavelet, that generate 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 space). Hence a function can be represented as a linear combination of shifted copies of the scaling function and the wavelet. The coefficients for the approximation and for 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). The properties of an MRA allow to decompose the scaling function as a linear combination of scaled copies of the scaling function itself and scaled copies of the wavelet. Hence the function decomposition can be iterated, which allows to obtain the function representation with a fine to coarse approach: the process is iterated on the approximation coefficients in order 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.

In [131], a MRA based approach for not regularly spaced data is presented. The data considered is 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 [151] is applied.

In [65] another *fine to coarse* technique based on the wavelet 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.

2.2.5 Integration

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 this case the acquisition has to be performed from different points of view. The information captured from different sensors has to be integrated in order to calculate a single representation. This operation can be divided in two steps:

Registration the different information are reframed with respect to the same reference system;

Fusion a single model is generated combining the data of different views in the overlapping regions.

Registration

The *registration* is an operation used in many fields. Whenever the data, coming from different sources of information, have to be transformed in a single reference system, a registration procedure has to be performed. In particular, in the case of three-dimensional reconstruction, the registration can be needed for different reasons:

Sessions the acquisition is performed at different times, for example if the number of sensors is not enough to acquire the object in a single acquisition;

Sensors different sensors have to be used to acquire different features of the same object;

Subject if a comparison of different object's models has to be performed.

The registration procedure can be simplified if there are reference points in the different acquisition or if there is information about the calibration of the acquisition device. A typical case is the registration of data collected at different times. In this case, the point of view changes in time (or the point of view is the same, but the object is moving). The registration of different views corresponds, then, to the inversion of the function that describes the motion of the object with respect to the sensor (e.g., if the object is placed on a turn table). If there are not common reference points in the different views, some features have to be found in order to compute the transformations

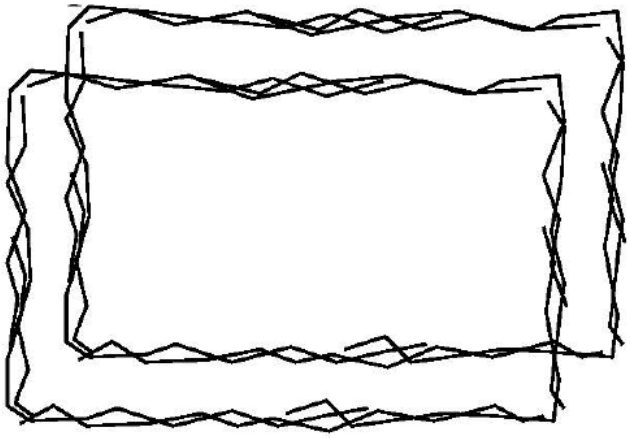


Figure 2.24: Example of a bad registration due to the presence of two subsets poorly registered among each other.

that relate the different acquisitions. As, generally, an overlapping region for each pair of different views is not available, some views can be accurately registered among each other, but poorly registered with respect to other views (fig. 2.24). The formation of cliques of views can introduce registration error which results in abrupt variations in the fused dataset. Hence, the registration has to solve mainly two kinds of problems:

- a good estimation of the object features;
- a uniform diffusion of the registration error.

The algorithm *Iterative Closest Point*(ICP)[20] is one of the most used to realize the registration. The effectiveness of ICP has been proved for *every* representation modality [20]. The algorithm is based on the definition of a distance function among points of the registration sets. For a set of points, a rototranslation is operated. This rototranslation is such that the distance among the set of points and the closest correspondence 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.

In [165], a modified version of ICP is proposed. The views are represented as *range images* and a polygonal surface is computed for each of them. The vertices are weighted

with a value that represents the reliability of the measure. The ICP is applied just to the vertex with a correspondence in 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 (it 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 [179] the registration is performed in two steps from the surfaces corresponding to 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 law is computed aligning these curves.

Fusion

The calibration and reconstruction errors can cause the surfaces belonging to different views, after the registration, not to overlap perfectly. The simple unification of the registered data or the surface would produce a poor representation of the object surface, with artifacts in the overlapping regions. The aim of the fusion procedure is to improve the representation of the surface in the overlapping regions (fig. 2.25).

Generally, each acquisition device has an error distribution which depends on the features of the acquired object. For example, the uncertainty on the data in the regions close to a border is, typically, greater than the uncertainty in the inside regions, or it may depend on the relative orientation of the surface and the sensor. This information, acquisition device dependent, can be exploited in the fusion procedure to weight the reliability of the data.

In [50], where each view is a *range image*, each pixel of the view is weighted with its distance from the closest border pixel. The depth of each pixel, in the resulted merged image, is computed using a weighted average. The border pixels are adapted in a smooth way.

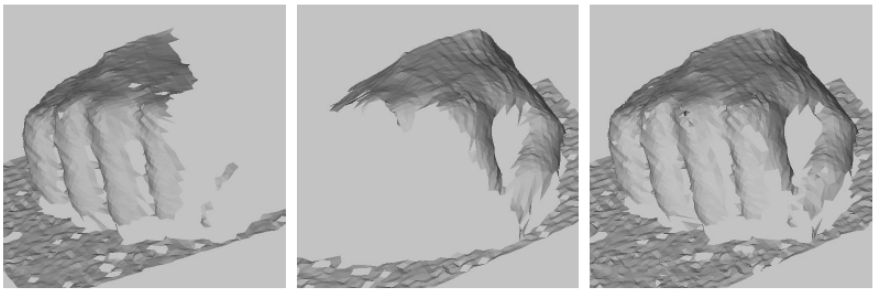


Figure 2.25: The first two models, on the left and center, are merged, on the right.

The procedure in [165] is realized in two steps. Initially, a mesh with the same topology as the object is computed. Then, the geometrical accuracy is recovered with an adaptation of the mesh itself to the registered data. In the first step, the parts of the mesh in the overlapping regions are removed and new edges that connect the vertices in these regions are computed. As the procedure can create small triangles in the border regions, a further step performs their elimination. Although the resulting mesh would be acceptable from the topological point of view, its geometry can present artifacts. Hence, for each vertex the set of vertices that lie in its spherical neighborhood in the original meshes is considered for estimating the normal at the vertex as the average of the normals of the selected vertices. Then, the vertex is moved in the average of the intersections between the computed normal and the original meshes.

Among the other techniques for the fusion problem, an interesting approach is presented in [126]. The integration of the information of volumetric images is here performed using the wavelet transform: the original images are decomposed in their components (high and low frequency), integrated in this domain and then composed back.

2.3 Optimization

A 3D model can be used in different applications. Depending on the features of the final application, a post-processing of the model can be required. For example, virtual reality applications usually requires models composed by a number of polygons as small as possible, while if the model have to be used in a movie it can be necessary to modify

the model for a better inclusion in a scene.

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

- conversion
- compression
- watermarking
- animation
- morphing

The conversion of a model is applied to obtain a representation of the information in a format suitable for the elaboration in other applications. A common conversion operation is the triangulation of the model's surface [79]. An interesting aspect of the conversion is the re-parameterization. In [101] it is presented a multiresolution method for the parameterization of a triangular mesh. The original mesh is simplified (in $O(N)$ levels where N is the number of vertices) till to obtain a mesh with few polygons that is used as base for the parameterization. The resulting description allows a reformulation of the mesh, in which the distribution of the triangles can be made more regular or the connectivity can be forced constant and a priori chosen.

The objective of 3D compression is allowing the description of larger and more complex models using a given budget of resources. This can be useful, for example, for models used in network applications. The compression can be realized using techniques utilized for the one-dimensional and two-dimensional cases as: wavelets, entropy coding, and predictive coding. Other approaches take advantage of the properties of 3D models, as: Edgebreaker [143], Subdivision Surfaces [121], and triangle strips [55]. Current techniques for 3D compression can be classified into three categories:

- Mesh-based approaches, that realize the compression considering the topological relation among the triangles of the polygon mesh representing an 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, the compression techniques try to minimize the loss of perceptible information with respect to a constraint that determines the quantity of resources usable (i.e., the maximum number of triangles).

The *watermarking* [138] is a procedure that is correlated to the compression. In general, it is a technique used to modify the representation of the object inserting information that is not perceptible. Only a particular procedure allows the extraction of this information, allowing the recognition of the model. Each model can be then marked with a code and the information can be used to recognize it, for example for author rights reasons:

The most critical feature of a *watermarking* procedure is its robustness. It has to resist to the common manipulative operations, such as scale variation, resampling and segmentation.

The *animation* and *morphing* are techniques to manipulate graphical objects commonly used in the field of computer graphics. The *animation* of a 3D model is obtained associating a motion law (as a function of time) to the surface points. However, in practice, it is not possible specifying the motion law for each point of a model. Instead, a very common technique uses a skeleton structure for the motion mapping. This structure is, generally, composed by hierarchically connected segments that have to respect geometrical constraints. Each point of the surface is correlated to one (or more than one) segment and the animation is obtained just as an effect of the skeleton animation. A hierarchical representation of the surface can improve the animation of the model: the computation of the skeleton can be obtained using a low resolution model [101] [100].

The *morphing* is the visual effect that is used to transform the graphical representation of an object in the representation of another object. The morphing is realized by

the computation of a smooth time function that relates the parameters of two different models. In order to obtain a better visual effect, only the coefficients that code for the same features have to be related. For example, 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 the realization of the morphing. In fact, as the most representative features of the objects should be present at large scale, the transformation can be performed using mainly the coefficients of the layers at low resolution.

2.4 Summary

The digitization of a real object is a complex operation, used in many fields with different aims. The real cases are characterized by a large variability of the conditions in which the digitization has to be performed. Although the creation of a 3D model can result from a very variegated sequence of activities, it can be described, at high level, as a well defined sequence of steps. These steps do not depend on the particular application.

The acquisition is the first phase and it is performed with the aim of collecting information about the features of the acquired object. The reconstruction phase puts together the data collected and computes a single object description. The optimization phase transforms the model reconstructed in a format more suited to the application that will make use of it.

The core of this thesis is related to the reconstruction phase. In particular, two multiresolution paradigms HRBF and HSVR are presented and analyzed in depth. In order to better explain the features of these paradigms, in the next chapter the class of problems that they are able to solve is presented, in a more general context with respect to the surface reconstruction.

3

Regression

The prediction of values of variables from observations is known in statistical literature as *regression*. This problem has been studied in several disciplines belonging to the computer science area and applied mathematics in general. In particular, the prediction of real valued variables is an important topic for machine learning. When the value to be predicted is categorical or discrete, the problem is called classification problem. Although in machine learning most of the research works have 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. Functional prediction, real value prediction, function approximation and continuous class learning are examples of the several names used to identify the regression problem. In this chapter, the name *regression* is used, because it is the historical one.

More formally, the regression problem can be defined as follow:

> Given a set of *training* 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 describes the dependency between x and y. The objective 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) never used in the solution of the regression problem.

It should be noted that the regression problem is more general than the problem in which the training samples are assumed to be noise free (for example, interpolation problems). Typically, the techniques developed for noise-affected data can perform good solution also in the case where the absence of noise is assumed. Vice versa the techniques developed for the noise free case are not able to filter the noise and the solution tends to reproduce the noise of the training samples. In this case, typically, the solution does not perform a good generalization. When the solution is characterized by a low error on the training samples and a high error on the others data, it is called overfitting solution. The techniques for regression can be considered more robust because they are more suitable for noise filtering and to avoid the overfitting problem.

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 allows the extraction of knowledge automatically. In fact, as the solution allows to predict the output corresponding to an input value never used in the configuration, it can be used to discover new expertise about a considered domain.

Many techniques have been developed for the solution of regressions problem. In the following, a review of some popular regression techniques is presented.

3.1 Parametric and 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 too tight choice for the parametric family can have the collateral effect of excluding the function that actually generates the dataset (which would be the "true" solution to the regression problem), this approach offers the advantage of a simpler solution (i.e., it is less computationally expensive than other methods). Simple linear regression in statistical analysis is an example of 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:

$$y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \cdots + \beta_d x_{i,D} + \epsilon_i, \quad i = 1, \ldots, n \tag{3.1}$$

The first subscript, i, denotes the observations or instances, while the second denotes the number of input variables. There are $D + 1$ parameters, β_j, $j = 0, \cdots, D$ to be estimated. The term ϵ_i represents the errors of the solution for the sample i, namely $\epsilon_i = y_i - \hat{y}_i$, where $\hat{y}_i = \hat{f}(x_i)$, the value of the solution (the hyperplane) for x_i. .

In the parametric model the structure of the function is given, and the procedure estimation regards just the parameters, $\{\beta_j\}$, according to a fitting criterion. Generally, the criterion used is based on the minimization of an error (or loss) function on the training points. A typical approach is the minimization of the sum of squares difference between the predicted and the actual value of the examples (least squares criterion).

When the distribution from which the training data have been sampled is known, the parametric family of the solution can be correctly chosen. In this case, the parametric methods perform an accurate estimation. Hence these methods can be applied in the case where there is a priori knowledge about the application domain. When, on the contrary, the a priori knowledge is poor or not available, a non-parametric approach is generally preferable.

The non-parametric approach aims to find a general structure able to model a large set of functions, requiring only very weak assumptions on the distribution underlying the data. In this case the solution is not found as optimization on a set of parameters but as a function of the training points. These methods are generally based on the use of locally weighted average of the training data to solve the regression problem. An example of non-parametric approach is the locally weighted regression that will be presented in the section (3.3).

The choice of the possible functions set in which the solution has to be searched is known as model selection problem. It is a critical task due to the *bias-variance dilemma*. 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: $\mathrm{Var}(M) + \mathrm{Bias}(M) + \sigma_\epsilon$.

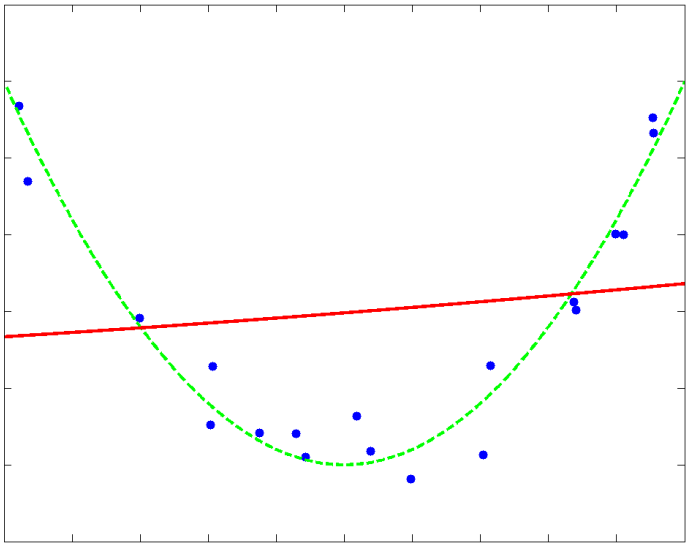


Figure 3.1: The dot line line represents the target function, the solid line is a solution computed by a linear model.

The first term, $\mathrm{Var}(M)$, 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, $\mathrm{Bias}(M)$, describes how the best approximation achievable by the model is a good estimation 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. 3.1), the variance can be small (the solution can be robust with respect different training sets), but the bias is large (the best, in some sense, solution for the model is not a good estimation of the regression function). On the contrary if the model is able to perform non linear solutions (fig. 3.2), the variance can be large (as the solution tends to overfit the data, it will change for different training sets), but the bias is small (as the model is able to compute a quadratic polynomial, then a good estimation 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 parametric and non-parametric approaches mainly

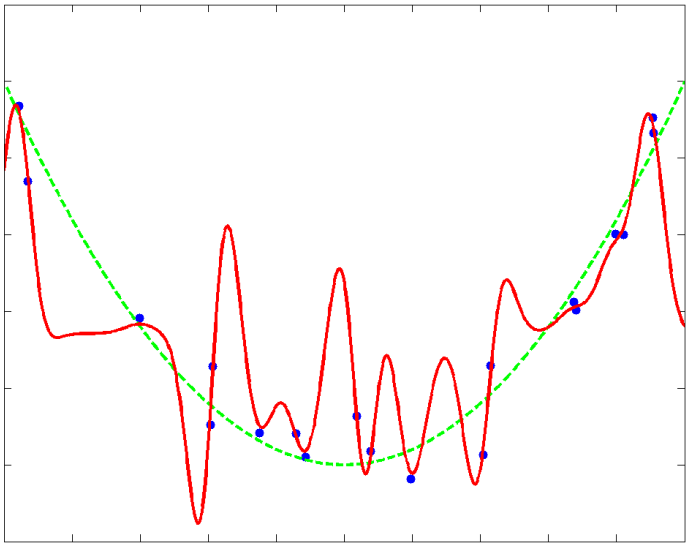


Figure 3.2: The dot line represents the target function, the solid line is a solution computed by a model with small bias and large variance.

using two methods: local methods, that base the search of the solution by using local averaging on the data, and global methods, in which the solution is found solving a global optimization problem.

3.2 Instance-based regression

Instance-based approach is a family of methods that instead of configuring a model for explicit generalization uses a strategy for the prediction based only on the direct comparison of new instance 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 feature, such methods are called also *lazy learning* algorithms. This class of algorithms is derived from the *nearest neighbor classifier* [41]. They were initially devoted to the classification [45][8] and then they have been extended to the regression [93] case.

In the instance-based techniques for regression, the examples are composed by a set of input variables, which values are either categorical or numerical, and the output

variables are continuous. The predicted label value for an instance is computed as a function of the training set elements. For example, the *nearest neighbor method* considers the output variables of the most similar training samples (called neighbors) of the new instance, and it compute the values of the output variable of the new instance as a function (typically the average) of the output variables in this 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 the most similar training set elements:

$$\hat{f}(x) = \sum_{x_j \in T_i} \frac{S(x, x_j)}{\sum_{x_j \in T_i} S(x, x_j)} y_j \tag{3.2}$$

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

This method performs good approximations only for a sufficiently large training set and for locally linear regression functions. 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 [8].

The complexity of this class of methods depends on the number of nearest neighbours. The number of nearest neighbours for a specific problem can be found using different heuristic techniques as cross-validation (see Glossary 6.2.3) [96].

3.3 Locally weighted regression

Locally weighted methods [12] belong, as the instance-based ones, 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 closer training set elements, the locally weighted methods perform the prediction using a locally estimated model. The local models are generally linear or non-linear parametric functions [12]. These methods are also based on the weighting of the data for giving more importance to relevant examples and less importance to irrelevant 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 the new instance and each training point. The functions used to weight the training points contribution are called *kernel*; one of the most used is the Gaussian function. For example the prediction for the instance x_t can be obtained using the Nadaraya-Watson estimator as:

$$\hat{f}(x) = \sum_i \frac{y_i K_\sigma(x_i, x)}{K_\sigma(x_i, x)} = \sum_i \frac{y_i e^{\frac{-||x_i - x||^2}{\sigma^2}}}{e^{\frac{-||x_i - x||^2}{\sigma^2}}} \tag{3.3}$$

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:

$$L = \sum_{i=1}^{n} E(\hat{f}(x_i), y_i) \tag{3.4}$$

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:

$$L(x) = \sum_{i=1}^{n} [(\hat{y} - y_i)^2 K(x_i, x)] \tag{3.5}$$

It can be shown that the best estimate $\hat{y}(x_t)$ that will minimize the loss function $L(x_t)$ is $\hat{y} = \hat{f}(x_t)$, where $\hat{f}(x_t)$ is computed as in (3.3).

Different weighting functions can be applied to this approach [12] and this choice can produce very different performances. Locally weighted methods have shown higher flexibility and interesting properties, such as smoothness and statistical analyzability, 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, a *k-d tree* data structure [12] can be 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.

3.4 Rule induction regression

Rule induction regression methods have been developed for studying regression approaches 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 [174]. Rule induction models have been initially presented for classification problem [174] and then extended for regression [175].

These approaches have features similar to those of *decision tree* [28], as both the models induce a recursive subdivision of the input space. The rule induction algorithms, generally, provide models with better explanatory capabilities since the rules that they found are not mutually exclusive. Furthermore, the rules are more compact and predictive than trees [175]. Regression trees show better performances when there are many higher order dependencies among the input variables [68]. They are extremely suited to find the relevant input variables in high dimensional cases when only a small subset of them is meaningful. A disadvantage of regression tree models is that partitioning of the input space can cause discontinuity 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:

$$\text{if } x \in R_k \text{ then } \hat{f}(x) = a_k = \text{median}\{y_i^k\} \tag{3.6}$$

where R_k is a region of the input space, c_k is a constants, and $\{y_i^k\}$ is the set of training points that lie in the region 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 in the region R_k. This is a common choice, as the median is the minimizer of 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 at 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 for eliminating the nodes that cause overfitting.

In regression tree, a single partition 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 disjoint: a single sample can satisfy several rules. Hence, a strategy to choose the rule to be applied from those that are satisfied is needed. In [174], 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 that is satisfied is selected:

$$\text{if } h < k \text{ and } x \in R_h \text{ and } x \in R_k \text{ then } \hat{f}(x) = c_h \tag{3.7}$$

The rule-based regression model can be built, as in 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 it covers 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 [118] [40].

In [175] a rule-induction regression algorithm based on the classification algorithm *Swap-1* [174] 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 of predicting the values of the output variables of the examples that satisfy the rule condition. Each time that 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 in categorical ones. This is realized using a variant of *K-means* algorithm [91]:

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

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 [175].

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

3.5 Projection pursuit regression

The local averaging methods suffer of poor accuracy when the input space has a large numbers of attributes (*curse of dimensionality*). In fact, when a training set is distributed in a space, the density of the set decreases exponentially with the increasing of number of dimensions [69] [77]. In practice, this means that in higher dimension the number of training points is never enough in order to compute local averaging.

The Projection Pursuit Regression approach [95] is more suited for 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 modelled in the previous steps. The model is represented by the sum of the smooth approximation S_m determined at each iteration:

$$\hat{f}(x) = \sum_{m=1}^{M} S_m(\beta_m^T \cdot x) \tag{3.8}$$

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 search of the β_m, the vector of parameters, at each iteration and the a priori choice of the smooth functions as well. The β_m vector is found in accordance with a fitting criterion such that the minimization of a loss function is achieved. In particular β_m has to maximize the following:

$$\beta_m = \max_{\beta} R(\beta) = \max_{\beta} 1 - \frac{\sum_{i=1}^{n}(r_i - S_m(\beta^T \cdot x_i))^2}{\sum_{i=1} r_i^2} \tag{3.9}$$

where r_i represents the error of i-th training point at m-th iteration, $r_0 = y_i$. 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 [95][142][141]. In [142] the performances of the algorithm are compared for three different smooth functions: smoothing spline, super-smoother, and polynomial. Also smooth functions based on local averaging of the residuals [69] can be used.

3.6 Multivariate Adaptive Regression Splines

As mentioned in Section 3.4, the tree-based recursive partitioning regression suffers of the problem of lack of continuity in the partition boundaries, which affects the accuracy. Furthermore the tree methods are not able to provide good approximation for some classes of target functions (e.g., linear or additive functions). Multivariate Adaptive Regression Splines (MARS) addresses these two problems of the recursive partitioning regression to increase accuracy [68]. The solution computed has the following form:

$$\hat{f}(x) = c_0 + \sum_{m=1}^{M} c_m B_m(x) \tag{3.10}$$

where c_i is a constant and B_m is a basis function. 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.

The MARS set of basis functions is:

$$\mathbb{B} := \{[x^j - \tau_i^j]_+, [\tau_i^j - x^j]_+ | \, \tau_i^j = x_i^j, j \in \{1, 2, \cdots, D\}, i \in \{1, 2, \cdots, n\}\} \tag{3.11}$$

where 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.

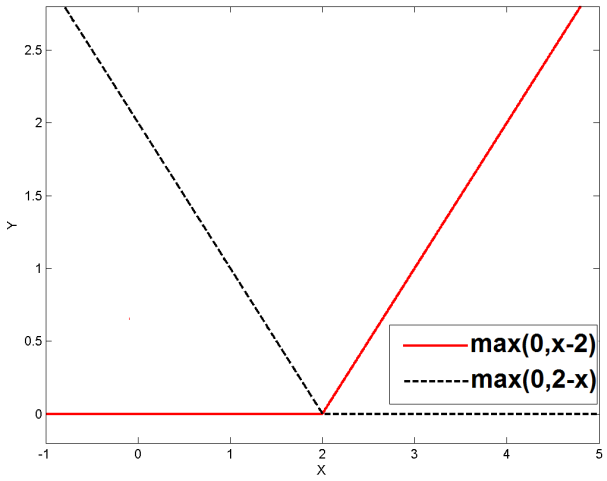


Figure 3.3: Two examples of hinge functions.

Hinge functions are a key part of MARS models. In fig. 3.3 are reported two examples of hinge functions. A pair of hinge functions as that in fig. 3.3 is called *reflected pair*. As the hinge function is zero for a part of its range, it can be used to partition the data into disjoint regions, each of which can be processed independently.

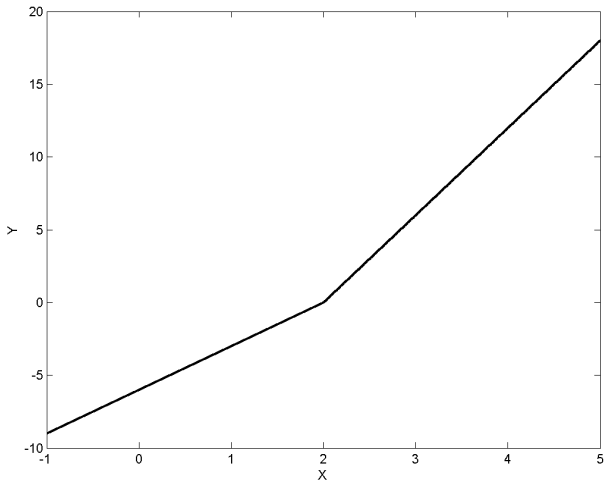


Figure 3.4: Piecewise linear function. It is defined as $f(x) = 6[x-2]_+ - 3[2-x]_+$

In particular the use of hinge function allows building piecewise linear and non-linear functions (fig. 3.4). MARS operates by constructing reflected pairs (as those in fig. 3.3) knots at each training points for each input variables.

The MARS algorithm presented in [68] is composed by two procedures:

Forward stepwise The basis functions are searched starting by the constant basis function, the only present initially. At each step, the split that minimize a given approximation criterion among all the possible splits on 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 estimation. In order to choose the best estimated model a cross-validation technique can be applied.

In [162] it is proposed a variation of the MARS algorithm in which the backward procedure is not used. In substitution, a penalized residual sum of squares for the

forward procedure is introduced, and the problem is reformulated as a *Tikhonov regularization problem.*

3.7 Artificial Neural Networks

The Artificial Neural Networks (ANN) is a family of models belonging to the class of parametric approaches [22] [52] [81] [136], which have been inspired by the computational model of human brain neural network. Among the models of the 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 scope:

$$B(z) = \frac{1}{1 + \exp(-z)} \tag{3.12}$$

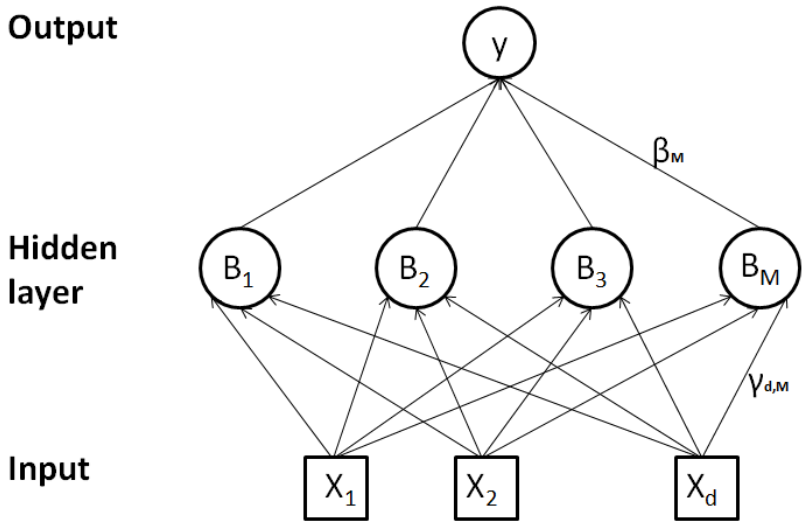


Figure 3.5: A scheme of single hidden layer feedforward neural network model

In fig. 3.5, a scheme of the MLP model is depicted. The MLP processes the data layerwise: each layer receives the input from the output of the previous layer and provides the input to the next layer. 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 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 by linear neurons, while the intermediate layers, called hidden layers, are composed by non-linear (commonly logistic) neurons. The scheme in fig. 3.5 represents a MLP with 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 [86]. However the addition of extra hidden layers may allow to perform a more efficient approximation with fewer units [22].

The output of the network of fig. 3.5 is computed as:

$$\hat{f}(x) = \sum_{m=1}^{M} \beta_m B(\gamma_m^T \cdot x) \tag{3.13}$$

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 in fig. 3.5. For example, in recurrent neural networks the information flow is allowed to loop back to previous layers.

The number of the hidden units of a network determines the trade-off between bias and variance of the model (3.1). More hidden units are used in a network, better the training data are fit. However if a too large number of units is used, the model can show overfitting on the training data, determining poor predictive performance.

The parameters of a neural network are, generally, found minimizing a loss function (as in 3.3) that measures the difference between the network output and the values of the output variables. The loss function usually used is the squared one.

Since MLP output function 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 an 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 for the computation of the gradient, due to its computational efficiency, is the *backpropagation* algorithm [146]. 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 and the recursive application of the above mentioned property allows to propagate the computation of the gradient with respect to all 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 step of length η in opposite direction with respect the gradient of the loss function. The choice of the value of η, called learning rate, is generally, difficult. A too small value of η can determine many steps to reach a minimum, vice versa a too large value can determine oscillating behavior in which a minimum is never reached. To speed up the convergence, the η parameter varies during the learning phase and many heuristics have been proposed to choose the shape of this variation. The simple gradient descend is, generally, an inefficient method and the second order gradient approaches are preferred.

The second order methods are based on the fact that in a minimum of a function, its derivative is null. As a consequence, the Taylor expansion of the function computed with respect to the minimum will not have first order term. Hence, for points close to the minimum, the function can be effectively approximated by using only the second order derivative (and the value of the function in the minimum). It results that the vector that separate from the minimum can be computed directly as the ratio between the derivative of the function in the considered point and the its second order derivative in the minimum.

As the value of the second derivative in the minimum is not known it has to be approximated. In general, this term is the Hessian matrix of the loss function and there are several iterative techniques based on the use of its approximation as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [31] and the Levenberg-Marquardt (LM) algorithm [103] [111]. The BFGS requires that the starting parameter vector is close to a minimum to be effective. Other techniques (e.g., based on the simple gradient descent) can be applied before it in order to reach such starting point. The LM algorithm does

not suffer of this problem, but it requires more memory than BFGS as it has to store some matrices. Hence, for large networks, it becomes inefficient.

3.8 Wavelet regression

An interesting approach to perform function approximation is *wavelet regression* [46] [47] [49]. This approach derives from the unification and development of many ideas from the field of filtering theory, physics and applied mathematics. The *wavelet* transformation is a representation of a function $f(x) \in L^2(\mathbb{R})$ using a linear combination of basis function. The basis functions, called *wavelet*, 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 $\psi_{a,b}(x)$ 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 succession of approximating function $\{P_j[f(x)]\}_{j\in\mathbb{Z}}$ that converges to the function self. The function $P_j[f(x)]$ is a linear combination of functions obtained by translating a single function 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. It has been proved [110] that there exists a sequence $\{h_k\}$ such that:

$$\phi(x) = 2\sum_k h_k \phi(2x - k) \tag{3.14}$$

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:

$$f(x) = \sum_j Q_j[f(x)] = \sum_{j,k} \gamma_{j,k}\psi_{j,k}(x) \tag{3.15}$$

Then the function $f(x)$ can be obtained also as linear combination of detail functions. As for the scaling function, it has been proved [110] that there exists a sequence $\{g_k\}$ such that:

$$\psi(x) = 2\sum_k g_k \psi(2x - k) \tag{3.16}$$

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:

$$\lambda_{j,k} = \langle f, \phi_{j,k} \rangle \Rightarrow P_j[f(x)] = \sum_k \langle f, \phi_{j,k} \rangle \phi_{j,k}(x) \tag{3.17}$$

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

$$\gamma_{j,k} = \langle f, \psi_{j,k} \rangle \Rightarrow Q_j[f(x)] = \sum_k \langle f, \psi_{j,k} \rangle \psi_{j,k}(x) \tag{3.18}$$

The orthogonality 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 $\{g_k\}$ (3.14, 3.16) can be seen as the coefficients of a pair of digital filters (more precisely, a Finite Impulse Filter). These filters are then characterized by a finite number of not null 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 constraints by defining an MRA which makes use of biorthogonal basis allows obtaining two pair of filters. The first one can be used for the transformation (i.e., for computing the coefficients), while the second one can be used for the inverse transformation (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:

$$P_j[f(x)] = \sum_k \langle f, \tilde{\phi}_{j,k} \rangle \phi_{j,k}(x) \tag{3.19}$$

$$Q_j[f(x)] = \sum_k \langle f, \tilde{\psi}_{j,k} \rangle \psi_{j,k}(x) \tag{3.20}$$

The discrete wavelet transformation assumes the form:

$$f = \sum_{j,k} \langle f, \tilde{\psi}_{j,k} \rangle \psi_{j,k} \tag{3.21}$$

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

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* [156] [157] [158] 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 computational cost that becomes typically unfeasible when the dimension of the space of the problem increases. Another disadvantage of this approach is that the noise on the data can determine problems for the estimation of the coefficients. Despite the regression is based on MRA, the computation of the solution is computed using a *fine to coarse* approach. Hence to perform an efficient noise filtering, the amount of noise on the data has to be known.

3.9 Summary

The estimation of a function from a set of noise affected samples is a problem that can be addressed using many different techniques. In this chapter, a review of the main approaches presented in the literature has been given.

In general, there is not an approach that can be considered superior to the others. Each one shows good or bad performances depending on the particular application. The a priori knowledge about the application problem typically determines which approach to use.

In general, a single taxonomy for the regression methods does not exist. The different approaches can be classified using different criteria. These criteria are based on either the kind of solution found or on the kind of strategy used to find the parameters of the solution. The most used features for classifying the regression methods are:

1. parametric and non-parametric;

2. linear and non-linear;

3. global and local.

There are methods that are difficult to classify and can be considered hybrid methods. For example, the piecewise linear models perform solutions that are locally linear (i.e., the sub-solutions are linear), but the regression function is not linear.

In the following two chapters, specific regression paradigms will be analyzed in depth. They can be considered both parametric non-linear machine learning approaches. The first one, based on a particular family of neural network, is called Hierarchical Radial Basis Function model and can be considered a local approach. The second one, called Support Vector Machine, can be considered a global approach.

The global approaches compute the solution as an optimization of a functional. The local approaches perform the estimation of the parameters by using local models on the data. In general the first methods are not fast but are less sensitive to the problem of sparseness of the data that occurs when the dimension of the input space increases. On the contrary the latter allow fast and accurate solution when there is a large amount of data but are more sensitive to outliers and are not able to produce accurate solutions for high dimension of input space.

4

Hierarchical Radial Basis Functions Networks

In this chapter, a particular kind of neural network, which belongs to the perceptron family [81], is considered. The models of the perceptron family are characterized for the activation function of their neural units and their connection schema. In particular, the units with radial symmetry, generally, allow obtaining good approximation with a limited number of units. Each unit is characterized by a *radius* that defines its influence region: only the input inside the influence region produces a significant activation of the unit. The 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, the locality allows to avoid the interference of examples in distant regions during the configuration of the network parameters that can cause collinear error and may stuck the learning procedure in local minima. The networks based on these units are called *Radial Basis Functions* (RBF) Networks [120] [136] [134].

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

$$\hat{f}(x) = \sum_{k=1}^{M} w_k G(x; \mu_k, \sigma_k) = \sum_{k=1}^{M} w_k \frac{e^{-\frac{||x-\mu_k||^2}{\sigma_k^2}}}{\sqrt{\pi^D}\sigma_k^D} \tag{4.1}$$

where M is the number of Gaussians 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 the po-

sition in the input domain of the k-th Gaussian, D is the number of input variables (or input space dimension). σ_k determines the size of influence region of the k-th Gaussian: in fact, this Gaussian assumes a value significantly higher than zero in the spherical region centered in μ_k and radius proportional to σ_k. The Gaussians have unitary norm.

The learning problem can be formulated as: 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 problem is ill-posed, because the surfaces that pass through the given points are infinite. In order to choose the best approximation among the possible solutions, some constraints have to be introduced, as well as a criterion for evaluating the suitability of each solution. The smoothness of the solution is a natural constraint: for small variations of a point have to correspond to small variations of the surface. The selection of optimal parameters is a non-trivial problem as it implies a non linear optimization where there are many local minima. Some heuristics have been proposed for the simplifying the search of a good, although not the optimal, solution. The constraints and the heuristics have the aim to incorporate in the learning process of the network some a priori knowledge about the target function.

In the theory of regularization, the learning problem can be formulated introducing a smoothness term in the cost function that has to be minimized [135] [74] [180]. 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 [173] [123]. Techniques based on stochastic gradient [74] suffer of local minima, which may prevent optimal solution.

In order to find the global minimum, algorithms specialized and computationally intensive have to be used [18] [37] [75]. The solution can be searched operating an optimization on all the parameters [94] [124] [21]. Although, in principle, these techniques are able to find the global minimum, they are based on the exhaustive exploration of parameters space and then they are computationally expensive.

A different strategy, called *hybrid learning* [30] [120], derives from the observation that the parameters in (4.1) can be divided in two classes depending by their role: structural parameters and synaptic weights. The parameters that influence the Gaussians behavior, (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 for determining their values. With these techniques the position of the units $\{\mu_k\}$ can be determined by using clustering algorithms (e.g., [112]) and the number of units, M, can be chosen a priori. The parameters $\{\sigma_k\}$, define the behavior of the function in regions among the samples and they are determined using heuristics [120] [127] [134] [154]. When the structural parameters are set, the equation (4.1) describes a linear system where the unknowns are the weights, $\{w_k\}$. Although they can be computed solving the system, for networks with high number of units this solution can imply numerical or memory allocating problems. Hence a local approach for the weights estimation can be preferable.

The growing structures [70] [71] [134] are an improvement of hybrid schemes. The number of units is not a priori given, but the units are added to network one by one till a criterion is satisfied. These schemes are iterative and the estimation of good parameters, generally, requires a time consuming learning phase.

Another approach, used in *computer vision*, is based on the use of regularly spaced Gaussians placed on a lattice covering the input space [132] [149]. 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 missing details in other regions. Furthermore, the presence of overfitting would imply a waste of resources due to a use of too many units. This problem can be solved in several ways. For example, in [39] a methods based on the computation of an orthogonal model basis that allows both improving the generalization ability of the original model and reducing the number of the units is proposed. The weights of the orthogonal basis are found by an iterative minimization of a functional. 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) [25], [59]; it is based on the idea of inserting units only where they are needed without using iterative procedure to find the weights.

HRBF combines the characteristics of growing structures and considerations grounded in signal processing, allowing a fast and robust estimation of both the structural parameters and the weights of the network. Because part of the work of the thesis is related to HRBF model, it is presented in detail in section 4.1, where the concepts illustrated in [25] [59] are summarized.

The configuration of this model is performed using the entire dataset. When the dataset is not completely available at the moment of the reconstruction, i.e., the model have to be configured while the dataset is still acquired, the HRBF model cannot be applied efficiently. In order to update the already configured model with respect the new points the entire configuration procedure has to be repeated. This is generally not suitable for the cases in which the configuration of the model has to be performed in real-time with respect to the data acquisition. In section 4.2, an innovative configuration algorithm for HRBF online learning is presented [63]. It allows to achieve accuracy comparable to that of the original HRBF algorithm.

4.1 Batch HRBF

4.1.1 Gaussian Regular RBF network and Gaussian Filter

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

$$\hat{f}(x) = \sum_{k=1}^{M} w_k G(x; \mu_k, \sigma_k) = \sum_{k=1}^{M} w_k \frac{e^{-\frac{||x-\mu_k||^2}{\sigma_k^2}}}{\sqrt{\pi}\sigma_k} \tag{4.2}$$

In the following, a RBF network composed of regularly spaced Gaussian units which have the same standard deviation, σ, will be referred as regular RBF network. Since the size of the region of the domain input where each Gaussian affects the approximation is proportional to σ, this parameter describe also the scale at which the RBF operates (and σ is also termed the scale parameter). In this case, (4.2) can be reframed as the convolution of the weights sequence $\{w_k\}$ with a Gaussian function $G(x)$ (actually, with a regular sampling of G). Filtering theory concepts can then be applied to find

an efficient algorithm for computing the values of the structural parameters and the weights of regular RBF networks. In the following, some concepts for characterizing the Gaussian filter will be introduced in section 4.1.2. Then, in sections 4.1.3 and 4.1.4, the regular RBF network model will be generalized with models that, although they cannot be used in practical cases, can simplify the theoretical derivation of the HRBF algorithm, that will be illustrated in section 4.1.6.

4.1.2 Gaussian Filter

The ideal low-pass filter keeps all the components at frequency lower than $\nu_{\text{cut-off}}$ and the components at higher frequency are deleted. 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. Defining two thresholds suitable δ_1 and δ_2, $\nu_{\text{cut-off}}$ and $\nu_{\max}$ are identified by the following:

$$\delta_1 \leq |\tilde{F}(\nu)| \leq 1 \quad \text{for} \quad 0 \leq \nu \leq \nu_{\text{cut-off}} \tag{4.3}$$

$$0 \leq |\tilde{F}(\nu)| \leq \delta_2 \quad \text{for} \quad \nu_{\max} \leq \nu \leq \infty \tag{4.4}$$

These values define three bands in the frequency domain: *Pass Band*, *Stop Band* and *Transition Band*. In the first band, the frequency components are kept almost unchanged, in the second are almost deleted and in the last are attenuated progressively. The Gaussian filter (fig. 4.1) is formulated as:

$$G(x, \sigma) = \frac{1}{\sqrt{\pi}\sigma} e^{-\frac{x^2}{\sigma^2}} \tag{4.5}$$

Its Fourier transform is:

$$\tilde{G}(\nu, \sigma) = e^{-\pi^2\sigma^2\nu^2} \tag{4.6}$$

The following relations describe as σ and the frequencies in equations (4.3) and (4.4) are related:

$$e^{-\pi^2\sigma^2\nu^2_{\text{cut-off}}} = \delta_1 \quad \Rightarrow \quad \nu_{\text{cut-off}} = \frac{\sqrt{-\log\ \delta_1}}{\pi\sigma} \tag{4.7}$$

$$e^{-\pi^2\sigma^2\nu^2_{\max}} = \delta_2 \quad \Rightarrow \quad \nu_{\max} = \frac{\sqrt{-\log\ \delta_2}}{\pi\sigma} \tag{4.8}$$

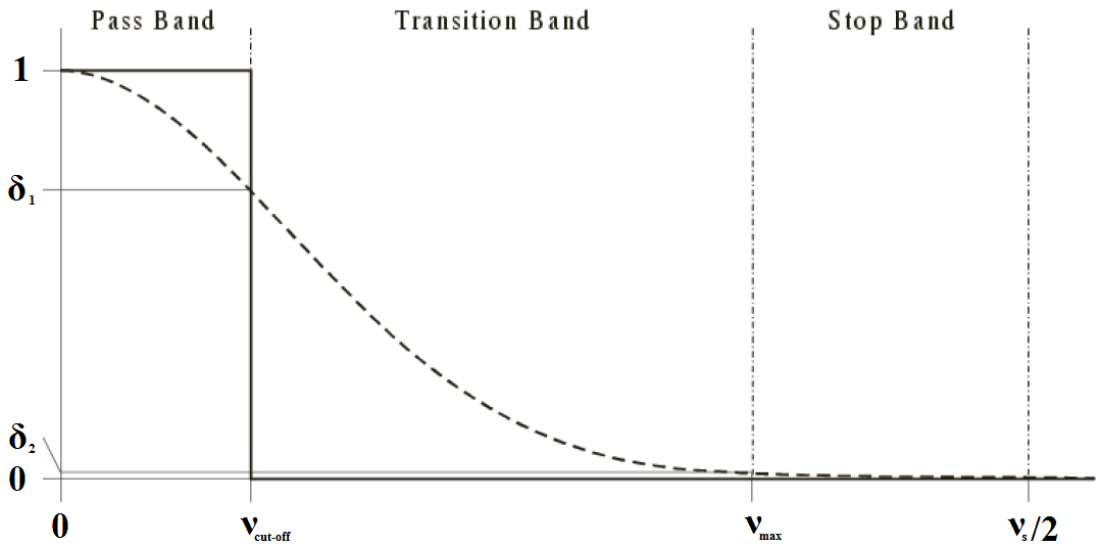


Figure 4.1: Gaussian filter

4.1.3 Regular continuous RBF network

Equation (4.2) describes a neural network that, in practical case, has to be constituted of a finite number of units. However it is possible to consider a generalization constituted of an infinite number of contiguous units, which will be referred in the following as the continuous RBF network. The structural continuity of this 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 (4.2)) and a function $\sigma(x) : \mathbb{U} \to \mathbb{R}^+$. If the distance between the Gaussians tends to zero, the equation (4.2) can be generalized by using a regular continuous RBF network in which $\mathbb{U} \equiv \mathbb{R}$ and $\sigma(x) = \sigma$:

$$f(x) = \int_{\mathbb{R}} w(c)\, G(x - c|\sigma)\, \mathtt{d}c \; = w(x) * G(x; \sigma) \tag{4.9}$$

Due to the convolution theorem [90], (4.9) implies:

$$\tilde{f}(\nu) = \tilde{w}(\nu)\tilde{G}(\nu; \sigma) \tag{4.10}$$

Here, given a function $w(x)$, the function $f(x)$ is obtained. In the regression problem, however, it is the inverse. In the following, a rule for obtaining a function $w(x)$ that provide a good estimation of $f(x)$ is investigated. If $w(x)$ were replaced by the function

$f(x)$ itself, the function $\hat{f}(x)$ would be obtained:

$$\hat{f}(x) = \int_{\mathbb{R}} f(c)\, G(x - c|\sigma)\, \mathtt{d}c \; = f(x) * G(x;\sigma) \tag{4.11}$$

and in the frequency domain:

$$\tilde{\hat{f}}(\nu) = \tilde{f}(\nu)\tilde{G}(\nu;\sigma) \tag{4.12}$$

From the (4.12) it is clear that $\hat{f}(x)$ is 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 Gaussian filter. If $f(x)$ does not contain significant frequency components over ν_M, in order to obtain a good approximation of $f(x)$, the parameter σ in (4.7) have to be set such that the following constraint is satisfied:

$$\nu_{\text{cut-off}} > \nu_M \tag{4.13}$$

4.1.4 Regular discrete RBF network

In real cases there is a limited number, N, of samples. Now it is considered 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:

$$\hat{f}(x_i) = \sum_{k=1}^{N} f_k\, G(x_i; x_k, \sigma)\Delta x \;=\; \frac{\Delta x}{\sqrt{\pi}\,\sigma} \sum_{k=1}^{N} f_k e^{-\frac{(x_i - x_k)^2}{\sigma^2}} \tag{4.14}$$

Equation (4.14) can be extended on the whole real line using the following interpolation:

$$\hat{f}(x) = \sum_{k=1}^{N} f_k\, G(x; x_k, \sigma)\Delta x \;=\; \frac{\Delta x}{\sqrt{\pi}\,\sigma} \sum_{k=1}^{N} f_k e^{-\frac{(x - x_k)^2}{\sigma^2}} \tag{4.15}$$

In (4.15), also the Gaussian filter is sampled, with sampling frequency $\nu_s = \frac{1}{\Delta x}$. This fact, for the sampling theorem, introduces another constraint:

$$\nu_{\max} < \frac{\nu_s}{2} \tag{4.16}$$

Relations (4.7, 4.8) can be modified as follows:

$$e^{-\pi^2\sigma^2\nu^2_{\text{cut-off}}} \;=\; \delta_1 \tag{4.17}$$

$$e^{-\pi^2\sigma^2\nu_{\max}^2} = \frac{\delta_2}{2} \tag{4.18}$$

δ_2 has been halved because, when $\nu_{\max} = \nu_s/2$, the Gaussian receives the same contribution from the main lobe and from the closest replica (aliasing effect). Other Gaussians contribution can be considered, but they are 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 $\nu_M^\star$, where:

$$\nu_M^\star = \sqrt{\frac{\log\ \delta_1}{\log\ \delta_2/2}}\ \frac{\nu_s}{2} \tag{4.19}$$

Equation (4.19) is comparable with the sampling theorem. The latter asserts that the maximal frequency that can be reconstructed by a sampled signal is equal to the half of the sampling frequency. As $\delta_1 > \delta_2$, the maximum reconstructed frequency by the model in (4.15) will be under that indicated by the Shannon theorem.

If δ_1 is set to $\sqrt{2}/2$, according with the common practice (attenuation of 3 dB), and δ_2 is set to 0.01, the following is obtained from (4.19):

$$\nu_s = 7.829\ \nu_M^\star \tag{4.20}$$

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, the ratio tends to two. In this way the quality of the reconstruction decreases. In particular if δ_2 increases there will be an increase of aliasing effect; if δ_1 decreases there will be greater attenuating of higher frequency components.

Using the above stated relations and setting the maximal frequency $\nu_M^\star$, it is possible to configure a regular discrete RBF network, setting the number, M, the positions, $\{\mu_k\}$, and the width, σ of the Gaussians (compatibly with the sampling frequency ν_s). The weights $\{w_k\}$ will be derived from the subsampling of the input data.

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 (4.20)). 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: $\mu_k = x_k,\ M = n$. From the constraints about the frequencies and set $\Delta x = 1/\nu_s$, it can be derived that the σ parameter cannot be less than $1.465\Delta x$. The weights correspond to the values assumed by the sampled points: $w_k = y_k$. It should be noted that for matching a given target maximum frequency (below $\frac{1}{8\Delta x}$) a suitable subsampling of the data set can be applied. In this case, the number of Gaussians, their centers, and their width have to be changed accordingly to the subsampling.

4.1.5 Hierarchical approach

The previous technique is inefficient when the function f presents different frequency content in different domain regions. The presence of high frequency details in just some parts of the domain would require the use of 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 objective of realizing a general approximation of the function, $a_1(x)$. The value of the cut-off frequency of this network is chosen relatively small, say ν_1. For the estimation 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 residual set r_1 can be defined for formalizing the information that is not described by a_1:

$$r_1 \ = \ \{y_i - a_1(x_i)\} \tag{4.21}$$

A second network characterized by a cut-off frequency ν_2, $\nu_2 > \nu_1$, is used for obtaining an approximation of the residual r_1. In this way a network that realizes the function $a_2(x)$ is obtained. 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. The representation at l resolution is given by the sum of the output of the first l networks.

4.1.6 Generalization: non regular noise-affected data

The procedure described in section 4.1.4 supposes regularly spaced noise-free data sampled in the same position of the Gaussian centers. If this hypothesis is not verified, a technique for the estimation of the value assumed by the sampled function, f, in the Gaussian centers is needed. The presence of noise, however, is a minor problem, because the Gaussian filter attenuates the high frequency components of the dataset (which are mainly due to noise). It should be noted that this estimation is not the approximation that will be carried out by the network, but only a rough estimation of the values required by the learning procedure and missing in the dataset. To stress this concept, this rough estimation will be indicated using $\check{f}$.

The estimation of $\check{f}_k = \check{f}(\mu_k)$ will be performed locally: a subset A_k of the dataset will be selected using the distance from μ_k as selection criterion; then the value in A_k will be combined for providing the value f_k. The region from which A_k belong to is called the *receptive field* of the k-th Gaussian. The simple criteria used for the samples selection are mainly two. The first one is based on the number of elements used in the estimation: for each Gaussian the n samples closer to its center, μ_k are selected. The second one is based on the distance: for each Gaussian every point closer than a given threshold, ρ, to μ_k is selected. Both the methods have advantages and disadvantages.

In [25], an estimation criterion [12] based on a weighted average of points belonging to the set A_k, is used, where the weight of each sample decreases with the distance of the sample from the center of the Gaussian. In particular, the weight function used is a Gaussian function. The implicit assumption is that the points close to the center are able to provide a reliable estimation with a normal probability distribution. The estimation in the Gaussian center, μ_k, is hence:

$$\check{f}(\mu_k) = \frac{\sum_{x_r \in A_k} y_r e^{-\frac{(x_r - \mu_k)^2}{\sigma_w^2}}}{\sum_{x_r \in A_k} e^{-\frac{(x_r - \mu_k)^2}{\sigma_w^2}}} \tag{4.22}$$

where the estimation set, A_k, is:

$$A_k = \{x_r \mid ||x_r - \mu_k|| < \rho\} \tag{4.23}$$

The parameter σ_w is equal to σ of the considered RBF network (although $\sigma_w < \sigma$ would be more precautionary, as it avoid to attenuate too much the high frequency

components of the dataset), while ρ is set equal to Δx, which is the spacing between two consecutive Gaussians ($\mu_r - \mu_{r-1} = \Delta x$, $\forall r$).

Furthermore, the selection of a local dataset allows to evaluate if the considered Gaussian is required for the approximation. In fact, if in A_k the value of f is below the measurement noise, ϵ, no Gaussian need to be inserted (otherwise, overfitting may be obtained). Hence, the k-th Gaussian is inserted (i.e., w_k is set to $\Delta x\, f_k$) if and only if the average value of the output variable, R_k, in the spherical neighborhood of μ_k is over a given threshold ϵ, namely if the following is satisfied:

$$R_k = \frac{\sum_{x_r \in A_k} ||y_r||}{|A_k|} > \epsilon \tag{4.24}$$

Otherwise, w_k is set to zero. Since in a hierarchical scheme, the output variable used to estimate the weights is the residual, R_k is termed *local residual*. Besides, if in A_k there are too few samples to provide a reliable estimation an alternative criterion have to be applied for estimating w_k. A simple approach can be set w_k to zero, but also an iterative procedure that increases the value of ρ until A_k is enough populated (e.g., if $|A_k| < \alpha$, for a suitable α) can be devised.

4.1.7 Batch HRBF configuration algorithm

The concepts and the relations illustrated in the previous sections allow to design a non iterative algorithm for setting up the parameters of an HRBF networks in order to approximate a function from a finite sampling of the function itself. This procedure exploits the knowledge of the entire input data set and adopts local estimates to setup the network parameters. This produces a very fast configuration algorithm, which is also suitable to be parallelized.

The HRBF configuration algorithm can be summarized as following:

1. Given a dataset $\{(x_i,\, y_i) | i = 1, \ldots, n\}$.

2. A set of L cut-off frequencies is chosen, $\{\nu_l |\ l = 1, ..., L\}$. Each frequency corresponds to a layer of RBF.

3. From the set of frequencies, the values of σ and Δx for each layer are computed:

$$\sigma_l = \frac{\sqrt{\log 2/2}}{\pi \nu_{\text{cut-off}}} \tag{4.25}$$

$$\Delta x_l = \frac{\sigma_l}{1.465} \tag{4.26}$$

4. The first residual is set as: $r_0 = \{r_0(x_k) = y_k\}$

5. For $l = 1, \ldots, L$, the l-th layer is considered.

6. The Gaussians centers of the l-th layer, $\{\mu_{l,j} | \mu_{l,j} - \mu_{l,j-1} = \Delta x_l\}$, are set such that they cover the range covered by the dataset. The number, M_l of the Gaussians of the l-th layer can be set as $M_l = \left\lceil \frac{\max(x_i) - \min x_i}{\Delta x_l} \right\rceil$.

7. For each Gaussian, j, the subset $A_{l,j}$ is selected, where the receptive field is a spherical neighborhood of the (l,j)-th Gaussian, centered in $\mu_{l,j}$, with radius proportional to Δx_l.

8. If (4.24) is satisfied, the coefficient $w_{l,j}$ is computed as:

$$w_{l,j} = \Delta x_l \frac{\sum_{x_r \in A_{l,j}} y_r \; e^{-\frac{(x_r - P_{l,j})^2}{\sigma_l^2}}}{\sum_{x_r \in A_{l,j}} e^{-\frac{(x_r - P_{l,j})^2}{\sigma_l^2}}} \tag{4.27}$$

9. The set $a_l = \{a_l(x_i) = \sum_j w_{l,g} G(x_i; \mu_{l,j}, \sigma_l)\}$ is computed.

10. The residual is computed as: $r_l = \{r_{l-1}(x_i) - a_l(x_i)\}$

11. If $l < L$, jump back to step 5.

The output function results:

$$\tilde{f}(x) = \sum_{l=1}^{L} \sum_j w_{l,j} G(x; \mu_{l,j}, \sigma_l) \tag{4.28}$$

L reconstructions with different resolution are then available:

$$\tilde{a}_l(x) = \sum_{h=1}^{l} a_h(x) = \sum_{h=1}^{l} \sum_j w_{h,j} G(x; \mu_{h,j}, \sigma_h) \tag{4.29}$$

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, the first step of the algorithm above described is not always needed. For some applicative problems, the values of σ can be set directly. Furthermore, it is not required to set the number of layers a priori. In fact, other termination criteria can be used: for example, the procedure can be iterated until the residual is under a given threshold on the entire domain.

Since the hierarchical scheme requires the computation of the output of each layer in order to compute the data set for the next layer, when the width of the layer is small with respect to the extent, most of the computational cost of the output of the layer is wasted. This happens because the Gaussians function is defined for every real value. However, due to its exponential decay, the Gaussian assumes significant values only in the region close to its center. This allows to limit the contribution of each Gaussian to the output of the layer only in a suitable neighborhood of it center. This region, I_k, will be termed the *influence region* of the Gaussian. It can be formalized through its radius, τ:

$$G_{l,k} = \begin{cases} \frac{1}{\sqrt{\pi}\sigma_{l,k}} \exp\left(-\frac{||x-\mu_{l,k}||)^2}{\sigma^2}\right), & ||x-\mu_{l,k}|| < \tau_{l,k} \\ 0, & \text{elsewhere} \end{cases} \tag{4.30}$$

Summarizing, the batch HRBF network hyperparameters (i.e., the parameters of the procedure that set up the HRBF network) are reviewed in the following, with some annotations on how their value can be set up:

- the width of the weighting function of (4.22), σ_w;
- the radius of the receptive field, ρ, (4.23);
- the radius of the influence region, τ, (4.30);
- the minimum number of samples, α, for a reliable estimation of the weights;
- the set of the centers of the units of each layer, $\{\mu_{l,k}\}$;
- the set of the width of the units of each layer, $\{\sigma_l\}$;
- the number of layers, L;
- the error threshold ϵ, (4.24).

Some hyperparameters (namely σ_w, ρ, τ, and α) can be considered as hard-wired parameters, since the behaviour of the training algorithm is not very sensitive to their value. Only for very fine tuning purposes, they should be considered. The parameters $\{\mu_{l,k}\}$, $\{\sigma_l\}$, and L are related to the structure of the HRBF network. Although reasonable criteria can be devised for setting their value, for particular applications their value can be forced by the user. The error threshold ϵ depends heavily by the noise on the data and the accuracy that is required to the solution performed by the HRBF network. These three groups of hyperparameters require different levels of knowledge on the HRBF operation. In particular, the direct setup of σ_w, ρ, τ, and α requires a low level knowledge of the HRBF algorithm, while the set up of $\{\mu_{l,k}\}$, $\{\sigma_l\}$, and L requires only a grasp on the RBF theory, and the set up of ϵ requires only a priori knowledge on the dataset (i.e., on the applicative domain).

As mentioned above, $\sigma_w \leq \sigma_l$, for each layer. A reasonable choice can be: $\sigma_w = \sigma_l/2$. The hyperparameter ρ controls the width of the receptive field, and, hence, it should be proportional to the width of the units of the layer, σ_l, or to the spacing, Δx_l. Hence, reasonable choice can be: $\rho = \sigma_l$ or $\rho = 2\Delta x_l$. A too small value of ρ can make the estimation unreliable due to lack of samples, but, on the other hand, large value of ρ can make the estimation computationally expensive (and also unreliable, due to excessive averaging). The radius of the influence region affects both the accuracy and the computational cost of the training procedure. Since the value of the Gaussian is negligible for points more than 3 σ away from the center, this can be a reasonable value, although, for practical purposes, even a lower proportionality factor can provide an accurate enough approximation. The minimum number of samples required for a reliable estimation depends both from the variability of the function and from the 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 position of the units of each layer can be easily set up by covering the range of the data. In the $\mathbb{R}^D \rightarrow \mathbb{R}$ case, however, this choice can be inefficient. Some a-priori knowledge on the dataset can be exploited for limiting the placement of the Gaussians only where the data are effectively located (e.g., using the convex hull, or the alpha shape, of the dataset). The width of the Gaussian units of each layer $\{\sigma_l\}$, can be set using some a-priori knowledge on the frequency content of the function. However, when this information is missing, the width of the Gaussians can be halved at each

layer. A reasonable value for σ_1 can be the length of the interval where the data belong to. When multivariate approximation is operated, this value can be the maximum of the length of the intervals containing the input variables. The number of layers, L, can be a-priori determined considering the computational resources available (e.g., the maximum number of units), or can be adapted considering the error achieved by the last configured layer (e.g., through cross validation).

The error threshold is the most critical hyperparameter as it determines the accuracy of the approximation of the whole HRBF network. It should be set using a-priori knowledge on the dataset (i.e., the noise on the data), if available, but can be otherwise determined with cross validation. For instance, this threshold can be obtained 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.

4.1.8 Extension to the two-dimensional case

Due to the property of the Gaussian function, the extension to the multivariate case is quite easy. In particular, the two-dimensional case is considered here, since it is of interest for the applicative problem of surface reconstruction. Besides, the formal explanation is clearer and can be easily generalized. In the $\mathbb{R}^2 \to \mathbb{R}$ case, the Guassian filter results:

$$G(x;\sigma) = \frac{1}{\pi\sigma^2} e^{-\frac{||x||^2}{\sigma^2}} \tag{4.31}$$

where $x \in \mathbb{R}^2$. As $x = (x_1, x_2)$, the (4.31) can be written as:

$$G((x_1,x_2);\sigma) = \frac{1}{\pi\sigma^2} e^{-\frac{x_1^2+x_2^2}{\sigma^2}} = \frac{1}{\pi\sigma^2} e^{-\frac{x_1^2}{\sigma^2}} \frac{1}{\pi\sigma^2} e^{-\frac{x_2^2}{\sigma^2}} = G(x_1;\sigma)G(x_2;\sigma) \tag{4.32}$$

and its Fourier transform can be written as:

$$\hat{G}(\nu;\sigma) \;=\; e^{-\pi^2\sigma^2\nu_1^2}\, e^{-\pi^2\sigma^2\nu_2^2} \;=\; e^{-\pi^2\sigma^2||\nu||^2} \tag{4.33}$$

as the Fourier transform of a separable function is equal to the product of the one-dimensional transform of its components. Then, all the considerations and relations for one-dimensional case are valid also for two-dimensional one. There is only a change in step 8 of the HRBF configuration algorithm (section 4.1.7): the proportionality

constant between the estimated value of the function and the coefficient of the Gaussian, Δx_l, becomes Δx_l^2.

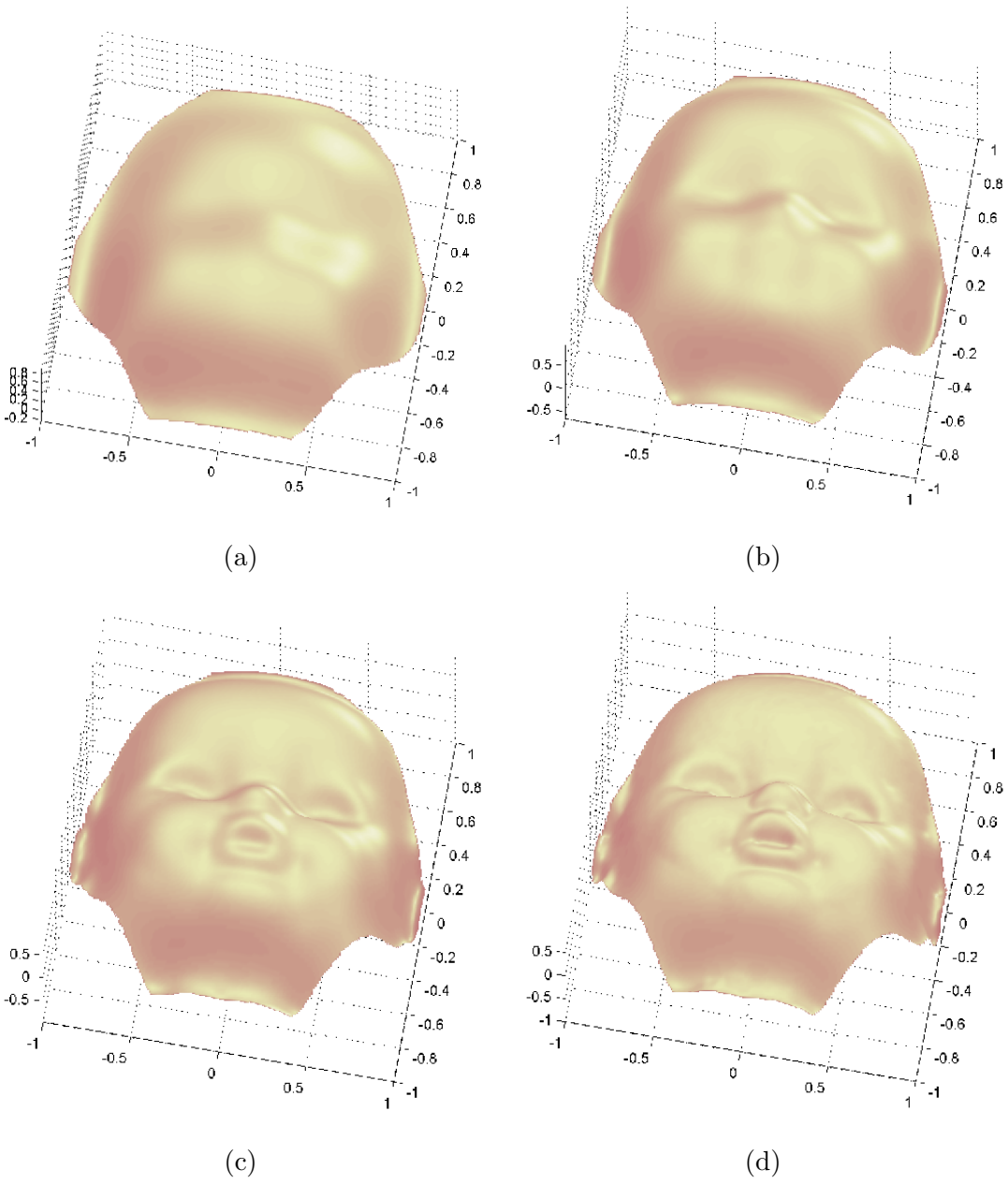


Figure 4.2: Reconstruction of a doll with HRBF model using (a) 5 layers, (b) 6 layers, (c) 7 layers, and (d) 8 layers. The hierarchical structure of the model allows to choose the number of layers that match the required resolution.

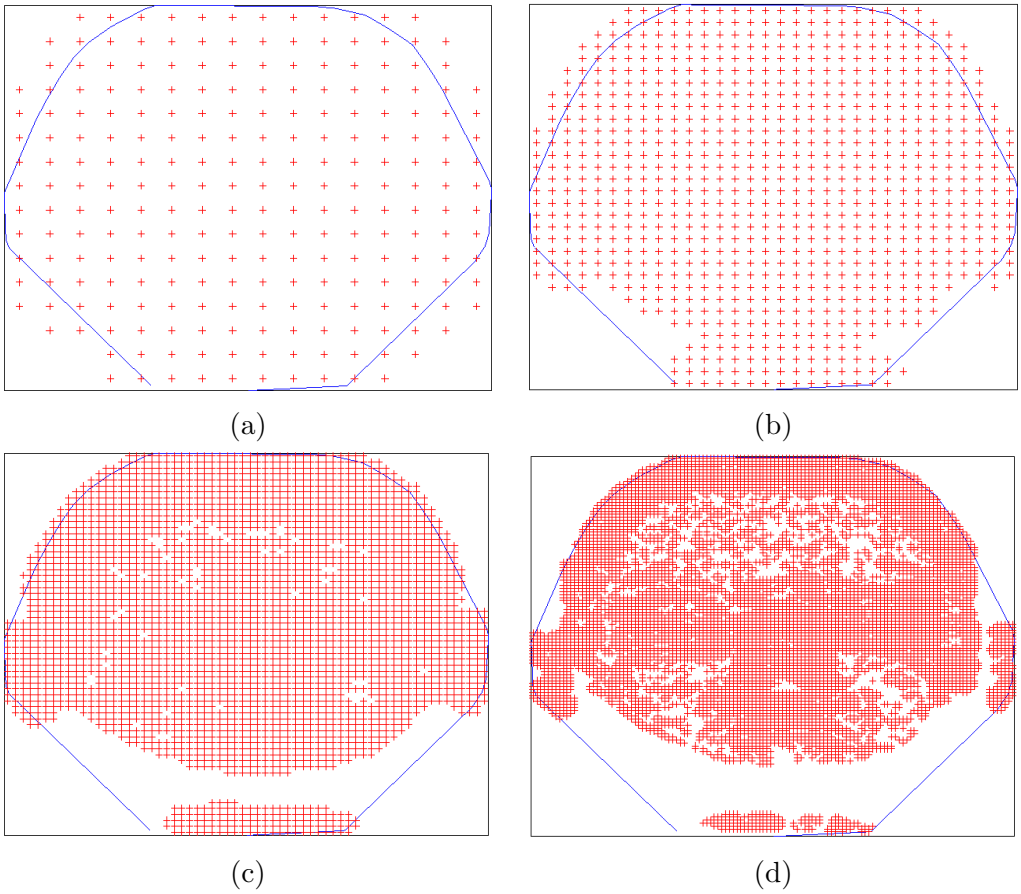


Figure 4.3: The grids of Gaussians for the 5th (a), 6th (b), 7th (c), and 8th (d) layers of the HRBF model used for the reconstruction in fig. 4.2.

In fig. 4.2 an example of reconstruction realized using the HRBF model is reported. The dataset has been acquired by the systems described in [27]. 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 scale are added. The hierarchical approach allows the multi-scale reconstruction at different levels of detail. In fig. 4.3 the grids of Gaussians (i.e., the regular lattice where the Gaussian are placed) are depicted. The centers of the Gaussians effectively used (i.e., those that have not null weight) of the level are marker in red. It can be noticed that in the last layers some Gaussians are missing due to the lack of the samples in their receptive field or because, locally, the approximation already match the required accuracy.

4.1.9 Batch HRBF network approximation properties

In [59], the approximation properties of the HRBF model have been investigated. The basis used for the regular discrete RBF has been shown to be a Riesz basis for the space it spans. This implies that the decomposition that it operates is unique and stable. Furthermore, the degree of stability (i.e., the influence of errors in the data on the approximation) depends only on the ratio $\frac{\sigma}{\Delta x}$. Since $\sigma_l = 1.465\Delta x_l$ for all the layers, the stability is the same for all the scales at which the HRBF network operates.

The approximation property holds for regular RBF networks: it can be shown that, for a large class of functions (the compactly supported infinitely differentiable functions), there exists a scale factor σ_j such that the approximation performed by the layer is arbitrarily closer to the considered function. Since the function to be approximated changes at every layer (as it is the residual left by the previous layers), the convergence of the HRBF approximation scheme can be questioned. However is has been proved that the residual becomes smaller and smaller as the layers are added to the HRBF network, and the convergence of the HRBF approximation is uniform for the compactly supported functions in $\mathcal{C}^\infty$ that have equilimited derivatives (i.e., there exists a supremum of the norms of the derivatives of function).

4.2 On-line HRBF

On-line learning algorithms compute the value of the parameters of an approximation model (such as a neural network) by considering 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 method efficiently. This is the case, for instance, of linear models, where the matrix inversion required computing the optimal value of the model's parameters cannot be computed due the excessive memory needed. Besides, the peculiarity of the application may require on-line learning. For instance, when the approximation model have to be updated due to the non-stationarity of the process that provide the data, or when a partial approximation is required during the acquisition of the training set.

For the HRBF networks, the scheme described in the previous section cannot be applied in this scenery. 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 estimate in

(4.22) becomes out of date for the first layer. Then $\check{f}(\mu_k)$ has to be reestimated for all the Gaussians by using the new data set constituted of the old data set and 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.

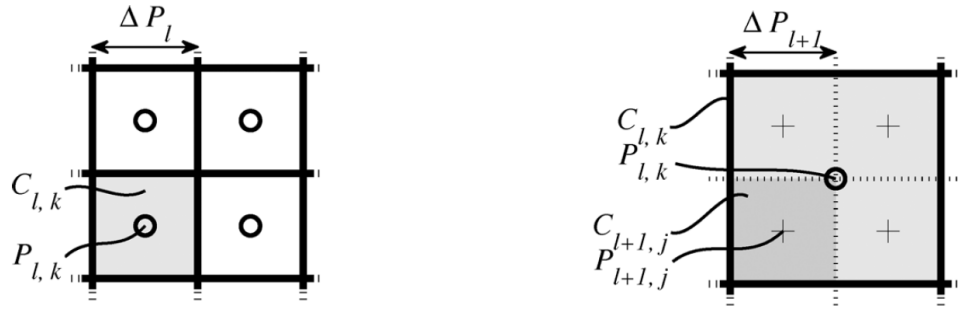
One possibility is to reconfigure the entire network computing all the parameters every time a new point is added to the input data set. This solution is computationally expensive and unfeasible for real-time configuration. To avoid this, a few approximations have to be introduced.

The most limiting factor to real-time operation is the shape of the receptive field: as the Gaussians have radial symmetry, their receptive field comes out with a spherical shape that does not allow an efficient partitioning of the data points. To overcome this problem, the receptive field is approximated with a cubic region [60]; this approximation can be accepted as far as the input space has a low dimensionality [61].

Cubic approximation allows organizing the incoming data points and the HRBF parameters into an efficient data structure. For each layer l, the input space is partitioned into non-overlapping regular boxes $C_{l,k}$, each centered in a different Gaussian center $\mu_{l,k}$. As shown in fig. 4.4, for a $\mathbb{R}^2 \to \mathbb{R}$ mapping, the input domain is partitioned into squares $C_{l,k}$, where each $C_{l,k}$ is called the *close neighborhood* of the (l,k)-th Gaussian $G_{l,k}$. The vertices of each $C_{l,k}$ are shifted of $\frac{1}{2}\Delta x_l$ with respect to the grid centers.

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 (4.27). 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



(a) (b)

Figure 4.4: 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 panel (a). The close neighborhoods tessellate 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 quad-tree decomposition, as shown in panel (b). Each $C_{l+1,j}$ has a side half the length of $C_{l,k}$, and it is centered in a Gaussian $\mu_{l+1,j}$ positioned in "+".

the $(l+1)$-th layer (*children*). This relationship, depicted in fig. 4.4(b), 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.

The configuration algorithm is structured as a sequence of steps for updating the weights followed by a single step in which residual is evaluated and Gaussians possibly inserted. These two phases, depicted in the scheme in fig. 4.5, are iterated until new points are added.

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

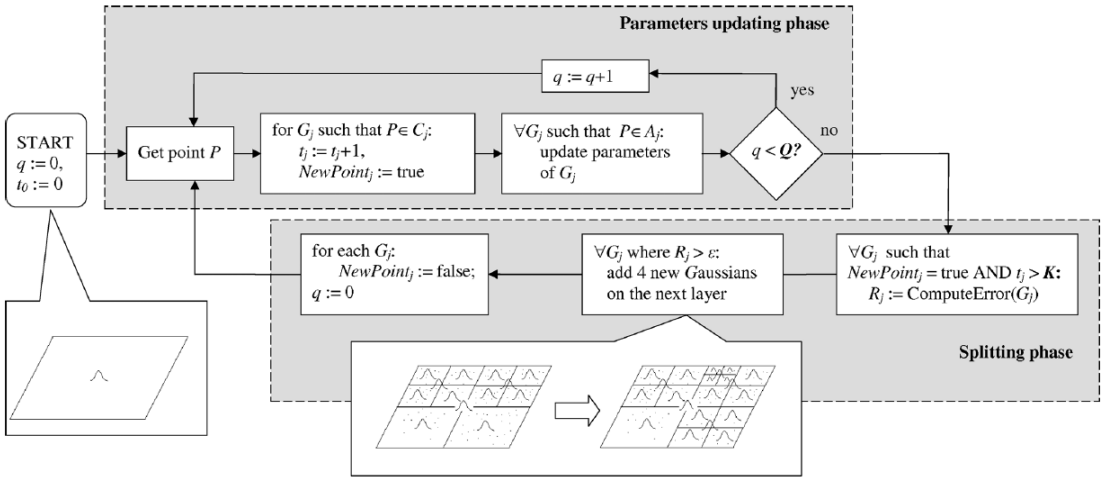


Figure 4.5: Schematic representation of the on-line HRBF configuration algorithm

4.2.1 First Learning Phase: Updating of the Parameters

When a new point x_{new} is given, the quantities $n_{l,k}, d_{l,k}$, and $w_{l,k}$ (4.27), associated to the Gaussians such that $x_{\text{new}} \in A_{l,k}$, are updated

$$n_{l,k} \;:=\; n_{l,k} + r_{l-1}(x_{\text{new}}) e^{-||\mu_{l,k} - x_{\text{new}}||^2/(\sigma_l/2)^2} \tag{4.34}$$

$$d_{l,k} \;:=\; d_{l,k} + e^{-||\mu_{l,k} - x_{\text{new}}||^2/(\sigma_l/2)^2} \tag{4.35}$$

$$w_{l,k} \;=\; \frac{n_{l,k}}{d_{l,k}} \cdot \Delta x_l^D \tag{4.36}$$

where $r_{l-1}(x_{\text{new}})$ is computed, like in (4.21), 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 (4.34)–4.36) should be recomputed for the next layer for all the (already acquired) data points inside the influence region of $G_{l,k}$. This would require recomputing the residual of the next layer and so forth up to the last configured layer.

However, this would lead to an excessive computational load, and, in the updating phase, the terms in (4.34, 4.35, 4.36) are modified only by adding the contribution of

x_{new} to $n_{l,k}$ and $d_{l,k}$. The rationale is that increasing the number of points, (4.36) tends to (4.27). The residual is then recomputed only in x_{new}, which is sufficient to obtain a good estimate of (4.27).

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

4.2.2 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, 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 candidates for splitting. Let us call J their ensemble.

For each Gaussian of J, the local residual $R_{l,k}$ is reevaluated for all the points inside its close neighborhood using the present 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. 4.4(b)].

It is remarked 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.

As a result, the parameters of an inserted Gaussian $G_{l,k}$, $n_{l,k}$, $d_{l,k}$ and $w_{l,k}$ in (4.34, 4.35, 4.36) are computed using all the points contained in its close neighborhood; for this new Gaussian, no distinction is made between the points sampled in the earlier acquisition stages and the last Q sampled points. The quantities $n_{l,k}$, $d_{l,k}$ and $w_{l,k}$ are set to zero when no data point is present inside $C_{l,k}$ and the Gaussian $G_{l,k}$ will not contribute to the network output.

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

4.2.3 Proof of convergence

In [59], the capability of a single-layer HRBF, with a scale σ adequate to approximate a given function $f(\cdot)$, has been proven showing that the residual can be made arbitrarily smaller. Moreover, it has been 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 proven.

The on-line and the batch scheme differ for both the computation of the weights [(4.34)–(4.36) versus (4.27)] and for the rule of insertion of new Gaussians (in the batch scheme, this occurs layerwise, 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 operation such that

$$a_1(\cdot) = f_p \star G(\cdot; \sigma_1) \tag{4.37}$$

is 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 (4.37) converges to the value computed in (4.28) 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:

$$\lim_{p\to\infty} w^{\mathrm{b}}_{1,k} = \frac{\sum_{m=1}^{p} y_m \, e^{-||\mu_{1,\,k} - x_m||^2/(\sigma_1/2)^2}}{\sum_{m=1}^{p} e^{-||\mu_{1,\,k} - x_m||^2/(\sigma_1/2)^2}} \Delta x_1^D = \lim_{p\to\infty} w^{\mathrm{o}}_{1,k} \tag{4.38}$$

where $w^{\mathrm{b}}_{1,k}$ are the weights computed in the batch algorithm by (4.27) and $w^{\mathrm{o}}_{1,k}$ are those computed in the on-line algorithm by (4.34)–(4.36).

It follows that:

$$\lim_{p\to\infty} \Delta r_1(x_i) = \lim_{p\to\infty} r_1^{\mathrm{b}}(x_i) - r_1^{\mathrm{o}}(x_i) = 0 \tag{4.39}$$

where $r_1^{\mathrm{b}}(x_i)$ is the residual at the point x_i computed by (4.27), and $r_1^{\mathrm{o}}(x_i)$ is the same residual computed by (4.34, 4.35, 4.36).

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

$$n_{2,\,k} := n_{2,\,k} + (r_1^{\mathrm{o}}(x_p) + \Delta r_1(x_p)) \cdot e^{-||\mu_{2,\,k} - x_p||^2/(\sigma_2/2)^2} \tag{4.40}$$

$$d_{2,\,k} := d_{2,\,k} + e^{-||\mu_{2,\,k} - x_p||^2/(\sigma_2/2)^2} \tag{4.41}$$

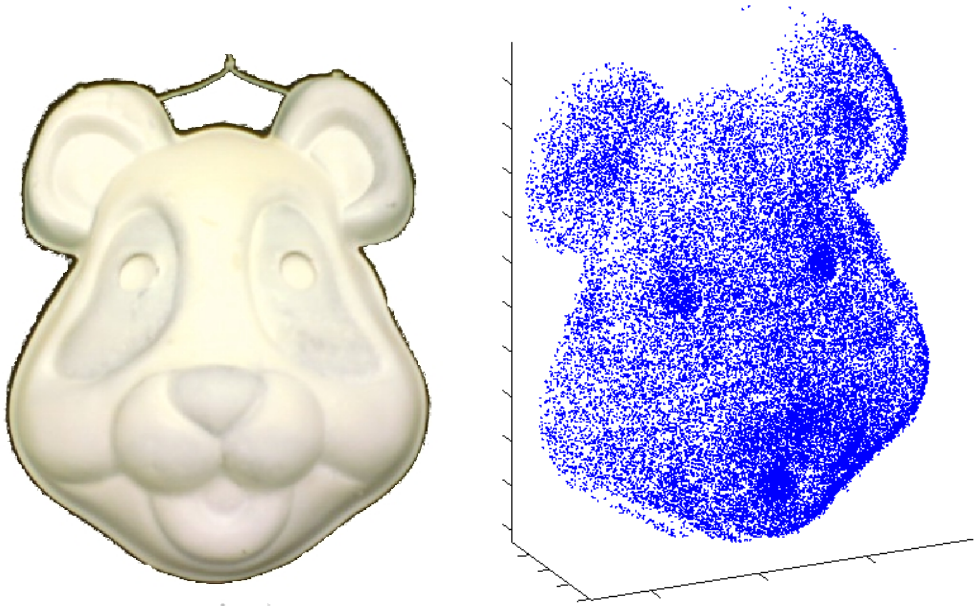
Since $\lim_{p\to\infty} \Delta r_1(x_p) = 0$ and $d_{2,\,k}$ always increases with p, the contribution of the initially sampled data points becomes negligible as p increases. As a result, $\lim_{p\to\infty} w_{2,k}^{\mathrm{b}} = \lim_{p\to\infty} w_{2,k}^{\mathrm{o}}$, and also the approximation of 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.

4.2.4 Experimental results

For the experimental results on-line HRBF model has been extensively applied to 3D scanning. Digitization was performed by the Autoscan system [27][26] which allows sampling more points inside those regions which contain more details: a higher data density can therefore be achieved in those regions that contain higher spatial frequencies. To this aim real-time feedback of the reconstructed surface is of paramount importance as shown in [6].

A typical set of sampled data is reported in fig. 4.6b: it is constituted of 33 000 points sampled over the surface of the artifact (a panda mask) in fig. 4.6a. As can be seen in Figs. 4.7 the reconstruction becomes better and better with the number of points sampled. Acquisition was stopped when the visual quality of the reconstructed surface was judged sufficient and no significant improvement could be observed when new points were added (compare fig. 4.7e and fig. 4.7f).



(a) (b)

Figure 4.6: A typical data set acquired by the Autoscan system [27]. The panda mask in (a) is reconstructed starting from 33 000 3D points automatically sampled on the surface by the Autoscan system; these constituted the input to the HRBF network. Notice the higher point density in the mouth and eyes regions.

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 [42]; from the dataset in fig. 4.6b, 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 l_1-norm, ϵ_{mean}, and in l_2-norm, root mean squared error, RMSE, as:

$$\epsilon_{\text{mean}} = \frac{1}{N} \sum_{i=1}^{N} |r_T(x_i)| \tag{4.42a}$$

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} r_T(x_i)^2} \tag{4.42b}$$

where $r_T(x_i)$ is the reconstruction error on the i-th point of the test set, $i = 1, \ldots, n$. To avoid border effects, (4.42) 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%.

The experiments have been carried out on a machine equipped with Intel Pentium4, 2.4 GHz, 512 KB cache, 512 MB RAM.

4.2.5 Comparison with the batch HRBF

Results of the comparison with the batch HRBF approach are reported in table 4.1. 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. The final reconstruction error of 0.391 mm, is very close to ϵ; a total of 9 222 Gaussians were allocated over the eight layers, and produce a sparse approximation (cf. fig. 4.8d–f).

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

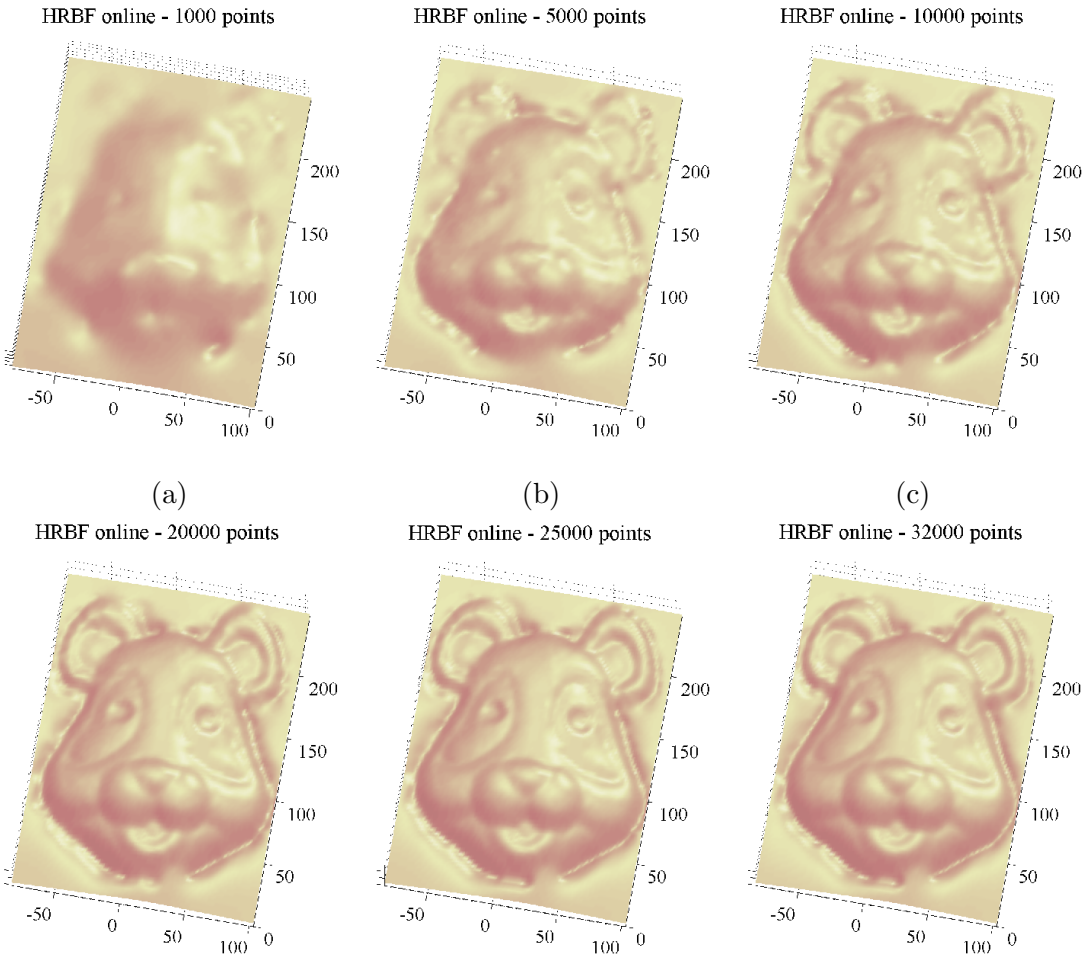


Figure 4.7: Panels show the reconstruction with on-line HRBF after 1 000, 5 000, 10 000, 20 000, 25 000, and 32 000 points have been sampled.

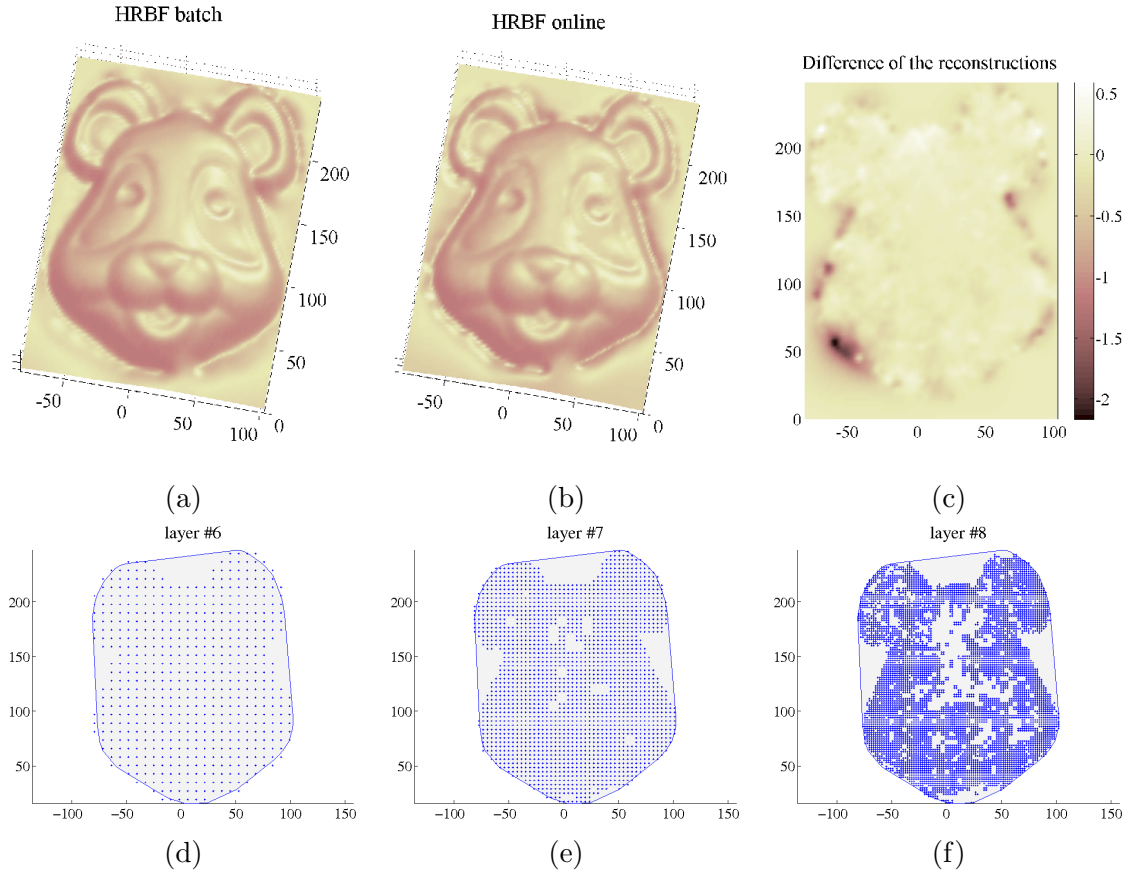


Figure 4.8: Reconstruction with (a) HRBF batch and (b) HRBF on-line. The difference between the two surfaces is shown in panel (c). In panels (d)–(f) the center of the Gaussians allocated by the on-line algorithm in the last three layers is shown.

Two batch modalities have been considered. In the first one, *pure batch*, the configuration procedure described in section 4.1.7 [59] 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 (4.27), considering all the data points inside the receptive field of each Gaussian, as described in [25].

As shown in fig. 4.8, 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. 4.8c and 4.9). This was obtained at the expense of a larger

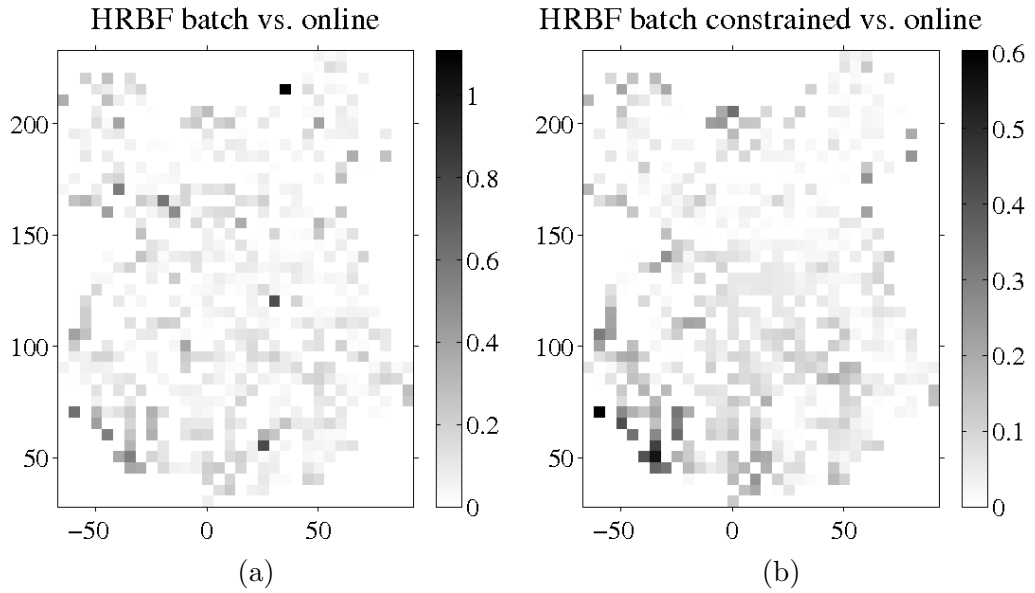


Figure 4.9: Difference in the reconstruction error on the points of the test set: on-line vs. pure batch (a), on-line vs. batch constrained (b).

number of Gaussians: about 12.7% more than those used in the on-line approach, being 10 393 versus 9 222 (table 4.1). Despite the difference in the computational resources used, 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%).

It is remarked here that acquisition was stopped when the visual appearance of the model (reconstructed with the on-line approach) was considered sufficient.

Therefore, it has been investigated if there was room for further accuracy improvement by acquiring more data points. To this aim, it is plotted in fig. 4.10 the rate of Gaussians allocation and of error decrease as a function of the number of data points. As it is clearly shown, the batch version grows and converges faster than the on-line version: it achieves ϵ_{mean} of 0.391 mm using only 8 500 data points. The figure shows also that the error of the on-line model can be slightly lowered adding new points down to 0.381 mm, closer to the batch approach. To achieve such an error only 99 more Gaussians are required.

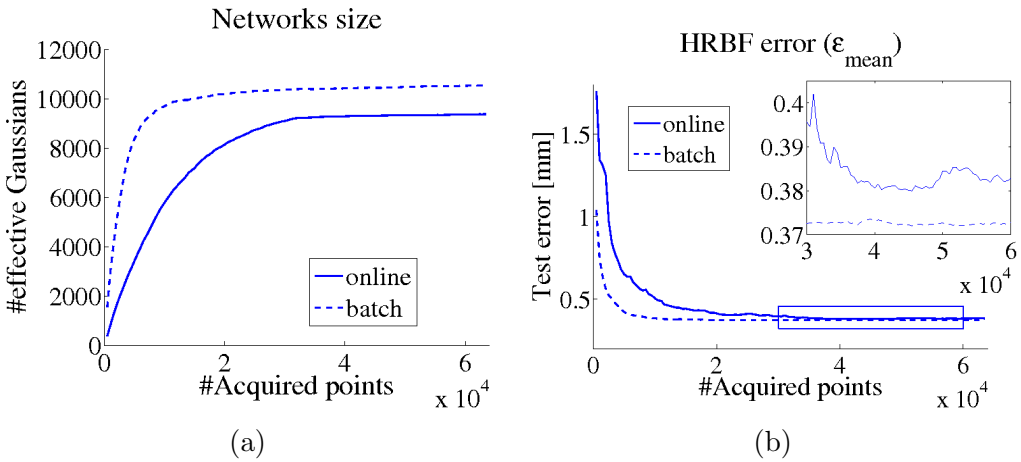


Figure 4.10: Number of Gaussian units (a) and mean error (b) as a function of the number of data points used to configure the networks. Data set is grown of 500 data points at a time.

Table 4.1: Accuracy and Parameters of Each Layer of the HRBF Networks

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

Table 4.2: Reconstruction with several datasets

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

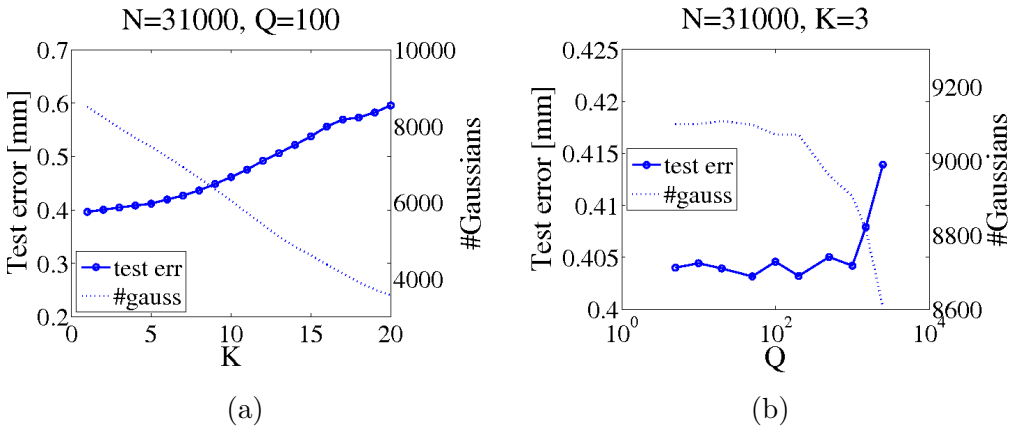


Figure 4.11: Test error and number of Gaussian units with respect to K with Q fixed to 100 (a), and with respect to Q with K fixed to 3 (b).

4.2.6 Discussion

Results are consistent for different artifacts (cf. fig. 4.13); real-time visualization of the actual surface has been of great value in directing the laser spot more frequently 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, as the Gaussians of the first layers 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. Alternatively, data acquisition could be stopped when splitting does not occur anymore. The error was measured as the mean value of the test error averaged over ten randomizations on the same dataset.

The on-line configuration algorithm contains two hyperparameters: K and Q (Section 4.2.2); their influence has been experimentally assessed by training the model with different combinations of K and Q. This gives an insight on the general behavior of the on-line algorithm as a function of these parameters.

Figure 4.11a 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,

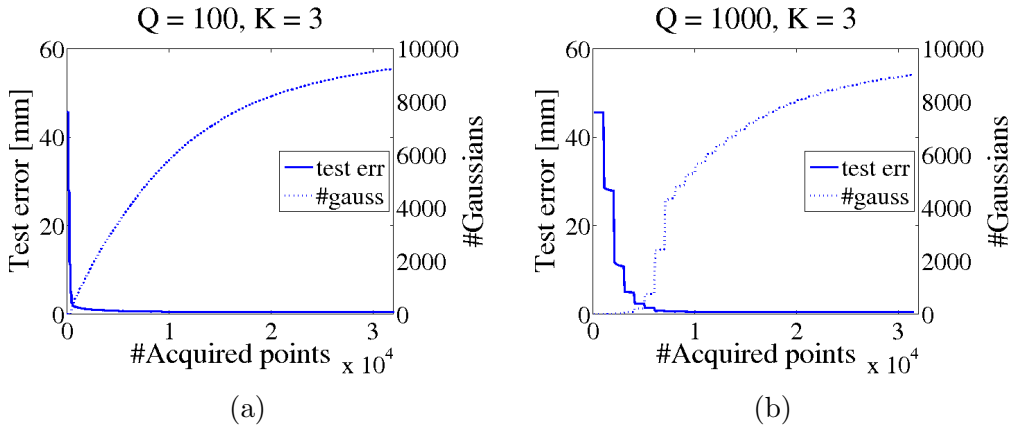


Figure 4.12: Test error and number of Gaussian units reported with respect to N for small and large values of Q: in (a), Q was fixed to 100, and, in (b), Q was fixed to 1 000.

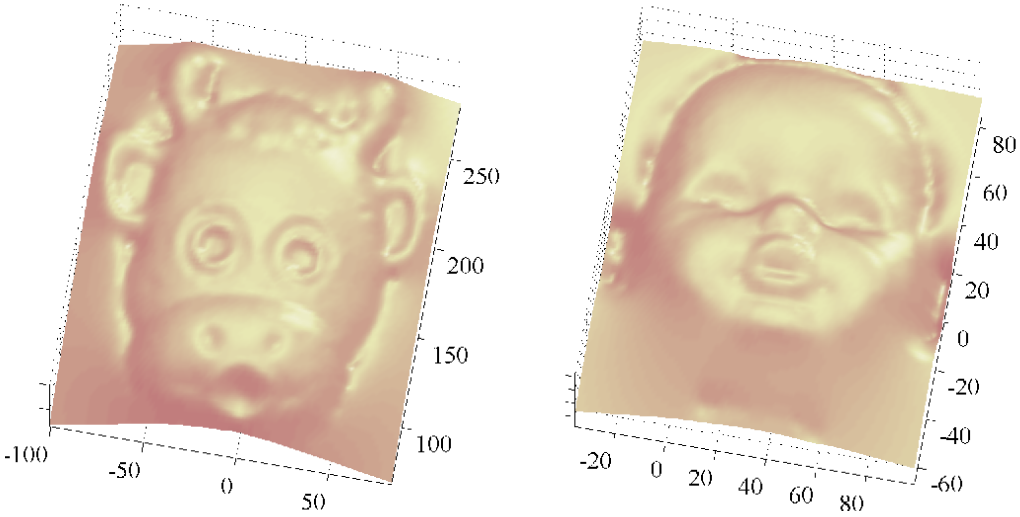
given the same total number of points, although the single weights can be estimated better, a smaller number of split operations would occur. This suggests using a very low value of K; in fact $K = 1$ produces the smallest test error, of 0.378 mm. This is paid with a larger number of Gaussians that reaches 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 $K = 1$.

The behavior of the test error as a function of Q is shown in fig. 4.11b. For small values of Q, the behavior is almost the same: 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 (Section 4.2.2). Afterward, its weight is updated considering only the points inserted inside its *receptive field*, as in (4.34, 4.35, 4.36). 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 occur, and the reconstruction becomes poorer. This situation is depicted in fig. 4.11b where for a relatively large value of Q, the test error tends to increase as an effect of the decrease in the number of the allocated Gaussians.

From the curve in fig. 4.11, it can be derived that the optimal value of Q would be $\approx 1\,000$, 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 the loss of interactivity. In fact, when Q increases, the split operation occurs less frequently. As this operation produces the largest decrease in the reconstruction error, a long time has to elapse before the user can see a large change in the appearance of the reconstructed model, and the improvement in the model quality is not smooth in time. Moreover, the reconstruction error decreases less quickly; this is well captured in fig. 4.12. Hence, the use of large values of Q is not interesting for such applications where interactive on-line learning is required and the value of $Q = 100$ has been adopted here as it is approximately twice the data sampling rate, that is of 60 points/s.

Therefore, although the value of K and Q may be subjected to optimization with respect to network accuracy or size, the resulting values may not be adequate for real-time applications. In particular, as Q produces a very similar test error over a wide range of values, it has been set here according to the data sampling rate to guarantee interactivity; this value is lower than the one that would produce the smallest network. This value could be even lowered, but at the price of a large overhead, as the split phase is the most computationally expensive. Therefore a value of Q related to data rate seems the most reasonable. However, it is remarked that using more Gaussians than the minimum required is not very sensitive to overfitting, because of the mechanism of local computation of the weights, provided that enough points have been sampled inside the receptive field of the Gaussians. This is shown in Figs. 4.12 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 estimation of the weights can be recovered in the higher layers thanks to the incremental optimization construction. For this reason, a very low value of K can be chosen: a value of $K < 5$ worked well in all our experiments and produced a good reconstruction with a reasonably low number of Gaussians.



(a) (b)

Figure 4.13: HRBF on-line reconstruction of the dataset (a) *cow* (33861 points, 9501 Gaussians), and (b) *doll* (15851 points, 6058 Gaussians).

The parameter L decides the level of detail in the reconstructed surface as it sets the smallest value of σ that is related to the local spatial frequency. It could be set in advance when this information is available to avoid the introduction of spurious high frequency components; otherwise, L is incremented until the error in (4.42) goes under threshold or a maximum number of Gaussians is inserted. It should be remarked that in this latter case, if Q were too small, L could increase more than necessary.

The mechanism used in the weight update phase, that does not require the reconfiguration of the whole network, produces a slight bias in the weights. This can be appreciated in table 4.1, 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 output value in $\mu_{l,k}$ (4.27) is estimated 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. However, this value is equal to that in the batch model only for the first layer (where all the Gaussian weights are computed using the height of all sampled points inside their *receptive field*). In the higher layer, where the residual in the already acquired points is not updated, the estimate of the weights may contain a bias. This, in turns, may produce a reconstruction error. However, in the splitting

phase, the bias in the weights and the reconstruction error are corrected in the *close neighborhood* of the split units, as the residual is recomputed there, using all the data points inside the $C_{l,k}$. Moreover, due to the non-orthogonality of the Gaussian basis function, the HRBF model is able to compensate the reconstruction error in one layer, with the approximation achieved by the next layer [58].

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: a Gaussian can be split before the neighbor Gaussians of the same layer. This is the case when higher frequency details are concentrated inside the *receptive field* of that Gaussian, as the parameters of each Gaussian are updated independently from those of the others. Therefore one branch of the network can grow before another branch.

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 weights in these regions may contain a bias that produces a local poor reconstruction. This error affects also the close neighborhood of the adjacent units 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 4.3).

Differently from other growing network models [9][117][70], pruning is not adopted here, as all the units participate in the reconstruction because of the configuration mechanism; pruning can be considered useful when dealing with time-variant systems or when some of the data are not pertinent, but the problem addressed here does not belong to this class, and the additional complexity for managing pruning does not seem justified here.

On-line HRBF shares with other growing networks models the criterion for inserting new units: the insertion is decided on the basis of a local error measure, which is fundamental to achieve real-time operation. The other element which allows real-time

Table 4.3: 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

operation is the grid support, which guides Gaussians positioning. This is shared also by [9]. However, in [9] all the weights are recomputed after the insertion of new Gaussians, while here only a subset of the weights is recomputed thanks to the hierarchical *close neighborhood* structure. This produces a large saving, especially for large networks. It should be noticed that this growing strategy implicitly implements an active learning procedure [78], as the data points that participate in the configuration of the new Gaussians are only those that carry an over-threshold error. Grid support has been also adopted by [149][113]; however, in their approach global optimization is used, that makes the configuration procedure computationally heavy.

4.3 Summary

The RBF network model is an interesting paradigm, which, thanks to the locality property, allows for obtaining a good approximation with limited number of units.

In the case of reconstruction of function characterized by different frequency content in different domain regions, the HRBF approach represents an efficient model for the computation of the solution.

This model is particularly suited to 3D scanning problems in which the surface of an object has to be computed from a set of noise-affected points sampled on the surface itself. The HRBF model can be configured in two modalities:

- batch, where all the points have to be acquired before the configuration;
- on-line, where the configuration is performed one sample at a time.

Both configuration algorithms have been discussed in this chapter. The on-line modality is an innovative contribution proposed in this thesis. It is particularly important because it allows a more effective scanning procedure, providing the surface

obtained up to that time as a feedback to the operator. The on-line procedure, derived from the batch one, produces results comparable to the latter, and can be effectively used in all the domains of low dimensionality. A proof of the equivalence between the two modalities in the asymptotic case has been provided.

5

Hierarchical Support Vector Regression

This chapter contains the second key point of the thesis: an innovative multi-scale approach based on Support Vector Machines (SVM) for solving regression problems. Thanks to the hierarchical structure, the model can be better applied to 3D surface reconstruction giving a new, more robust and faster configuration procedure.

The SVM approach is based on a non-linear generalization of the *Generalized Portrait* model developed in the sixties by Vapnik and Lerner [169]. As *Generalized Potrait* model, the SVMs are based on a framework of statistical learning theory, called also Vapnik Chervonenkis (VC) theory, which has been developed from 1960's to 1990's by Vapnik and Chervonenkis [168]. This theory is essentially related to the analysis of the conditions for which at the minimization of the training error corresponds the minimization of the generalization error.

Initially, this approach was applied just for classification problems; in particular the first works were focused on optical character recognition (OCR). 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 number of input variables. Thanks to the good performances with respect to other methods, SVMs rapidly became popular for all areas where statistical classification tasks must be performed [32] [33].

The SVMs have been more recently extended to regression [152], 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 independent from the number of input variables. As the SVM for classification, this approach has shown good performances for the solution of many applicative problems [122] [53] [155].

The smoothness of the solution computed by SVR paradigm is determined by a pool of parameters, generally, selected using a trial and error strategy. When the data are characterized by different frequency content over the input domain, the search of a combination of parameters that produces a good solution (over the entire input domain) can become particularly inefficient or unfeasible. The use of a hierarchical structure of SVRs that realize different scale reconstructions addresses this problem. The core of this chapter is an innovative paradigm based on SVR allowing multi-scale reconstruction and its use for 3D surface reconstruction problem.

In order to better understand the working principles of the Hierarchical Support Vector model, the SVM approach is considered initially in section 5.1. Starting from the linear classification problem, the explanation is extended to non-linear case and then to the regression problem. In section 5.2, the hierarchical regression approach is treated and a new model, termed Hierarchical Support Vector Regression (HSVR), is introduced.

5.1 SVM

5.1.1 Linear classification

The binary classification problem [168] can be defined as follow:
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 [167].

In the SVM paradigm, the $\hat{f}(x)$ is in the form of $\Theta(h(x))$, where $h(x)$ is a linear function of x:

$$\hat{f} = \Theta(h(x)) = \Theta(\omega \cdot x + b) \tag{5.1}$$

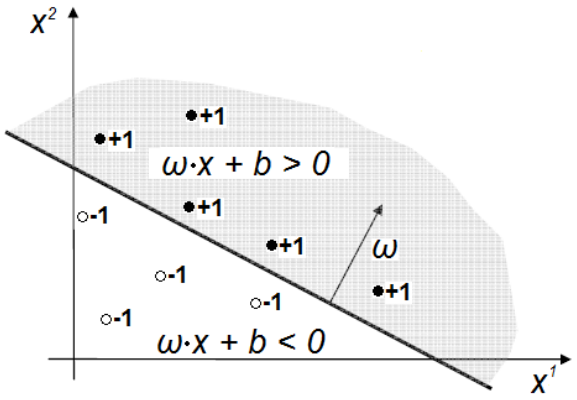


Figure 5.1: Binary linear classificator. x^i denotes the i-th input variable.

$$\Theta(z) = \begin{cases} -1 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases} \tag{5.2}$$

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. 5.1).

Then the classification problem can be formulated as the searching of the separator hyperplane parameters, ω and b. If the examples are linearly separable, the separator hyperplanes are in infinite number. Since the problem is ill-posed, the solution is not unique. The SVM algorithm allows finding the hyperplane that maximizes the margin. The margin is defined as the distance between the hyperplane and the closest examples (fig. 5.2). By the maximization of the margin the problem became well-posed, and if the solution exists it is always unique. Furthermore, 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. 5.3). The distance can be obtained multiplying the signed distance by the label y_i (5.2). Every separator

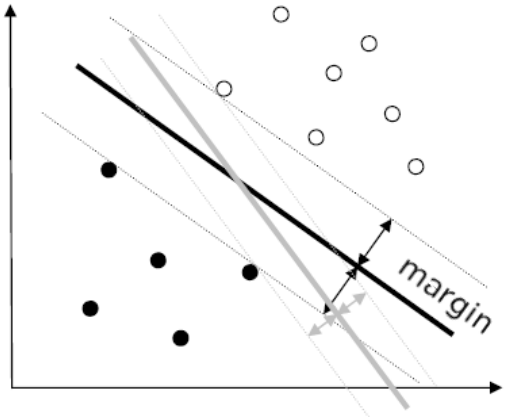


Figure 5.2: The separator hyperplanes are in infinite number. SVM algorithm finds that one that maximizes the margin.

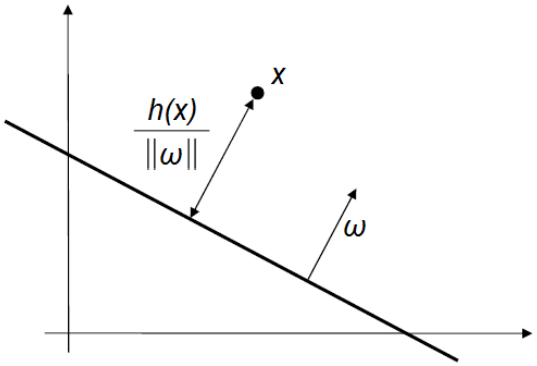


Figure 5.3: Distance between hyperplane and a point x.

hyperplane satisfies the following:

$$\frac{y_i h(x_i)}{||\omega||} \geq M > 0, \forall\, i = 1, \ldots, n \tag{5.3}$$

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

$$M^* = \max_{||\omega||=1,b} \min_i \; y_i h(x_i) \tag{5.4}$$

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:

$$\min_i \; y_i h(x_i) = 1 \tag{5.5}$$

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:

$$\begin{array}{rl} \min_{\omega,b} & \frac{1}{2}||\omega||^2 \\ \text{subject to} & y_i(\omega \cdot x + b) \geq 1,\ i = 1, \ldots, n \end{array} \tag{5.6}$$

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 (5.6):

$$L(\omega, b, \alpha) = \frac{1}{2}||\omega||^2 - \sum_{i=1}^{n} \alpha_i (y_i(\omega \cdot x_i + b) - 1) \tag{5.7}$$

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:

$$\min_{||\omega||,b} \max_{\alpha} \; L(\omega, b, \alpha) \tag{5.8}$$

the solution of this problem is the same of the solution of the primal problem (5.6).

Let (ω^*, b^*) be a pair of value for the problem (5.6). If (ω^*, b^*) do not satisfy all the constraints of (5.6) then the $\max_\alpha \; L(\omega, b, \alpha)$ tends to infinite and hence (ω^*, b^*)

is not a solution of (5.8). If (ω^*, b^*) satisfy all the constraints of (5.6) then the $\max_\alpha \ L(\omega, b, \alpha) = \frac{1}{2}||\omega||^2$, hence the solution of (5.8) is equal to the solution of (5.6).

Necessary conditions for a point (ω, b) to be a minimum of the primal problem (5.6) are the following:

$$\begin{array}{ll} \frac{\delta L(\omega,b,\alpha)}{\delta b} & = 0 \Rightarrow \sum_{i=1}^{r} \alpha_i y_i = 0 \\ \frac{\delta L(\omega,b,\alpha)}{\delta \omega} & = 0 \Rightarrow \omega = \sum_{i=1}^{r} \alpha_i y_i x_i \end{array} \tag{5.9}$$

The Lagrangian (5.7) can be rewritten as:

$$L(\omega, b, \alpha) = \frac{1}{2}(\omega \cdot \omega) - \sum_{i=1}^{r} \alpha_i (\omega \cdot x_i) - b \sum_{i=1}^{r} \alpha_i y_i + \sum_{i=1}^{r} \alpha_i \tag{5.10}$$

Using the strong duality theorem [51]:

$$\min_{||\omega||,b} \max_{\alpha} \ L(\omega, b, \alpha) = \max_{\alpha} \min_{||\omega||,b} \ L(\omega, b, \alpha) \tag{5.11}$$

Substituting the conditions (5.9) in the right part of (5.11), the dual problem is then obtained:

$$\begin{array}{ll} \max_\alpha & W(\alpha) = \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \\ \text{s.t.} & \\ \alpha_i \geq 0, & \forall \ i = 1, \ldots, r \end{array} \tag{5.12}$$

Therefore, the minimum of the primal problem (5.6) coincides with the maximum of the dual problem (5.12).

The optimization problem (5.12) (as the (5.6)) is a quadratic programming problem (convex quadratic functional and linear constraints). The solution is unique and can be found using standard numerical software.

Let α^* be the solution of the dual problem. The second condition of (5.9) is $\omega^* = \sum_{i=1}^{r} \alpha_i^* y_i x_i$ hence ω^* is a linear combination of training set vectors. Furthermore, from the Kuhn-Tucker theorem [98] it is known that the solution has to satisfy:

$$\alpha^* (y_i (\omega^* \cdot x_i + b^*) - 1) = 0, \quad \forall \ i = 1, \ldots, n \tag{5.13}$$

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 vectors 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

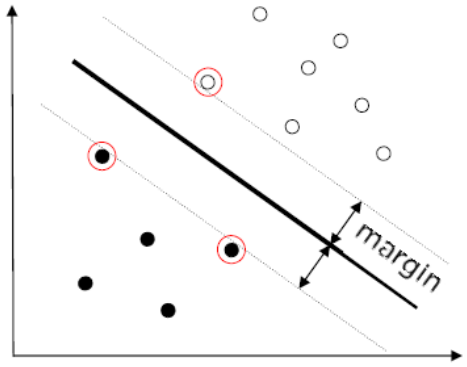


Figure 5.4: The circled points are the SVs that determine the hyperplane.

active ($y_i(\omega^* \cdot x_i + b^*) = 1$), their distance from the hyperplane is equal to the margin (fig. 5.4).

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

$$\omega^* = \sum_{SV} y_i \alpha_i^* x_i \tag{5.14}$$

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

$$y_i(\omega^* \cdot x_i + b^*) = 1 \Rightarrow b^* = y_i - \sum_{SV} y_j \alpha_j^* (x_j \cdot x_i) \tag{5.15}$$

Substituting in the decision function the expression of ω^*, it is obtained:

$$\hat{f}(x) = \Theta\left(\left(\sum_{SV} y_i \alpha_i^* x_i\right) \cdot x + b^*\right) = \Theta\left(\sum_{SV} y_i \alpha_i^* (x_i \cdot x) + b^*\right) \tag{5.16}$$

5.1.2 Soft margin classification

If the training set is non-linearly separable (fig. 5.5), 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 misclassification number and in the concurrent maximization 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 (5.6) are

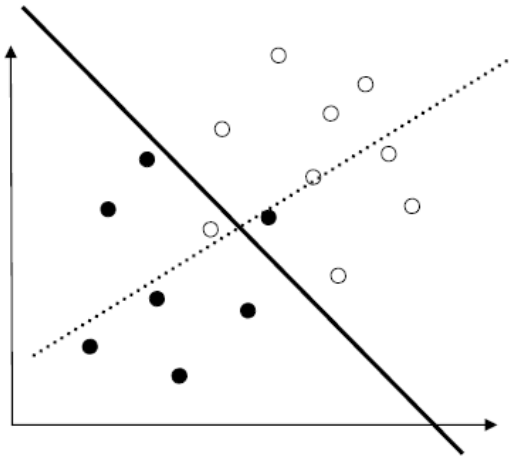


Figure 5.5: Example of a nonlinearly separable set.

reformulated as:

$$y_i(\omega \cdot x_i + b) \geq 1 - \xi_i, \forall\, i = 1, \ldots, n \tag{5.17}$$

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:

$$\min \frac{1}{2}(\omega \cdot \omega) + C \sum_{i=1}^{n} \xi_i^1 \tag{5.18}$$

where p has been chosen equal to one in order to maintain the computational tractability. C is a regularization constant that determines the trade-off between misclassified samples and the maximization of the margin.

The corresponding Lagrangian is:

$$L(\omega, b, \xi, \alpha, q) = \frac{1}{2}(\omega \cdot \omega) + C \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i \left(y_i(\omega \cdot x_i + b) - 1 + \xi_i\right) - \sum_{i=1}^{n} q_i \xi_i \tag{5.19}$$

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:

$$\begin{array}{rcll} \frac{\delta L(\omega,b,\xi,\alpha,r)}{\delta \omega} & = & \omega - \sum_{i=1}^{r} y_i \alpha_i x_i & = 0 \\ \frac{\delta L(\omega,b,\xi,\alpha,r)}{\delta \xi_i} & = & C - \alpha_i - q_i & = 0 \\ \frac{\delta L(\omega,b,\xi,\alpha,r)}{\delta b} & = & \sum_{i=1}^{r} y_i \alpha_i & = 0 \end{array} \tag{5.20}$$

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

$$\begin{aligned} &\max_\alpha W(\alpha) = \tfrac{1}{2}\textstyle\sum_{i,j} \alpha_i\alpha_j y_i y_j (x_i \cdot x_j) \\ &\text{s.t.} \\ &\textstyle\sum_{i=1}^{r} \alpha_i y_i = 0, \\ &0 \leq \alpha_i \leq C, \forall\, i = 1, \ldots, r \end{aligned} \tag{5.21}$$

(5.21) differs from (5.12) only for the constraints on the multipliers.

In this case, the KKT conditions are:

$$\begin{aligned} \alpha_i \left(y_i(\omega \cdot x_i + b) - 1 + \xi_i\right) &= 0 \\ \xi_i(\alpha_i - C) &= 0 \end{aligned} \tag{5.22}$$

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 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 (5.16).

5.1.3 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. 5.6). 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. 5.7).

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

$$\hat{f}(x) = \Theta(\omega \cdot \phi(x) + b) = \Theta\left(\sum_{j=1}^{D} \omega_j \phi_j(x) + b\right) \tag{5.23}$$

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.

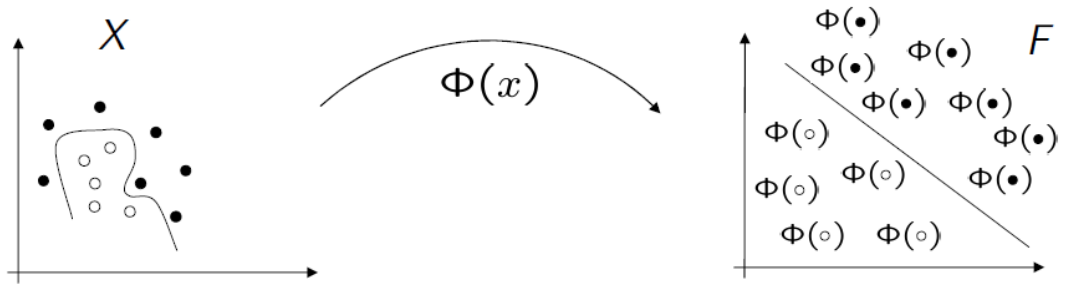


Figure 5.6: The function ϕ maps the data in other space called feature space.

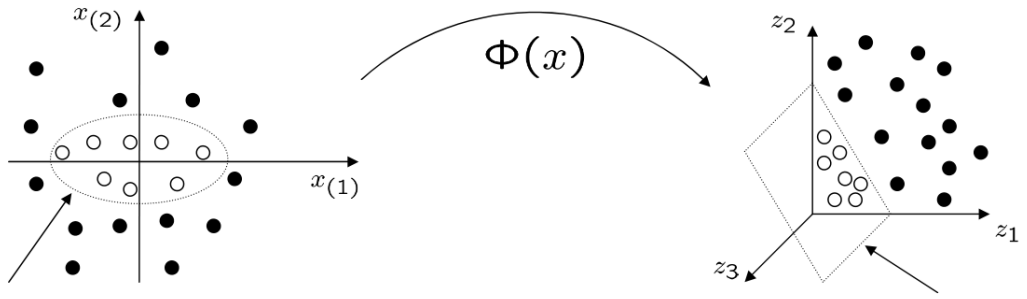


Figure 5.7: Thanks to ϕ the data become linearly separable, and the SVM algorithm can be applied.

The dual formulation would become:

$$\begin{aligned} &\max_\alpha W(\alpha) = \tfrac{1}{2}\textstyle\sum_{i,j} \alpha_i \alpha_j y_i y_j (\phi(x_i) \cdot \phi(x_j)) \\ &\text{s.t.} \\ &\textstyle\sum_{i=1}^{n} \alpha_i y_i = 0, \\ &0 \leq \alpha_i \leq C, \forall\, i = 1, \ldots, n \end{aligned} \tag{5.24}$$

and the relative decision function would be:

$$\hat{f}(x) = \Theta\left(\sum_{i=1}^{n} y_i \alpha_i (\phi(x_i) \cdot \phi(x))) + b\right) \tag{5.25}$$

It is noted 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.

5.1.4 Kernel

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

$$k(x, x') = \phi(x) \cdot \phi(x'), \qquad \forall\, x, x' \in X \tag{5.26}$$

where ϕ is a map from the input space X to space (having dot product) F.

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 ("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:

$$k(x, z) = k(z, x), \forall\, x, z \in X \tag{5.27}$$

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

$$K = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_r) \\ \vdots & \vdots & \vdots & \vdots \\ k(x_r, x_1) & k(x_r, x_2) & \cdots & k(x_r, x_r) \end{bmatrix} \in \mathbb{R}^{r \times r} \tag{5.28}$$

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:

$$\begin{aligned} &\max_\alpha W(\alpha) = \tfrac{1}{2} \textstyle\sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \\ &\text{s.t.} \\ &\textstyle\sum_{i=1}^{r} \alpha_i y_i = 0, \\ &0 \leq \alpha_i \leq C, \forall\ i = 1, \ldots, r \end{aligned} \tag{5.29}$$

The decision function becomes:

$$\hat{f}(x) = \Theta \left(\sum_{i=1}^{r} y_i \alpha_i k(x_i, x) + b \right) \tag{5.30}$$

5.1.5 Linear regression

As illustrated more formally in chapter 3, 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 $\hat{f}(x) : X \to Y$ is able to approximate the given samples, $\hat{f}(x_i) \approx y_i$.

The solution is searched using a *loss function* that measures the difference between the prediction and the real value of the samples. For instance, two common loss functions are the following:

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

where $\zeta = y - \hat{f}(x)$ (fig. 5.8).

In classification problem 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 the sparsity of the solution also in the regression case Vapnik introduced a family of loss function called *ε-insensitive*. In the case of 1-norm loss (fig. 5.9) the ε-insensitive version results:

$$L_1^\varepsilon(\zeta) = \begin{cases} 0 & \text{if } |\zeta| \leq \varepsilon \\ |\zeta| - \varepsilon & \text{otherwise} \end{cases} \tag{5.31}$$

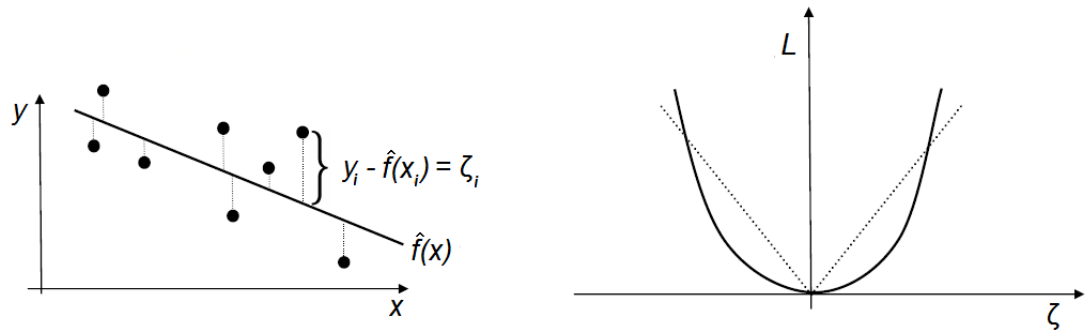


Figure 5.8: The goodness of an approximation, $\hat{f}$, is evaluated on the distance between point and its approximation (left panel) by means of a loss function, which, for instance, can be quadratic or linear (right panel).

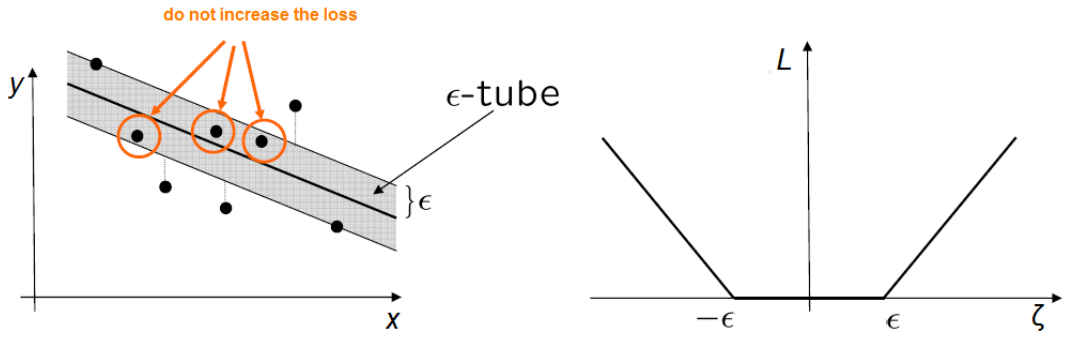


Figure 5.9: The points inside the ε-tube do not contribute to the training error (on the left), while the points outside the ε-tube contribute linearly to the error due to the shape of the loss function (on the right).

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

$$\min_{\omega,b} \frac{1}{2}(\omega \cdot \omega) + C\frac{1}{n}\sum_{i=1}^{n} L_1^{\varepsilon}(|y_i - (\omega \cdot x_i) - b|) \tag{5.32}$$

The functional of optimization problem can be seen as composed of two terms: the first, $\frac{1}{2}(\omega \cdot \omega)$, 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 vari-

ables the problem becomes:

$$\begin{aligned}
&\min_{\omega,b} \tfrac{1}{2}(\omega\cdot\omega) + C\sum_{i=1}^{n}(\xi_i+\xi_i^*)\\
&\text{s.t.}\\
&y_i - (\omega\cdot x) - b \le \varepsilon + \xi_i, \forall\, i=1,\dots,n\\
&(\omega\cdot x) + b - y_i \le \varepsilon + \xi_i^*, \forall\, i=1,\dots,n\\
&\xi_i, \xi_i^* \ge 0, \forall\, i=1,\dots,n
\end{aligned} \tag{5.33}$$

The constant C is a term that controls the trade-off between the minimization of the slope and the deviation from the insensitivity zone.

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

$$\begin{aligned}
L(\omega,b,\alpha,\alpha^*,\eta,\eta^*) \quad &= \tfrac{1}{2}||\omega||^2 + C\sum_{i=1}^{n}(\xi_i+\xi_i^*) -\\
&\sum_{i=1}^{n}\alpha_i(\varepsilon+\xi_i - y_i + (\omega\cdot x_i + b)) -\\
&\sum_{i=1}^{n}\alpha_i^*(\varepsilon+\xi_i^* + y_i - (\omega\cdot x_i - b)) -\\
&\sum_{i=1}^{n}(\eta_i\xi_i + \eta_i^*\xi_i^*)
\end{aligned} \tag{5.34}$$

The minimum conditions result:

$$\begin{aligned}
&\tfrac{\delta L}{\delta b} = 0 \quad \Rightarrow \quad \textstyle\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0\\
&\tfrac{\delta L}{\delta \omega} = 0 \quad \Rightarrow \quad \omega = \textstyle\sum_{i=1}^{n}(\alpha_i - \alpha_i^*)x_i\\
&\tfrac{\delta L}{\delta \xi_i} = 0 \quad \Rightarrow \quad \eta_i = C - \alpha_i\\
&\tfrac{\delta L}{\delta \xi_i^*} = 0 \quad \Rightarrow \quad \eta_i^* = C - \alpha_i^*
\end{aligned} \tag{5.35}$$

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

$$\begin{aligned}
&\max_{\alpha_i,\alpha_j} \textstyle\sum_{i,j=1}^{n}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)(x_i\cdot x_j) - \quad \varepsilon\sum_{i=1}^{n}(\alpha_i+\alpha_i^*) +\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textstyle\sum_{i=1}^{n} y_i(\alpha_i+\alpha_i^*)\\
&\text{s.t.}\\
&\textstyle\sum_{i=1}^{n}(\alpha_i-\alpha_i^*) = 0\\
&\alpha_i, \alpha_i^* \in [0, C], \forall\, i=1,\dots,n
\end{aligned} \tag{5.36}$$

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.

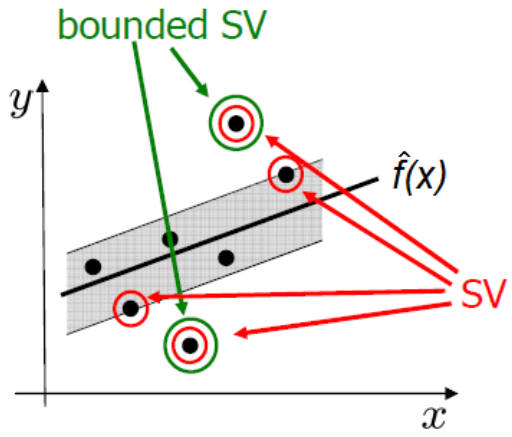


Figure 5.10: Only the points that do not lie in the ε-tube are SVs. Every point outside the ε-tube is a bounded SV, and the value of their α_i is C.

For this problem, the KKT conditions result:

$$\begin{aligned} &\alpha_i(\varepsilon + \xi_i - y_i + (\omega \cdot x_i) + b) = 0 \\ &\alpha_i^*(\varepsilon + \xi_i^* + y_i - (\omega \cdot x_i) - b) = 0 \\ &(C - \alpha_i)\xi_i = 0 \\ &(C - \alpha_i^*)\xi_i^* = 0 \end{aligned} \tag{5.37}$$

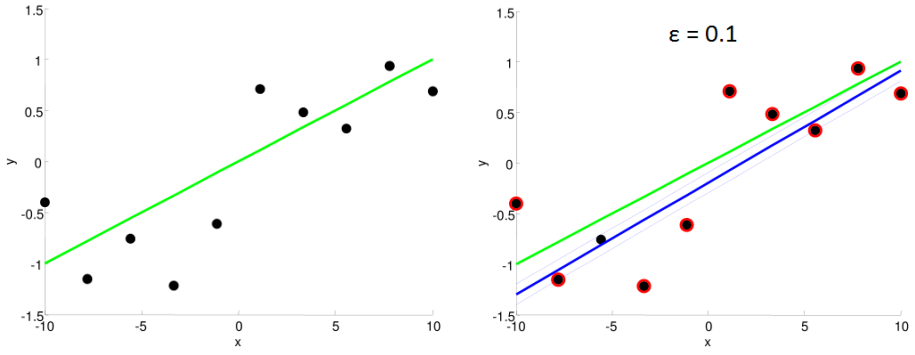
The first two conditions assure that the Lagrangian multipliers are zero for each point lying in the ε-tube. Then the solution is sparse. The last two conditions say that each point lying outside the ε-tube is a bounded support vector, namely $\xi_i > 0 \Rightarrow \alpha_i = C$. Each point lying on 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. 5.10).

The solution can be expressed as:

$$\hat{f}(x) = \sum_{SV}(\alpha_i - \alpha_i^*)(x_i \cdot x) + b \tag{5.38}$$

The value of b is computed using the KKT conditions.

In fig. 5.11, an example of linear regression is shown for several value of ε. The training set is obtained sampling a linear function (fig. 5.11-a) and a random uniform quantity has been added to the samples.



(a) (b) (c) (d)

Figure 5.11: In (a), a training set is obtained sampling a linear function in 10 points and a random uniform quantity has been added to the samples. In (b), (c), and (d) the SVR approximation obtained with ε-tube respectively of 0.1, 0.5, and 0.7. The points circled in red are the SVs.

5.1.6 Non-linear regression

Since the SVR are linear machines, then 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 (5.36) and in the decision function (5.38) the training set vectors appear only as the dot product of pairs of them. The kernel function can be used to compute the results of these dot products.

By using the kernel, the regression optimization problem has the form:

$$\begin{array}{l} \max_{\alpha_i,\alpha_j} \ \sum_{i,j=1}^{n}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)k(x_i,x_j) - \quad \varepsilon\sum_{i=1}^{n}(\alpha_i+\alpha_i^*) + \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \sum_{i=1}^{n} y_i(\alpha_i+\alpha_i^*) \\ \text{s.t.} \\ \sum_{i=1}^{n}(\alpha_i-\alpha_i^*)=0 \\ \alpha_i,\alpha_i^* \in [0,C], \forall\ i=1,\ldots,n \end{array} \tag{5.39}$$

The decision function becomes:

$$h(x)=\sum_{SV}(\alpha_i-\alpha_i^*)k(x_i,x)+b \tag{5.40}$$

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

$$\begin{array}{l} \max_{\beta} \ \sum_{i,j=1}^{n}\beta_i\beta_j k(x_i,x_j) - \quad \varepsilon\sum_{i=1}^{n}|\beta_i| + \sum_{i=1}^{n} y_i\beta_i \\ \text{s.t.} \\ \sum_{i=1}^{n}\beta_i=0 \\ -C\le\beta_i\le C, \forall\ i=1,\ldots,n \end{array} \tag{5.41}$$

The decision function becomes:

$$h(x)=\sum_{SV}\beta_i k(x_i,x)+b \tag{5.42}$$

For the KKT conditions the β_i coefficients have to satisfy the following relationships:

$$|\beta_i| = \begin{cases} 0, & |y_i - f(x_i)| < \varepsilon \\ [0,\, C], & |y_i - f(x_i)| = \varepsilon \\ C, & |y_i - f(x_i)| > \varepsilon \end{cases} \tag{5.43}$$

In practice, a tolerance threshold δ, is introduced to allow finding a reasonable number of points on the tube boundary, and (5.43) becomes:

$$|\beta_i| = \begin{cases} 0, & |y_i - f(x_i)| < \varepsilon - \delta \\ [0,\, C], & \varepsilon - \delta \leq |y_i - f(x_i)| \leq \varepsilon + \delta \\ C, & |y_i - f(x_i)| > \varepsilon + \delta \end{cases} \tag{5.44}$$

The Gaussian function is one of the most used as kernel. If it is used, (5.42) assumes exactly the form of the function computed by the RBF network (4.2). Hence, the SVRs, as the RBF nets, are characterized by a solution that is a linear combination of kernel functions.

5.2 Multi-scale SVR

5.2.1 Single kernel regression

A single kernel function is used in the standard SVR approach. This features a predefined shape characterized by a set of parameters. Like other methods based on kernels, the quality of the solution depends on the choice of the kernel function and of its parameters; these must be suitable to the current data. In general, this choice, also known as kernel selection [99], is a difficult task: the function is often chosen by trial and error, genetic optimization or domain knowledge of the user.

Besides this, the choice of a single kernel function can be questioned. In fact, when the data are characterized by a space varying frequency content, the use of a single kernel is not able to produce accurate solutions and approaches based on multiple kernels have been recently investigated [140][172]. In these approaches the kernel is defined as a mixture of predefined basic kernels, where the mixing coefficients are computed during the optimization phase. In this case, the form of the output is:

$$\hat{f}(x) = \sum_{i=1}^{n} (\beta_i \sum_{j=1}^{m} \mu_j k_j(x,\, x_i)) + b \tag{5.45}$$

where the type and the number (m) of kernels used in the linear combination have to be chosen a priori and the coefficients μ_j are determined in the optimization phase.

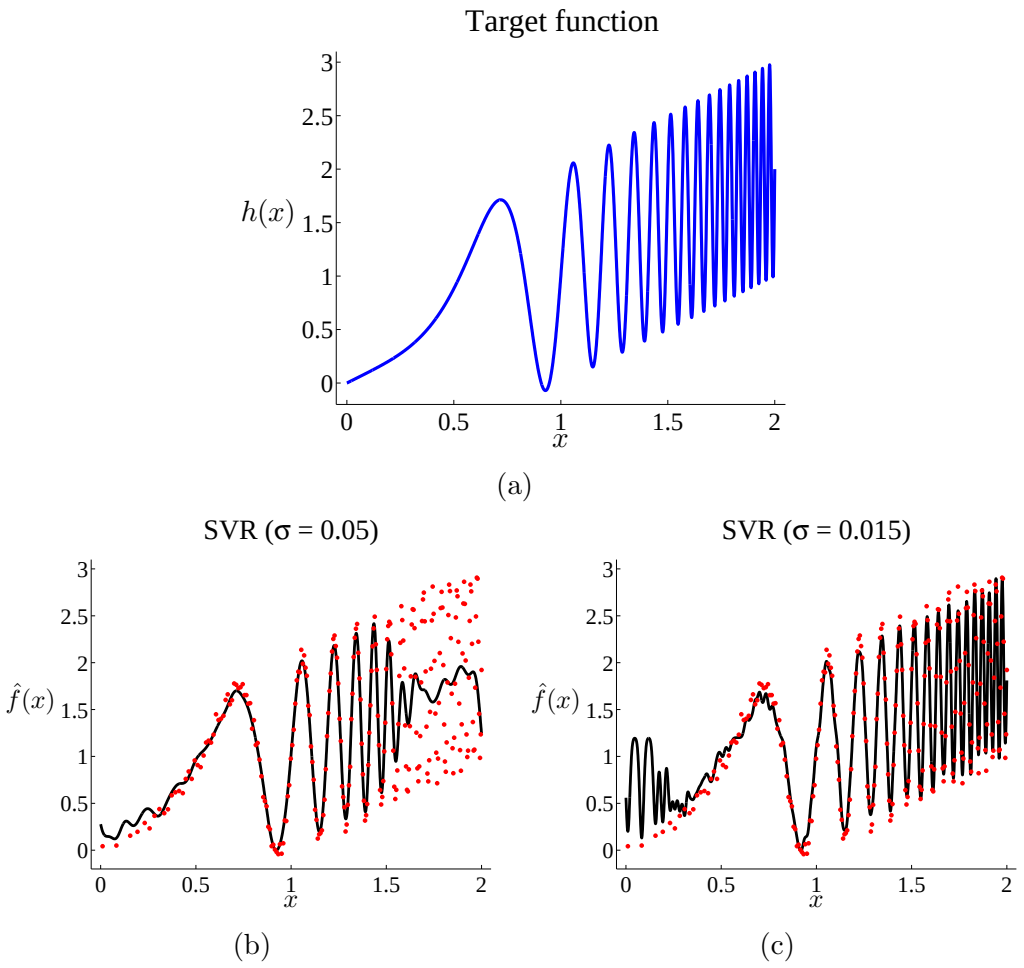


Figure 5.12: (a) A function with non-stationary frequency content, and (b)–(c) two SVR using a single Gaussian kernel with two different scale parameters, σ. (b) A large scale kernel provides a smooth solution, but is unable to reconstruct the details, while (c) a small scale kernel suffers of overfitting providing poor generalization.

However, even in this case, the solution is a linear combination of copies of the same function.

The problem of using a single kernel is highlighted in the examples reported in fig. 5.12. The data points have been sampled on the curve $h(\cdot)$

$$h(x) = \sin(2\pi x^4) + x \tag{5.46}$$

whose local frequency content increases with x. The sampling step is decreased with the local frequency according to $\frac{1}{120x}$. The solution computed with a large kernel fails in reconstructing the details as shown in fig. 5.12b. On the other side, using a kernel with a small scale, such as the one used in fig. 5.12c, the solution will be prone to overfitting and lack of generalization in scarcely sampled regions.

In the following it is presented a novel approach that allows adapting automatically the kernel parameters to the local frequency content of the data. This approach is based on a hierarchical structure, where each layer contains SVs that have the same kernel function, but the SVs in the different layers do feature different parameters and therefore the resulting model can be considered a multi-kernel approach. The model has been termed Hierarchical Support Vector Regression (HSVR).

5.2.2 Multi-kernel by hierarchical structure

The output of the HSVR model is obtained as the sum of the output of several layers, each constituted of single-kernel SVRs, $\{a_l(\cdot)\}$ characterized by a different scale:

$$\hat{f}(x) = \sum_{l=1}^{L} a_l(x, \sigma_l) \tag{5.47}$$

where L is the number of layers and σ_l determines the scale of the kernel of the l-th layer. The different layers are organized as a hierarchy where the scale decreases when the layer number increases, that is $\sigma_l \geq \sigma_{l+1}$ holds. When the kernel is the Gaussian function, the output of each layer can be written as:

$$a_l(x; \sigma_l) = \sum_{k=1}^{M_l} \beta_{l,k}\, G(||x - x_{l,k}||; \sigma_l) + b_l = \sum_{k=1}^{M_l} \beta_{l,k}\, e^{((x-x_{l,k})^2/\sigma_l^2)} + b_l \tag{5.48}$$

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. Therefore, the l-th SVM layer realizes a reconstruction of the

approximation up to a certain scale, determined by σ_l. HSVR configuration proceeds adding and configuring one layer at a time, proceeding from the layer featuring the largest scale to that featuring the smallest one.

The first layer is trained such that the distance between the function produced by the first layer itself and the data is minimized (5.32). All the other layers are trained to approximate the residual, which is the difference between the original data and the output of the HSVR model produced by the layers configured up to that stage. The measure of the residual for each layer, r_l, is given as:

$$r_l(x_i) = r_{l-1}(x_i) - a_l(x_i) \tag{5.49}$$

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

$$S_l = \{(x_1, r_{l-1}(x_1)), \ldots, (x_n, r_{l-1}(x_n))\} \tag{5.50}$$

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

The value of the scale parameter of the first layer, σ_1, is somehow arbitrary; for instance it can be chosen proportional to the size of the input domain (e.g., the length of the diagonal of the data bounding box). New layers are added during training until a given stopping criterion is satisfied (e.g., when the validation error does not decrease anymore).

Two other parameters are defined for each layer: C_l, that controls the trade-off between the regression error and the smoothness of the solution, and ε, that controls the amplitude of the ε-insensitivity tube around the solution itself.

Although few attempts to theoretically determine a proper value of C [171] [161] in regularization theory have been proposed, it is usually experimentally set by trial and error. In this work, C_l has been chosen, for each layer, as J times the standard deviation of the residuals used to configure that layer:

$$C_l = J \,\mathrm{std}(\, r_{l-1}(x_i)) \tag{5.51}$$

with the following motivation. First, it is noticed that as C_l is the value assumed by the Lagrange multipliers associated to the SVs of the l-th layer ((5.42) and (5.43)), its value represents the maximum weight that can be associated to each Gaussian in (5.42). For the regions of the input space where the Gaussians associated with the SVs have no significant overlap (this depends both on the Gaussian scale parameter and on the

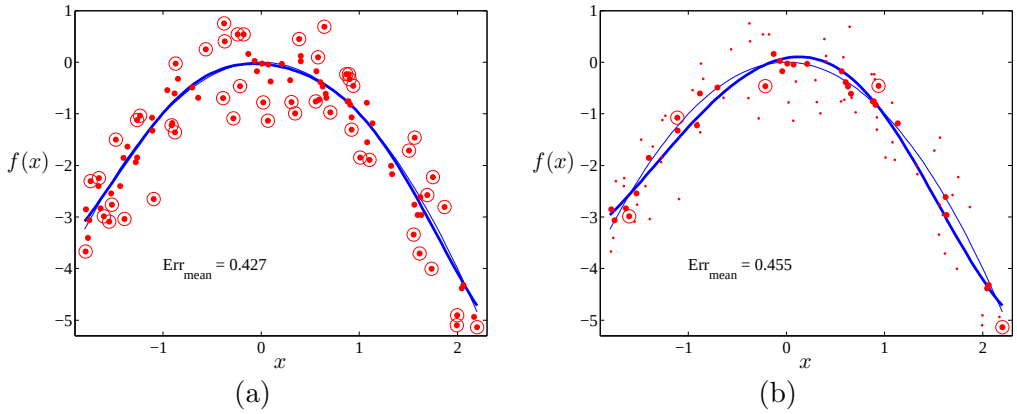


Figure 5.13: Data points reduction. A smooth function is shown in thin line in both panels. A set of 100 points have been randomly sampled over it and a Gaussian random quantity has been added to them. These points are displayed as dots. Thick dots represent all the points used by the optimization engine to determine the regression represented as a thick line. Circled dots represent the SVs. $\mathrm{Err}_{\mathrm{mean}}$ 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 ($\varepsilon = 0.416$, $C = 9.67$, $\sigma = 1.66$, $\mathrm{Err}_{\mathrm{mean}} = 0.427$) and in panel (b) the solution obtained considering only the points in S_l' (5.52). Notice that in the latter case only 32 data 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.

data density), the value of C_l is approximately the maximum value that the solution can assume in those regions (the Gaussian kernel maximum amplitude is equal to one, (5.48)). For this reason, C_l should be large enough to allow the solution reaching the maximum or minimum value of the data points inside the whole input domain. On the other hand, a too large value of C_l could favor overfitting. Experimental results on different data sets have suggested to choose the value of J in the interval (0, 5]. This represents a trade-off between these two requirements. However, inside the above range, results are largely independent on the value of J.

Hence, the only parameter that cannot be determined from the data set is ε. This should be proportional to the accuracy required for the regression, as its value is linearly related to noise amplitude [153].

5.2.3 Training set reduction

The drawback of the previous scheme is the total number of SVs, which is significantly higher than in standard SVR. Moreover, in HSVR the layers with a larger value of σ have a number of SVs similar to those layers with a smaller σ. This appears in contrast with common sense, as fewer units should be required to realize a reconstruction at a larger scale, but it is due to the fact that all the data points distant from the solution by more than ε are selected as SVs (5.43). Hence, in the first layers, where the HSVR output has a low frequency content, many data points will lie far from the computed function and they will be selected as SVs, thus leading to an unnecessary high number of SVs for the layer.

To avoid this, after each layer has been configured, a selection step devoted to reduce the number of the SVs is carried out. Then, the loss function (5.39) is minimized a second time, considering only the reduced training set to obtain the final approximation for that layer.

To reduce the number of the SVs, it is first noticed that the distance of a training point from the computed function measures the suitability of the function to describe the information conveyed by that data point. In this sense, points too distant from the solution cannot be "explained" by the solution itself and their utility can be questioned: they can be regarded as outliers. For these reasons, an acceptable approximation of the solution should be obtained using only those points that lie close to the solution itself.

This intuition has been confirmed experimentally. It has been observed that the quality of the approximation at a given scale does not degrade significantly by considering only points close to the ε-tube (cf. fig. 5.13). In particular, for the l-th layer, let us define as S'_l the set constituted only of those SVs which lie on the border of the ε-tube and those that lie far from the computed function for less than $\varepsilon/2$:

$$S'_l = \left\{ (x_i,\, r_{l-1}(x_i)) \text{ s.t. } |r_l(x_i)| - \varepsilon| < \delta \vee |r_l(x_i)| < \frac{\varepsilon}{2} \right\} \tag{5.52}$$

where δ is the tolerance parameter that determines the thickness of the ε-tube margin (5.44). It should be noticed, that S'_l is produced by means of the evaluation of the residual, $r_l(\cdot)$, obtained when (5.39) is maximized considering all the data points (first optimization step), while the final output of the layer and therefore the associated

residual for that layer, is obtained maximizing (5.39) only with the points in S'_l (second optimization step).

Therefore, the configuration phase of each layer is structured in two sequential steps: the first one provides the approximation at the considered scale considering all the training points, while the second one realizes an efficient representation of the approximation by considering only a selected subset of these points.

In order to cope with the diminished points density in S'_l, the value of the parameter C_l is increased proportionally in the second optimization step:

$$C'_l = C_l \frac{|S_l|}{|S'_l|} = J \,\mathrm{std}(\, r_{l-1}(x_i)) \frac{|S_l|}{|S'_l|} \tag{5.53}$$

It should be remarked that, similarly to the HRBF model [59][63] (described in section 4.1), any reconstruction error induced by the information loss due to the reduction of the training set will flow into the residual, r_l, that is used to configure the next layer of the architecture. Therefore such error is not critical, as it will be taken care by the next layer. The reduction procedure can be applied also to standard SVR. However, as in standard SVR there is no chance to recover the error introduced by the pruning, more care should be taken in the selection of the reduced training set. In section 5.2.5, the reduction procedure is discussed and compared with pruning techniques found in the literature [106][125].

5.2.4 Experimental results

The results obtained using the HSVR model on both simulated and real data are reported here and compared with those obtained with standard SVR [152], in terms of the number of SVs, the computational time, and the accuracy. The accuracy is assessed through the root mean square error (RMSE), and the mean absolute error ($\mathrm{Err}_{\mathrm{mean}}$) and its standard deviation ($\mathrm{Err}_{\mathrm{std}}$). These are computed over a test set, different from the training one. A third set, different from both, called validation set, is used for choosing the optimal value of the hyperparameters, which are ε, σ, and C for the standard SVR model, and of only ε and C for the HSVR model. For the latter, C is set according to (5.51) and (5.53) and the value of σ for the first layer is set equal to the size of the input domain. The validation set is used also to decide when the growth of the HSVR model has to be stopped: no new layer is added when the validation error

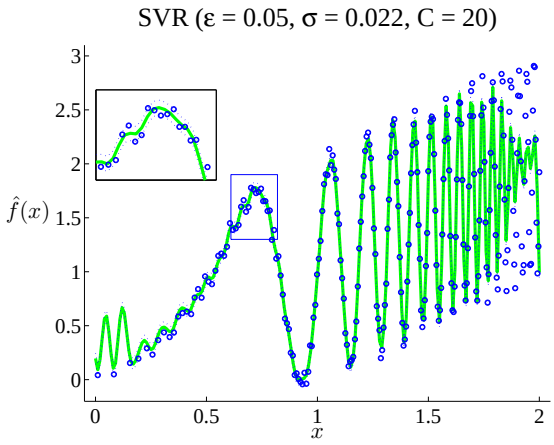


Figure 5.14: Reconstruction provided by standard SVR with the best hyperparameters ($\varepsilon = 0.05$, $\sigma = 0.022$, $C = 20$). Notice the poor approximation on the right side of the curve and spurious oscillations on the left.

does not decrease anymore and the last layer is discarded, as it can easily contribute to overfitting.

The optimization problem in (5.39), that arises for both the hierarchical and the standard SVR approach, was solved through LibCVM Toolkit Version 2.2 [164]. This software has shown the same accuracy of SVM^{light} [89] (that is one of the most used software packages for SVM) with a substantial saving of computational time. This was measured on a machine equipped with an Intel Pentium 4, at 2.40 GHz, 512 KB cache, and 512 MB of memory.

5.2.4.1 Regression on synthetic data

The space-varying function, $h : \mathbb{R} \to \mathbb{R}$, defined in (5.46) and plotted in fig. 5.12a is used. It is a synthetic dataset that allows to stress the limits of the single scale SVR. The training dataset has been obtained by sampling (5.46) in 252 points such that the sampled data density is proportional to the local frequency content, and adding a random uniform quantity in $[-0.1, 0.1]$ to simulate sampling error. The solution has been evaluated using a test set and a validation set, each composed of 500 points sampled from $h(\cdot)$ with a uniform distribution.

The accuracy of standard SVR was evaluated on the validation set for all the possible

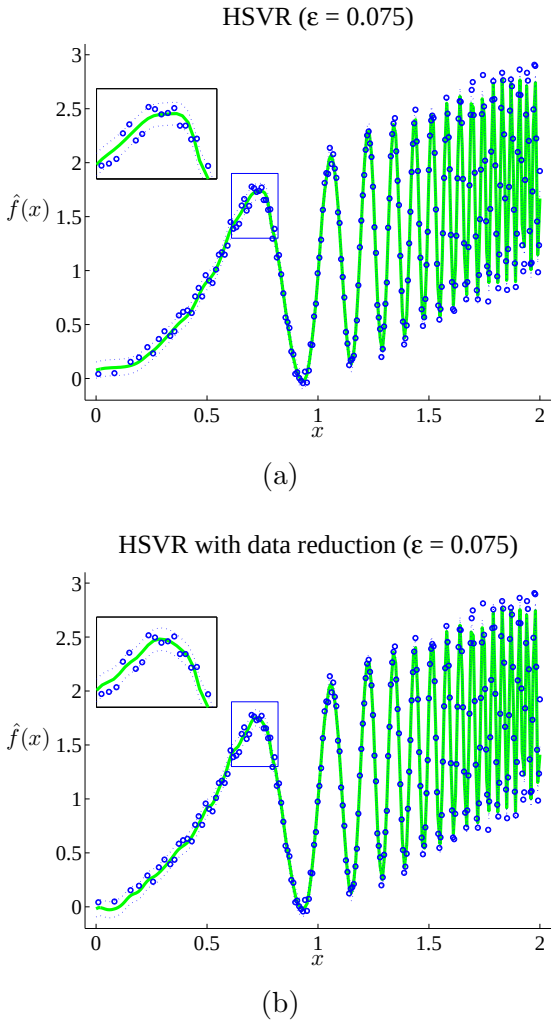


Figure 5.15: Reconstruction provided by HSVR (a) and HSVR with data points reduction (b) with $\varepsilon = 0.075$. The dashed lines limit the ε-insensitive region (i.e., the data points that lie inside this region do not increase the loss function value (5.32)).

Table 5.1: Accuracy on synthetic dataset

	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

combinations of the following values of the hyperparameters, ε, σ, and C:

$$\varepsilon \in \{0.0, 0.005, 0.01, 0.025, 0.05, 0.075, 0.1, 0.2\} \tag{5.54}$$

$$\sigma \in \{0.015, 0.022, 0.0313, 0.0625, 0.125, 0.25\} \tag{5.55}$$

$$C \in \{0.5, 1, 1.5, 2, 5, 10, 20\} \tag{5.56}$$

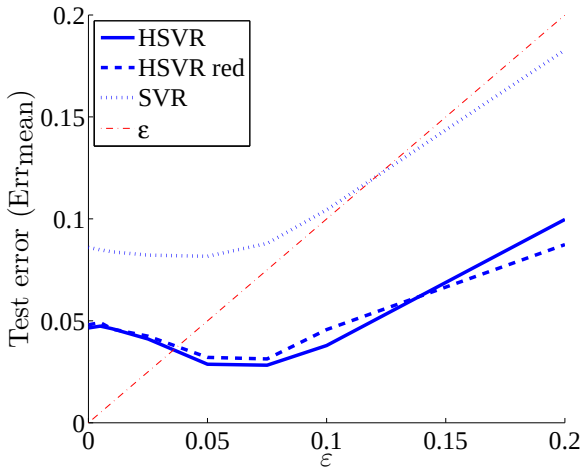
and the best SVR was used in the comparison. This was obtained with $\varepsilon = 0.05$, $\sigma = 0.022$, and $C = 20$ and it is shown in fig. 5.14. Notice the poor approximation on the right side of the curve and the spurious oscillations on the left. These are not present in the curve provided by the HSVR model shown in figs. 5.15.

This is supported by the quantitative data reported in table 5.1 that reports the accuracy of the different models.

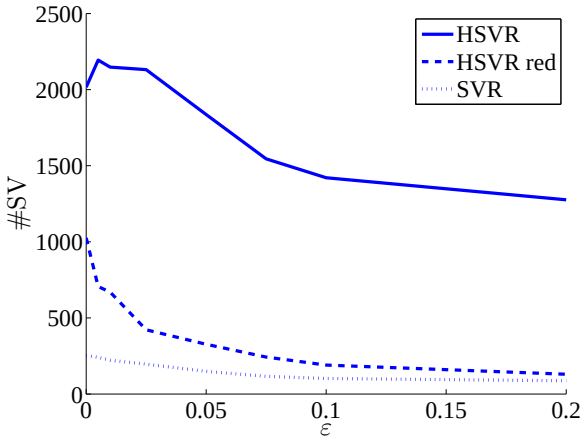
The computational time for the HSVR model is referred to the entire process of configuring the 9 layers required before the growth stops, while for the SVR model the configuration time does not consider the process for searching for optimum value of ε, C and σ, but is referred only to the computation of the solution with $\varepsilon = 0.05$, $\sigma = 0.022$, and $C = 20$.

If we considered also the search for the optimal value of the parameters, the total configuration time would increase significantly to 43.8 s. For sake of comparison, the configuration time for HSVR increases to 3.27 s and 4.49 s for HSVR without and with reduction, to test the eight different values of ε.

Enlarging the search space of C up to 100,000, the accuracy of SVR improves. In fact, the test error decreases from 0.0816 down to 0.0517, although it remains higher



(a)



(b)

Figure 5.16: (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.

Table 5.2: 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)

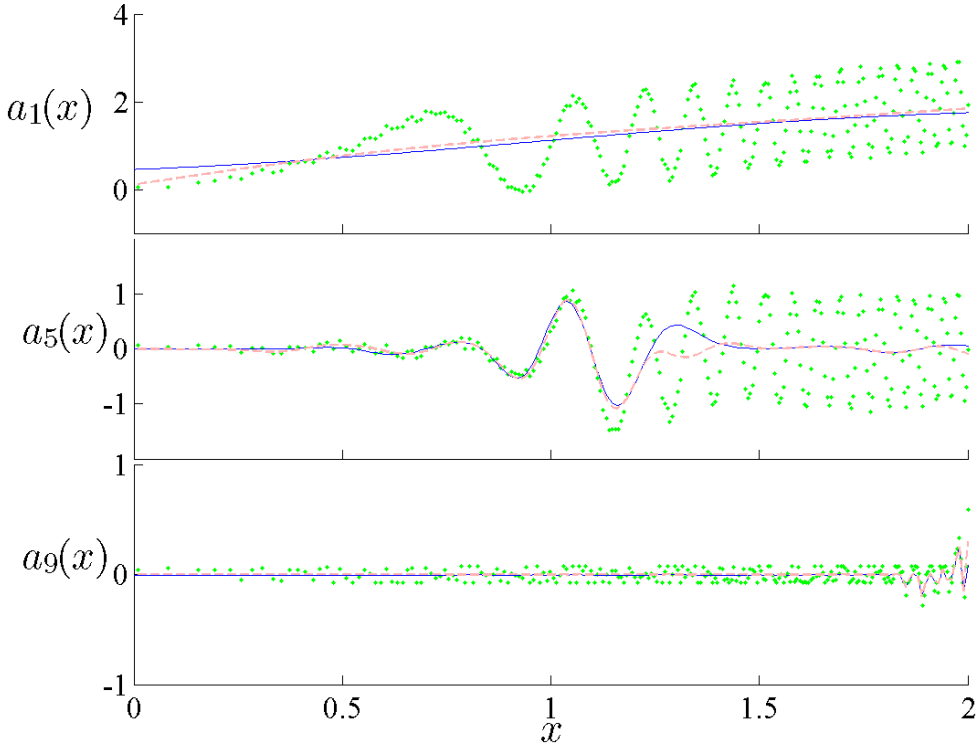


Figure 5.17: Reconstruction operated by the 1-th, 5-th and 9-th layer of the HSVR model when all the data points are considered (dashed line) and when only the points in S'_l are considered (continuous line). The residual of each layer (i.e., the training points for that layer) are reported as dots. 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.

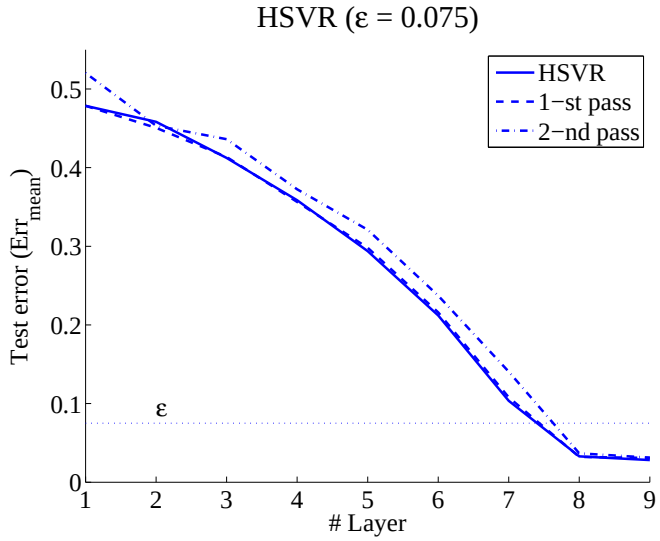
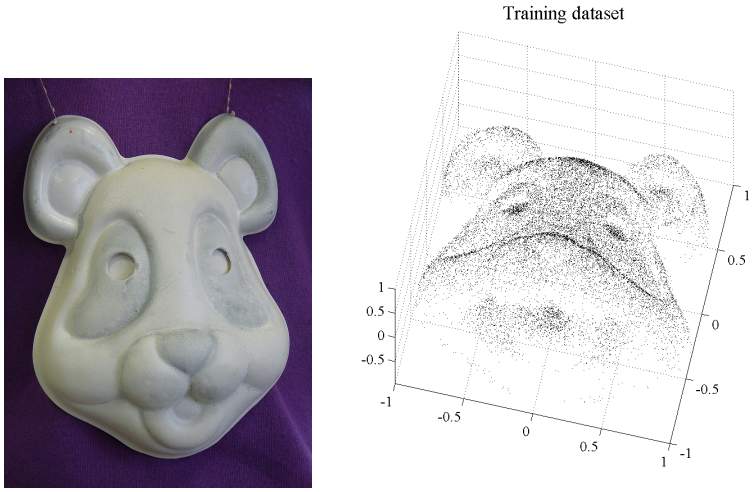


Figure 5.18: Evolution of the test error ($\mathrm{Err}_{\mathrm{mean}}$) of the HSVR models as new layers are inserted. The continuous line represents the error of the HSVR model with no data reduction. The dashed line represents the error of the model when data reduction was applied in all the previous layers, but not in the current one in which all the data points are passed to the optimization procedure (1-st optimization pass). The dot dashed line represents the error of the HSVR model when data reduction is applied to all the layers (2-nd optimization pass is applied to the configuration of all the layers).

than that of HSVR. However, the time required to compute this solution increases enormously to 3,129 s.

The role of ε has been investigated. As it can be seen in fig. 5.16, the test error produced by the HSVR model is below ε for $\varepsilon > 0.05$. This is much smaller of the optimal value of 0.075. This means that the data points, on average, lie inside the ε-tube around the curve. Instead, for SVR the optimal value of ε is 0.05, which is smaller than the test error achieved (0.0816). This means that a relatively large number of data points are not contained inside the ε-tube as can be seen in fig. 5.14. Moreover, as expected, the number of SVs decreases with the increase of ε.

The increase in the detail of the model is shown in fig. 5.17, where the output of three layers is shown. Notice that data reduction has the effect of smoothing the solution in the first layers, but smoothing tends to disappear in the last layer. This is highlighted in fig. 5.18 where the test error obtained with data reduction is superimposed to that obtained without data reduction. Quantitative figures are summarized in table 5.2.



(a) (b)

Figure 5.19: Panel (a) shows the artifact (a panda mask) that has been digitized obtaining the data for the experiment. In panel (b) the training data set is reported.

5.2.4.2 Regression on real data

Figure 5.19 shows a typical data set sampled from a real artifact (a panda mask) through a 3D scanner [63]. As the points have been sampled from a single point of view, they belong to a 2.5D surface that can be described as a $\mathbb{R}^2 \to \mathbb{R}$ function and they are therefore suitable to SVM regression. The data set is composed of $22,000$ points, of which $18,000$, randomly chosen, are used for training, $2,000$ for validation, and $2,000$ for testing. Each coordinate of the points was normalized to fit inside $[-1, 1]$. Besides, in order to limit border effects, validation and test error has been computed in the inner region of the data set, considering only points distant from the closest boundary by more than 0.1.

SVR was computed with all the combinations of the following values of the parameters (ε, σ, and J):

$$\varepsilon \in \{0,\, 0.0025,\, 0.005,\, 0.01,\, 0.02\} \tag{5.57}$$

$$\sigma \in \{0.188,\, 0.0938,\, 0.0469,\, 0.0234\} \tag{5.58}$$

$$J \in \{0.5,\, 1,\, 2,\, 5\} \tag{5.59}$$

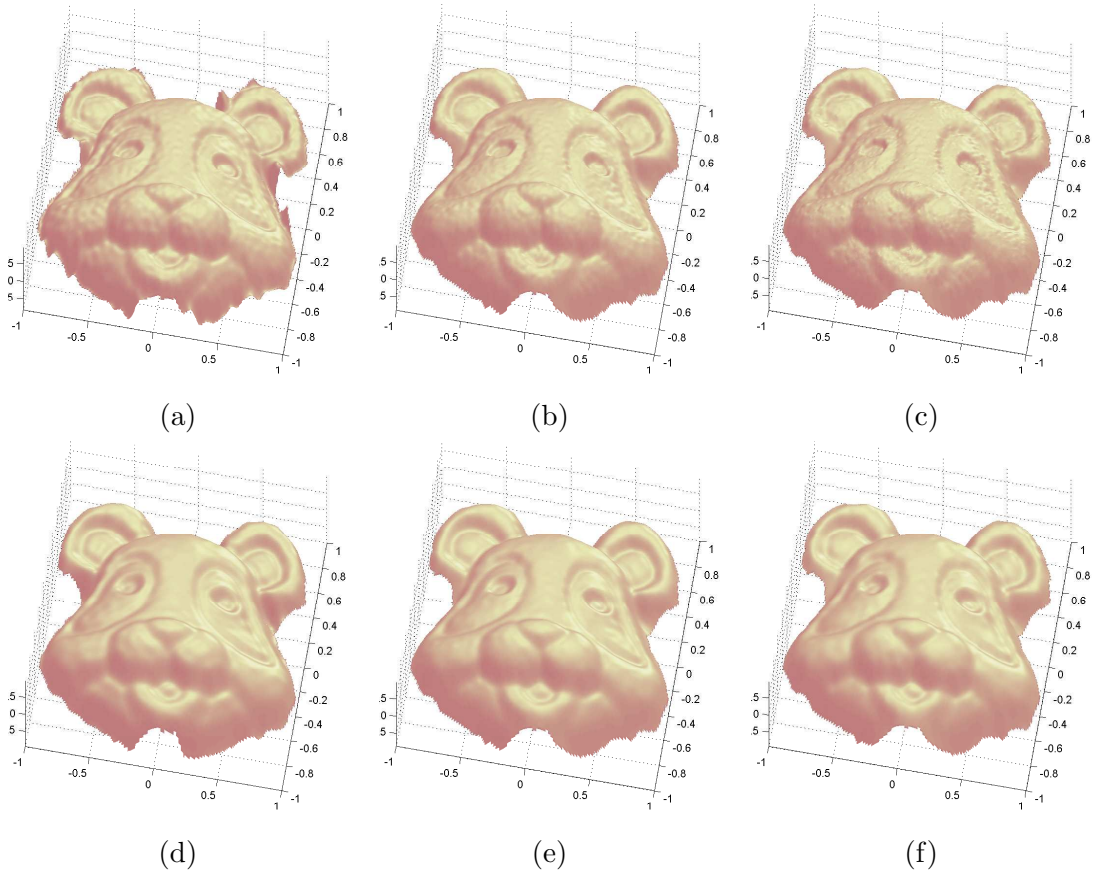


Figure 5.20: Panels (a), (b), and (c) show the surfaces that determine the lowest test error for SVR, HSVR and HSVR with reduction. The parameter used were respectively $J = 0.5$, $\varepsilon = 0.005$ and $\sigma = 0.0469$ for SVR, $J = 0.5$ and $\varepsilon = 0.01$ for HSVR, and $J = 1$ and $\varepsilon = 0.01$ 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)).

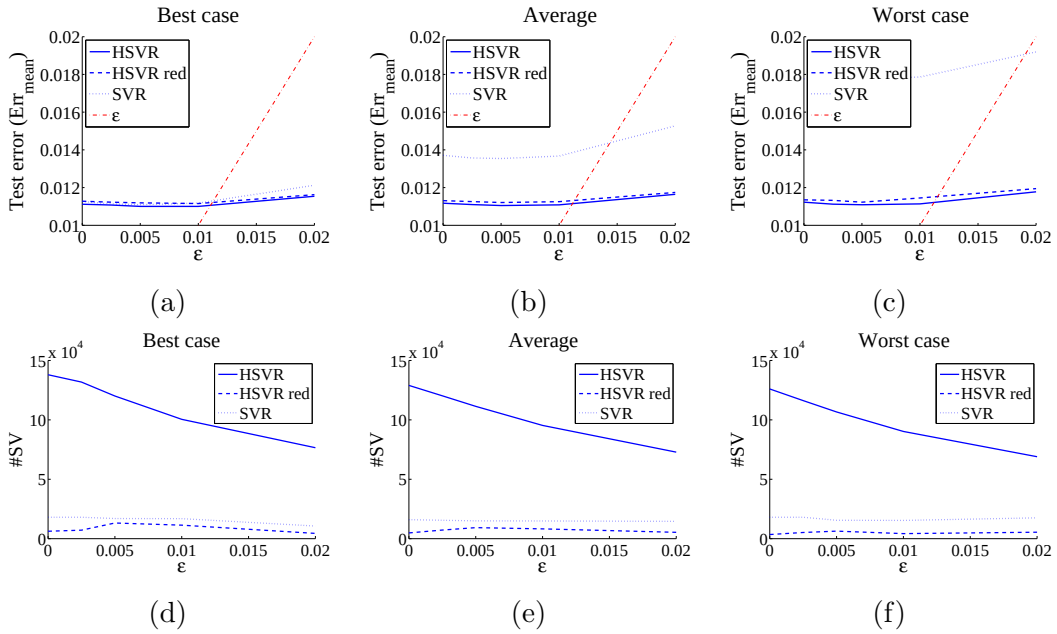


Figure 5.21: Test error (a–c) and number of SVs (d–f) used by the SVR and HSVR model for the Panda dataset as a function of ε are reported. Results for the best, average and worst cases are plotted. For reference, the value of ε has been reported as a dot-dashed line.

C was set proportionally to the standard deviation of the output variable, y, of the points through the proportionality factor J:

$$C = J\,\mathrm{std}(y) \tag{5.60}$$

similarly to (5.51).

The HSVR was computed using all the combinations of the values of J in (5.59) and ε in (5.57). The surface associated to the lowest test error is shown in fig. 5.20a–c for SVR and HSVR with and without reduction. However, although surface variability lies inside the ε-tube region, the results are not nice due to high frequency variability. A better visual appearance can be obtained only from HSVR models, discarding a few of the last layers. This is clearly evident in fig. 5.20e–f.

The test error and number of SVs for HSVR and SVR, averaged over five different randomizations of the data set, are reported in fig. 5.21. The best case results are reported in table 5.3 that shows that the best test error is similar in all the three

Table 5.3: 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

models (all in the range $[0.0110, 0.0112]$). This consideration can be extended to all the best models for each value of ε (fig. 5.21a).

On the contrary, the average and worst case test error (computed over all the parameters combinations) are much higher for SVR than for HSVR. 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.

Although the test error for all the three optimal models is very similar, the configuration time is very different: 682 s for HSVR, 1,104 s for HSVR with reduction, and 382 s for SVR. This large difference becomes smaller when suboptimal configuration parameters are considered. In fact, in the average cases the computational time to configure one model is 1024 s, 1241 s, and 1093 s for HSVR, HSVR with reduction and SVR.

The time required to explore the space of the hyperparameters has to be considered. We remark that the dimensionality of this space is smaller for HSVR, as σ does not need to be determined in advance. In the case here considered, 80 combinations have been used for SVR and 20 for HSVR, with a time saving of 25.4%. In fact, for SVR a total configuration time of 87,410 s has been used, while HSVR and HSVR with reduction required a total configuration time of 16,845 s and 22,168 s respectively.

From figs. 11d–f and table 5.3 it is evident that HSVR uses about ten times the SVs of SVR. However data reduction allows reducing this number slightly less than SVR (allowing a saving of 9.61%, for the optimal case).

5.2.5 Discussion

The HSVR model is based on two key elements: a multi-scale incremental approximation and a reduction of the number of the data points passed to the optimization

procedure and of those that contribute to the model output (the SVs).

As clearly shown in fig. 5.14 and table 5.1 the SVR with a single kernel function is not able to perform a solution of good quality when data frequency content and local data density is variable. In particular, spurious oscillations are produced in the region of low frequency content when a high frequency kernel is adopted to follow higher frequency oscillations (fig. 5.12c). In the multi-scale approach presented in this chapter, instead, the function in the low frequency region is reconstructed by the first layers while the higher layers improve the reconstruction of the function in the high frequency region (fig. 5.17).

The better reconstruction achieved by HSVR does not depend on a particular value of ε as shown in figs. 5.16a and 5.21a–c: the test error of the HSVR model is well below the error of standard SVR for a wide range of ε values, even considering the best combination of ε, C, and σ for SVR.

The reduction of the SVs that effectively contribute to the output of each layer does not degrade the reconstruction (cf. fig. 5.18) and it allows to produce a very similar output with much less SVs (243 vs. 1545 for the synthetic dataset and 11,351 vs. 100,448 for the panda mask data set). The same is true for the output of the intermediate layers as shown in fig. 5.18 and table 5.2. As the solution accuracy does not degrade, most of the SVs employed especially in the first layers (cf. table 5.2) are wasted in the vain attempt of approximating details with a kernel that operates at a too large scale. These SVs can therefore be pruned without degrading the output quality of the model.

Figure 5.18 shows also the residual error that can be attributed to data points reduction (this is the height difference between the error "2-nd pass" and the error "HSVR"). This error is recovered in each next layer, as can be seen by comparing the error produced when no data reduction is applied to the current layer, but it was applied in all the previous layers ("1-st pass") with that produced when no data reduction is applied ("HSVR"): the two errors are almost coincident.

The reduction procedure is not limited to the SVs: data points are subdivided according to (5.52) in points internal to the ε-tube, that are used to constrain the function close to the unknown true function from which data have been extracted, and points on the margin or external to the ε-tube, that are effectively used to represent the solution (5.52); only these are called SVs (cf. fig. 5.13). All the data points, internal

and not internal to the ε-tube, are passed to the optimization procedure, while only a subset of both groups of points is passed when the reduction step is implemented (cf. fig. 5.13). This procedure is particularly efficient when data sampling is relatively dense; if a few data points are present and these are sparse, reduction might turn out not as efficient.

The problem of obtaining a SVM model with few SVs has been studied in several works [106] [72] [92] [76] [181]. These approaches can be classified into two main families. The first redefine the optimization problem in order to control the number of SVs inserted [72] [92]. The second family realizes the reduction of SVs in two steps: the standard training of the SVM is performed first and then a pruning procedure is applied. Training set reduction presented here belongs to the second family.

In [106] the set of kernel matrix row vectors is partitioned using K-means algorithm. The pruning operates clusterwise: a vector is deleted if it differs from its orthogonal projection in the space spanned by the others vectors of the cluster for less than a given threshold. After the pruning the coefficients of the SVs of the model are updated in order to recover part of the lost information. From the point of view of the ratio between the accuracy lost and the number of SVs deleted, this approach, generally, performs better reductions than that based on (5.52). Conversely, from the point of view of the computational cost, the reduction based on (5.52) is convenient. The pruning method presented in [106], in fact, requires a computational time similar to that of the training phase, while the approach presented here requires, generally, a small fraction of the time of the training phase (considering, also, the second step of training).

In [125] the hyperplane found by the SVM is seen from a mechanical point of view. Each SV exerts a force on the hyperplane. The minimum of the optimization problem determines that the system is in an equilibrium state. The pruning is operated by substituting two SVs with a SV that exerts an equivalent force, the system is, again, in an equilibrium state. The new SVs, generally, do not exert exactly the equivalent force of those deleted and an approximation is used. The selection of the pairs of SV for substitution is performed by means of a greedy heuristic procedure: the pair such that their substitution determines the minimum deviation of the solution is selected. The stop criterion is based on monitoring of the accuracy lost for each replacement. When the lost accuracy is over a given threshold, the pruning procedure is stopped. This approach has the advantage of controlling the lost accuracy of each pruning step

but, as the previous one, it is more computational expensive than the training selection based on (5.52).

Both the approaches [106] and [125] are applied to the classification case, but they may be adapted to the regression case. Vice versa the pruning based on (5.52) cannot be applied to the classification case, because it makes use of the ε-tube definition. The pruning methods, are generally characterized, by the aim of found a good trade-off between SVs reduction and accuracy lost. The computational time spent to find the pruned solution can be recovered in the phase of computation of the reduced model output. For the configuration of HSVR layers it is more important save time of the pruning procedure than preserve the accuracy. In fact, it should be noticed that, similarly to the HRBF model [59] [63], any reconstruction error induced by the information loss due to the reduction of the training set will flow into the residual, r_l, that is used to configure the next layer of the architecture. Therefore such error is not critical, as it will be taken care by the next layer.

Taking a closer look at fig. 5.13a and fig. 5.13b a small difference between the solution obtained with and without data point reduction can be observed. However, this difference is incorporated in the residual of that layer and it can be recovered by higher layers (table 5.2). This is clearly shown in fig. 5.17. Moreover, this difference becomes smaller and smaller as new layers are added, and, in the last layer, the two curves are almost coincident.

This consideration is confirmed in fig. 5.20 and 5.21, where it is evident that the model with and without reduction perform a very similar reconstruction, both from the point of view of test error minimization (fig. 5.21a–c) and from the point of view of visual quality evaluation (fig. 5.20e–f).

The saving in the number of SVs obtained using the data reduction, is paid with an increase in the computational time (table 5.3). However, as shown in table 5.3, the increase of 61.9% in the configuration time is worth a 88.7% saving in the number of SVs. Moreover, both the number of SVs and the computational time for the two pass optimization compares well with the corresponding figures of the traditional SVR.

In this respect, the number of layers may be critical. Different strategies can be used to stop the learning procedure. If no a-priori information is available, the use of validation error guarantees that a good generalization is obtained. The process of adding new layers can be stopped when the validation error for new layers does not

decrease anymore. Otherwise, learning can be stopped when the error on the residual drops below a given threshold: for instance, this can be associated to measurement noise [63]. Other criteria, depending on the applicative context, can be adopted to stop the learning. For example, for graphical applications it is preferable an over smooth surface (with a greater test error, fig. 5.20d–f) than a surface that tends to overfit the data (although with a lower test error, fig. 5.20a–c). New layers can be inserted until the reconstruction does not present over fitting and the layers performing over fitting can be discarded. For standard SVR instead the selection of a smoother model needs the exploration of the parameters space and a qualitative evaluation of the solution resulting from each trial.

In the standard SVR approach, the quality of the solution depends critically on the value of the parameters, C, σ, and ε. This makes the computation of the standard SVR regression extremely time consuming, as the space of the three hyperparameters should be explored with the optimization procedure.

The HSVR approach, instead, is much less sensitive fig. 5.21a–c) and its configuration time is comparable to the configuration time of the standard SVR approach with a fixed set of hyperparameters (the configuration average time for panda mask dataset is 1024 s, 1241 s, 1093 s for HSVR, HSVR with reduction and SVR). This is due to the strategy adopted by the HSVR model to set the parameters.

As kernels with a different σ are chosen in the different layers, the criticality in choosing a single value of σ, adequate for the data, disappears: starting from a large σ and halving it in each layer, guarantees that a value of σ adequate to the data is found as shown in fig. 5.17. The hyperparameter ε is left to the optimization procedure although it can be set according to the measurement noise on the data. The last parameter, C, is set proportional to the standard deviation of the data points value (5.51) through the factor J. This has been experimentally set analyzing different data set in the range [0.1, 20], but low values of J are usually sufficient for achieving the best results. Even if its optimal value may vary with different data sets, it has been verified that results are robust with respect to variations of J for different data sets as shown in fig. 5.21a–c, where best, average and worst cases are almost similar.

In principle, it is possible to realize an accurate regression of a high frequency content function using a linear combination of large scale kernels (characterized by low frequency content) [74]. However, in general, this result is not applicable for real

cases, because the computational effort needed is too high. In fact, the use of large scale kernels involve large coefficients (high value of C), which may cause numerical instability of the solution due to numerical precision effects and noise on the input data. Besides the configuration time tends to increase with the value of C. Hence, the use of high value of C for real cases is almost unfeasible.

In principle, this learning scheme can work with kernels other than the Gaussians. However, the Gaussian kernel has two main properties: the scale parameter, σ, allows shaping the kernel such that the SVs are sensitive to different frequency ranges, and the non orthogonality, which allows to recover in the next layers the reconstruction error possibly left by the previous layers. Although in principle other kernels that enjoy the above mentioned properties could be used, adequate optimization engines should be developed, different from LibCVM Toolkit, which goes beyond the aim of this work. The use of different kernels will be investigated in future works.

5.3 Summary

The SVR paradigm is a global approach to solve regression problems. Thanks to the formulation as a convex optimization problem, a solution of that problem exists always and is unique. Furthermore the solution can be found using standard optimization software. Thanks to these features the SVM have become popular in many applicative fields including 3D scanning.

In the standard version of the SVR the solution is computed as a linear combination of a single kernel function. In this chapter it has been shown that this feature can limit the accuracy of regression for dataset characterized by different frequency content in different domain regions. The accuracy of the solution can be increased using the HSVR model proposed here.

The HSVR performs the reconstruction using a set of standard SVR with different hyperparameters of the same kernel functions. In this sense, HSVR can be considered a multi-kernel approach.

In order to reduce the total number of SVs used in HSVR, another model based on a second step of configuration for each layer, has been proposed. This version, called HSVR Red, allows solutions characterized by a degree of accuracy comparable to that of the HSVR and a number of SVs generally smaller than the standard SVR.

The HSVR and HSVR Red models have been applied to the 3D scanning problem. The results have shown that these innovative approaches are more robust than the standard SVR. Furthermore, the hyperparameter selection problem has been strongly reduced. Hence the hierarchical models allow to obtain more accurate surfaces using a faster configuration procedure.

6

Conclusion

Modelling is an essential activity applied in the most part of the scientific and real-life problems. The objective of the modelling is to obtain a simplified representation of phenomena, objects, and physical processes which captures the essential relations among the quantities and variables of interest. When the interest is oriented to the study of a physical object, the three-dimensional digital model of the object is a representation that is useful for several reasons.

Three-dimensional models are used in many application fields, encompassing, e.g., design, archeology, medicine, and entertainment. The use of digital 3D model has various advantages. For example, in medicine the 3D models of internal organs are used for diagnosis or to create virtual reality environments used for the training of surgeons. The entertainment is probably the field in which the 3D models are most used: almost all video games are based on the use of 3D models; often they are composed of digital characters impersonating real persons, like in sport game. Also in the cinematography the use of digital characters is constantly growing.

3D models essentially come from two distinct ways: CAD modeling in which the model is built manually by a user and physical object measurement in which the modeling is realized by means of 3D scanners. The latter is the fastest way to realize a 3D model of a physical object. The model presents, generally, higher accuracy and more realism with respect to that one obtained with CAD.

The output of 3D scanners is typically a noise-affected points cloud sampled on the object surface. An important step of the process of creation of the 3D model is the computation of a continuous description of the surface starting from the points

cloud. This step can be seen as a function approximation problem from points sampled on the function itself. Machine learning views the function approximation as a supervised learning problem. The points represent a set of examples and the surface to be reconstructed represents the process that has generated them.

The core of this thesis regards the development of innovative models for surface computation based on supervised learning approaches. In particular two kinds of paradigms for surface computation have been considered:

On-line models On-line models are particularly important for the surface reconstruction because these models allow real-time reconstruction with respect to the acquisition of the points from real object. For example, in active 3D scanning where a user drives a laser source over an artifact to sample data points over its surface, a real-time feedback of the current reconstructed surface would be of great help to sample points in the regions where the details are still missing in the reconstructed surface. This improves 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 hierarchical model is computed as the sum of the output of each sub-model. Hierarchical models are important tools for surface reconstruction problems because they, generally, allow a robust and efficient reconstruction.

The starting point of the work is represented by two approaches used for this kind of problems: Hierarchical Radial Basis Function network (section 4.1.7) and Support Vector Machine (section 5.1). These two paradigms have been chosen because they are two different approaches to the problem of surface reconstruction from sampled points. In particular the first one is a local method, as the function approximation is based on a local averaging of the data, while the second one is a global method, for which the surface reconstruction is formulated as an optimization problem of a functional.

In particular, the contribution of the thesis can be summarized in two key points: the development of an innovative on-line model for Hierarchical Radial Basis Function (HRBF) neural network [63] [17], presented in section 4.2, and the development of an original Hierarchical model for Support Vector Machines [62] [64], called Hierarchical Support Vectior Regression (HSVR), presented in section 5.2.

Table 6.1: Accuracy on "panda mask" dataset

	Err_{mean}	Err_{std}	RMSE	#SVs	time [s]
HSVR	0.0110	0.0115	0.0160	100 448	681
HSVR (red.)	0.0112	0.0119	0.0163	11 351	1 104
HRBF	0.0112	0.0111	0.0158	14 679	49

The results obtained have shown that the on-line HRBF can effectively perform a real-time reconstruction and the accuracy of the solution tends to the accuracy of the HRBF model configured with batch approach. The accuracy of the two models become comparable after the acquisition of a reasonable amount of points (4.7 and 4.8).

The results obtained with HSVR have shown that the hierarchical model allows a more accurate and robust reconstruction due to the use of a set of kernels at different scales. Furthermore the problem of choosing the hyperparameters is heavily reduced. Thanks to the use of the second step of configuration for each layer, the hierarchical model performs good approximation saving computational resources with respect to the standard SVR model.

6.1 HRBF vs. HSVR

The techniques developed in this thesis are evolutions of two common approaches for regression: Hierarchical Radial Basis Function Network and Support Vector Machine for Regression. These two paradigms represent two different approaches to deal with the problem of regression.

In HRBF the computation of the solution is based on the estimation of the weights of each unit computed as weighted average of the points that lie in a neighborhood of the unit self. The model can be considered a local approach because is based on local averaging.

In SVR the computation of the solution is formulated as an optimization problem. A functional, composed of two terms, is minimized in order to find the solution: the first term controls the training error and the second controls the smoothness of the solution. As the solution is performed by the global optimization of a functional, this model can be considered a global approach.

The local approaches have the advantages that being based on local averaging they do not need iterative configuration procedures. This, in practice, means that the configuration of local models is, generally, faster than the configuration of global models. This is confirmed by the data in table 6.1: the time for the configuration of the HRBF is 7.20% of the time of the HSVR and 4.44% of the time of HSVR red.

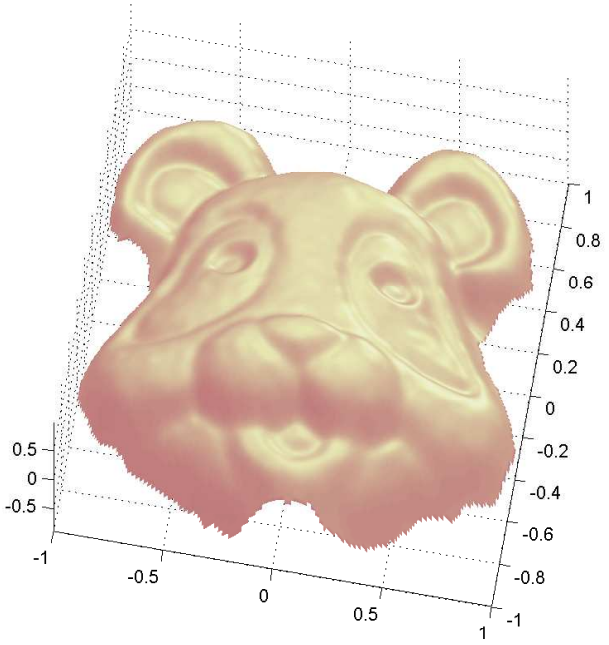
On the other hand the local approaches are more sensitive to the presence of outliers. A local estimation can be strongly influenced by the presence of an outlier. Vice versa the global training error can be slightly changed by the presence of the same outlier.

The local approaches are, also, more sensitive to the sparseness of the data. Sparser are the data fewer elements are used in the local averaging. This means less robust estimations. This is the reason whereby, generally, the local approaches do not show good performances for problems with many input variables. Typically, the sparseness of the data increases exponentially with the increase of the number of input variables.

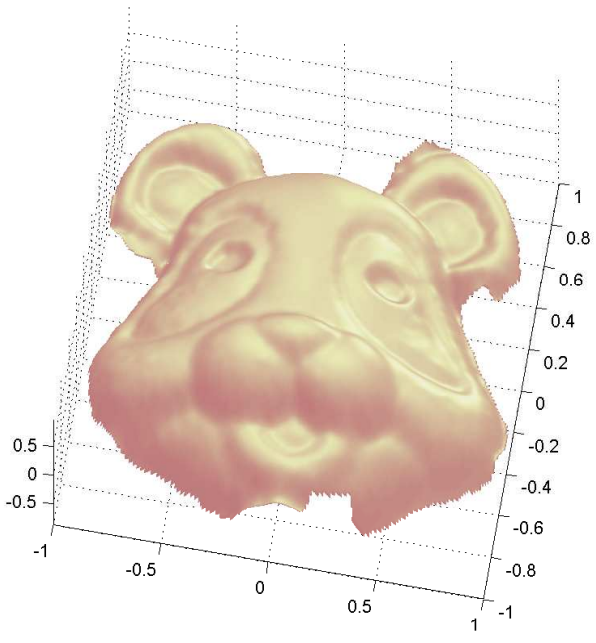
In particular the SVR (and HSVR) model has the advantage that the computational complexity of the configuration phase does not depend on the number of input variables. The number of input variables is hidden by the use of the kernel functions. The only information used in the configuration procedure is the value of the kernel function for each pair of training points. It can be computed once before the optimization of the functional.

With the HRBF model, instead, when the number of input variables increases, the computational complexity increases as well. For a large number of input variables both the computation of the positions of the units' centers and distances between points and units centers imply more complex geometrical operations. Furthermore, for two input variables the on-line procedure approximates the influence region of each Gaussians as a square region (close neighborhood); only the points that lie in this region are used for the estimate of the unit's weight. This approximation is not suitable for a large number of input variables. In fact, the difference between the volume of a hypercube (close neighborhood) and the volume of the circumscribed hypersphere (influence region) grows exponentially with the number of input variables.

If the reconstruction is performed for the case $\mathbb{R}^2 \to \mathbb{R}$ and the dataset is sufficiently dense, both methods reach the same level of accuracy (cf. table 6.1). Despite the fact that, for the two approaches, the test error is very similar, from the point of view of the visual quality, the surface computed by the HRBF model is quite better than that



(a)



(b)

Figure 6.1: (a) The surface computed by the HSVR model. (b) The surface computed by the HRBF model.

computed by HSVR (fig. 6.1). 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, instead 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.

6.2 Future works

Three possible research directions of the thesis are presented in this section. The first one regards to the development of on-line models for Hierarchical Support Vector Machines. This could be interesting for the 3D scanning problems but, also, for all the applications in which the solution has to be performed in real-time with respect the data acquisition.

The second extension is relative to the implementation on parallel hardware of the on-line hierarchical models. Exploiting the computational performances of the parallel hardware some approximations needed in the on-line model could be eliminated improving the accuracy of the models.

The third possible extension is the development of a Hierarchical Support Vector Machine for the classification problem. This problem presents many features in common with the regression case. Also in classification the solution is obtained using the mapping performed by the kernels function. Hence, using a set of kernels at different scales can be profitable also for this kind of problems. In the following the three research direction are discussed.

6.2.1 On-line HSVR

When the entire dataset is not available all together but the data comes sequentially, the HSVR model cannot be applied efficiently. In order to update the already configured model with respect to new points the entire configuration procedure has to be repeated. This is generally not suitable for the cases in which the configuration of the model has to be performed in real-time with respect to the data acquisition. On-line configurations of SVM models for classification have been proposed in [159] [44]. More recently on-line configurations of SVR models have been developed [109] [184].

In [109] an incremental algorithm based on updates of SVR approximation whenever a new sample is added to the training set is proposed. The weight associated to the 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.

The on-line configuration of HSVR has to consider that the updating of a layer determines the change of the residuals used in the configuration of the next layers. The procedure of updating, hence, has to be repeated for each layer. This is very time consuming and some approximation strategies have to be developed in order to allow real-time configuration.

Since the good performances of HSVR, procedures like that one proposed in [109] should be investigated to realize the on-line configuration of hierarchical model too.

6.2.2 Implementation on parallel Hardware of On-line hierarchical models

The surface reconstruction is, generally, a computationally expensive procedure: using strategies to save computational time can be important. The computational power saved can be devoted to improve the accuracy of the solution. A common way to perform a task in less time is, if possible, dividing it in subtasks and perform them simultaneously.

Parallel computing is the simultaneous use of multiple computational resources to solve a computational problem. In fact, often, a computational task can be divided into small subproblems and they can be solved concurrently. Parallel processing is an established approach for high-performance computing. The usage of highly parallel hardware has become commonplace in many areas, encompassing, e.g., high-energy physics, image processing, multimedia and life science. The development of parallel architectures and tools for parallel processing continues to challenge computer architects and software engineers. The parallel computing offers to scientist an effective tool to solve complex problems.

An example of popular parallel hardware used in research community is the Graphics Processing Unit (GPU) [108]. GPU is a dedicated graphics rendering device for a personal computer, workstation and game console. GPUs are co-processors that have

been heavily optimized for computer graphics processing. Computer graphics processing is a field dominated by data parallel operations, particularly linear algebra matrix operations. These chips are not only a powerful graphics engine but they are, also, high parallel programmable processors. Furthermore, the GPUs are, nowadays, the most powerful computational hardware for dollar present on the market [128] [129]. The GPU's rapid increase in both programmability and capability has brought the research community to use these systems to solve many kinds of problems.

The HRBF configuration procedure, both on-line and batch, is characterized by local operations, which can be performed in parallel. If the configuration of a layer is considered, the estimation of the weights of the units can be realized completely in parallel because the estimates are independent among each other.

The use of GPUs can then be well applied to the HRBF model. Furthermore for the 3D scanning, the GPU can be used both for the configuration of the model parameters and for the rendering of the surface computed by the model. Some preliminary results seem to be satisfactory and promising.

The configuration procedure of the SVR is characterized by the minimization of a convex problem. The optimization is performed on the space of the coefficients of input points. This optimization is generally realized using Sequential Minimal Optimization (SMO) that performs the optimization considering two coefficients at a time and freezing the others [43]. Despite the fact that this formulation cannot be implemented on parallel hardware, some variants of the SMO algorithm have been proposed and they have been implemented on parallel hardware [36] [34]. The implementation presented in [34] has been realized for CUDA architecture and shows a speed-up of 10 to 70 times over libSVM (one of the most popular SVM implementation).

Since the good results obtained with the CUDA architecture for SVM, an implementation of HSVR (and On-line HSVR) on CUDA may be interesting.

6.2.3 Hierarchical Support Vector Machine for Classification

The most works in literature regarding the Support Vector Machines are relative to their use for classification problems. As in the regression case the solution is computed as linear combination of a single kernel function. Also in this case one of the most common kernel functions is the Gaussian. The benefits found by the use of hierarchical model for regression could be well applied for the classification case.

Obviously some adaptations in the hierarchical scheme have to be made. For example the means of the residual in classification case is strongly different. In regression it is clear that the difference between the output and the target value can be considered as reconstruction error. This error can be directly used as input for the next layer. In multi classification case the error cannot generally be measured as the difference between predicted class and real class. This difference can be used as measure of error just in rare cases, for example, if a particular correlation between the labels of the classes and the closeness (from a geometric point of view) of the classes themselves is present.

It is noted that SVM algorithm for classification solves just binary classification. The extension to the multi classification case is performed transforming the multi classification task in a binary classification. This is obtained computing for each class a classification of the kind: the selected class vs. all the others. In this way it is possible to compute a multi classification using a binary classification algorithm.

Considering the hierarchical model for binary classifiers, the residual of each layer will be 0 for points that are correctly classified and $+1$ or -1 otherwise. Should the examples already correctly classified be used for the configuration of the next layer? This is the first problem that the hierarchical model has to solve. Using only examples misclassified in the previous layer would decrease the dataset size. The consequence of that may be a decrease of the configuration time and an increase of the classification error due to the lack of important training input. The second problem is how to use the information of misclassification of the examples for the configuration of the next layer. The last problem regards the computation of the output of the hierarchical model. In the regression case the output is merely the sum of the outputs of each layer, but in this case probably another strategy should be developed.

Since the absence of the ϵ hyperparameter for SVR classification, if the hyperparameters reduction of HSVR (from (ϵ, C, σ) for SVR to (ϵ, J) for HSVR) is confirmed also in the classification case, the model selection should be performed only on J parameter. This would determine a great improvement in the computational time needed to find the best solution.

Glossary

α-shape The α-shape is a generalization of Delaunay triangulation [14]. The α-shape of a set of points P is obtained from its Delaunay triangulation removing the elements (edges, triangles and tetrahedrons) that cannot be inscribed in a sphere of radius α.

Cross-validation The cross-validation is a technique used to estimate the predictive performance of a model. A dataset is partitioned into two subsets. The model is configured with one of the subsets and it is evaluated with the other one. The process is iterated on several partitioning. The model performances are evaluated averaging the results of the iterations.

Convex hull The *convex hull* of a set of points $P \subset \mathbb{R}^D$ is the intersection of convex subsets of $\mathbb{R}^D$ that contain P. For a finite set of points in $\mathbb{R}^2$, the convex hull is the convex polygon of minimum area that contains the points. The vertices of convex hull are a subset of P.

Voronoi diagram The Voronoi diagram of a set of points $P \subset \mathbb{R}^D$ is the partition in regions determined by P, such that the region corresponding to $p \in P$ contains all the points of $\mathbb{R}^D$ closer to p than any other points of P.

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 the same as the open unit ball in $\mathbb{R}^n$).

Marching Cubes The *Marching Cubes* [107] 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 other under threshold, the isosurface of value equal to threshold pass through the cube. Determining the vertices of the cube that are intersected by 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 then generated connecting the obtained triangular patches.

Octree An *octree* is a tree data structure where each node has eight children. Each child represents an eighth of the volume of the parent node. It is a data structure used for the search of points belonging to a region of the 3D space. This structure allows a logarithmic access to the data of the set.

Delaunay triangulation The Delaunay triangulation of a set of points P is composed by triangles (tetrahedrons in three dimensions) that have as vertices the points of P, such that there are not other vertices lying in the spheres circumscribed to the triangles.

References

[1] URL http://www.isite3d.com/4400lr.html. 26

[2] URL http://www.faro.com/quantum. 17

[3] URL http://www.leica-geosystems.com/en/79411.htm. 27

[4] URL http://en.wikipedia.ork/wiki/Graphics_processing_unit. 30

[5] URL http://www.faro.com/content.aspx?ct=us&content=pro&item=3. 27

[6] URL http://www.dti.unimi.it/ferrari/hrbf_online/hrbf_online.wmv. 102

[7] URL http://www.zcorp.com/Products/3D-Scanners/ZScannerandtrade-700/spage.aspx. 10

[8] D. Aha and D. Kiblerand M. Albert. Instance-based learning algorithms. *Machine Learning*, pages 37–66, 1991. 63, 64

[9] A. Alexandridis, H. Sarimveis, and G. Bafas. A new algorithm for online structure and parameter adaptation of RBF networks. *Neural Networks*, 16(7):1003–1017, 2003. 113, 114

[10] M. E. Algorri and F. Schmitt. Surface reconstruction from unstructured 3d data. In *Computer Graphics Forum*, volume 15, pages 47–60, 1996. 40, 45

[11] Y. Aloimonos. Detection of surface orientation from texture i: the case of plane. *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 584–593, 1986. 25

[12] G. C. Atkenson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11:11–73, 1997. 64, 65, 66, 88

[13] A. Baader and G. Hirzinger. A self-organizing algorithm for multisensory surface reconstruction. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, volume 1, pages 81–89, 1994. 46

[14] C. L. Bajaj, F. Bernardini, and G. Xu. Automatic reconstruction of surfaces and scalar fields from 3d scans. In *SIGGRAPH 95 Conference Proceedings*, pages 109–118, 1995. 40, 167

[15] J. Batlle, E. Mouaddib, and J. Salvi. Recent progress in coded structured light as a technique to solve the correspondence problem: A survey. *Pattern Recognition*, 31(7):963–982, 1998. 29

[16] P. N. Belhumeur, D. J. Kriegman, and A. L. Yuille. The bas-relief ambiguity. *IJCV*, pages 33–44, 1999. 31

[17] F. Bellocchio, S. Ferrari, V. Piuri, and N.A. Borghese. Online training of hierarchical RBF. In *Proceedings of IJCNN 2007 (IEEE International Joint Conference on Neural Networks)*, pages 2159–2164, August 2007. 158

[18] A. B. Berg. Locating global minima in optimisation problems by a random-cost approach. *Nature*, 361:708–710, 1993. 80

[19] F. Bernardini, C. L. Bajaj, J. Chen, and D. R. Schikore. Automatic reconstruction of 3d cad models from digital scans. *Int. J. on Comp. Geom. and Appl.*, 9(4-5): 327–370, 1999. 43

[20] P. Besl and N. McKay. A method for registration of 3-d shapes. *Trans. PAMI*, 17(8), 1992. 53

[21] S. A. Billings and G. L. Zheng. Radial basis function network configuration using genetic algorithms. *Neural Networks*, 8(6):877–890, 1995. 80

[22] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995. 10, 72, 73

[23] E. Bittar, N. Tsingos, and M. P. Gascuel. Automatic reconstruction of unstructured 3d data: Combining medial axis and implicit surfaces. *Computer Graphics Forum*, 14(3):457–468, 1995. 44

[24] J. D. Boissonat. Geometric structures for threedimensional shape representation. *ACM Trans. Graphics*, 3(4):266–286, 1984. xii, 42

[25] N. A. Borghese and S. Ferrari. Hierarchical RBF networks and local parameter estimate. *Neurocomputing*, 19(1–3):259–283, 1998. 48, 82, 88, 106

[26] N. A. Borghese and S. Ferrari. A portable modular system for automatic acquisition of 3d objects. *IEEE Transaction on Instrumentation and Measurement*, 49 (5):1128–1136, 2000. 48, 102

[27] N. A. Borghese, G. Ferrigno, G. Baroni, S. Ferrari, R. Savaré, and A. Pedotti. AUTOSCAN: a flexible and portable 3D scanner. *IEEE C. G. & A.*, 18(3):38–41, 1998. xiv, 10, 95, 102, 103

[28] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth, 1984. 66

[29] M. Brooks. Two results concerning ambiguity in shape from shading. *AAAI-83*, pages 36–39, 1983. 31

[30] D. S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988. 81

[31] C. G. Broyden. The convergence of a class of double-rank minimization algorithms. *Journal of the Institute of Mathematics and Its Applications*, 6:76–90, 1970. 74

[32] B. Schölkopf C. Burges and V. Vapnik. Extracting support data for a given task. *Proceedings of First International Conference on Knowledge Discovery and Data Mining*, 1995. 117

[33] B. Schölkopf C. Burges and V. Vapnik. Incorporating invariances in support vector learning machines. *Artificial Neural Networks ICANN 96*, 1112:47–52, 1996. 117

[34] A. Carpenter. cuSVM: a CUDA implementation of support vector classification and regression. In *Technical Report*, 2009. 164

[35] R. Beltran Catalan, E. Islas Perez, and B. Zayas Perez. Evaluation of 3d scanners to develop virtual reality applications. *Fourth Congress of Electronics, Robotics and Automotive Mechanics*, pages 551–556, 2007. 35

[36] B. C. Catanzaro, N. Sundaram, and K. Keutzer. Fast support vector machine training and classification on graphics processors. In *Technical Report No. UCB/EECS-2008-11*, 2008. 164

[37] B.C. Cetin, J. Barhen, and J.W. Burdick. Terminal repeller unconstrained subenergy tunnelling (trust) for fast global optimization. *Journal of Optimization Theory and Application*, 77(1):97–126, 1993. 80

[38] Y. T. Chan. *Wavelet Basics*. Kluwer Academic Publishers, 1995. 51

[39] S. Chen, X. Hong, and C. J. Harris. Sparse kernel density construction using orthogonal forward regression with leave-one-out test score and local regularization. *IEEE Trans. on System, Man, and Cybernetics*, 34(4):1708–1717, 2004. 81

[40] P. Clark and T. Niblett. The cn2 induction algorithm. *Machine Learning*, 3: 261–283, 1989. 67

[41] T.M. Cover and P. E. Hart. Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, pages 21–27, 1967. 63

[42] P. Craven and G. Wahba. Smoothing noisy data with spline functions. Estimating the correct degree of smoothing by the method of generalized cross-validation. *Numer. Math.*, 31(4):377–403, 1978/79. ISSN 0029-599X. 104

[43] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000. 164

[44] L. Csato and M. Opper. Sparse representation for gaussianprocess models. *Advances in neural information processing systems*, 13:444–450, 2001. 162

[45] B.V. Dasarathy. Nearest neighbor (nn) norms: Nn pattern classifcation techniques. *IEEE Press*, 1991. 63

[46] I. Daubechies. Orthonormal bases od compactly supported wavelets. *Comm. on Pure and Applied Mathematics*, 41:909–996, 1988. 75

[47] I. Daubechies. *Ten lectures on Wavelets*. Philadelphia: SIAM, 1992. 75

[48] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Computer Graphics, 30(Annual Conference Series)*, pages 11–20, 1996. 37

[49] D. L. Donoho and I. M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 1993. 75

[50] C. Dorai, G. Wang, A. K. Jain, and C. Mercer. Registration and integration of multiple object views for 3d model construction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):83–89, 1998. 54

[51] W. S. Dorn. Duality in quadratic programming. *Quarterly of Applied Mathematics*, 18(2):155–162, 1960. 122

[52] G. Dreyfus. *Neural Networks: methodology and applications*. Springer, 2005. 72

[53] H. Drucker, C.J.C. Burges, L. Kaufman A. Smola, and V. Vapnik. Support vector regression machines. *Advances in Neural Information Processing Systems 9*, pages 151–161, 1997. 118

[54] H. Edelsbrunner and E. P. Mucke. Three-dimensional alpha shapes. *ACM Trans. Graphics*, 13(1):43–72, 1994. 41

[55] F. Evans, S. Skiena, and A. Varshney. Optimizing triangle strips for fast rendering. In *IEEE Visualization 96 Proceedings*, volume 2, pages 319–326, 1996. 56

[56] J. Fan, N. Dimitrova, and V. Philomin. Online face recognition system for videos based on modified probabilistic neural networks. In *Proceedings of the 2004 International Conference on Image Processing*, volume 3, pages 2019–2022, 2004. 10

[57] L. A. Feldkamp, L. C. Davis, and J.W. Kress. Practical conebeam algorithm. *J. Opt. Soc. Am. A*, 1:612–619, 1984. 18

[58] S. Ferrari, N. A. Borghese, and V. Piuri. Multiscale models for data processing: an experimental sensitivity analysis. *IEEE Trans. on I. & M.*, 50(4):995–1002, 2001. 113

[59] S. Ferrari, M. Maggioni, and N. A. Borghese. Multi-scale approximation with hierarchical radial basis functions networks. *IEEE Trans. on Neural Networks*, 15(1):178–188, January 2004. 11, 82, 96, 101, 106, 140, 153

[60] S. Ferrari, I. Frosio, V. Piuri, and N. A. Borghese. Automatic multiscale meshing through hrbf networks. *IEEE Trans. Instrum. Meas.*, 54(4):1463–1470, 2005. 97

[61] S. Ferrari, G. Ferrigno, V. Piuri, and N. A. Borghese. Reducing and filtering point clouds with enhanced vector quantization. *IEEE Trans. Neural Netw.*, 18 (1):161–177, 2007. 97

[62] S. Ferrari, F. Bellocchio, V. Piuri, and N. A. Borghese. A hierarchical scheme for multi-kernel support vector regression. *Submitted to IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2010. 158

[63] S. Ferrari, F. Bellocchio, V. Piuri, and N. A. Borghese. A hierarchical RBF online learning algorithm for real-time 3-D scanner. *IEEE Trans. on Neural Networks*, 21(2):275–285, February 2010. 82, 140, 147, 153, 154, 158

[64] S. Ferrari, F. Bellocchio, V. Piuri, and N.A. Borghese. Multi-scale support vector regression. In *Proceedings of IJCNN 2010 (IEEE International Joint Conference on Neural Networks)*, pages 2159–2164, July 2010. 158

[65] C. Ford and D.M. Etter. Wavelet basis reconstruction of nonuniformly sampled data. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 45(8):1165–1168, 1998. 51

[66] D. R. Forsey and R. H. Bartels. Hierarchical b-spline refinement. *Computer Graphics*, 22(4):205–212, 1988. 49

[67] D. R. Forsey and D. Wong. Multiresolution surface reconstruction for hierarchical b-splines. *Graphics Interface*, pages 57–64, 1998. 49

[68] J. Friedman. Multivariate adaptive regression splines. *Annals of Statistics*, 19 (1):1–141, 1991. 66, 69, 71

[69] J. H. Friedman and W. Stuetzle. Projection pursuit regression. *Journal of the American Statistical Association*, 76:817–823, 1981. 68, 69

[70] B. Fritzke. Growing cell structures — A self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441–1460, 1994. 81, 113

[71] B. Fritzke. Growing grid — A self-organizing network with constant neighborhood range and adaptation strength. *Neural Processing Letters*, 2(5):9–13, 1995. 81

[72] G. M. Fung, O. L. Mangasarian, and A. J. Smola. Minimal kernel classiers. *J. Mach. Learn. Res.*, 3:2303–321, 2002. 152

[73] M. C. Gambino, R. Fontana, G. Gianfrate, M. Greco, L. Marras, M. Materazzi, E. Pampaloni, and L. Pezzati. *A 3D Scanning Device for Architectural Relieves Based on Time-Of-Flight Technology.* Springer Berlin Heidelberg, 2005. ISBN 978-3-540-22996-4. 26

[74] F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219–269, 1995. 80, 154

[75] D. Gorse, A. Sheperd, and J.G. Taylor. Avoiding local minima by a classical range expansion algoritm. In *In Proceedings of ICANN 94*, 1994. 80

[76] J. Guo, N. Takahashi, and T. Nishi. An efcient method for simplifying decision functions of support vector machines. *IEICE Trans. Fund.*, (10):2795–2802, 2006. 152

[77] P. Hall. On projection pursuit regression. *Annals of Statistics*, 17(2):573–588, 1989. 68

[78] M. Hasenjäger and H. Ritter. New learning paradigms in soft computing. In *Active learning in neural networks*, pages 137–169. Physica-Verlag GmbH, 2002. ISBN 3-7908-1436-9. 114

[79] P. S. Heckbert and M. Garland. Optimal triangulation and quadric-based surface simplification. *Computational Geometry*, 14:49–65, 1998. 56

[80] C. Hernandez, G. Vogiatzis, and R. Cipolla. Multiview photometric stereo. *IEEE Trans. on PAMI*, 30(3):548–554, 2008. 32

[81] J. Hertz, A. Krogh, and R. G. Palmer. *An Introduction to the Theory of Neural Computation*. Addison Wesley, 1991. 72, 79

[82] T. Higo, Y. Matsushita, N. Joshi, and K. Ikeuchi. A hand-held photometric stereo camera for 3d modeling. In *Proc in Computer Vision and Pattern Recognition*, pages 1234–1241, 2009. 32

[83] H. Hoppe. *Surface Reconstruction from Unorganized Points*. PhD Thesis, Dept. of Computer Science and Engineering, University of Washington, 1994. 8

[84] H. Hoppe, T. DeRose, T. Duchamp, J. Mc-Donald, and W. Stuetzle. Surface reconstruction from unorganized points. In *Computer Graphics (SIGGRAPH 92 Proceedings)*, volume 26, pages 71–78, 1992. 40, 43

[85] B. Horn. Obtaining shape from shading information. *The Psychology of Computer Vision*, 1975. 31

[86] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991. 73

[87] P.S. Huang and S. Zhang. Fast three-step phase shifting algorithm. *Applied Optics*, 45(21):5086–5091, 2006. 30

[88] M. Idesawa, T. Yatagai, and T. Soma. Scanning moir method and automatic measurement of 3-d shapes. *APPLIED OPTICS*, 16(8):2152–2162, 1977. 33

[89] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11, pages 169–184. MIT Press, Cambridge, MA, 1999. 141

[90] Y. Katznelson. *An introduction to Harmonic Analysis*. Dover, 1976. 84

[91] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data – An Introduction to Cluster Analysis*. Wiley, 1990. 68

[92] S. S. Keerthi, O. Chapelle, and D. De Coste. Building support vector machines with reduced classier complexity. *J. Mach. Learn. Res.*, 7:1493–1515, 2006. 152

[93] D. Kibler, D. W. Aha, and M. K. Albert. Instance-based prediction of real-valued attributes. *Comput. Intell.*, pages 51–57, 1989. 63

[94] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimisation by simulated annealing. *Science*, 220:671–680, 1983. 80

[95] S. Klinke and J. Grassmann. Projection pursuit regression and neural network. Technical report, Humboldt Universitaet Berlin, 1998. 68, 69

[96] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, volume 2, pages 1137–1143, 1995. 64

[97] T. Kohonen. *Self-Organizing Maps*. Springer, Berlin, Heidelberg, 1995. 45

[98] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Proceedings of 2nd Berkeley Symposium*, pages 481–492. Berkeley: University of California Press, 1951. 122

[99] G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semi-definite programming. *Journal of Machine Learning Research*, 5:27–72, 2004. 134

[100] A. Lee, H. Moreton, and H. Hoppe. Displaced subdivision surfaces. In *SIGGRAPH 2000 Conference Proceedings, Annual Conference Series*, pages 85–94, 2000. 57

[101] A. W. F. Lee, W. Sweldens, P. Schroder, L. Cowsar, and D. Dobkin. Maps: Multiresolution adaptive parameterization of surfaces. In *SIGGRAPH 98 Conference Proceedings, Annual Conference Series*, pages 95–104, 1998. 56, 57

[102] S. Lee, G. Wolberg, and S. Y. Shin. Scattered data interpolation with multilevel b-splines. *IEEE Transactions on Visualization and Computer Graphics*, 3(3): 228–244, 1997. 49

[103] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *The Quarterly of Applied Mathematics*, 2:164–168, 1944. 74

[104] A. Levin, R. Fergus, F. Durand, and W. T. Freeman. Image and depth from a conventional camera with a coded aperture. *ACM Transactions on Graphics*, 26 (3):70–1–70–9, 2007. 25

[105] M. Levoy, S. Rusinkiewicz, M. Ginzton, J. Ginsberg, K. Pulli, D. Koller, S. Anderson, J. Shade, B. Curless, L. Pereira, J.David, and D. Fulk. The digital michelangel project: 3d scanning of large statues. *Proc. ACM SIGGRAPH 2000*, 2000. xi, 5

[106] X. Liang. An effective method of pruning support vector machine classiers. *IEEE Trans. on Neural Networks*, 21(1):26–38, 2010. 140, 152, 153

[107] W. E. Lorensen and H. E. Cline. Marching cubes: a high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987. 168

[108] D. Luebke and G. Humphreys. *How gpus work*. IEEE Computer, 2007. 163

[109] J. Ma, J. Theiler, and S. Perkins. Accurate on-line support vector regression. *Neural Computation*, 15:2683–2703, 2003. 162, 163

[110] S. Mallat. A theory for multiscale signal decomposition: The wavelet representation. *IEEE Trans. on Pattern and Machine Intelligence*, 11(7):674–693, 1989. 51, 75

[111] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11:431–441, 1963. 74

[112] T.M. Martinetz, S.G. Berkovich, and K.J. Schulten. neural-gas network for vector quantization and its application to time-series prediction. *IEEE Trans. on Neural Networks*, 4(4):558–568, 1993. 81

[113] J. I. M. Martínez. Best approximation of gaussian neural networks with nodes uniformly spaced. *IEEE Trans. on Neural Networks*, 19(2):284–298, 2008. 114

[114] T. J. Mckinley, M. McWaters, and V. K. Jain. 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*, pages 148–155, 2001. 22

[115] R. Mencl and H. Muller. Interpolation and approximation of surfaces from three-dimensional scattered data points. In *STAR State of The Art Report Eurographics 98*, pages 51–67, 1998. 39

[116] R. Mencl and H. Muller. Graph-based surface reconstruction using structures in scattered point sets. In *Proceedings of the Conference on Computer Graphics International*, pages 298–311, 1998. 46

[117] Q. Meng and M. Lee. Error-driven active learning in growing radial basis function networks for early robot learning. *Neurocomputing*, 71(7-9):1449–1461, 2008. ISSN 0925-2312. doi: http://dx.doi.org/10.1016/j.neucom.2007.05.012. 113

[118] R. Michalsky, I. Mozetic, J. Hong, and N. Lavrac. The multi-purpose incremental learning syste, aq15 and its testing application to three medical domains. In *In proceedings of AAAI-86*, pages 1041–1045, 1986. 67

[119] J. V. Miller, D. E. Breen, W. E. Lorensen, R. M. OBara, and M. J. Wozny. Geometrically deformed models: a method for extracting closed geometric models form volume data. *Computer Graphics*, 25(4):217–226, 1991. 44

[120] J. Moody and C. J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–294, 1989. 79, 81

[121] F. Morcin and N. Garcia. Hierarchical coding of 3d models with subdivision surfaces. In *Image Processing, 2000*, volume 2, pages 911–914, 2000. 56

[122] K.-R. Müller, A. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik. Predicting time series with support vector machines. *Artificial Neural Networks ICANN 97*, 1327:999–1004, 1997. 118

[123] K. S. Narendra and K. Parthasarathy. Gradient methods for the optimization of dynamical systems containing neural networks. *IEEE Transactions on Neural Networks*, 2(2):252–262, 1992. 80

[124] K. S. Narendra and M. A. L. Thathachar. *Learning Automata - An Introduction*. Prentice Hall, 1989. 80

[125] D. Nguyen and T. Ho. An efficient method for simplifying support vector machines, 2005. 140, 152, 153

[126] S.G. Nikolov, D.R. Bull, C.N. Canagarajah, M. Halliwell, and P.N.T. Wells. Image fusion using a 3-d wavelet transform. In *In Seventh International Conference on Image Processing And Its Applications*, pages 235–239, 1999. 55

[127] M. J. L. Orr. Regularization in the selection of radial basis function centers. *Neural Computation*, 7(3):606–623, 1993. 81

[128] J. D. Owens, D. Luebke, N. G. M. Harris, J. Kruger, A. E. Lefohn, and T. J. Purcell. A survey of general-purpose computation on graphics hardware. In *Eurographics*, pages 21–25, 2005. 164

[129] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips. Gpu computing. In *Proceedings of the IEEE*, volume 96, pages 879–899, 2008. 164

[130] K. Park, I. D. Yun, and S. U. Lee. Automatic 3-d model synthesis from measured range data. *IEEE Transaction on Circuits and Systems for Video Technology*, 10 (2):293–301, 2000. 41

[131] L. Pastor and A. Rodriguez. Surface approximatio of 3d objects from irregularly sampled clouds of 3d points using spherical wavelets. In *International Conference on Image Analysis and Processing*, pages 70–75, 1999. 51

[132] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990. 81

[133] L. Piegel and W. Tiller. *The NURBS Book*. Springler-Verlag, 1997. 4

[134] J. Platt. A resource-allocating network for function interpolation. *Neural Computation*, 3:213–225, 1991. 79, 81

[135] T. Poggio. A theory of how the brain might work. In *Technical Report AIM-1253, Massachusetts Institute of Technology*, 1990. 80

[136] T. Poggio and F. Girosi. Network for approximation and learning. In *Proceedings of the IEEE*, volume 78, pages 1481–1497, 1990. 72, 79

[137] M. Potmesil. Generating octree models of 3d objects from their silhouettes in a sequence of images. *CVGIP*, (40):1–29, 1987. 24

[138] E. Praun, H. Hoppe, and A. Finkelstein. Robust mesh watermarking. In *SIGGRAPH 99 Conference Proceedings, Annual Conference Series*, pages 49–56, 1999. 57

[139] F. P. Preparata and M. I. Shamos. *Computational Geometry - An Introduction.* Springer-Verlag, 1985. 39

[140] S. Qiu and T. Lane. Multiple kernel learning for support vector regression. Technical report, Computer Science Department, The University of New Mexico, Albuquerque, NM, USA, 2005. 134

[141] L. Rejtö and G. Walter. Remarks on projection pursuit regression and density estimation. *Stochastic Anal. Appl.*, 10:213–222, 1992. 69

[142] C. Roosen and T. Hastie. Automatic smoothing spline projection pursuit. *Journal of Computational and Graphical Statistics*, 3:235–248, 1994. 69

[143] J. Rossignac. Edgebreaker: Connectivity compression for triangle meshes. *GVU Technical Report GIT-GVU-98-35*, 1998. 56

[144] G. Roth and E. Wibowoo. An efficient volumetric method for building closed triangular meshes from 3-d image and point data. In *Graphics Interface*, pages 173–180, 1997. 40

[145] A. Rubaai, R. Kotaruand, and M.D. Kankam. Online training of parallel neural network estimators for control of induction motors. *IEEE Transaction on Industry Applications*, 37(5):1512–1521, 2001. 10

[146] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, 1:318–362, 1986. 74

[147] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy. Real-time 3d model acquisition. In *Proceeding of the*, pages 438–446. ACM Press, 2002. 10, 30

[148] D. Saad. *Online Learning in Neural Networks*. Cambridge University Press, 1998. 10

[149] R. M. Sanner and J. E. Slotine. Gaussian networks for direct adaptive control. *IEEE Transactions on Neural Networks*, 3(6):837–863, 1992. 81, 114

[150] T. Schreiber and G. Brunnett. Approximating 3d objects from measured points. *In Proceedings of 30th ISATA*, 1997. 42

[151] P. Schroder and W. Sweldens. Spherical wavelets: Efficiently representing functions on the sphere. In *SIGGRAPH 95 Conference Proceedings*, pages 161–172, 1995. 51

[152] A.J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, 2004. 117, 140

[153] A.J. Smola, N. Murata, B. Schölkopf, and K.-R. Müller. Asymptotically optimal choice of ϵ-loss for support vector machines. In *Proceedings of the 8th International Conference on Artificial Neural Networks, Perspectives in Neural Computing*, pages 105–110. Springer Verlag, 1998. 138

[154] D. F. Specht. Probabilistic neural networks. *Neural Networks*, 3:109–118, 1990. 81

[155] M. Stitson, A. Gammerman, V. Vapnik, V. Vovk, C. Watkins, and J. Weston. Support vector regression with anova decomposition kernels. *Advances in Kernel MethodsSupport Vector Learning*, pages 285–292, 1999. 118

[156] W. Sweldens. The lifting scheme: A new philosophy in biorthogonal wavelet constructions. *Wavelet applications in signal and image processing III, volume 2569 of Proceedings of SPIE*, pages 68–79, 1995. 77

[157] W. Sweldens. The lifting scheme: A custom-design construction of biorthogonal wavelets. *Appl. Comput. Harmon. Anal.*, 3(2):186–200, 1996. 77

[158] W. Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM J. Math. Anal.*, 29(2):511–546, 1997. 77

[159] N. A. Syed, H. Liu, and K. K. Sung. Incremental learning with support vector machines. In *In Proceedings of the Workshop on Support Vector Machines at the International Joint Conference on Artificial IntelligenceIJCAI-99*, 1999. 162

[160] R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. *Computer Graphics*, 26(2):185–194, 1992. 44

[161] Yaohua Tang, Weimin Guo, and Jinghuai Gao. Efficient model selection for support vector machine with gaussian kernel function. In *Computational Intelligence and Data Mining, 2009. CIDM '09. IEEE Symposium on*, pages 40–45, 2009. 137

[162] P. Taylan and G.-W. Weber. Multivariate adaptive regression spline and continuous optimization for modern applications in science, economy and techology. Technical report, Humboldt Universitaet Berlin, 2004. 71

[163] D. Terzopoulos and D. Metaxas. Dynamic 3d models with local and global deformations: deformable superquadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):703–714, 1991. 45

[164] I.W. Tsang, J.T. Kwok, and Pak-Ming Cheung. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6: 363–392, 2005. 141

[165] G. Turk and M. Levoy. Zippered polygon meshes from range images. In *Computer Graphics, 28(Annual Conference Series)*, pages 311–318, 1994. 53, 55

[166] R. Vaillant and O. Faugeras. Using extremal boundaries for 3d object modelling. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2(14):157–173, 1992. 24

[167] V. Vapnik. *Statistical Learning Theory.* Wiley-Interscience, 1989. 9, 118

[168] V. Vapnik and A. Chervonenkis. Theory of pattern recognition. *Nauka, Moscow*, 1974. 117, 118

[169] V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24:774–780, 1963. 117

[170] D. Visintini, A. Spangher, and B. Fico. The vrml model of victoria square in gorizia (italy) from laser scanning and photogrammetric 3d surveys. *Proc in Web3D 2007*, pages 165–169, 2007. 27

[171] Dong Wang, Xiang-Bin Wu, and Dong-Mei Lin. Two heuristic strategies for searching optimal hyper parameters of c-svm. In *Proceedings of the Eighth International Conference on Machine Learning and Cybernetics*, pages 3690–3695, 2009. 137

[172] Zhe Wang, Songcan Chen, and Tingkai Sun. MultiK-MHKS: A novel multiple kernel learning algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):348–353, 2008. ISSN 0162-8828. doi: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2007.70786. 134

[173] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning Journal*, 8(3), 1992. 80

[174] S. Weiss and N. Indurkhya. Optimized rule induction. *IEEE Expert*, 8(6):61–69, 1993. 66, 67

[175] S. Weiss and N. Indurkhya. Rule-based machine learning methods for functional prediction. *Journal of Artificial Intelligence Research*, 3:383–403, 1995. 66, 67, 68

[176] C. Wohler. *3D Computer Vision: Efficient Methods and Applications.* Springer, 2009. 21

[177] R. J. Woodham. Photometric method for determining surface orientation from multiple images. *Optical Engineering*, 19(1):139–144, 1980. 32

[178] C. Wust and D. W. Capson. Surface profile measurement using color fringe projection. *Mach. Vision Appl.*, 4:193–203, 1991. 30

[179] R. Yang and P. K. Allen. Registering, integrating and building cad models from range data. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-98)*, pages 3115–3120, 1998. 54

[180] A. L. Yuille. A computational theory for the perception of coherent visual motion. *Nature*, 333:71–74, 1988. 80

[181] X. Zeng and X. Chen. Smo-based pruning methods for sparse least squares support vector machines. *IEEE Trans. on Neural Networks*, 16(6):1541–1546, 2005. 152

[182] H. Zha, T. Hoshide, and T. Hasegawa. A recursive fitting-and-splitting algorithm for 3-d object modeling using superquadrics. In *Proceedings of the International Conference on Pattern Recognition*, volume 1, pages 658–662, 1998. 47

[183] S. Zhang, D. Royer, and S. Yau. Gpu-assisted high-resolution, real-time 3-d shape measurement. *OPTICS EXPRESS*, 14(20):9120–9129, 2006. 30

[184] Y. Zhenhua, F. Xiao, and L. Yinglu. Online support vector regression for system identification. *Advances in Natural Computation*, 3611:627–630, 2005. 162